Explicit Building Block Multiobjective Evolutionary Computation: Methods and Applications

Richard O. Day

EXPLICIT BUILDING BLOCK MULTIOBJECTIVE EVOLUTIONARY COMPUTATION: METHODS AND APPLICATIONS

DISSERTATION

Richard Orison Day, Captain, USAF

AFIT/DS/ENG/05-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENG/05-03

EXPLICIT BUILDING BLOCK MULTIOBJECTIVE EVOLUTIONARY COMPUTATION: METHODS AND APPLICATIONS

DISSERTATION

Presented to the Faculty of the
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Richard Orison Day, B.S.C.E., M.S.C.E.
Captain, USAF

June, 2005

AFIT/DS/ENG/05-03

EXPLICIT BUILDING BLOCK MULTIOBJECTIVE EVOLUTIONARY COMPUTATION: METHODS AND APPLICATIONS

Richard Orison Day, B.S.C.E., M.S.C.E.

Captain, USAF

Approved:

Gary B. Lamont

Gary B. Lamont, Ph.D
Research Committee Chairman

13 June 2005
date

Will P. Baker

William P. Baker, Ph.D.
Research Committee Member

13 Jun 05
date

Gilbert L. Peterson, Ph.D.
Research Committee Member

15 Jun 05
date

David A. Van Veldhuizen

Major David Van Veldhuizen, Ph.D.
Research Committee Member

17 Jun 05
date

Michael A. Marciniak, Ph.D.
Dean's Representative

20 Jun 05
date

Accepted:

29 Jun 05

Robert A. Calico, Jr. Date
Dean, Graduate School of Engineering and Management

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

Acknowledgements

I thank God for blessing me with this opportunity and for the wisdom gifted to my mentors.

> . . . My son, beware of anything beyond these. Of making many books there is no end, and much study is a weariness of the flesh. [215]

Table of Contents

List of Figures

Figure Page

Figure Page

Figure Page

List of Tables

Table Page

Table Page

Table Page

List of Algorithms

List of Abbreviations

Abbreviation		Page
MOPs	Multiobjective Optimization Problems	2
USAF	United States Air Force	2
EAs	Evolutionary Algorithms	2
EC	Evolutionary Computation	2
OR	Operations Research	2
NPC	NP-complete	2
VMP	Vector Maximum Problem	3
VEGA	Vector Evaluated Genetic Algorithm	5
MOEAs	Multiobjective Evolutionary Algorithms	5
BB	Building Block	5
MOMGA-II	Multiobjective messy Genetic Algorithm	7
DM	Decision Maker	7
MPI	Message Passing Interface	8
PVM	Parallel Virtual Machine	8
pfmGA	parallel fast messy Genetic Algorithm	8
CHARMm	Chemistry at HARvard Molecular mechanics	9
BBB	Building Block Builder	19
CIs	Confidence Intervals	29
GMBB	Good Multiobjective Building Blocks	41
mGA	messy GA	57
fmGA	fast messy GA	57
IEA	Intelligent Evolutionary Algorithm	57
PMBGAs	Probabilistic Model-Building GAs	57
EDA	Estimation of Distribution Algorithms	57

Abbreviation		Page
BOA	Bayesian Optimization Algorithm	57
PCI	Probabilistic Complete Initialization	58
BBF	Building Block Filtering	58
IGC	Intelligent Gene Collector	59
OED	Orthogonal Experimental Design	59
MED	Main Effect Difference	59
MOMGA	Multiobjective messy GA	59
PEI	Partially Enumerative Initialization	59
CTMS	Competitive Template Managing System	61
BSC	Bit-Based Simulated Crossover	62
PBIL	Population-Based Incremental Learning	62
UMDA	Univariate Marginal Distribution Algorithm	62
GBaBB	Good Bayesian Building Block	67
BaBB	Bayesian Building Block	67
BD	Bayesian-Dirichlet	68
BIC	Bayesian information criteria	68
PPDP	Preserve, Progress, Diversity and Provide	83
ALP	Advanced Logistics Problem	85
MMOKP	Modified Multiobjective Knapsack Problem	85
ESTM	Evolutionary Solution Tracing Mechanism	88
TV	Target Vector	88
PFS	Pareto Front Structure	94
OMOEA	Orthogonal Multiobjective Evolutionary Algorithm for Multiobjective Optimization	108
HFC	Hierarchical Fair Competition	123
ER	Error Ratio	134
GD	Generational Distance	135

Abbreviation		Page
HV	Hypervolume	135
S	Spacing	136
ONVG	Overall Non-dominated Vector Generation	137
ONVGR	Overall Non-dominated Vector Generation Ratio	137
ME	Maximum Pareto Front Error	137
KWtest	Kruskal-Wallis	142
ANOVA	Analysis of Variance	142
df	degrees of freedom	143
PSP	Protein Structure Prediction Problem	191
MET	Met-Enkephalin	193
MLPNN	Multilayered Perceptron Neural Network	194
RBFNN	Radial Basis Function Neural Network	195
MOfmGA	Generic Multiobjective fmGA	196
OPLS	Optimized Potentials for Liquid Simulations	199
REM	Random Energy Model	199
OAV	Organic Air Vehicles	218
IUAV	Interrogator UAV	219
UAV	Unmanned Aerial Vehicle	219
TUAV	Transport UAV	220
OAVFFP	OAV flight formation problem	220
LS	Local Search	223
SOP	Single Objective Problem	238
RF	radio frequency	271
ES	Evolutionary Strategies	328
TUB	Technical University of Berlin	328
GAs	Genetic Algorithms	328
EP	Evolutionary Programming	330

Abbreviation		Page
GP	Genetic Programming	331
ssGA	steady state GA	332
sGA	simple Genetic Algorithm	332
EVOPs	Evolutionary Operators	332
CT	Competitive Template	335
IDEAs	Iterated Density Estimation Algorithms	341
ECGA	Extended Compact GA	341
MDL	Minimum Description Length	342
MPMs	Marginal Product Models	343
cGA	Compact GA	343
SA	Simulated Annealing	344
MIMIC	Mutual Information Maximizing Input Clustering	345
COMIT	Combining Optimizers with Mutual Information Tress	345
BMDA	Bivariate Marginal Distribution Algorithm	345
FDA	Factorized Distribution Algorithm	345
EBNA	Estimation of Bayesian Network Algorithm	345
MOGA	Multiobjective Genetic Algorithm	348
NPGA	Niched-Pareto Genetic Algorithm	349
NSGA	Non-dominated Sorting Genetic Algorithm	350
SPEA	Strength Pareto Evolutionary Algorithm	353
PAES	Pareto Archived Evolution Strategy	355
PESA	Pareto Envelop-based Selection Algorithm	357
MOSGA	Multiobjective Struggle Genetic Algorithm	357
OMEA	Orthogonal Multiobjective Evolutionary Algorithm	359
GENMOP	General Multiobjective Evolutionary Algorithm	360
AFIT	Air Force Institute of Technology	360
IMOEA	Intelligent Multiobjective Evolutionary Algorithm	368

Abbreviation		Page
mBOA	Multiobjective Bayesian Optimization Algorithm	370
LPOPs	Large Parameter Optimization Problems	373
CORBA	Common Object Request Broker Architecture	379
ORB	Object Request Broker	379
DFFT	Discrete Fast Fourier Transform	380
IFFT	Inverse FFT	382
FPGAs	Field Programmable Gate Array	385
DoD	Department of Defense	386
BADD	Battlefield Awareness Data Dissemination	386
APK	amplitude and phase keying	388
SNR	signal-to-noise ratio	388
PSK	phase-shift keying	388
DOE	Design of Experiments	403
NP-hard	Nondeterministic Polynomial Time Hard Problem	420

List of Symbols

Symbol		Page
$g_j(\mathbf{X})$	Additional side constraint function j within an MOP	3
k	Number of objectives within an MOP	3
$f_j(\mathbf{X})$	Objective function j within an MOP	3
n	Number of decision variable within an SOP or MOP	4
m	Number of side constraints within an SOP or MOP	4
ϵ	$\epsilon \in \mathbb{R}^n$ where $\epsilon > 0$	10
Ω	Set of all possible solutions	11
$\preceq$	Right side Pareto Dominates Left side	11
$\mathcal{P}^*$	Pareto Optimal Set	11
$\mathcal{PF}^*$	Pareto Front	11
$\underset{\epsilon}{\preceq}$	Left side Pareto epsilon Dominates Right side	12
$\mathcal{P}^*_\epsilon$	Pareto epsilon (ϵ) Optimal Set	12
$\mathcal{PF}^*_\epsilon$	Pareto epsilon (ϵ) Front	12
$\ddot{n}$	$\ddot{n}$ represents a countable or infinite countable set	13
$\ddot{u}$	Represents an uncountable set	13
P_{true}	Computational true Pareto Optimal Set	14
PF_{true}	Computational true Pareto Front set	14
P_{known}	Computational Pareto Optimal Set best found	14
PF_{known}	Computational Pareto Front set found	14
$\mathrm{P}^{\epsilon}_{true}$	Computational true Pareto ϵ Optimal Set.	15
$\mathrm{PF}^{\epsilon}_{true}$	Computational true Pareto ϵ Front set	15
$\mathrm{P}^{\epsilon}_{known}$	Computational Pareto ϵ Optimal Set found	15
$\mathrm{PF}^{\epsilon}_{known}$	Computational Pareto ϵ Front set found	15
$\mathcal{A}$	The alphabet set	21

Symbol		Page
ℓ	Number of locus positions in a complete solution	21
$\mathbf{o}$	$\mathbf{o}$ is designated as the size of a BB ($\tilde{b}$)	21
$\tilde{b}$	$\tilde{b}$ is an order-$\mathbf{o}$ building block	21
l_i	Represents the loci position for the ith bit in the BB	21
Λ	The set of loci positions available in a complete solution	23
$\mathbb{P}$	Denotes the set of every possible solution in the search space	25
$\mathbb{P}$	Denotes the power set.	25
$\mathcal{C}$	Indicates a single complete solutions (fully specified)	27
R	Indicates the Replacement Operator ($R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\}$)	27
$\mathcal{M}$	Matrix of all permutations of a $\mathbf{o}$ length variable string	27
$\mathbb{BB}^{(o)}$	Set of all order-$\mathbf{o}$ BBs in $\mathbb{P}$	28
$\mathcal{W}$	set of all order-$\mathbf{o}$ BBs with one BB, $\tilde{b}$, removed	28
$\mu_{\tilde{b}}$	Mean of all fitness values resulting from overlaying $\tilde{b}$ onto each member of the entire population $\mathbb{P}$	28
$\mu_{\mathcal{W}}$	Mean of all fitness values resulting from overlaying $\mu_{\mathcal{W}}$ onto each member of the entire population $\mathbb{P}$	28
c_{κ_1}	Lower CI value with α significance level for sample data κ	31
c_{κ_2}	Upper CI value with α significance level for sample data κ	31
α	The *significance level* for a confidence interval	31
κ	Represents a sample pool of data or a single datum point	31
$\mathcal{W}'$	A sampled set of at least 33 BBs from $\mathcal{W}$	31
$\mathcal{N}'$	Sample size of a population	31
$z_{1-\frac{\alpha}{2}}$	$(1-\frac{\alpha}{2})$-quantile of a unit normal variate	31
$\mathcal{N}$	The size of the entire population	31
$\mathbb{P}'$	A sampling of $\mathbb{P}$	32
$\overline{\mu}_{\tilde{b}}$	Sample mean of $\tilde{b}$ after being overlaid onto a sampling of $\mathbb{P}$	32
$\overline{\mu}_{\mathcal{W}'}$	Sample mean of $\mathcal{W}'$ after each is overlaid onto a sampling of $\mathbb{P}$	32

Symbol		Page
$\overline{\sigma}_{\tilde{b}}$	Sample standard deviations for $\tilde{b}$	32
$\overline{\sigma}_{\mathcal{W}'}$	Sample standard deviations for $\mathcal{W}'$	32
$t_{[\frac{1-\alpha}{2};\nu]}$	$(1-\frac{1}{\alpha})$-quantile of a t-variate with ν degrees of freedom	33
ν	t-variate degrees of freedom	33
c_{t1}	Lower confidence interval value result from the student-t test	33
c_{t2}	Upper confidence interval value result from the student-t test	33
$\mathcal{BB}'$	The *sampled* set of order-**o** BBs	34
$\mu_{\tilde{b}}$	Mean of fitness values resulting from overlaying $\tilde{b}$ onto $\mathbb{P}$	35
$\mu_{\mathcal{W}'}$	Mean of fitness values resulting from overlaying each BB in $\mathcal{W}'$ onto $\mathbb{P}$	35
ζ	The set of objectives available in an MOP	40
$\vec{\eta}^{\,(\tilde{b})}$	Centroid for $\tilde{b}$	41
$\vec{\eta}^{\,(\mathcal{W})}$	Centroid for $\mathcal{W}$	41
RF	indicates Ranked Fitness	50
$\mathcal{O}$	Set of BBs having a variety (can be the same) sizes	52
$\mathcal{O}^r$	A set of BBs of various sizes where $\mathcal{J}(\mathcal{O}^{\nabla}) \triangleq P_{true}$	52
$\mathcal{O}^*$	A minimal set of maximal ordered BBs that satisfies the $\mathcal{O}^r$ criteria	52
Ω_t	Sum of expected unique BBs found up to and including generation t	54
ω_t	Number of expected unique BBs found in generation t	54
t	Indicates the generation number of an EA or MOEA	54
$Gain_{a(t)}$	Available BB Gain	54
q	Indicates the population size of randomly generated BBs.	55
$E(q)$	Formal equation for expected number of unique BBs found in q BBs	55
$\hat{r}$	Number of regular CTs used in the MOMGA-IIa	61

Symbol		Page
$\hat{i}$	Number of inverse CTs used in MOMGA-IIa	61
$\hat{o}$	Number of orthogonal CTs used in MOMGA-IIa	61
$\mathbb{C}$	Set of complete solutions	66
$\mathcal{G}$	Total number of generations to complete one experiment	74
ℓ'	The string length hold within the computer (can be longer than number of genes)	77
$\hat{c}(\alpha)$	Given an alpha, the acceptable amount of error	77
β^2	The signal-to-noise ratio estimate	77
p_f	Probability of filtering	80
p_m	Probability of mutation	80
$\hat{d}$	Number of bits to randomly delete from population members	80
$t\hat{v}s$	Number of target vector slots available for assignment	90
$c\hat{t}s$	Total number of CTs used by MOMGA-IIa	93
$t\vec{v}_i$	Denotes target vector i within $t\hat{v}s$ target vectors	99
$f_\tau(\vec{x}_{max})$	Indicates $\vec{x}_{max}$ is the best w.r.t. objective τ	102
$f_\tau(\vec{x}_{min})$	Indicates $\vec{x}_{min}$ is the worst w.r.t. objective τ	102
$\triangle$	Void-factor where no incoming point for target vector slot assignment can lie after being normalized to the normal 0-1 objective space	102
$\angle_{min}$	Half the minimum angular distance between any two target vectors partitioning the objective space	106
$\vec{w}$	An incoming point to CTMS to check for target vector slot assignment	106
$a\vec{t}_i$	A currently assigned point to the i^{th} target vector slot	106
$\prec_t$	Indicates that a left side vector is target vector dominated over a right side vector (*i.e.*, $\vec{A} \prec_t \vec{B}$ or $\vec{B}$ target vector dominates $\vec{A}$)	107

Symbol		Page
$area_i$	Area between the origin and vector vec_i	135
$\bar{\mathbf{d}}$	Mean of all $\mathbf{d_i}$	136
$\mathbf{d_i}$	Distance of neighboring vectors in PF_{known}	136
vol_i	Volume of the calculated between the origin and vector vec_i	138
u	Utility function	140
$\tilde{\lambda}$	User specified number of scalarizing vectors	140
$\mathtt{R}$	Reference set containing PF_{true}	140
$\tilde{R}_h$	h^{th} rank of a sample within a set, $\mathcal{N}$, of metric results	143
H	Kruskal-Wallis H test	143
χ^2	Indicates the Chi squared distribution	151
$\notin$	Indicates that the groups tested under the Kruskal Wallis test are not within the same population	151
F	Indicates the F distribution used with Kruskal Wallis test	151
$\in$	Indicates that the groups tested under the Kruskal Wallis test are within the same population	151
n_a	Multiplier for calculating population size for fmGA	176
E_i	Indicates an energy value for i where i is the name of the energy	197
K_b	Force constant determining the strength of an atom's bond	197
b	Atom bond length	197
b_0	Ideal atom bond length	197
K_θ	Force constant of measured angle θ	197
θ_0	Ideal angle θ for the force constant	197
D	Effective dielectric function	198
r	Distance between two atoms	198
q_i	Charge value for atom i	198
W	Total number of weights	204
N_{min}	Number of data records	204

Symbol		Page
$SLICE$	Number of bits representing one variable	225
P_b	Probability of bit error	236
$\frac{Eb}{No}$	Signal to Noise ratio	236
Eb	Energy per bit	237
No	Spectral Noise Density	237
$\hat{d}$	The number of dimensions	239
s	Indicates signal	240
$\mathcal{H}$	Hamming code distance matrix	241
$\mathbf{D}_{i,j}$	Symbol distance matrix (Euclidean distance from symbol i to symbol j	243
K	Constant used in an m-ary symbol set design problem	244
$\mathcal{S}$	contains the symbols in the entire constellation	245
$\mathcal{G}$	Contains the values for $\mathcal{S}$	245
$\mathcal{P}_{err}^{(v)}(i,j,h)$	Probability of error when symbol v is transmitted and symbol h is reconstructed at grid position (i,j)	246
σ	Standard Deviation	248
$\hat{R}_i$	Amplitude of a symbol (also called the radius of the symbol position)	248
$\mathcal{E}_s$	Energy of an entire digital symbol constellation	248
$\mathcal{Q}(x)$	Half the complement to the error function	248
L	Represents the number of amplitude levels in a single dimension	258
P_{matrix}	A transitional probability matrix used within Markov Chain Optimization	326
$\breve{E}_j$	Indicates state j within a Markov Chain P_{matrix}	326
$W(x,x')$	Computes the transition probability of state $x \rightarrow x'$	328
$\breve{\mu}$	Indicates the total number of parents in an ES	329
$\breve{\rho}$	Marks the number of parents that are to be recombined in an ES	329

Symbol		Page
$\check{\lambda}$	Stands for the number of offspring	329
$\check{\gamma}$	The isolation number within an ES	329
$\vec{d_i}$	Represents one population member/vector within an ES's $\mathbb{P}'$	330
$\mathcal{EVOP}(\mathcal{S})$	Perform a set of evolutionary operators on the population members in $\mathcal{S}$	333
$\mathcal{S}$	An arbitrary selection operator	333
$\mathcal{V}$	An arbitrary selection operator	333
ℓ'	Total number of genes	337
$\hat{m}_g$	Inverse of the probability for selecting a gene combination	337
$\mathcal{N}_g$	fmGA suggested subpopulation size	337
β	Difference between sampled BB means	339
p_s	Probability of selection	341
I_{best}	This is the set of —I— best population w.r.t.a SO function	341
$p_i^{(t)}$	Number of individuals dominating individual i at generation t	348
$\mathbb{E}'$	External subpopulation set $\lvert\mathbb{E}'\rvert < \lvert\mathbb{P}'\rvert$	354
$\mathcal{T}(\mathbb{P}'$	Indicates binary tournament selection operated on $\mathbb{P}'$	354
$\mathcal{MP}$	Indicates a multi-set union of populations	354
$\mathbb{A}$	Dynamically growing archive of solutions	355
χ	Used as a niche	358
$\Psi_{\mathcal{N}}$	Group of sub niches	358
$N_{\mathcal{E}_{max}}$	Capacity of elite pool for the IMOEA	368
$\mathcal{E}$	Set of elite for IMOEA	370
$\mathcal{E}'$	Set of temporarily elite for IMOEA	370
φ	Predetermined or learned threshold value	381
$\bar{\mathbb{P}}$	Indicates that each set of decision variables in the population is lined up into one long vector (assuming binary decision variables)	381

AFIT/DS/ENG/05-03

Abstract

This dissertation presents principles, techniques, and performance of evolutionary computation optimization methods. Concentration is on concepts, design formulation, and prescription for multiobjective problem solving and explicit building block (BB) multiobjective evolutionary algorithms (MOEAs). Current state-of-the-art explicit BB MOEAs are addressed in the innovative design, execution, and testing of a new multiobjective explicit BB MOEA. Evolutionary computation concepts examined are algorithm convergence, population diversity and sizing, genotype and phenotype partitioning, archiving, BB concepts, parallel evolutionary algorithm (EA) models, robustness, visualization of evolutionary process, and performance in terms of effectiveness and efficiency. The main result of this research is the development of a more robust algorithm where MOEA concepts are implicitly employed. Testing shows that the new MOEA can be more effective and efficient than previous state-of-the-art explicit BB MOEAs for selected test suite multiobjective problems (MOPs) and Untied States Air Force applications. Additional contributions include the extension of explicit BB definitions to clarify the meanings for good single and multiobjective BBs. A new visualization technique is developed for viewing genotype, phenotype, and the evolutionary process in finding Pareto front vectors while tracking the size of the BBs. The visualization technique is the result of a BB (solution) tracing mechanism integrated in the new MOEA to enable a researcher to determine the required BB sizes and assign an approximation *epistasis* level for solving a particular problem. The culmination of this research is explicit BB state-of-the-art MOEA technology based on the MOEA design, BB classifier type assessment, solution evolution visualization, and insight into MOEA test metric validation and usage as applied to the following: test suite, deception, bioinformatics, unmanned vehicle flight pattern, and digital symbol set design MOPs.

EXPLICIT BUILDING BLOCK MULTIOBJECTIVE EVOLUTIONARY COMPUTATION: METHODS AND APPLICATIONS

I. Introduction

Finding optimal solutions to multicriteria problems requires a non-ambiguous procedure (algorithm) to effectively implicitly or explicitly search all possibilities. The choice of such an algorithm can depend upon a determination of the problem domain characteristics as well as a study of all the various search techniques. For low dimensional[1] multicriteria problems, deterministic search techniques are able to find optimal solutions efficiently (if they exist). For higher dimensional problems, stochastic techniques are generally employed in order to find acceptable solutions.

Humans naturally optimize low dimensional tasks without specifically defining the entire system, mathematically or symbolically. In fact, much is written about optimizing human tasks in scientific and non-scientific literature. In the novel *Cheaper by the Dozen*, daily tasks for a household of 14 are attempted to be optimized by parents Frank and Lillian Gilbreth [85]. It is the parents' desire to optimize every task in their lives in order to make their existences more harmonious and productive. One of the most common tasks optimized in daily life is the route driven to and from work. This optimization problem is easily accomplished given that it has a small set of possible solutions that can be enumerated. However, most real-world optimization problems are more complex as they have multiple objectives and a non-determinable metric or measure of solution quality.

[1] A low dimensional problem is define as having a number of steps to solve the problem bounded by a polynomial (see Definition 30 in Appendix M on page 419).

This research is focused on solving or finding good solutions for higher dimensional multicriteria or multiobjective optimization problems (MOPs) such as those found in relevant United States Air Force (USAF) real-world applications and benchmark test suite MOPs using a stochastic multiobjective evolutionary search algorithm.

1.1 Overview

Evolutionary Algorithms (EAs) are a part of a more general field of algorithmic study called Evolutionary Computation (EC). EC research consists of computational techniques that are based to some degree on the evolution of biological life in the natural world. Today's researchers use EAs to search single and multiobjective landscapes to find near optimal solutions for problems having many different characteristics (continuous, discrete, multimodal, ...)

In fact, EA design is employed in many fields of application study. Examples include the use of EAs as a problem solver in Computer Science, Computer Engineering, and Operations Research (OR). Each field classifies EAs in their own terms. In the OR field, optimum seeking methods are classified as *mathematical programming* techniques; thus, EAs as an Evolutionary Computation (EC) method are also regarded as a subset of algorithms or heuristics [186] (see Table 82 in Appendix M on page 420). Within the Computer Science and Computer Engineering fields, EAs fall into the advanced algorithm techniques for complex problem solving. EAs have a run time that is polynomial[2]; yet, the problems that EAs generally solve are NP-Complete (NPC) problems (see Definitions 30, 31, 32, 33, and 34 in Appendix M on page 420). Furthermore, EA designers accept that a suboptimal solution found in polynomial time is an acceptable compromise to finding the optimal solution by deterministically searching the entire solution space. The reason for the compromise is because the search space is normally too large to exhaustively search due to the exponential time characteristics of such problems. Examples of such problems

[2] Polynomial run times occur if and only if the EA has a stopping criteria.

include the Travelling Salesman problem and the Graph Coloring problem; more examples are presented in Table 83 in Appendix M on page 421. These problems have many different objective and decision variable characteristics ranging from discrete quantities having a range of $\{0, 1\}$ to continuous real valued quantities having ranges limited only by the word size of the computer solving the problem.

1.2 Historical Overview of Multicriteria Optimization

Evidence of multiobjective optimization can be traced back to 1895 when Georg Ferdinand Ludwig Philipp Cantor (1895-1906)[3] and Felix Hausdorff (1906-1919) forged the mathematical foundations for infinite dimensional ordered spaces. Concurrently, in 1896, Vilfredo Pareto defined Pareto optimal and labelled solutions evaluating to vectors found on the Pareto optimal front as The Edgeworth-Pareto optimum. William Karush, in 1939, completed his thesis work on multiple variable function optimization with side constraints which led to Harold W. Kuhn and Albert W. Tucker's introduction of the vector maximum problem (VMP). Equation 1 describes Kuhn and Tucker's VMP where k denotes the number of objective functions to be maximized. In some cases, objective functions for the VMP are required to be constrained prior to being optimized. These other objectives, for convenience, represent some other technical constraints and are represented by $g_j(\mathbf{X}) \leq 0$ in Equation [193]. It should be noted that this same problem can be called the vector *minimization* problem when minimizing for $\{f_1(\mathbf{X}),\ f_2(\mathbf{X}),\ \ldots,\ f_k(\mathbf{X})\}$[4] [186]. Kuhn and Tucker's VMP made multiobjective optimization a mathematical discipline.

[3] These years indicate research years within the field.

[4] A value from $f_j(\mathbf{X})$ is called within the objective space or phenotype[5]. domain whereas the $\mathbf{X}$ used in $f_j(\mathbf{X})$ is called within the decision variable space or genotype domain.

$$\begin{aligned} &\textit{Find } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \\ &\textit{which optimizes} \quad f_1(\mathbf{X}),\ f_2(\mathbf{X}),\ \ldots,\ f_k(\mathbf{X}) \\ &\textit{subject to} \quad g_j(\mathbf{X}) \leq 0, j = 1, 2, \ldots, m \end{aligned} \tag{1}$$

Kuhn and Tucker's seminal paper describing the concept of *proper efficiency*[6] proved that the multipliers of all components of the objective function in the necessary optimality conditions are strictly positive [139]. With this research, Kuhn and Tucker then came to be known as having the first attempt to derive a multiobjective optimization theory. Their work derived necessary and sufficient conditions for the optimal solution of programming problems and laid the foundations for many nonlinear programming and multicriteria studies such as Kenneth Arrow et al. who, in 1953 used the term *admissible* instead of *efficient* points. In 1951, following Kuhn and Tucker, Tjalling C. Koopmans published a paper concerning an activity analysis of production and allocation theory. Then, in 1960, Leonid Hurwicz generalized Kuhn and Tucker's results for vector spaces. Multicriteria optimization research continued and began to shift to bigger applications. In 1951, Koopmans applied multiobjective optimization to production theory, Zadeh claimed the first engineering research in the early 1960s, and Marglin optimized water resource plans in 1967. At the same time that Zadeh is claimed the first multicriteria engineering research, Rosenberg had

[6] w.r.t. the VMP a point, x^0, is said to be efficient if $x^0 \in X$ and there exists no other feasible point x such that $f(x) \geq f(x^0)$ and $f(x) \neq (x^0)$. Furthermore, x^0 is *properly efficient* if it meets the efficient condition (above) and if there exists a scalar $J > 0$ such that, for each i, the result is

$$\frac{f_i(x) - f_i(x^0)}{f_j(x^0) - f_j(x)} \leqslant J$$

for some j such that $f_j(x) < f_j(x^0)$ whenever $x \in X$ and $f_i(x) \geq f_i(x^0)$. [84]

thoughts on EA multiobjective applications. His foresight is brought to a reality by Schaffer in the 1980s (the actual multiobjective EA implementation was the Vector Evaluated Genetic Algorithm (VEGA)). The VEGA research launched a quest for good evolutionary computation methods to solve extremely difficult multicriteria problems. The following sections outline the goal and supporting objectives of this dissertation research that extends state-of-the-art Multiobjective Evolutionary Algorithms (MOEAs) research. [31, 34, 144]

1.3 Research Goal and Objectives

This research builds on previous building block (BB)[7] MOEA research [222,244] and embraces the conjecture that every optimization problem (both single and multiobjective) has an inherent set of BBs that can be combined together to create all optimal solutions for a particular problem. In addition, explicit BB based MOEAs have shown to be a good choice for solving many different multicriteria applications [41, 125, 244]. It is for these reasons that the goal of this research is to *extend* BB multicriteria evolutionary algorithm understanding, develop a more robust algorithm capable of solving a larger range of NPC applications, and apply the newly designed algorithm to several USAF applications. Table 1 lists a shortened version of this main goal and supporting objectives.

Goal Development of a more robust multiobjective evolutionary algorithm capable of solving larger range of NPC applications, which are relevant to United States Air Force's real-world applications. In addition, to advance the understanding design and application of BB based MOEAs. This goal is met using the following objectives.

Objective 1: Using accepted MOEA techniques and an innovative design, a new more robust and scalable algorithm is developed. The new algorithm is tested

[7] A building block is composed of pieces of an entire solution.

using test suite problems of a variety of characteristics to show it is scalable over previous designs. In addition, state-of-the-art MOEA metrics are evaluated.

Objective 2: There are many Air Force applications requiring multiobjective problem solving. Sensor management and vehicle routing are just two examples of military applications that have been positively affected by MOEA research [50,51,53]. The explicit BB MOEA is applied to some of these problems to cover the range of different fitness landscapes identified in Table 84 in Chapter IV.

Table 1: Goal continued and supporting Objectives

Goal	**Objectives**
Development of a more robust multiobjective evolutionary algorithm capable of solving larger range of NPC applications, which are relevant to United States Air Force's real world applications.	**1:** Develop a new algorithm that is more robust (effective) in solving a larger range of MOPs. Test and valid metrics and statistical methods are selected for comparing MOEAs.
	2: Indicate how MOEA research advantageously applies to Air Force applications
To advance the understanding, design and application of BB based MOEAs.	**3:** Extend MOEA understanding by extending the meaning of BBs in an explicit search algorithm
	4: Indicate that explicit BB MOEA approaches are statistically similar in solving deception problems; however, different in search methods.
	5: Use MOEA generic objectives as preserve, progress, and diversity for extending an explicit BB MOEA.
	6: Develop possible MOEA techniques that can be employed in an efficient parallel design and implementation.

Objective 3: Previous BB definitions are evaluated and new, more precise, definitions are derived. New BB definitions are tied to defining what a good BB definition is for different multiobjective explicit BB algorithms. Following these, a new definition for an Optimal BB set is suggested.

Objective 4: Current innovative MOEA techniques include a Bayesian network integration to probabilistically distinguish good BBs associated with a prescribed search space for specified objective functions. The multiobjective messy genetic algorithm (MOMGA-II) has a similar technique for identifying good BBs and has at least the same or better effectiveness when solving deception problems than the Bayesian Optimization Algorithm. Results show a limitation of the MOMGA-II when solving larger scaled deception problems. However, with the addition of an innovative competitive template management system into the MOMGA-II, the new MOEA (MOMGA-IIa) exceeds results found by the multiobjective Bayesian approach algorithm.

Objective 5: MOEA fundamentals are the following: an MOEA design must preserve *good* solutions, progress toward the optimal solutions, maintain diversity, and provide the best, but limited, set of solutions to the decision maker (DM). The new MOEA design follows these fundamentals allowing for the new explicit BB MOEA to become more robust. New competitive template modifications are applied. The multiple competitive template generation techniques have been studied; however, often times generation of many competitive templates in the wrong area within the objective space cements the search into a specific area because the generated competitive templates are too close in the phenotype domain. It is thought that competitive template generation should encompass a check for phenotype distribution to be sure that multiple competitive templates are not locking the search into one area. Objective space value normalization and an even partitioning of the known space is used to accomplish phenotype and genotype domain diversity. Competitive template spacing is then tied to the reduction of the actual size of BBs required for a good search. In addition, an archive is added to the MOEA to preserve and allow for an acquired diversity. Finally, the MOEA is allowed to progress toward the optimal solution or solutions by preserving solutions that evaluate to non-dominated vectors - especially those solutions evaluating as the best found for each objective.

Objective 6: Many design issues exist when parallelizing an EA [24]. The following are three general parallel models that are common in the MOEA field: island, diffusion, and master-slave. Deciding on which model to implement within an EA is dependent upon the hardware architecture employed. Many different hardware attributes are important; including processing and memory capabilities. Some possible hardware configurations include multi-computer, multi-processor, homogeneous, and heterogeneous systems. The memory may be shared or distributed and the network topology may be a mesh, hypercube, or ring structure. The parallelization may be of the GA itself, operators, fitness evaluation, or the population pool, with synchronous or asynchronous communication calls. Finally, choosing the parallel library to use is also of importance, with the Message Passing Interface (MPI) [174] and Parallel Virtual Machine (PVM) being just two possibilities. Previous research into high performance computing [157,164] led to the decision to use MPI with an island model implementation on many different applications. This was decided due to the variety of heterogeneous and homogeneous systems that are utilized in conjunction with this research. Most of our parallel fast messy Genetic Algorithm (pfmGA) research has concentrated on the use of distributed and heterogeneous systems where MPI allows for somewhat *easy* portability between systems. The genetic operators were not parallelized due to the minor improvements in performance [80]. The parallelization instead centers on the fitness evaluations and an island model implementation that yields *good* speedups in conjunction with use of the pfmGA.

This research builds on the previous implementation of a farming and island model by adding a preprocessing technique that reduces the run time associated with calculating a computational intensive fitness function; namely the Protein Structure Prediction force field approximate model based on the Chemistry at HARvard Molecular mechanics (CHARMm) model ver 22 and the Monte Carlo Simulation model for digital symbol set design testing. These preprocessing techniques approximate the fitness functions using a neural network and other mathematical models. Results show that these are efficient time savers; however, the modeling loss due to neural network fitness function replacement is not an acceptable and the mathematical models do not accurately represent the Monte Carlo simulation model. Mach-up timing reveals that these methods, once implemented, in some cases yield more than 16 times speedup over the normal fitness function. However, due to the lack of periodicity within the Protein Structure Prediction problem's fitness function and limitations of the mathematical models the approximation techniques tested are observed to mislead the new MOEA to incorrect areas in the search space resulting is less effectiveness and suboptimal solutions.

Also, a new parallel model is suggested for implementation using the new MOEA designed. The new model is based on a recently tested Hierarchical Fair Competition (HFC) framework for parallelizing evolutionary algorithms in [108]. The new model suggests more divisions of chromosomes into factions determined by a normalized objective space, as well as a hierarchical fitness level subpopulation for bettering an EA's future civilizations.

This section summarized the goal and supporting objectives of this research. The following section provides the research approach and scope.

1.4 Approach and Scope

The approach taken for this research includes the analysis of previous developed MOEAs in comparison with a newly developed MOEA. Test suite MOPs and selected Air Force applications are used to test both the previous state-of-the-art MOEA and the new innovative MOEA described in this research in support of the objectives. MOEA metrics and statistics are used to support or compare to initial test hypotheses and final results. Tentative explanations are given where possible, and insight gained through this research is then employed in future algorithm design. Experiments are designed and executed using state-of-the-art metrics. Results are analyzed statistically.

The research goal and supporting objectives are defined in Section 1.3. The following chapters support these objectives and ultimately meet the goal of this research. A few assumptions are made about the background of the readers of this document. Readers should have a undergraduate level understanding of evolutionary algorithms, supporting operators, high performance computer parallel concepts, chemistry, biology, probability theory, digital communications, and mathematics. Many innovative ideas new to the MOEA research field are addressed in this study – each in support of meeting the supporting objectives. Also addressed are several real-world and pedagogical MOPs. Next, background definitions are presented. It is suggested that the reader reference these definitions as needed while reading this document.

1.5 Basic Definitions

The reader may resort to these definitions when needed to clarify discussions. The basic definitions required for an understandable reading are the general MOP, Pareto dominance, Pareto optimality, Pareto optimal set, Pareto front, and Pareto epsilon (ϵ) dominance. Each are defined as the following:

Definition 1 (General MOP [31, 34, 216]): *A general MOP minimizes (or maximizes)* $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$ *subject to* $g_i(\vec{x}) \leq 0,\ i = \{1, \ldots, m\},\ \vec{x} \in \Omega$. *An MOP solution minimizes (or maximizes) the components of a vector* $F(\vec{x})$ *where* $\vec{x}$ *is a n-dimensional decision variable vector* $\vec{x} = (x_1, \ldots, x_n)$ *from some universe* Ω. *It is noted that* $g_i(\vec{x}) \leq 0$ *represents some other technical constraints that also must be met while minimizing (or maximizing)* $F(\vec{x})$ *and* Ω *contains all possible* $\vec{x}$ *that can be used to satisfy an evaluation of* $F(\vec{x})$. □

Definition 2 (Pareto Dominance [31, 34, 216]): *A vector* $\vec{u} = (u_1, \ldots, u_k)$ *is said to dominate another vector* $\vec{v} = (v_1, \ldots, v_k)$ *(denoted by* $\vec{u} \preceq \vec{v}$*) if and only if* $\vec{u}$ *is partially less than* $\vec{v}$, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. □

Definition 3 (Pareto Optimality [31, 34, 216]): *A solution* $\vec{x} \in \Omega$ *is said to be Pareto Optimal with respect to (w.r.t.)* Ω *if and only if (iff) there is no* $\vec{x'} \in \Omega$ *for which* $\vec{v} = F(\vec{x'}) = (f_1(\vec{x'}), \ldots, f_k(\vec{x'}))$ *dominates* $\vec{u} = F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$. *The phrase* ***Pareto Optimal*** *is taken to mean with respect to the entire decision variable space unless otherwise specified.* □

Definition 4 (Pareto Optimal Set [31, 34, 216]): *For a given MOP,* $F(\vec{x})$*, the Pareto Optimal Set,* $\mathcal{P}^*$*, is defined as:*

$$\mathcal{P}^* := \{\vec{x} \in \Omega \mid \neg\exists\ \vec{x'} \in \Omega\ \ F(\vec{x'}) \preceq F(\vec{x})\}. \tag{2}$$

□

Definition 5 (Pareto Front [31, 34, 216]): *For a given MOP,* $F(\vec{x})$*, and Pareto Optimal Set,* $\mathcal{P}^*$*, the Pareto Front (*$\mathcal{PF}^*$*) is defined as:*

$$\mathcal{PF}^* := \{\vec{u} = F(\vec{x}) \mid \vec{x} \in \mathcal{P}^*\}. \tag{3}$$

□

Definition 6 (Pareto epsilon (ϵ) Dominance): *A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to epsilon-dominate another vector $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \underset{\epsilon}{\preceq} \vec{v}$) if for some $\epsilon > 0$ u_i is partially less than $v_i + \epsilon$,* i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq (v_i + \epsilon) \wedge \exists i \in \{1, \ldots, k\} : u_i < (v_i + \epsilon)$ *where $\epsilon > 0$.* □

Definition 7 (Pareto epsilon (ϵ) Optimality): *A solution $\vec{x} \in \Omega$ is said to be Pareto epsilon Optimal with respect to Ω if and only if there is no $\vec{x'} \in \Omega$ for which $\vec{v} = F(\vec{x'}) = (f_1(\vec{x'}), \ldots, f_k(\vec{x'}))$ epsilon dominates $\vec{u} = F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$. The phrase **Pareto epsilon Optimal** is taken to mean with respect to the entire decision variable space unless otherwise specified.* □

Definition 8 (Pareto epsilon (ϵ) Optimal Set): *For a given MOP, $F(\vec{x})$, the Pareto epsilon Optimal Set, $\mathcal{P}^*_\epsilon$, is defined as:*

$$\mathcal{P}^*_\epsilon := \{\vec{x} \in \Omega \mid \neg\exists\ \vec{x'} \in \Omega\ \ F(\vec{x'}) \underset{\epsilon}{\preceq} F(\vec{x})\}. \tag{4}$$

□

Definition 9 (Pareto epsilon (ϵ) Front): *For a given MOP, $F(\vec{x})$, and Pareto epsilon Optimal Set, $\mathcal{P}^*_\epsilon$, the Pareto epsilon Front ($\mathcal{PF}^*_\epsilon$) is defined as:*

$$\mathcal{PF}^*_\epsilon := \{\vec{u} = F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x})) \mid \vec{x} \in \mathcal{P}^*_\epsilon\}. \tag{5}$$

□

The assumption that $\mathcal{P}^*$ and $\mathcal{PF}^*$ represent sets of values having infinite word length makes *strictly* using these sets as the *goal* set of Pareto optimal solutions and associated optimal PF vectors difficult for all MOPs mainly due to the computational limitation gap between using an uncountable infinite set (theoretical values) and countable/finite set (computational values) to represent decision variables and associated objective vector values.

Table 2 lists the three types of theoretical relationships between the $\mathcal{P}^*$ and $\mathcal{PF}^*$ set sizes that must be addressed depending upon MOP characteristics.

Table 2: Relationships between the $\mathcal{P}^*$ and $\mathcal{PF}^*$ set size.

	$\lvert\mathcal{P}^*\rvert$		$\lvert\mathcal{PF}^*\rvert$	Mappings of sets having size 1, $\ddot{n}^a$, and $\ddot{u}^b$
1.	Countable	→	Countable	$\{(1 \rightarrow 1), (\ddot{n} \rightarrow 1), (\ddot{n} \rightarrow \ddot{n})\}$
2.	Uncountable	→	Countable	$\{(\ddot{u} \rightarrow 1), (\ddot{u} \rightarrow \ddot{n}), (\ddot{u} \rightarrow \ddot{n})\}$
3.	Uncountable	→	Uncountable	$\{(\ddot{u} \rightarrow \ddot{u})\}$

[a] $\ddot{n}$ represents a countable/infinite or finite set.
[b] $\ddot{u}$ represents an uncountable set.

$\mathcal{P}^*$ and $\mathcal{PF}^*$ can represent the goal sets for a MOEA search algorithm in certain circumstances. Any goal set having an uncountable $|\mathcal{PF}^*|$ cannot be solved by a deterministic Turing Machine; thus, #3 above cannot be solved by digital computers because of word length restriction. Furthermore, only under certain circumstances can a deterministic Turing Machine find the right solutions that evaluate to a countable $|\mathcal{PF}^*|$ set when the $|\mathcal{P}^*|$ is uncountable infinite; thus, #2 can be solved only under certain circumstances by digital computers. Finally, if the set of values contained in $|\mathcal{P}^*|$ and $|\mathcal{PF}^*|$ are subsets of values[8] that can be represented by a deterministic machine used to solve an MOP then relationship #1 can be solved by digital computers. Therefore, when defining the goal set it is important to have a set that can be found by the MOEA. The goal set is referred to as P_{true}.

The following terminology is used to distinguish between the real-world's and mathematical world's representation of solutions and associated PF vectors when solving MOPs. In the cases where $\mathrm{P}_{true} \nsubseteq \mathcal{P}^*$, P_{true} and PF_{true} are in the proximity[9] of $\mathcal{P}^*$ and $\mathcal{PF}^*$ respectfully. The three cases listed under each term are related to the relationships found in Table 2.

[8] These values must be a *computational number*; otherwise, a digital computer could not represent the goal sets. A computational number is a number that can be represented within a digital computer.

[9] Distance of optimal solutions and associated PF vectors to the theoretical true depends upon word length restriction and characteristics of the problem domain.

P_{true} This term is given as the MOP's computational true Pareto Optimal Set (decision variables). Under the following conditions P_{true} is a subset of $\mathcal{P}^*$ because the decision variables are to be discrete.

1. $\begin{cases} \texttt{All computational numbers in } \mathcal{P}^*, & \mathrm{P}_{true} \subseteq \mathcal{P}^* \\ \texttt{At least one non-computational number in } \mathcal{P}^*, & \mathrm{P}_{true} \nsubseteq \mathcal{P}^* \end{cases}$
2. $\mathrm{P}_{true} \nsubseteq \mathcal{P}^*$
3. $\mathrm{P}_{true} \nsubseteq \mathcal{P}^*$

PF_{true} This term is given as the MOP's computationally true Pareto Front set (objective values). Under the following conditions PF_{true} is a subset of $\mathcal{PF}^*$ due to the discrete decision variables and associated PF vectors within the computer.

1. $\begin{cases} \texttt{All computational numbers in } \mathcal{PF}^*, & \mathrm{PF}_{true} \subseteq \mathcal{PF}^* \\ \texttt{At least one non-computational number in } \mathcal{PF}^*, & \mathrm{PF}_{true} \nsubseteq \mathcal{PF}^* \end{cases}$
2. $\mathrm{PF}_{true} \nsubseteq \mathcal{PF}^*$
3. $\mathrm{PF}_{true} \nsubseteq \mathcal{PF}^*$

P_{known} This term defines the Pareto Optimal set best found by the MOEA (decision variables). P_{known} often times does not represent true Pareto Optimal Set; however, it represents the best set found by an MOEA for a particular MOP.

PF_{known} This term, PF_{known}, defines a Pareto Front set found by the MOEA that may be an intermediate Pareto Front optimal set for the MOEA (*i.e.* objective values are not as good as objective values in the final Pareto Front set found by the MOEA).

The following terminology is used to distinguish between the real-world's and mathematical-world's representation of solutions and associated Pareto epsilon front vectors when solving MOPs. In the cases where $\mathrm{P}^{\epsilon}_{true} \nsubseteq \mathcal{P}^*_{\epsilon}$, $\mathrm{P}^{\epsilon}_{true}$ and $\mathrm{PF}^{\epsilon}_{true}$ are in the proximity of $\mathcal{P}^*_{\epsilon}$ and $\mathcal{PF}^*_{\epsilon}$ respectfully. The three cases listed under each term are related to the relationships found in Table 2.

$\mathrm{P}^{\epsilon}_{true}$ This term is given as the MOP's computational true Optimal epsilon Set (decision variables). Under the following conditions $\mathrm{P}^{\epsilon}_{true}$ is a subset of $\mathcal{P}^*_{\epsilon}$ because the decision variables for the MOP must be discrete.

1. $\begin{cases} \texttt{All computational numbers in } {\mathcal{P}_{\epsilon}}^*, & \mathrm{P}^{\epsilon}_{true} \subseteq {\mathcal{P}_{\epsilon}}^* \\ \texttt{At least one non-computational number in } {\mathcal{P}_{\epsilon}}^*, & \mathrm{P}^{\epsilon}_{true} \nsubseteq {\mathcal{P}_{\epsilon}}^* \end{cases}$
2. $\mathrm{P}^{\epsilon}_{true} \nsubseteq {\mathcal{P}_{\epsilon}}^*$
3. $\mathrm{P}^{\epsilon}_{true} \nsubseteq {\mathcal{P}_{\epsilon}}^*$

$\mathrm{PF}^{\epsilon}_{true}$ This term is given as the MOP's computationally true Pareto epsilon Front set (objective values). Under the following conditions $\mathrm{PF}^{\epsilon}_{true}$ is a subset of $\mathcal{PF}^*_{\epsilon}$ due to making the decision variables and objective vectors of the MOP discrete within the computer.

1. $\begin{cases} \texttt{All computational numbers in } {\mathcal{PF}_{\epsilon}}^*, & \mathrm{PF}^{\epsilon}_{true} \subseteq {\mathcal{PF}_{\epsilon}}^* \\ \texttt{At least one non-computational number in } {\mathcal{PF}_{\epsilon}}^*, & \mathrm{PF}^{\epsilon}_{true} \nsubseteq {\mathcal{PF}_{\epsilon}}^* \end{cases}$
2. $\mathrm{PF}^{\epsilon}_{true} \nsubseteq {\mathcal{PF}_{\epsilon}}^*$
3. $\mathrm{PF}^{\epsilon}_{true} \nsubseteq {\mathcal{PF}_{\epsilon}}^*$

$\mathrm{P}^{\epsilon}_{known}$ This term defines Pareto epsilon Optimal Set found by the MOEA (decision variables). $\mathrm{P}^{\epsilon}_{known}$ often times does not represent the true Pareto epsilon Optimal Set; however, it should represent the best Pareto epsilon Optimal Set found by an MOEA for a particular MOP.

$\mathrm{PF}^{\epsilon}_{known}$ This term defines Pareto epsilon Front set found by the MOEA. $\mathrm{PF}^{\epsilon}_{known}$s may be an intermediate Pareto epsilon Front for the MOEA (*i.e.* a set that is not as good as the final Pareto epsilon Front set found by the MOEA).

A good discussion of these definitions and terminology can be found in Appendix H on page 391 [34]. The last section in this chapter summarizes what was covered and outlines by chapter upcoming discussions .

1.6 Reader Qualifications

It is assumed that the reader has a fair background in optimization techniques and in particular evolutionary computation including genetic algorithms, evolutionary strategies, evolutionary programming, and genetic programming. A summary of these optimization techniques is provided in Appendix E on page 324; however, this appendix is only an introduction to each method and thus other references are recommended [5, 89, 227]. Exposure to multicriteria optimization is also advantageous for understanding of concepts and problems discussed.

1.7 Summary of Contributions

Enumerated in this section are the contributions achieved by this research. Each contribution is briefly summarized, but this is by no means a replacement for the detail summary found in Chapter IX on page 261. Yet, it does provide the reader with a contributions path to facilitate in the understanding of this document.

1. Explicit BB definitions are extended in Chapter II to include clarifying the meaning of good single and multiobjective BBs.
2. A more robust algorithm is developed where MOEA concepts are implicitly designed within the new algorithm.
3. A BB (solution) tracing mechanism is integrated within the new algorithm to enable a BBB researcher to evaluate required BB sizes for solving a particular problem.

   BB size is not all that can be measured, but also the stability of variables within a problem can be measured. This includes identifying the usefulness and sensitivity within each decision variable.
4. The tracing mechanism allows for the development of a new metric serving the explicit BB MOEA community with a epistasis metric.

5. A new visualization technique is developed for the viewing of the genotype, phenotype and evolutionary process of the new algorithm finding solutions evaluating to Pareto front vectors while tracking the size of the BB required for finding each solution.

6. A new way to display metric results is identified. This method should be picked up by future MOEA researchers for quick and easy analysis between MOEA metric results.

7. Application of this MOEA on Air Force applications during this research and in future research is also a contribution.

8. software is written in a way that can be useful. New MOPs can be *Plugged* into the algorithm without integrating the new code directly into the algorithm.

   Matlab code is written to assist in the post mortem visualization of the BB trace.

This section is provided only to give a quick summarized list of each contributions provided by this research.

1.8 Summary

In this chapter, the research goal and supporting objectives are introduced. This document is organized as follows. Chapter II describes BBs and extends BB understanding by analyzing and revising the accepted single and multiobjective BB definitions. In Chapter III, a detailed description of MOEA development is given – Appendix II on page 19 directly supports this chapter by giving a brief summary of today's state-of-the-art EAs and MOEAs. MOEA fundamentals are presented and related to the development of the new MOEA. Chapter IV discusses MOEA metrics, MOP test suites, and a short design-of-experiments section. Within this chapter, the MOMGA-II and MOMGA-IIa are compared using the MOP test suites. This comparison leads to the application of the newly designed MOEA to real-world

Air Force Applications. Chapters V-VIII describes the problem domain including all the applications and pedagogical problems used in this investigation. Finally, Chapter IX concludes with a summary of the results, research contributions, future design ideas, and future applications.

II. Building Blocks and Building Block Builders

Today, optimization researchers have many different search algorithms available for solving a variety of problems including single and multiobjective. Each and every algorithm can be classified either as an implicit or explicit BB builder (BBB)[1]. Furthermore, these BBBs search for good BBs for each particular problem - but not the complete solution all at once. In addition, each problem has associated with it a set of BBs that can be juxtaposed in some way to build each and every optimal solution for that particular problem. It is the position of this research that explicit BBBs can find these good BBs better than implicit BBBs. This statement begs the question, "How can one determine if one BBB is better than another?" It can be said that a *good* BB building algorithm is one that can statistically, over time and for a number of different problems, identify more good BBs than the competitors. Researchers require BB metrics in order to measure a BB building algorithm's capabilities - making a measurement of this kind difficult. This chapter describes BBs in a novel approach to define exactly what it means to be a *good BB* with respect to a BBB. Using previous BB definitions, new more rigorous definitions are delineated to formulate a more precise set of criteria for labeling a BB as good or bad. In addition, the significance of a BB definition to BBB designers is considered. It is shown that old BB definitions do not precisely fit any of today's explicit BBB's BB decision criteria; in fact, it is conjectured that good BBs should be identified using a classifier system specific to each BBB.

This BB investigation is important to many researchers because many assume that the solving ability within an evolutionary computation method comes from its ability to improve solutions by assembling "partial solutions," the so-called building

[1] The term building block builder is describing stochastic evolutionary search algorithms. It is the position of this document that algorithms search either implicitly or explicitly for good building blocks that make up the optimal solutions (optimal solutions that evaluate to non-dominated vectors).

blocks [231]. Formally, this conjecture is known as the *Building Block Hypothesis* (BBH)[2] and it suggests that *beneficial properties* of a parent are aggregated in (relatively) small code blocks at several locations within the chromosome. The BBH does not specify how to identify BB having these *beneficial properties* providing motivation for this avenue of study.

2.1 Overview

This chapter begins by presenting the previous state-of-the-art BB definitions. These definitions are clarified with alternative, more rigorous, definitions that allow for mathematical testing of the definition. Testing of these definitions concludes that the previous definition does not accurately define good BBs with respect to the focal explicit BB search algorithm (MOMGA-IIa). A suggested correction to the definition is given to make it work better when applied to deception problems; however, even the suggested modification does not describe the MOMGA-IIa's process by which it identifies good BBs. Discussed are the concepts of optimal BB sets, BBB operating modes, MOEA convergence, and real-world BBBs (MOEAs). In relation to these real-world examples of explicit BBBs, conjectures on BB classification systems are given as a replacement to the good BB definitions previously accepted by BB researchers. The BB Hypothesis is reviewed and an advancement of a Multiobjective BB Hypothesis is then presented right before new good BB definitions are delineated. The chapter ends with the No Free Lunch Theorem emphasizing the fact that no single BBB is better than another across all MOPs.

2.2 Introduction

BBs are elements that represent only part of a whole solution. BBs are made up of sub-elements and can be any size smaller than the whole. When BBs are

[2] A detailed discussion of the Building Block Hypothesis and Multiobjective Building Block Hypothesis can be found in sections 2.11 and 2.12 on pages 63 and 65 respectfully.

put together, they can make up a complete entity. In this study, a binary solution representation is chosen; therefore, the BBs are sets of bits, and the sub-elements are the bits making up the BBs. To that end, a few assumptions are made: a builder's alphabet, $\mathcal{A}$, is the set $\{0, 1\}$, a complete solution has ℓ locus[3] positions and only one allele value assigned to each locus position (allele values are assigned from the alphabet). Complete solutions are fully specified solutions where every locus position in the bit string is assigned a value from the alphabet. It follows that a BB is a non-empty set of $\mathbf{o}$ locus positions (l_i) each assigned an allele value (a_i) subject to these three conditions:

1. $0 < \mathbf{o} < \ell$;
2. $\forall_{\{i \in (1,\ldots,\mathbf{o})\}} l_i \in \{1, \ldots, \ell\}$
3. $\forall_{\{i \in (1,\ldots,\mathbf{o})\}} a_i \in \{1, 0\}$

An example of BB $\tilde{b}$ is given in Equation 6.

$$\tilde{b} \equiv \{(a_1, l_1), \ldots, (a_\mathbf{o}, l_\mathbf{o})\} \tag{6}$$

This BB representation suggests how a complete solution might be formulated and gives rise to the idea that bits need not necessarily be contiguous[4]. Other builders use a contiguous representation of BBs [102]; however, their representation hinders their capability to identify a high epistasis associated with non-contiguous bits.

Finally, a BB cannot be evaluated by itself. A BB needs a helper to fill unspecified loci positions. To evaluate a BB, a BB borrows a complete solution's alleles that are specified at loci positions matching the missing loci positions within the BB.

[3] A locus, l_i, position is a number specifying the bit location within a string of bits representing the solution.

[4] According to the definition, a BB's bits using the representation given here do not need to be located next to each other.

This is formally known as *overlaying* a BB onto a complete solution and is defined in Definition 12 on page 25. Decision variables produced by overlaying a BB onto a complete solution and associated objective vectors are assumed to be computational numbers. Furthermore, it is assumed that an optimal BB is a BB that, once overlaid onto a complete solution, is an optimal solution that can be found in P_{true} and must evaluate to a vector in PF_{true} (see footnote 8 on page 13 for a description of computational number).

2.3 Building Block Definitions

Previous research has produced a few BB definitions [243]. In fact, in 1996, an order-**o** potential BB definition was published by Larry Merkle (see Definition 10 on page 23) [155]. The structure of this definition fits well in the description of how a particular BB is represented. This definition also reveals that the representation may result in an over-specification of bit positions (*i.e.*, the same locus position might be defined twice in the same BB). This particularity is overcome by setting each over-specified locus position to the same value (*i.e.*, for all l_i and l_j, $i = j \Leftrightarrow l_i = l_j$). It is important to notice that the structure of the constraints allows for flexibility in a BB object, but the definition described by Merkle is indeed more rigorous. Therefore, a final constraint is added, #4 (listed below), to make BBs described in this study consistent with Merkle's[5].

4. $\forall_i \forall_j \left\{ i = j \Leftrightarrow l_i = l_j \right\}_{\{(a_i,l_i),(a_j,l_j)\} \subseteq \tilde{b}}$

Juxtaposed[6] BBs (see Figure 2.3 on page 24 for an illustration of a juxtaposition operator juxtaposing BBs) that are overspecified in one locus position must also be

[5] The definition of BB within this chapter is restricted to not have loci positions specified more than once within that same BB (*i.e.,* $\{(1,1),(1,2),(0,1)\}$ is not a valid BB within this chapter.) It should be noted that in the real-world of explicit BBBs this is not realistic and a rule to take the first allele value specified at loci value i when scanning a BB from left-to-right is kept - all other alleles specified at loci value i are ignored. Using this rule, the BB example within this footnote becomes $\{(1,1),(1,2)\}$ which consequently is a valid BB for this chapter.

[6] Juxtaposing BBs means to put different BBs together into a larger BB continuously until finally the BB becomes a complete solution (fully specified solution).

adjusted. The adjustment made is a simple one. BBs are scanned from left-to-right, and the first value for any locus position is kept for BB determination. This particular fix to adhere to this final BB constraint is a simple fix; however, it is not to say that this is the best method for fixing overspecified strings.

Definition 10 (Order-*o* Potential Building Block): *Let $\mathcal{A}$ be a non-empty set (the genic alphabet), $\ell \in \mathbb{Z}^+$ (*the *nominal string length),* $\Lambda \triangleq \{1, \ldots, \ell\}$ *(*the *loci),* and $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_{\mathbf{o}}, l_{\mathbf{o}})\} \in \mathbb{P}(\mathcal{A}x\Lambda)$ *a set of genes. If the loci of* $\tilde{b}$ *are distinct,* i.e., $\tilde{b}$ *satisfies* $i = j \Leftarrow: l_i = l_j$*, then it is called an* **order-o** *potential building block or simply a potential building block.* □

Following Merkle's order-**o** potential BB definition, Jesse Zydallis developed two general definitions describing good single objective BBs and good multiobjective BBs [243]. Definitions 11 on page 23 and 15 on page 39 present his account of what comprises a good BB in both single and multiobjective problems.

Definition 11 (Good Single Objective Building Block (SBB)): *A good single objective BB meets the requirements of Definition 10 on page 23 and the mean fitness value of the BB evaluates to a good fitness value over a number of different allelic combinations placed in the unspecified loci.* □

Zydallis' aim was to generally give meaning to a good BB; however, he does not attempt to include a rigorous method, or a mathematical definition, for identifying how to distinguish a good BB from a bad BB. This missing method or definition is vital in determining on what side of the fence a particular BB might lie. Thus, the next section more rigorously revises Zydallis' good BB definitions. In addition, these extended definitions set up the frame work for the expansion of understanding to even more definitions and the proposal of an alternative idea that good BB identification may lie in both the particular BBB algorithm and the optimal solutions for a particular problem.

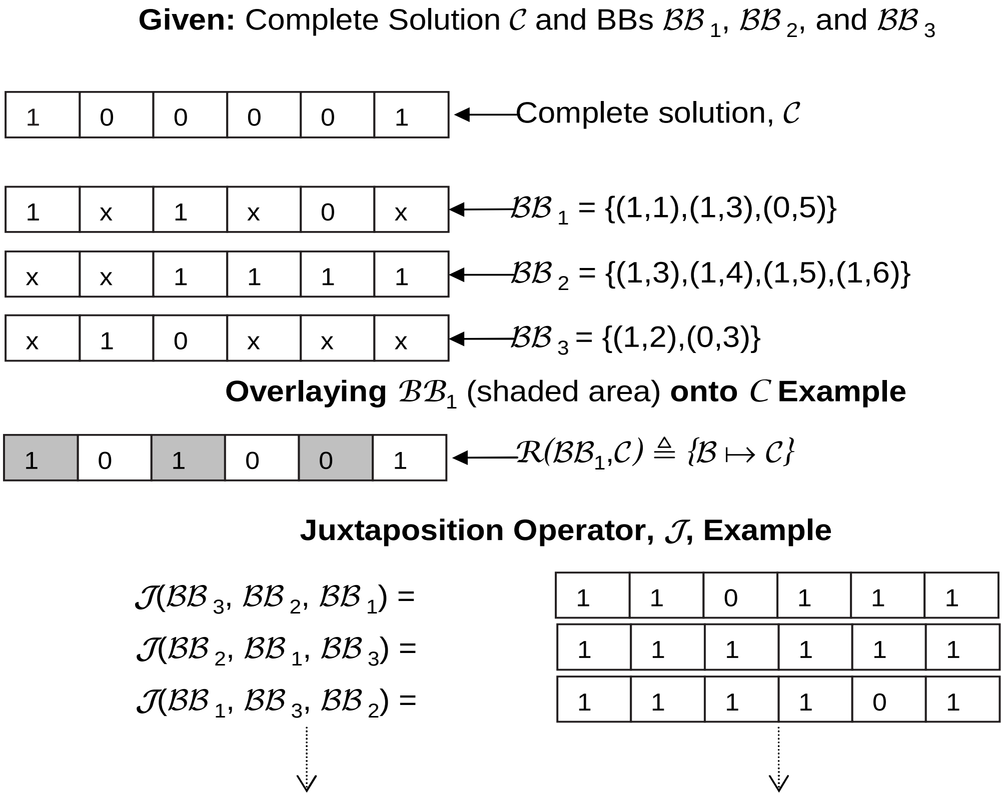


Figure 1: Illustrated are the results of creating a fully specified solution using a left-to-right static juxtapositional operator, $\mathcal{J}$, on three BBs. When $\mathcal{J}$ is applied to a list of BBs, the first allele values specified for loci i are kept no matter how may future alleles are specified to loci i by BBs that come later in the list. BBs are scanned from left-to-right in terms of which to process first with $\mathcal{J}$. In addition, an example of overlaying a BB onto a complete solution is illustrated.

2.4 Extended Good Building Block Definitions

In extending the good BB definition, it is essential to develop a more rigorous definition to be able to test BBs for goodness. Decidedly, there are two types of good BBs: *true* and *sampled*. A *true* order-**o** good BB (GBB_{true}) is one that is good with respect to the entire population of complete solutions and order-**o** BBs. A *sampled* good BB ($GBB_{sampled}$) is a BB that is *good* with respect to a sampling of the entire population of complete solutions[7] and order-$\mathbf{o}$ BBs[8]. Definition 13 defines how to mathematically determine if a BB is *truly* good and Definition 14 defines if a BB is *sampled* as good.

2.4.1 True *Good Building Block.* Some assumptions have been made to develop these extended definitions. First, a higher fitness value equates to a better solution. Secondly, fitness values associated with the entire population have no particular distribution. The third assumption is that the average fitness of a particular BB, $\tilde{b}$, is the average fitness obtained from overlaying[9] $\tilde{b}$ onto each member within the entire population of complete solutions, $\mathbb{P}$[10] (see Table 3 for an example of *overlaying*). In addition, the value that defines the border between good and bad BBs is the average fitness[11] value of all BBs having the same number of loci as $\tilde{b}$, excluding $\tilde{b}$, after being overlaid onto each member of $\mathbb{P}$. Finally, if $\tilde{b}$'s average fitness value is above the good-bad BB boundary[12], it is declared as a *true* good BB.

Definition 12 (Overlaying): *Let a BB be* $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_{\mathbf{o}}, l_{\mathbf{o}})\}$ *where* $\tilde{b}$ *is called the order-$\mathbf{o}$ BB to be overlaid. Let* $\Lambda \triangleq \{1, 2, \ldots, \ell\}$ *where* ℓ *is the length of a*

[7] The entire population of complete solutions is defined as every possible combination that can occur given the string size, ℓ, of a complete solution and the alphabet set $\mathcal{A}$. The matrix of the entire population of complete solutions is generated using a Cartesian product of ℓ copies of the alphabet set, $\mathcal{A}^{\ell}$, and denoted $\mathbb{P}$. For example, if $\ell = 2$, then $\mathbb{P} = \mathcal{A}^{\ell} = \mathcal{A}^2 = \mathcal{A} \times \mathcal{A} = \{0,1\} \times \{0,1\} = \{(0,0),(0,1),(1,0),(1,1)\}$.

[8] The entire population of order-$\mathbf{o}$ BBs, denoted $\mathbb{BB}^{(\mathbf{o})}$, is defined as every possible combination of bit patterns that can be formed using $\mathbf{o}$ bits placed at every combination of $\mathbf{o}$ bit locations in a string of size ℓ such that $\mathbf{o} < \ell$.

[9] *Overlaying* consists of copying allele values from a complete solutions that have loci positions in common with those that are missing within the BB. The copied allele values do not remain within the BB subsequent to its evaluation (see Figure 2.3 on page 24 for a graphical example or Table 3 on page 27 for a written example).

[10] $\mathbb{P}$ normally illustrates the power set; however, within this document it is considered to be every possible solution within the search space of a particular problem. This includes infeasible and feasible regions. The power set is indicated by $\mathbb{P}$.

[11] The average fitness values is used because the extended definitions use the original definitions as a base. The idea to extend the original definitions into a mathematically rigorous definition is to show that the original and extended definitions do not hold (not true) in all situations.

[12] The good-bad BB boundary for BB $\tilde{b}$ of order-$\mathbf{o}$ is located at the mean of all finesses resulting from overlaying all order-$\mathbf{o}$ BBs onto every possible complete solution.

complete solution and $\ell > 0$. *Let* $\mathcal{M}$ *be a matrix having* $\|\mathcal{A}\|^{\mathbf{o}}$ *rows and* $\mathbf{o}$ *columns. Each row of* $\mathcal{M}$ *is an element of* $\mathcal{A}^{\mathbf{o}}$ *(see footnote 7 on page 25 for an example of* $\|\mathcal{A}\|^{\mathbf{o}}$*.). Let* $\mathbb{BB}^{(o)}$ *be the set of all order-*$\mathbf{o}$ *BBs and defined as the following:*

$$\mathbb{BB}^{(\mathbf{o})} \triangleq \bigcup_{\lambda' \subseteq \Lambda, \|\lambda'\| = \mathbf{o}} \left\{ \bigcup_{j \in \{1, \ldots, \|\lambda'\|\}} \left\{ \bigcup_{i \in \{1, \ldots, \|\mathcal{A}\|^{\mathbf{o}}\}} \{(M_{i,j}, \lambda'_j)\} \right\} \right\}$$

Let the overlaying operator, $\mathcal{R}$*, be defined as*

$$\mathcal{R} : \mathbb{BB}^* \times \mathbb{P} \to \mathbb{P}$$

Given: $\mathcal{C}$ *is a complete solution such that* $\mathcal{C} \in \mathbb{P}$. *Let* $\mathcal{C} \triangleq \{(\tilde{a}_1, l_1), \ldots, (\tilde{a}_\ell, l_\ell)\}$, $\mathcal{C}_* \triangleq \{l_1, l_2, \ldots, l_\ell\}$ *and* $\tilde{b}_* \triangleq \{l_{\ell_1}, l_{\ell_2}, \ldots, l_{\ell_\wp}\}$ *where* $|\tilde{b}_*| = \mathbf{o}$.

$$\mathcal{R}(\tilde{b}, \mathcal{C}) = \tilde{b} \cup \left(\mathcal{C} \setminus \acute{\mathcal{C}}\right)$$

where $\acute{\mathcal{C}} = \left\{(\tilde{a}_{\acute{\ell}}, l_{\acute{\ell}}) \in \mathcal{C} : l_{\acute{\ell}} = l_{\ell_i} \texttt{ for some } l_{\ell_i} \in \tilde{b}_*, \acute{\ell} \in \Lambda\right\}$

For an English definition and an example, see Table 3 on page 27. □

It is important to note that when applying this definition, a BB sometimes overlays several different population members and the overlay function causes the new member to be the same as a previously observed member because the BB has replaced the bits making these population members different. In these cases, the BB is causing a pattern to be formed on these different population members and must be evaluated as if it is a new solution regardless of the fact that it is a repeat. This repeated member pattern makes the resulting average value gravitate toward the genuine good or bad nature of the specific BB *under inspection.* In addition, it would be a mistake to overlook the fact that a BB may cause the new member to become infeasible. In most cases, infeasible regions are unwanted and it is better to decode chromosomes around these areas; however, sometimes infeasible instances

Table 3: Example: Overlaying Order-3 BB, $\tilde{b}$, onto a Complete Solutions, $\mathcal{C}$.

The BB, $\tilde{b}$, meets the requirements of Definition 10. The Order-3 BB is overlaid onto the complete solution, $\mathcal{C}$. $\mathcal{C}$ has 5 loci, $\ell = 5$. Let the overlaid BB be

$$\tilde{b} \triangleq \{(a_1, 2), (a_2, 4), (a_3, 5)\}$$

and the complete solutions be

$$\mathcal{C} \triangleq \{(1,1), (1,2), (1,3), (1,4), (1,5)\}.$$

Let the replacement operator R and operator $\mapsto$ be defined as the following:

$$R : \mathbb{BB}^{(\mathbf{o})} \times \mathbb{P} \to \mathbb{P}; \Big\{ R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\} \Big\}$$

where γ meets the requirement of Definition 10, ρ is a complete solution, and the $\mapsto$ operator indicates that the left side alleles replace the right side alleles at the specified loci positions of the left side BB ρ; otherwise, the right side complete solution alleles remain unchanged. Thus, overlaying BB, $\tilde{b}$ onto $\mathcal{C}$ results in the following equality:

$$R(\tilde{b}, \mathcal{C}) \triangleq \{(1,1), (a_1, 2), (1,3), (a_2, 4), (a_3, 5)\}$$

are unavoidable and must be handled differently. Within this chapter, it is assumed that all solutions are feasible[13].

Definition 13 (*True* Good order-o Building Block ($\boldsymbol{GBB^{o}_{true}}$)): *A* true *good BB meets the requirements of Definition 10 and the following: Let a potentially* true *good BB be $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_{\mathbf{o}}, l_{\mathbf{o}})\}$ where $\tilde{b}$ is called the* BB under inspection*. Let $\Lambda \triangleq \{1, 2, \cdots, \ell\}$ where ℓ is the length of a complete solution and $\ell > 0$. Let $\mathcal{M}$ be a matrix having $\|\mathcal{A}\|^{\mathbf{o}}$ rows and $\mathbf{o}$ columns. Each row of $\mathcal{M}$ is an element of*

[13] It may seem to the reader that the assumption of having all feasible solutions in the search space is not realistic; thus, it is also acceptable to assume that BB overlaying that cause infeasible solutions warrant a non-existence clause or hole for that particular evaluation in the definition to determine BB classification. This means that no fitness values are held and that particular solution instance is removed completely from any good/bad/ugly BB calculation. In the case where that evaluation was going to be used as a sampling it does not count and another sample must be made. In the case where that evaluation is required for true BB classification, the evaluation is ignored and all BB summations using that evaluation is reduced by one.

$\mathcal{A}^{\mathbf{o}}$ (see footnote 7 on page 25 for an example of $\|\mathcal{A}\|^{\mathbf{o}}$.). Let $\mathbb{BB}^{(o)}$ be the set of all order-$\mathbf{o}$ BBs and defined as the following:

$$\mathbb{BB}^{(\mathbf{o})} \triangleq \bigcup_{\lambda' \subseteq \Lambda, \|\lambda'\|=\mathbf{o}} \left\{ \bigcup_{j \in \{1,\cdots,\|\lambda'\|\}} \left\{ \bigcup_{i \in \{1,\cdots,\|\mathcal{A}\|^{\mathbf{o}}\}} \{(M_{i,j}, \lambda'_j)\} \right\} \right\}$$

*Let $\mathcal{W} = \mathbb{BB}^{(\mathbf{o})} \backslash \tilde{b}$ where $\mathcal{W}$ is the set of all order-$\mathbf{o}$ BBs with the BB under inspection removed. Let $\mathbb{P} \triangleq \mathcal{A}^{\ell}$ where $\mathbb{P}$ is all combinations of possible complete solutions. Let $f_1(\vec{x})$ and **eval** be defined as the following function:*

$$f_1(\vec{x}) \in \mathbb{R}; \{f_1(\vec{x}) : f_1(\vec{x}) = \mathbf{eval}(\vec{x})\}_{\forall_{\vec{x} \in \mathbb{P}}} \ \texttt{where}\ \mathbf{eval} : \mathbb{P} \to \mathbb{R}$$

*where $f_1(\vec{x})$ is assigned to each complete solution, $\vec{x}$, in $\mathbb{P}$. Defined for completeness, **eval** identifies that there exists an arbitrary function (related to the MOP) that maps a complete solution from the genotype to phenotype domain: existence of the **eval** function and resultant is assumed. Let the replacement operator* R *and operator* $\mapsto$ *be defined as the following:*

$$R : \mathbb{BB}^{(\mathbf{o})} \times \mathbb{P} \to \mathbb{P}; \left\{ R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\} \right\}$$

where γ meets the requirement of Definition 10, ρ is a complete solution, and the $\mapsto$ operator indicates that the left side element parts replace the right side complete solution where the left side element is specified; otherwise, the right side complete solution remains unchanged. $\tilde{b}$ is a **true** *good BB if and only if $\left(\frac{\sum_{k \in \mathbb{P}} f(R(\tilde{b},k))}{\|\mathbb{P}\|} > \frac{\sum_{k \in \mathbb{P}} \sum_{j \in \mathcal{W}} f(R(j,k))}{\|\mathcal{W}\| \cdot \|\mathbb{P}\|} \right)$ which is the same as saying $(\mu_{\tilde{b}} > \mu_{\mathcal{W}})$[14] where $R(\tilde{b}, k)$ and $R(j, k)$ are fully specified solutions.* □

[14] *$\mu_{\tilde{b}}$ is the mean of all fitness values resulting from overlaying $\tilde{b}$ onto each member of the entire population $\mathbb{P}$ and $\mu_{\mathcal{W}}$ is the mean of all fitness values resulting from overlaying $\mu_{\mathcal{W}}$ onto each member of the entire population $\mathbb{P}$*

As it is, Definition 13 yields a concrete mathematical definition for identifying *truly* good and bad BBs. However, as the complete solution string size increases, it becomes infeasible to test if a BB is *truly* good. Instead, an alternate method for identifying good and bad BBs is required - one that is less laborious. Scientists, engineers, and cooks often sample a large group to get information about the entire group without checking each and every member of the group - for any respectable cook knows someone only need sample a spoonful of soup to know how the entire pot tastes. Being a cook myself, it is easy to recognize the merit to this methodology so in the next section a method for tasting the BB soup is statistically described and from it, one can determine a BB's goodness without checking the entire population.

2.4.2 Sampled Good Building Block. Given a set of complete solutions, the test for identifying a true good BB is uncomplicated; however, when using a sampling technique goodness testing becomes complicated because the calculations are estimates of what the entire population might yield resulting in confidence intervals (CIs) about the mean of the sampled data. These confidence intervals present an extra category for a tested BB to be labeled. For now, this category is called *equivalent*[15]. To summarize, a BB under inspection can now be labeled as *good*, *bad* or *equivalent*. As an example of how confidence intervals change the landscape for how to label (good, bad or equivalent) BB under inspection, Figure 2 is provided.

In Figure 2, two different BBs, BB1 and BB2, are tested for goodness against other same sized (order-**o**) BBs. The three possible cases are represented in this figure. In the first case, confidence intervals overlap the mean value of the other sample. This case is presented where BB_1 is compared to $\mathcal{W}_1'$ in Figure 2. Although it may look as if BB_1 is worse than $\mathcal{W}_1'$, because at least one of the confidence intervals overlaps the mean of the other, these two means are considered to be *equivalent*. The second case, where the confidence intervals overlap, but the confidence intervals do

[15] Equivalent BBs are BBs that are no worse than the mean or sample mean. For multiobjective BBs, equivalent is synonymous to the Pareto dominance (for non-dominance) definition.

not cross the mean value barrier of the other is illustrated in Figure 2 when BB_2 is compared to $\mathcal{W}'_1$. In this second case, it seems that BB_2 is better than $\mathcal{W}'_1$; however, a student t-test must be accomplished to establish the goodness of BB_2. The final case, where no confidence interval overlaps the other, is presented in Figure 2 when BB_2 is compared with $\mathcal{W}'_2$. Here, BB_2 visually looks better than $\mathcal{W}'_2$ and because the confidence intervals do not overlap, BB_2 is declared a *sampled* good BB.

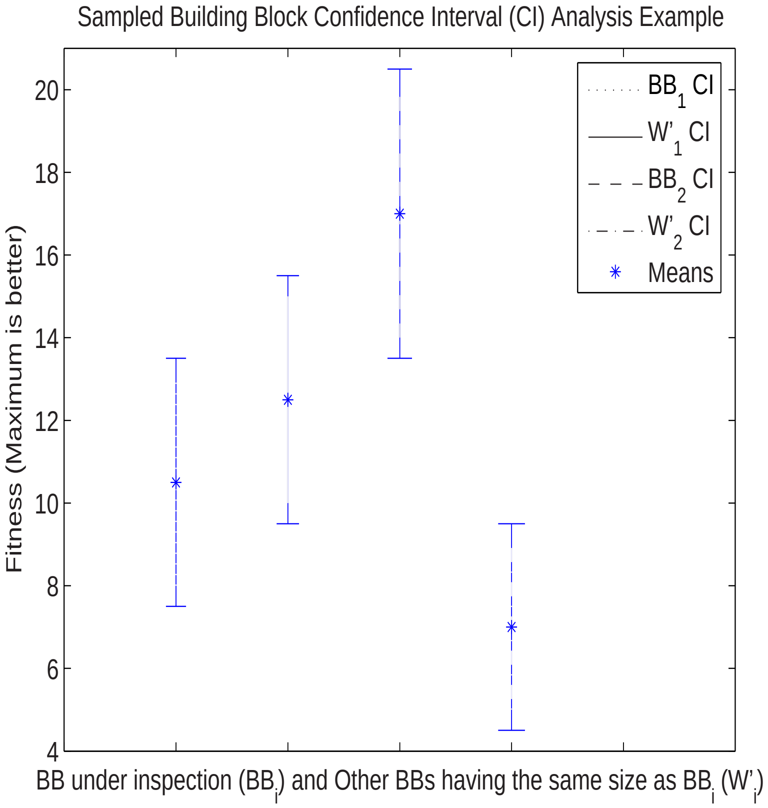


Figure 2: Example of confidence interval tests for differences between means. Within the graph, BB_1 and $\mathcal{W}'_1$ are equivalent, a student-t test must be accomplished to show a different of BB_2 with respect to (w.r.t.) $\mathcal{W}'_1$, and BB_2 is said to be better than $\mathcal{W}'_2$.

The calculation of the confidence intervals is essential for Definition 14 to be used. Thus, a confidence interval for a sampled population[16] mean within a significance level, α, is calculated according to Equation 7. c_{κ_1} is the lower confidence interval value and c_{κ_2} is the upper confidence interval value. The *significance level* translates to a *confidence level* $100(1-\alpha)$ for a particular interval w.r.t.a sampled mean. This can also be called the *confidence coefficient.* Equation 7 presents the confidence interval calculation required at a parameterized significance level, α.

$$\left(\overline{x}_\kappa - z_{1-\frac{\alpha}{2}}\frac{\overline{\sigma}_{\overline{x}_\kappa}}{\sqrt{\mathcal{N}'}}, \overline{x}_\kappa + z_{1-\frac{\alpha}{2}}\frac{\overline{\sigma}_{\overline{x}_\kappa}}{\sqrt{\mathcal{N}'}}\right) \equiv (c_{\kappa_1}, c_{\kappa_2}) \tag{7}$$

In Equation 7, $\overline{x}_\kappa$ is the sample mean and $\overline{\sigma}_{\overline{\mathbf{x}}_\kappa}$ is the sample standard deviation with respect to κ; κ is either the BB under inspection, $\tilde{b}$, or the other sample BBs of the same size, $\mathcal{W}'$. Furthermore, $\mathcal{N}'$ is the sample size and $z_{1-\frac{\alpha}{2}}$ is the $(1-\frac{\alpha}{2})$-quantile of a unit normal variate. When $\kappa \equiv \tilde{b}$, $\mathcal{N}' = \|\mathbb{P}'\|$ and when $\kappa \equiv \mathcal{W}'$ then $\mathcal{N} = \|\mathbb{P}'\| * \|\mathcal{W}'\|$.

In addition, a finite population size factor is applied to the variance because the population size of all possible $\forall_{i\in\mathcal{W}}\forall_{j\in\mathbb{P}}\{i \mapsto j\}$ is known to be $\|\mathcal{W}\| * \|\mathbb{P}\|$ for $\mathcal{W}$ and $\|\mathbb{P}\|$ for $\tilde{b}$. Equation 8 represents this correction where $\mathcal{N}$ is the entire population size and $\mathcal{N}'$ is the sample size taken. When the entire population of order-$\mathbf{o}$ BBs is consulted with respect to one BB, $\tilde{b}$, the variance correction goes to zero. This is a good check because at the point of taking every sample possible the *sampled* good BB definition becomes the same as the *true* good BB definition.

$$\overline{\sigma}_c^2 = \frac{\overline{\sigma}^2}{\mathcal{N}'} \cdot \frac{\mathcal{N}-\mathcal{N}'}{\mathcal{N}-1} \tag{8}$$

[16] Without loss of generality, confidence intervals are determined from the fitness values obtained by overlaying a BB (partial solutions) on top of at least 29 different complete solutions. Taking a sampling of 29 (29 degrees of freedom) makes the average of the t_α-distribution for the gathered data tend to be z_α-normal, even when the distribution from which the average is computed is decidedly non-normal. [166]

For an example of a 90% confidence interval (*i.e.*, $\alpha = 0.1$) the mean variance determination within the Good Sampled BB definition would have $z_{1-\frac{\alpha}{2}} = 1.645$. This confidence coefficient, α, is parameterized within the definition for sampled good BBs.

2.4.3 Student t-test for sampled BBs. Upon encountering overlapping confidence intervals having neither means of $\tilde{b}$ or $\mathcal{W}'$ overlapped by the other's confidence interval, a student t-test is required to identify if the $\tilde{b}$ is a *sampled* good, a bad, or an equivalent BB.

An example of this is presented in Figure 2 when comparing BB_1 and $\mathcal{W}'_2$. To conduct the test on the sample population, $\mathbb{P}'$, and sample BBs, $\mathcal{W}'$, the sample means of each must be calculated. Equation 9 can be used to evaluate the two sample means $\overline{\mu}_{\tilde{b}}$ and $\overline{\mu}_{\mathcal{W}'}$. Note that the notation for $f_1 R(\tilde{b}, \mathbb{P}'_i)$ and $f_1(R(\mathcal{W}'_j, \mathbb{P}'_i))$ is defined in Definition 13.

$$\overline{\mu}_{\tilde{b}} = \frac{1}{||\mathbb{P}'||} \sum_{i=1}^{||\mathbb{P}'||} f_1 \left(R(\tilde{b}, \mathbb{P}'_i) \right) \qquad \overline{\mu}_{\mathcal{W}'} = \frac{1}{||\mathcal{W}'|| \cdot ||\mathbb{P}'||} \cdot \sum_{j=1}^{||\mathcal{W}'||} \sum_{i=1}^{||\mathbb{P}'||} f_1 \left(R(\mathcal{W}'_j, \mathbb{P}'_i) \right) \tag{9}$$

Next is required the calculation of the sample standard deviations for both $\tilde{b}$ and $\mathcal{W}'$: $\overline{\sigma}_{\tilde{b}}$ and $\overline{\sigma}_{\mathcal{W}'}$. Equations 10 and 11 both are used to calculate the sample standard deviations. In addition, the finite population correction factor for this sampled standard deviation must also be applied. See Equation 8 for the corrective factor equation. Following the sample standard deviation calculations, the mean difference (Equation 12), standard deviation of the mean difference (Equation 13), and the effective number of degrees of freedom needs to be calculated (Equation 14).

$$\overline{\sigma}_{\tilde{b}} = \left\{ \frac{\left(\sum_{i=1}^{\|\mathbb{P}'\|} f_1 \left(R(\tilde{b}, \mathbb{P}'_i) \right)^2 \right) - \|\mathbb{P}'\| * \overline{\mu}_{\tilde{b}}^2}{\|\mathbb{P}'\| - 1} \right\}^{\frac{1}{2}} \tag{10}$$

$$\overline{\sigma}_{\mathcal{W}'} = \left\{ \frac{\left(\sum_{j=1}^{\|\mathcal{W}'\|} \sum_{i=1}^{\|\mathbb{P}'\|} f_1 \left(R(\mathcal{W}'_j, \mathbb{P}'_i) \right)^2 \right) - \|\mathcal{W}'\| \cdot \|\mathbb{P}'\| \cdot \overline{\mu}_{\mathcal{W}'}^2}{\|\mathcal{W}'\| \cdot \|\mathbb{P}'\| - 1} \right\}^{\frac{1}{2}} \tag{11}$$

$$\overline{\mu}_{diff} = \overline{\mu}_{\tilde{b}} - \overline{\mu}_{\mathcal{W}'} \tag{12}$$

$$\overline{\sigma}_{diff} = \sqrt{\frac{\overline{\sigma}_{\mathcal{W}'_c}^2}{\| \mathcal{W}'\| \cdot \|\mathbb{P}'\|} + \frac{\overline{\sigma}_{\tilde{b}_c}^2}{\|\mathbb{P}'\|}} \tag{13}$$

$$\nu = \frac{\left(\frac{\overline{\sigma}_{\mathcal{W}'_c}^2}{\|\mathcal{W}'\| \cdot \|\mathbb{P}'\|} + \frac{\overline{\sigma}_{\tilde{b}_c}^2}{\|\mathbb{P}'\|} \right)^2}{\frac{1}{\|\mathcal{W}'\| \cdot \|\mathbb{P}'\| + 1} \cdot \left(\frac{\overline{\sigma}_{\mathcal{W}'_c}^2}{\|\mathcal{W}'\| \cdot \|\mathbb{P}'\|} \right)^2 + \frac{1}{\|\mathbb{P}'\| + 1} \cdot \left(\frac{\overline{\sigma}_{\tilde{b}_c}^2}{\|\mathbb{P}'\|} \right)^2} - 2 \tag{14}$$

$$(\overline{\mu}_{diff} - t_{[\frac{1-\alpha}{2};\nu]} \cdot \overline{\sigma}_{diff}, \overline{\mu}_{diff} + t_{[\frac{1-\alpha}{2};\nu]} \cdot \overline{\sigma}_{diff}) = (c_{t1}, c_{t2}) \tag{15}$$

Finally, the confidence interval for the mean difference is required. Equation 15 presents the equation used to calculate this interval. The $t_{[\frac{1-\alpha}{2};\nu]}$ is the $(1 - \frac{1}{\alpha})$-quantile of a t-variate with ν degrees of freedom. In conclusion, if the confidence interval $(c_{t1}$,$c_{t2})$ includes zero, the difference is *not* significant at $100(1 - \alpha)\%$ confidence level. Otherwise, the BB can be considered a *sampled* good BB if $\overline{\mu}_{\tilde{b}} > \overline{\mu}_{\mathcal{W}'}$ or bad BB if $\overline{\mu}_{\tilde{b}} < \overline{\mu}_{\mathcal{W}'}$. This process is summarized in Definition 14.

Definition 14 (Sampled Good Building Block ($\boldsymbol{GBB_s^{o,\alpha}}$)): *A* sampled *good BB meets the requirements of Definition 10 and the following: Let a potential good BB be $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_o, l_{\mathbf{o}})\}$ where $\tilde{b}$ is labelled BB under inspection. Let $\Lambda \triangleq \{1, 2, \cdots, \ell\}$ where ℓ is the length of a complete solution and $\ell > 0$. Let $\mathcal{M}$ be a* *matrix having* $\|\mathcal{A}\|^{\mathbf{o}}$ *rows and* $\mathbf{o}$ *columns. Each row of* $\mathcal{M}$ *is an element of* $\mathcal{A}^{\mathbf{o}}$*. Let* $\mathbb{BB}^{(o)}$ *be the set of all order-*$\mathbf{o}$ *BBs,* λ*' is an ordered set and defined as the following:*

$$\mathbb{BB}^{(\mathbf{o})} \triangleq \bigcup_{\lambda' \subseteq \Lambda, \|\lambda'\| = \mathbf{o}} \left\{ \bigcup_{j \in \{1, \cdots, \|\lambda'\|\}} \left\{ \bigcup_{i \in \{1, \cdots, \|\mathcal{A}\|^{\mathbf{o}}\}} \{(M_{i,j}, \lambda'_j)\} \right\} \right\}$$

Let $\mathcal{BB}' \subseteq \mathbb{BB}^{(\mathbf{o})}$ *where* $\mathcal{BB}'$ *is the* sampled *set of order-*$\mathbf{o}$ *BBs. Let* $\mathbb{P} \triangleq \mathcal{A}^{\ell}$ *where* $\mathbb{P}$ *is the entire population of complete solutions. Let* $\mathbb{P}' \subseteq \mathbb{P}$ *where* $\mathbb{P}'$ *is a subset, or sampled set, of the entire population of complete solutions. Let* $f_1(\vec{x})$ *and* ***eval*** *be defined as the following function:*

$$f_1(\vec{x}) \in \mathbb{R}; \{f_1(\vec{x}) : f_1(\vec{x}) = \mathbf{eval}(\vec{x})\}_{\forall_{\vec{x} \in \mathbb{P}}} \ \texttt{where}\ \mathbf{eval} : \mathbb{P} \rightarrow \mathbb{R}$$

where $f_1(\vec{x})$ *is assigned to each complete solution,* $\vec{x}$*, in* $\mathbb{P}$*. Defined for completeness,* ***eval*** *identifies that there exists an arbitrary function (related to the MOP) that maps a complete solution from the genotype to phenotype domain: existence of the* ***eval*** *function and resultant is assumed. Let the replacement operator* R *and operator* $\mapsto$ *be defined as the following:*

$$R : \mathbb{BB}^{(\mathbf{o})} \times \mathbb{P} \rightarrow \mathbb{P}; R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\}$$

where γ *meets the requirement of Definition 10,* ρ *is a complete solution, and the* $\mapsto$ *operator indicates that the left side element parts replace the right side complete solution where the left side element is specified; otherwise, the right side complete solution remains unchanged.* $\tilde{b}$ *is defined as follows:*

$$\tilde{b} = \begin{cases} \texttt{possible good} & : (\overline{\mu}_{\tilde{b}} > \overline{\mu}_{\mathcal{W}'}) \\ \texttt{possible bad} & : (\overline{\mu}_{\tilde{b}} < \overline{\mu}_{\mathcal{W}'}) \\ \texttt{equivalent} & : \textit{otherwise} \end{cases} \tag{16}$$

$$\texttt{possible good} = \begin{cases} \texttt{good} & : \left(c_{\tilde{b}_1} > c_{\mathcal{W}'_2}\right) \\ \texttt{equivalent} & : \left(c_{\tilde{b}_1} > \overline{\mu}_{\mathcal{W}'} > c_{\tilde{b}_2}\right) \texttt{ or } \left(c_{\mathcal{W}'_1} > \overline{\mu}_{\tilde{b}} > c_{\mathcal{W}'_2}\right) \\ \texttt{ugly(good)} & : \textit{otherwise} \end{cases}$$

$$\texttt{possible bad} = \begin{cases} \texttt{bad} & : \left(c_{\tilde{b}_2} < c_{\mathcal{W}'_1}\right) \\ \texttt{equivalent} & : \left(c_{\tilde{b}_1} < \overline{\mu}_{\mathcal{W}'} < c_{\tilde{b}_2}\right) \texttt{ or } \left(c_{\mathcal{W}'_1} < \overline{\mu}_{\tilde{b}} < c_{\mathcal{W}'_2}\right) \\ \texttt{ugly(bad)} & : \textit{otherwise} \end{cases}$$

$$\texttt{ugly(x)} = \begin{cases} \texttt{equivalent} & : (c_{t1} < zero < c_{t2}) \\ \texttt{x} & : \textit{otherwise} \end{cases}$$

□

Two more examples of comparing $\tilde{b}$ and $\mathcal{W}'$ are illustrated in Figures 3 and 4. These figures illustrate a $\mu_{\tilde{b}}$ having a non-overlapping (Fig 3) and an overlapping (Fig 4) confidence interval when compared to $\mu_{\mathcal{W}'}$. These illustrations are important because they label the objective (fitness) area associated with good and bad BBs. If a BB falls into the good BB area, its confidence interval does not overlap the mean of the good-bad marker and it passes the student-t test requirement for having a different mean and it can theoretically be called a good BB. Conversely, if a BB falls into the bad BB area, its confidence interval does not overlap the mean of the good-bad marker and it passes any student-t test requirement for having a different mean, it can be called a bad BB.

One final sanity check for this definition would be to let $\|W'\| \rightarrow \|W\|$ and $\|\mathbb{P}'\| \rightarrow \|\mathbb{P}\|$ which is exactly what the *true* good BB definition reflects. The resulting *sampled* good BB test becomes identical to the *true* good BB test because the

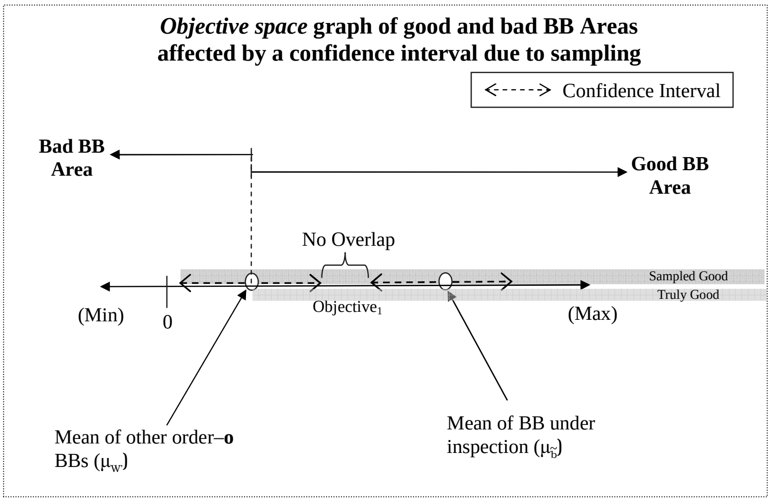

Objective space graph of good and bad BB Areas affected by a confidence interval due to sampling
<----> Confidence Interval
Bad BB Area
Good BB Area
No Overlap
Sampled Good
Truly Good
(Min)
0
Objective₁
(Max)
Mean of other order–o BBs (μw)
Mean of BB under inspection (μb̃)

Figure 3: This figure presents an example of a sampled single objective$_1$ BB test where the confidence intervals of the mean of order-**o** BBs and the BB under test do not overlap.

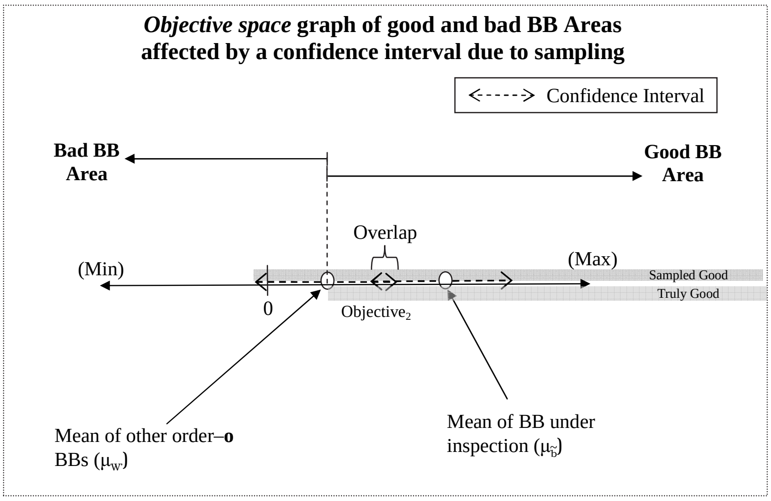

Objective space graph of good and bad BB Areas affected by a confidence interval due to sampling
<----> Confidence Interval
Bad BB Area
Good BB Area
Overlap
(Min)
(Max)
Sampled Good
Truly Good
0
Objective₂
Mean of other order–o BBs (μw)
Mean of BB under inspection (μb̃)

Figure 4: This figure presents an example of a sampled single objective$_2$ BB test where the confidence intervals of the mean of order-**o** BBs and the BB under test overlap.

finite population correction factor reduces the confidence intervals to zero making $(c_{\kappa_1} = c_{\kappa_2} = \mu_\kappa)$. In this case, $\mu_{\tilde{b}}$ is a good BB if and only if $\mu_{\tilde{b}} > \mu_{\mathcal{W}}$.

Finally, there is an adjustment to be made to the confidence coefficient because the confidence level is shared by k confidence intervals. Thus, the α is multiplied by the number of objective values, k, associated with each solution. Accordingly, if a 90% confidence and α is set to 0.1, the z values change to account for the shared confidence level. The new z values become: $z_{1-\frac{k\alpha}{2}} = (1.960, 2.326, 2.576, \cdots)$ where $k = (2, 3, 4, \cdots)$.

This concludes the single objective true and sampled good BB definitions section. Next, the multicriteria good BB definition is discussed.

2.5 Good Multiobjective Building Blocks

Multicriteria problems make selecting a best solution more difficult. In fact, instead of having a single best solution[17], multiple solutions are declared as the best or declared optimal (*equivalent*) solutions. These equivalent solutions are said to be Pareto optimal and evaluate to be in the non-dominated set of Pareto front vectors. Definition 2 on page 11 describes what it means to be a non-dominated vector. Similar to the single objective good BB definitions cited in Definitions 13 and 14), new multiobjective good BB definitions are now more rigorously defined.

It is the position of this research that there are two forms of good order-**o** BBs: *true* and *sampled*. *Sampled* order-**o** BBs have similar qualities as the solutions evaluating to the PF_{known} vectors. *True* order-**o** BBs are non-dominated w.r.t. all other order-**o** BBs and have similar properties to solutions evaluating to PF_{true} vectors. The decision variable set mapping to these PF_{true} are denoted as the *true* Pareto optimal set (see Definition 3). The members of the *true* Pareto optimal set are akin to the *true* good multiobjective BBs for this chapter. Identifying these *true* good multiobjective BBs requires a more rigorously extended Good Multiobjective BB definition because the current definitions, written by Zydallis in 2002 [243], say nothing of *true* or *sampled* BBs.

[17] This is not to say that some single objective problem do not have multiple optimal solutions. The difference lies in the fact these multiple single optimal solutions all have the same exact objective value (making it easy for one to declare all solutions evaluating to this optimal objective value to be optimal).

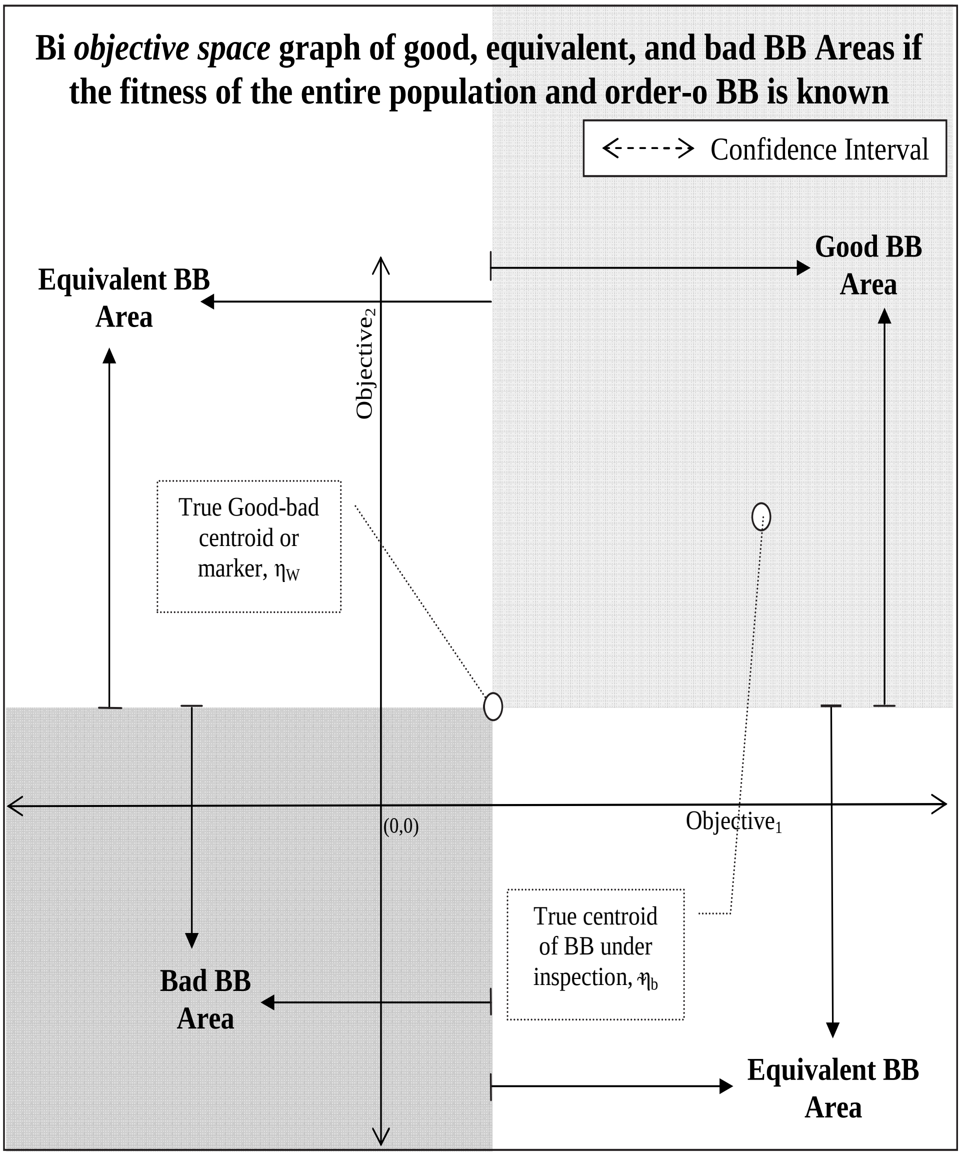

Bi *objective space* graph of good, equivalent, and bad BB Areas if the fitness of the entire population and order-o BB is known
<-----> Confidence Interval
Equivalent BB Area
Good BB Area
Objective$_2$
True Good-bad centroid or marker, η_W
(0,0)
Objective$_1$
True centroid of BB under inspection, η_b
Bad BB Area
Equivalent BB Area

Figure 5: Presented is an example of a true multiobjective BB test. Individually, each objective is shown in its single objective form in Figures 3 and 4. The areas to the upper right of $\mathcal{W}'$'s centroid indicates the area where good BBs are found, and the area to the bottom left of $\mathcal{W}'$'s centroid indicates the bad BBs. The other two unshaded areas indicate an area where equivalent BBs are found. These equivalent BBs are neither good nor bad - in fact, they represent all BBs that, after being overlaid into a complete solution, evaluate to have a non-dominated status w.r.t.$\mathcal{W}'$'s centroid, but they are labelled *equivalent*.

Definition 15 (Good Multiobjective Building Block (GMBB)): *A good multiobjective BB, meets the requirements of Definition 10 and the mean fitness value of the BB dominates (evaluates to a good fitness value as compared to) the fitness values of other BBs or population members in comparison testing based on Pareto dominance criteria. The evaluation of a multiobjective BB is conducted over a number of different allelic combinations placed in the unspecified loci.* □

Definition 15 generally describes the meaning of a good multiobjective BB; however, there is no attempt to have a rigorous description of how to test a BB for multiobjective goodness, in fact this definition needs to be modified to show how a mean fitness value is defined for a multicriteria BB (*i.e.*, one with more than one fitness value).

2.5.1 True *Good Multiobjective Building Blocks.* Several assumptions are required to provide a basis for the development of the revised definitions found in this section. First, a higher objective value equates to a better objective evaluation for a solution. Secondly, k is the number of objectives being optimized. Third, objective values associated with the entire population have no particular distribution. The fourth assumption is that the average fitness of BB, $\tilde{b}$, is calculated by taking the *centroid* of the objective values resulting from overlaying $\tilde{b}$ onto each member of the entire population of complete solutions ($\mathbb{P}$). In addition, the good-bad centroid (marker) is found in a similar manner. The centroid of the objective values resulting from evaluating each order-$\mathbf{o}$ BB, except $\tilde{b}$, after being overlaid on each member of the entire population of complete solutions is considered to be the good-bad marker.

There are two important differences between the single objective good-bad border and the multiobjective good-bad marker. First, the un-sampled single objective border divides the objective space into two distinct regions where the multiobjective marker divides the space into four regions or three distinct regions. Second, BBs can evaluate to be only good or bad in a single objective space where as BBs can evaluate to be good, equivalent (evaluates to be non-dominated), or bad in a multiobjective space. Figure 5 illustrates a bi-objective good-bad marker where the centroid divides the objective space into good, equivalent, and bad BBs. Finally, when the centroid of the evaluation of $\tilde{b}$, once overlaid into a complete solution, dominates the good-bad marker, the BB is identified as *truly* a GMBB. Definition 16 mathematically defines the criteria of what it means to be a *truly* good, bad or equivalent order-$\mathbf{o}$ BB.

Definition 16 (True Good Multiobjective Building Block ($\boldsymbol{GMBB^{o}}$)): *A* true *good MBB meets the requirements of Definition 10 and the following: Let a potentially* true *good BB be* $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_o, l_\mathbf{o})\}$ *where* $\tilde{b}$ *is labeled the BB under inspection. Let* $\Lambda \triangleq \{1, 2, \cdots, \ell\}$ *where* ℓ *is the length of a complete solution and* $\ell > 0$*. Let* $\zeta \triangleq \{1, 2, \cdots, k\}$ *where* k *is the number of optimization objectives. Let* $\mathcal{M}$ *be a matrix having* $\|\mathcal{A}\|^\mathbf{o}$ *rows and* $\mathbf{o}$ *columns. Each row of* $\mathcal{M}$ *is an element of* $\mathcal{A}^\mathbf{o}$*. Let* $\mathbb{BB}^{(\mathbf{o})}$ *be the set of all order-*$\mathbf{o}$ *BBs and defined as the following:*

$$\mathbb{BB}^{(\mathbf{o})} \triangleq \bigcup_{\lambda' \subseteq \Lambda, \|\lambda'\| = \mathbf{o}} \left\{ \bigcup_{j \in \{1, \cdots, \|\lambda'\|\}} \left\{ \bigcup_{i \in \{1, \cdots, \|\mathcal{A}\|^\mathbf{o}\}} \{(M_{i,j}, \lambda'_j)\} \right\} \right\}$$

Let $\mathcal{W} = \mathbb{BB}^{(o)} \setminus \tilde{b}$*. Let* $\mathbb{P} \triangleq \mathcal{A}^\ell$ *where* $\mathbb{P}$ *is all combinations of possible complete solutions. Let* $f_\tau(\vec{x})$ *and* ***eval*** *be defined as the following function:*

$$f_\tau(\vec{x}) \in \mathbb{R}; \{f_\tau(\vec{x}) : f_\tau(\vec{x}) = \mathbf{eval}_\tau(\vec{x})\}_{\vec{x} \in \mathbb{P}, \tau \in \zeta} \texttt{ where } \mathbf{eval}_\tau : \mathbb{P} \to \mathbb{R}$$

where $f_\tau(\vec{x})$*s are assigned to each complete solution,* $\vec{x}$*, in* $\mathbb{P}$*. Let the replacement operator* R *and operator* $\mapsto$ *be defined as the following:*

$$R : \mathbb{BB}^{(\mathbf{o})} \times \mathbb{P} \to \mathbb{P}; R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\}$$

where γ *meets the requirement of Definition 10,* ρ *is a complete solution, and the* $\mapsto$ *operator indicates that the left side element parts replace the right side complete* *solution where the left side element is specified; otherwise, the right side complete solution remains unchanged. Let* ${\eta_\tau}^{(\tilde{b})} = \left\{\frac{\sum_{\vec{x}\in\mathbb{P}} f_\tau(\vec{x})}{\|\mathbb{P}\|}\right\}_{\tau\in\zeta}$ *where* $\vec{\eta}^{\,(\tilde{b})}$ *is called* $\tilde{b}$*'s centroid. Let* ${\eta_\tau}^{(\mathcal{W})} = \left\{\frac{\sum_{\vec{x}\in\mathbb{P}}\sum_{j\in\mathcal{W}} f_\tau(R(j,\vec{x}))}{\|\mathcal{W}\|*\|\mathbb{P}\|}\right\}_{\tau\in\zeta}$ *where* $\vec{\eta}^{\,(\mathcal{W})}$ *is the centroid for* $\mathcal{W}$*.*

$$\tilde{b} = \begin{cases} \texttt{good} & : \forall_{\tau\in\zeta}\left({\eta_\tau}^{(\tilde{b})} > {\eta_\tau}^{(\mathcal{W})}\right) \\ \texttt{bad} & : \forall_{\tau\in\zeta}\left({\eta_\tau}^{(\tilde{b})} < {\eta_\tau}^{(\mathcal{W})}\right) \\ \texttt{equivalent} & : \textit{otherwise} \end{cases} \tag{17}$$

□

2.5.2 Sampled Good Multiobjective Building Blocks. Delineating decision boundaries for *sampled* good multiobjective BBs (GMBB) is not quite as simple as it is for the single objective case; however, the student-t test still applies. The large difference is in the confidence coefficient α being applied to the distribution. Equation 18 describes the confidence level adjustment that must be made for calculating confidence intervals for simultaneous confidence intervals with a family confidence coefficient $1-\alpha$. This adjustment is called Bonferroni's inequality and is a lower bound on the true (but often unknown) family confidence coefficient and is correct more than $(1-\alpha)100$ percent of the time.

$$P\left(\bigcap_{i=1}^{k}\bar{A}_i\right) \geq 1 - k\alpha \tag{18}$$

$$\left(\overline{x}_{\kappa_\tau} - z_{1-\frac{k\alpha}{2}}\frac{\sigma_{\overline{x}_{\kappa_\tau}}}{\sqrt{\mathcal{N}}}, \overline{x}_{\kappa_\tau} + z_{1-\frac{k\alpha}{2}}\frac{\sigma_{\overline{x}_{\kappa_\tau}}}{\sqrt{\mathcal{N}}}\right) \equiv \left(c_{\kappa_{\tau_1}}, c_{\kappa_{\tau_2}}\right) \tag{19}$$

Confidence intervals for k objectives are calculated using Equation 19 where τ indicates the objective number (see footnote 16 on page 31 for the minimum sample size requirement). The result of calculating k confidence intervals is a k dimensional shape representing an area that includes, with $100(1-k\alpha)\%$ confidence, the centroid of either $\tilde{b}$ or $\mathcal{W}'$ ($\kappa = \tilde{b}$ or $\mathcal{W}'$). The sampled MGBB is illustrated in Figure 6.

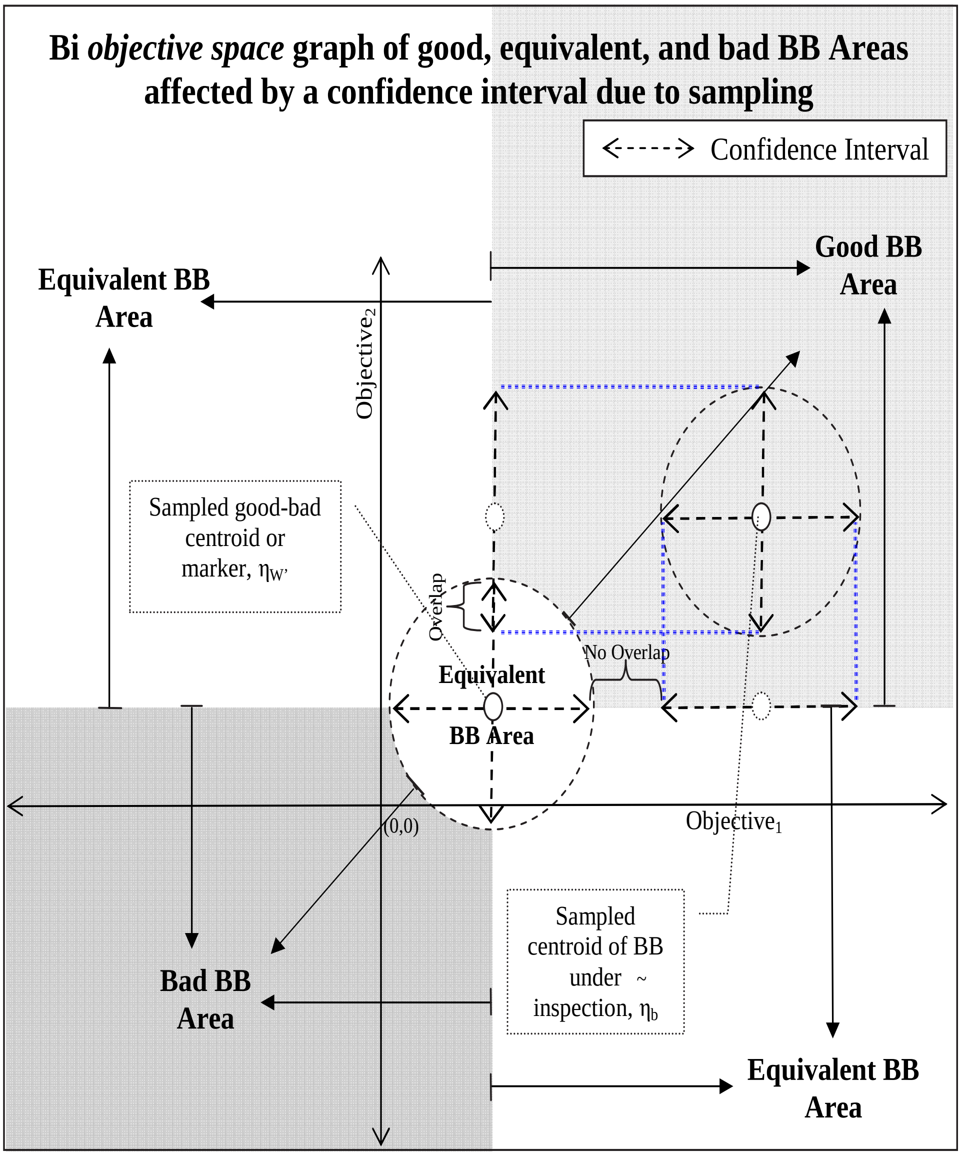

Bi *objective space* graph of good, equivalent, and bad BB Areas affected by a confidence interval due to sampling
<----> Confidence Interval
Good BB Area
Equivalent BB Area
Objective$_2$
Sampled good-bad centroid or marker, $\eta_{W'}$
Overlap
No Overlap
Equivalent BB Area
(0,0)
Objective$_1$
Bad BB Area
Sampled centroid of BB under ~ inspection, η_b
Equivalent BB Area

Figure 6: This figure presents an example of a sampled multiobjective BB test where the confidence intervals of the centroid of order-**o** BBs and the BB under test overlap are only in objective 2. Individually, each objective is shown in its single objective form in Figures 3 and 4. To the upper right and outside of the circle made by the confidence intervals of $\mathcal{W}'$'s centroid indicates the area where good BBs are found, and the area to the bottom left and outside of the circle made by the confidence intervals of $\mathcal{W}'$'s centroid indicates the bad BBs. The other two unshaded areas indicate an area where equivalent BBs are found. These equivalent BBs are neither good nor bad - in fact, they represent a BB that, after being overlaid onto a complete solution, evaluates to be non-dominated w.r.t. the good-bad centroid marker. These BBs evaluating to a non-dominated state are labelled *equivalent*. Note also that the area closed in by the confidence intervals is an *equivalent* area because the centroid's confidence interval would overlap anything found inside the circle.

There is one drawback to using the multidimensional confidence intervals in deciding if two centroids are different. The drawback comes in the form of how to decide if there is an overlap of confidence intervals. A researcher may use one of the following two methods to determine if an overlapped confidence interval exists. The first is a quick and dirty method and the second is a long method. In using either method, one must have the centroid of the BB under inspection located in the good BB area if attempting to call the BB good. This is illustrated by the big **X** in the upper right gray area of the objective space in Figure 7. Otherwise, the BB is either an equivalent BB or bad BB and does not make the requirements of a *sampled* good BB.

The two methods available to determine confidence interval overlap are discussed next. First, the quick method is discussed and a recursive mathematical definition is given that can be used for any objective space dimensionality. Secondly, the long method is given. The second method is the more rigorous and accurate method. Following the descriptions of these two methods, the sampled good multi-objective BB definition is given.

Quick

The quick and dirty method makes the assumption that the direct path (diagonal) confidence interval is an aggregate confidence interval that is equivalent to checking each objective dimension simultaneously for difference of centroids (means). The first step is the determining of each objective's confidence interval from sampling for the good-bad marker and incoming BB under inspection ($\tilde{b}$). This step uses Equation 19. The second step is determining the length of each confidence interval in the direction of each centroid. If either of the confidence intervals from $\mathcal{W}$ or $\tilde{b}$ is larger than the distance between the two centroids, it can be immediately concluded that there is no difference between the two centroids. In addition, if the sum of the confidence intervals is less than the distance between the two centroids then there is a difference between the centroids and $\tilde{b}$ can be said to be a good BB.

$$
\begin{aligned}
r^2(\tau) &= \frac{T_u(\tau)}{T_b(\tau)} \\
T_u(\tau) &= T_u(\tau-1) * a_1^2,\ T_b(1) = 1 \\
T_b(\tau) &= T_b(\tau-1) * a_\tau^2 \sin^2(\theta_{\tau-1}) + \left(\prod_{j=1}^{\tau-1} a_j^2\right) * \cos^2(\theta_{\tau-1}) \\
\mathit{where}\ a_\tau &= z_{1-\frac{k\alpha}{2}} * \frac{\sigma_{\overline{x}_\tau}}{\sqrt{\mathbf{n}}}, \theta_\tau = \arctan\left(\frac{\overline{x}_{\mathcal{W}_{\tau+1}} - \overline{x}_{\tilde{b}_{\tau+1}}}{\overline{x}_{\mathcal{W}_\tau} - \overline{x}_{\tilde{b}_\tau}}\right)
\end{aligned}
\tag{20}
$$

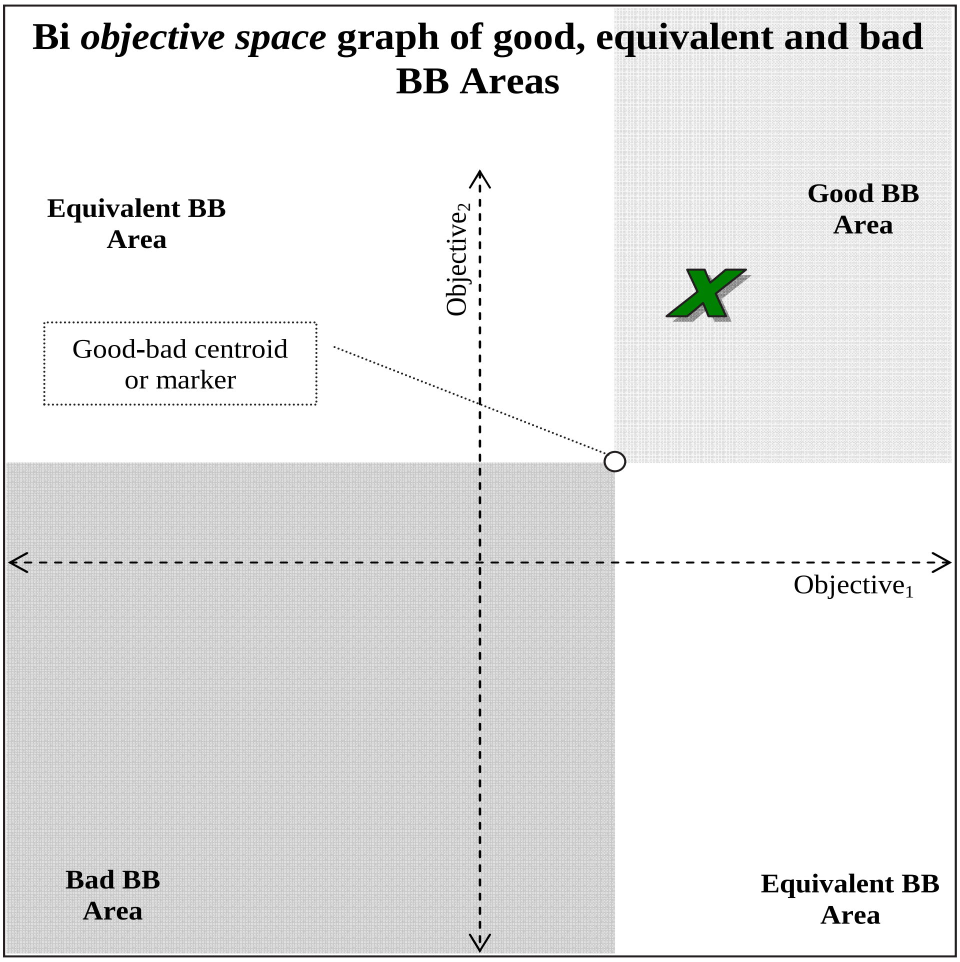

Bi *objective space* graph of good, equivalent and bad BB Areas
Equivalent BB Area
Good BB Area
Good-bad centroid or marker
Objective$_2$
Objective$_1$
Bad BB Area
Equivalent BB Area

Figure 7: Example of objective space good, bad and equivalent regions for a bi-objective problem. The good-bad centroid marker is represented with a small circle at the intersecting point for all regions. The large X marks the possible placement of a BB that would evaluate to being good by Definition 16 on page 40.

In order to calculate the direct path confidence interval, a recursive equation for calculating the outer radius of a k-dimensional ellipsoid must be used. Equation 20 represents such a formula. The confidence interval values, $c_{\kappa_{\tau_2}}$, minus the sampled mean, $\bar{x}_{\kappa_\tau}$, are used as the a_τ values in Equation 20. Figure 8 illustrates the direct path confidence interval between the BB under inspection, $\tilde{b}$, and the good-bad marker's, $\mathcal{W}$', centroid. Notice that in the figure the direct path confidence intervals indicated by the ellipse drawn around the centroids do not overlap. Yet, if each objective is taken individually, the confidence intervals overlap in objective 2. It is for this reason that the long method is the more developed and more accurate method.

Long

This method is the same as the single objective test, except that the test is repeated for each of the k objectives. If $\tilde{b}$'s centroid is located within the good BB area and it fails to be different in any one objective, it is called an *equivalent* BB - failing to be a *sampled* good BB. This method is also assumed to be the more accurate of the two difference checks discussed in this document. Furthermore, it is included in the *sampled* good multiobjective BB definition (Definition 17). Finally, Equations 21-27 are added to extend the single objective calculations to k dimensions. Each BB under inspection has k different confidence intervals of which the final calculation can be found in Equation 19. In the equations to calculate the k confidence intervals, τ is used as the distinguishing objective. Note that in these equations, Bonferroni's inequality is applied (see Equation 18), as well as the finite population correction factor found in Equation 8.

$$\overline{\mu}_{\tilde{b}_\tau} = \frac{1}{\|\mathbb{P}'\|} \sum_{i=1}^{\|\mathbb{P}'\|} f_\tau \left(R(\tilde{b}, \mathbb{P}'_i) \right) \qquad \overline{\mu}_{\mathcal{W}'_\tau} = \frac{1}{\|\mathcal{W}'\| \cdot \|\mathbb{P}'\|} * \sum_{j=1}^{\|\mathcal{W}'\|} \sum_{i=1}^{\|\mathbb{P}'\|} f_\tau \left(R(\mathcal{W}'_j, \mathbb{P}'_i) \right) \quad (21)$$

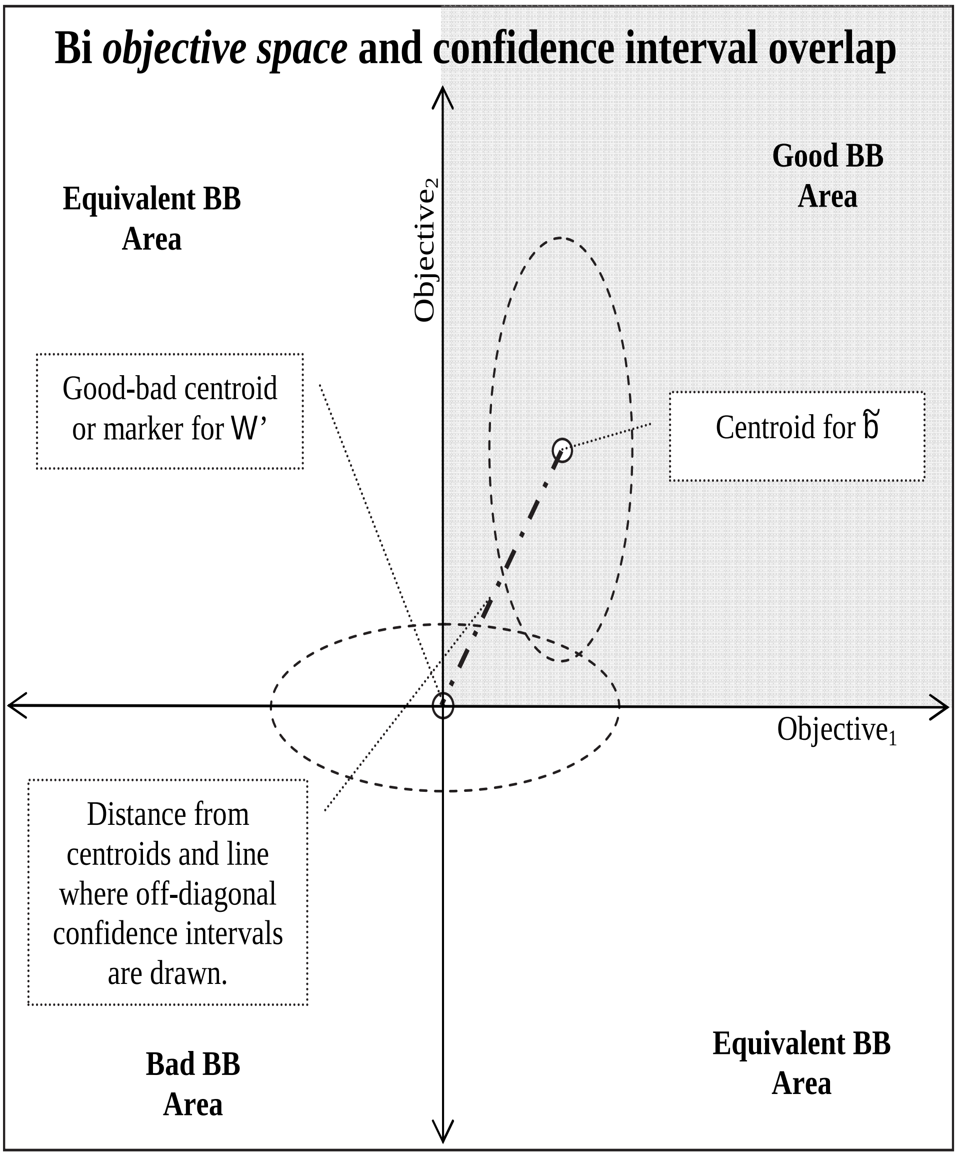

Bi *objective space* and confidence interval overlap
Good BB Area
Equivalent BB Area
Objective$_2$
Good-bad centroid or marker for W'
Centroid for $\tilde{b}$
Objective$_1$
Distance from centroids and line where off-diagonal confidence intervals are drawn.
Bad BB Area
Equivalent BB Area

Figure 8: This figure presents the ellipsoid confidence intervals that can be generated when solving an MOP. In addition, presented is the direct path confidence interval that would result between two centroids. The length of each interval is indicated by where the straight line distance line between the two centroids crosses the ellipse.

$$\overline{\sigma}_{\tilde{b}_\tau} = \left\{ \frac{\left(\sum_{i=1}^{\|\mathbb{P}'\|} f_\tau \left(R(\tilde{b}, \mathbb{P}'_i)\right)^2\right) - \|\mathbb{P}'\| * \overline{\mu}^2_{\tilde{b}_\tau}}{\|\mathbb{P}'\| - 1} \right\}^{\frac{1}{2}} \tag{22}$$

$$\overline{\sigma}_{\mathcal{W}'_\tau} = \left\{ \frac{\left(\sum_{j=1}^{\|\mathcal{W}'\|} \sum_{i=1}^{\|\mathbb{P}'\|} f_\tau \left(R(\mathcal{W}'_j, \mathbb{P}'_i)\right)^2\right) - \|\mathcal{W}'\| * \|\mathbb{P}'\| * \overline{\mu}^2_{\mathcal{W}'_\tau}}{\|\mathcal{W}'\| * \|\mathbb{P}'\| - 1} \right\}^{\frac{1}{2}} \tag{23}$$

$$\overline{\mu}_{diff_\tau} = \overline{\mu}_{\tilde{b}_\tau} - \overline{\mu}_{\mathcal{W}'_\tau} \tag{24}$$

$$\overline{\sigma}_{diff_\tau} = \sqrt{\frac{\overline{\sigma}^2_{\mathcal{W}'_{\tau_c}}}{\| \mathcal{W}'\| * \|\mathbb{P}'\|} + \frac{\overline{\sigma}^2_{\tilde{b}_{\tau_c}}}{\|\mathbb{P}'\|}} \tag{25}$$

$$\nu_\tau = \frac{\left(\frac{\overline{\sigma}^2_{\mathcal{W}'_{\tau_c}}}{\|\mathcal{W}'\|*\|\mathbb{P}'\|} + \frac{\overline{\sigma}^2_{\tilde{b}_{\tau_c}}}{\|\mathbb{P}'\|}\right)^2}{\frac{1}{\|\mathcal{W}'\|*\|\mathbb{P}'\|+1} * \left(\frac{\overline{\sigma}^2_{\mathcal{W}'_{\tau_c}}}{\|\mathcal{W}'\|*\|\mathbb{P}'\|}\right)^2 + \frac{1}{\|\mathbb{P}'\|+1} * \left(\frac{\overline{\sigma}^2_{\tilde{b}_{\tau_c}}}{\|\mathbb{P}'\|}\right)^2} - 2 \tag{26}$$

$$(\overline{\mu}_{diff_\tau} - t_{[\frac{1-k\alpha}{2};\nu_\tau]} * \sigma_{diff_\tau}, \overline{\mu}_{diff_\tau} + t_{[\frac{1-k\alpha}{2};\nu_\tau]} * \sigma_{diff_\tau}) = (c_{\tau_{t1}}, c_{\tau_{t2}}) \tag{27}$$

The equations above are used in the Sampled Good Multiobjective Building Block Definition (Definition 17). Next, the definition is given and following this definition declaration are a few examples.

Definition 17 (Sampled Good Multiobjective Building Block ($\boldsymbol{GMBB_s^{o,\alpha}}$)): *A good multiobjective BB meets the requirements of Definition 10 and the following: Let a potentially* sampled *good BB be $\tilde{b} \triangleq \{(a_1, l_1), \ldots, (a_o, l_{\mathbf{o}})\}$ where $\tilde{b}$ is labeled the BB under inspection. Let Lambda $\triangleq \{1, 2, \cdots, \ell\}$ where ℓ is the length of a complete* *solution and $\ell > 0$. Let $\zeta \triangleq \{1, 2, \cdots, k\}$ where k is the number of optimization objectives. Let $\mathcal{M}$ be a matrix having $\|\mathcal{A}\|^{\mathbf{o}}$ rows and* **o** *columns. Each row of $\mathcal{M}$ is an element of $\mathcal{A}^{\mathbf{o}}$. Let $\mathbb{BB}^{(o)}$ be the set of all order-***o** *BBs, λ' is an ordered set and defined as the following:*

$$\mathbb{BB}^{(o)} \triangleq \bigcup_{\lambda' \subseteq \Lambda, \|\lambda'\| = \mathbf{o}} \left\{ \bigcup_{j \in \{1, \cdots, \|\lambda'\|\}} \left\{ \bigcup_{i \in \{1, \cdots, \|\mathcal{A}\|^{\mathbf{o}}\}} \{(M_{i,j}, \lambda'_j)\} \right\} \right\}$$

Let $\mathcal{W} = \mathbb{BB}^{(o)} \setminus \tilde{b}$. Let $\mathcal{W}' \subseteq \mathcal{W}$. Let $\mathbb{P} \triangleq \mathcal{A}^n$ where $\mathbb{P}$ is all combinations of possible complete solutions. Let $\mathbb{P}' \subseteq \mathbb{P}$. Let $f_{i,\tau}$ and ***eval**$_\tau$ *be defined as the following function:*

$$\mathbf{F}(\mathbf{X}) = f_\tau \vec{x} : \mathbb{R}; \{f_\tau \vec{x} : f_\tau \vec{x} = \mathbf{eval}_\tau(\vec{x})\}_{\tau \in \zeta, \vec{x} \in \mathbb{P}} \texttt{ where } \mathbf{eval}_\tau : \mathbb{P} \rightarrow \mathbb{R}$$

where k $f_{\vec{x}}$s are assigned to each complete solution, $\vec{x}$, in $\mathbb{P}$. Let the replacement operator, R, be defined as follows:

$$R : \mathbb{BB}^{(o)} \times \mathbb{P} \rightarrow \mathbb{P}; R(\gamma, \rho) \triangleq \{\gamma \mapsto \rho\}$$

where γ meets the requirement of Definition 10, ρ is a complete solution, and the $\mapsto$ operator indicates that the left side element parts replace the right side complete solution where the left side element is specified; otherwise, the right side complete solution remains unchanged.

$$
\tilde{b} = \begin{cases} \texttt{possible good} & : \forall_{\tau\in\zeta}\left(\overline{\mu}_{\tilde{b}_\tau} > \overline{\mu}_{\mathcal{W}'_\tau}\right) \\ \texttt{possible bad} & : \forall_{\tau\in\zeta}\left(\overline{\mu}_{\tilde{b}_\tau} < \overline{\mu}_{\mathcal{W}'_\tau}\right) \\ \texttt{equivalent} & : otherwise \end{cases} \tag{28}
$$

$$
\texttt{possible good} = \begin{cases} \texttt{good} & : \forall_{\tau\in\zeta}\left(c_{\tilde{b}_{\tau_1}} > c_{\mathcal{W}'_{\tau_2}}\right) \\ \texttt{equivalent} & : \exists_{\tau\in\zeta}\left(c_{\tilde{b}_{\tau_1}} > \overline{\mu}_{\mathcal{W}'_\tau} > c_{\tilde{b}_{\tau_2}}\right) or \exists_{\tau\in\zeta}\left(c_{\mathcal{W}'_{\tau_1}} > \overline{\mu}_{\tilde{b}_\tau} > c_{\mathcal{W}'_{\tau_2}}\right) \\ \texttt{ugly(good)} & : otherwise \end{cases}
$$

$$
\texttt{possible bad} = \begin{cases} \texttt{bad} & : \forall_{\tau\in\zeta}\left(c_{\tilde{b}_{\tau_2}} < c_{\mathcal{W}'_{\tau_1}}\right) \\ \texttt{equivalent} & : \exists_{\tau\in\zeta}\left(c_{\tilde{b}_{\tau_1}} < \overline{\mu}_{\mathcal{W}'_\tau} < c_{\tilde{b}_{\tau_2}}\right) or \exists_{\tau\in\zeta}\left(c_{\mathcal{W}'_{\tau_1}} < \overline{\mu}_{\tilde{b}_\tau} < c_{\mathcal{W}'_{\tau_2}}\right) \\ \texttt{ugly(bad)} & : otherwise \end{cases}
$$

$$
\texttt{ugly(x)} = \begin{cases} \texttt{equivalent} & : \exists_{\tau\in\zeta}\left(c_{\tau_{t1}} < zero < c_{\tau_{t2}}\right) \\ \texttt{x} & : otherwise \end{cases}
$$

□

The *sampled* good MBB definition (Definition 17) ends with the inequality presented in Equation 28. The confidence intervals calculated from sampling the data represent most of the values within the final test cases. The confidence intervals do add to the equivalence area because the area is overlapped with the confidence intervals. Figure 5 illustrates the change in the good, equivalent, and bad BB areas.

The entire inequality in Equation 28 is elegant in that it relates back to the single objective case when the number of samples approaches the total number available. Furthermore, this final definition in its current state can be called the *sampled* good single and multiobjective BB definition because it works for all values of $k > 0$. In the following section, these definitions are used to extend BB and BB builder classification.

2.6 Adjusting the Fitness

A researcher using non-adjusted fitness values might run into problems when directly embedding these definitions within a BBB's BB selection operator. The reason is that the identification of good BBs is based on a fitness centroid which relies on the fact that the optimal fitness is located in a region where the majority of good fitness can be found. This can be the cause of problems with any algorithm that relies mostly on definitions that are using an averaging function to draw the line between good and bad solutions. To alleviate this, a ranking of solutions based on the fitness values is needed. Equation 29 defines, RF, a way to adjust the fitness landscape in a way to make the definitions work toward identifying BBs that are within the optimal solution as good BBs.

$$\begin{aligned} RF_{\mathcal{T}}\left(\vec{x}\ having\ worst\ f_{\mathcal{T}}(\vec{x})\right) &= 1 \\ RF_{\mathcal{T}}\left(\vec{x}\ having\ j^{th} best\ f_{\mathcal{T}}(\vec{x})\right) &= \mathcal{N} * RF_{\mathcal{T}}\left(\vec{x}\ having\ (j+1)^{th} best\ f_{\mathcal{T}}(\vec{x})\right) \end{aligned} \tag{29}$$

RF indicates Ranked Fitness.

Equal $f_{\mathcal{T}}(\vec{x})$s are grouped as the same j^{th} best.

$\mathcal{N}$ number of distinct $\vec{x}$ s being evaluated.

$f_{\mathcal{T}}(\vec{x})$ is defined in Definition 13 on page 27.

To explain this phenomenon further, a three-bit example can be found in Appendix L on page 407. The examples provide support and understanding of how to use these new GSBB and GMBB definitions in identifying if a BB is single or multiobjective good, equivalent, or bad. In addition, it is necessary to know the definition of a good BB even if these definitions are not used within a particular BBB. Certainly, there are many computations required to classify a BB when using these definitions. In some cases, it could be useful to know if a BBB algorithm is operating with good, equivalent or bad BBs. This ends the evaluation of the previous good BB definitions. Next, a supposition about the meaning of good and optimal BB sets is discussed.

2.7 Minimal of Maximal ordered BBs

BBBs are search algorithms that specifically look for good BBs. These good BBs are then put together to construct good complete solutions. BBBs, like these are working on the premise that a set of good BBs can be spliced together to *build* good solutions (see Figure 2.3 for an example of BBs being spliced together). In addition, the assumption is made based on the findings earlier in this chapter that it is better for a BBB to find larger good BBs than smaller BBs if the builder is using a non-modified fitness. From this, the conjecture can be made that a BBB is seeking the optimal BB set that can be used to build all optimal solutions for a problem.

Definition 18 (Optimal Building Block Set): *Let there be a set of BBs, $\mathcal{O}$, of various orders (sizes), $\mathbf{o_i}$, that can be put together using an operator, $\mathcal{J}$, in such a way to build the optimal set of solutions for a particular problem, P. The optimal BB set is the minimal set of maximally ordered BBs that can be put together using the operator $\mathcal{J}$ to build every optimal solution for P. Equation 31 minimally describes this definition.*

$$\mathcal{O}^r = max\left\{\sum_{i\in\mathcal{O}} exp(\|i\|)\right\}_{\mathcal{J}(\mathcal{O})=\mathcal{P}^*} \qquad (30)$$

$$\text{where } \mathcal{J} \text{ is a juxtapostional operator that builds complete solutions from a set of BBs,} \mathcal{O}.$$

$$\mathcal{O}^* = min\{\|j\|\}_{j\in\mathcal{O}^r} \qquad (31)$$

$$\text{where } \mathcal{O}^r \text{ is defined in Equation 30}$$

□

Definition 18 on page 51 abstractly defines what is meant when a BB set is said to be an *Optimal* BB Set. A Juxtapositional operator, $\mathcal{J}$, is defined to take a set of BBs, $\mathcal{O}$, and spliced them together to make a complete solution. Figure 2.3 illustrates an example of a left-to-right static juxtapositional operator. It is used to show the mapping of a set of BBs to complete solutions in the genotype space. Clearly, there exists a set of BBs, $\mathcal{O}^r$, that can be put together using the operator $\mathcal{J}$ in some way to make up the set of all optimal solutions. BB sets that can be put together to build all optimal solutions are called jBB sets. The optimal solution set may be unique or it may not. Next, in Equation 30, a maximally ordered jBB set, $\mathcal{O}^*$, is described to be better than lower ordered jBB sets based on the earlier discussion demonstrating how smaller BBs might be wrongly identified as good, when indeed they cannot be used to make the optimal solution. Thus, a set of larger good BBs is better than having smaller BBs. Therefore, as the order of jBBs go up for this Optimal jBB set, the better the set's contrived merit. Finally, the set of jBB sets that contain the minimal number of BBs within the entire set also becomes attractive because then the builder does not need to find as many BBs as a larger set; thus, Equation 31 defines the optimal set of BBs as the minimal set of Maximal ordered BBs. This Optimal BB Set may not be unique; however, each set that satisfies the definition is a smaller set of sub-elements. Hence, it can be expected that if a BBB algorithm seeks to find one of these Optimal BB sets, it results in a smaller number of required sets of sub-elements for the creation of all optimal solutions.

Using the definitions declaring good, equivalent and bad BBs in conjunction with what it means to be an Optimal BB set, some basic conclusions about the operations of BBBs can be drawn. The following is a list of conjectures about BBBs:

1. The meaning of a *good* BB is different within different BBBs. This depends on the statistical foundation of the algorithm. It is not the attempt of this research

to devise the different statistical models for BBBs; however, it is recognized that different models do exist.

2. BBBs are seeking at least one complete Optimal BB Set, jBB, contained in $\mathcal{O}^*$.
3. Once a builder has a jBB set, it must splice together the items in the set until it constructs the optimal solutions.
4. The success of a BB building algorithm is determined in two aspects:
    (a) Effectiveness in finding good BBs.
    (b) Effectiveness in putting good BBs together to make the Pareto optimal set of solutions.

Among these conjectures, there are two BB performance metrics that are revealed. The first, flagged above by $4.a$, is the effectiveness of a BBB in finding good BBs. The second, flagged above by $4.b$, is how well the BBB constructs optimum solutions from the good BB set found by the builder during the search procedure. These BBB gain metrics can be used to evaluate the operating mode or rate of good BB gain of a BBB algorithm.

In the next section, a definition for what it means to have positive BB gain is given. In addition, the gain can be used as an indicator to terminate a search or allow for an adaptive operator adjuster if the gain drops below a positive level.

2.8 BB Builder Operating Modes

BBB algorithms search mainly for good BBs. These BBs are then combined to find good complete solutions; however, the focus of a BBB's selection operator is specifically to select good BBs over bad ones. Furthermore, in accordance with Definition 18, if the algorithm finds only one of these optimal BB sets, the probability of putting together these BBs to make every optimal solution is increased.

BB gain is defined as a generational metric or a fluid[18] metric. Assume that a BBB discovers Ω_t total unique[19] BBs and ω_t unique good BBs in generation t.

The available BB gain is the possible number of unique BBs that can be found after an infinite number of generations over the number of unique BBs discovered after t generations. This stresses the importance of how many unique BBs can be expected given that a number of BBs are generated. This is significant sine the expected number of unique BBs limits the available gain a BBB can achieve in its search. The available BB gain given in Equation 32 as $Gain_{a(t)}$[20] and Equation 33 specifies the $lim_{j \rightarrow \infty}$ for ω_j $(E(\infty))$[21], thus, also limiting the available gain of the BBB.

$$Gain_{a(t)} = \frac{\sum_{j=1}^{\infty} \omega_j}{\sum_{h=1}^{t} \omega_h} \tag{32}$$

$$E(q) = \sum_{i=1}^{q} \forall_{u_i \in U_i} 1 * p(u_i) \tag{33}$$

This implies that there must exist an optimal set of BBs that is smaller than $E(\infty)$; if not, the BBB has a zero probability to succeed. Furthermore, it implies that BBBs should concentrate on the total number of BBs created generation after generation instead of the number of population members to create during a generation. Although this is a conjecture, it may have an impact on BBB design in the future.

[18] Fluid in this case is defined as: subject to change; variable; "a fluid situation fraught with uncertainty".

[19] A unique BB is a BB that has not been constructed by the builder in the current generation or any previous generation.

[20] More clearly stated, when the available BB gain is greater than one ($Gain_{a(t)} > 1$) the BBB still has a probability of finding more unique BBs and thus the probability to find more solutions.

[21] $E(q)$ is the expected number of unique BB a BBB can find after checking q BBs. As $q \rightarrow \infty$, E(q) must be larger than the minimum Optimal BB set size ($E(\infty) < min(\mathcal{O}^r)$).

Equation 33 is a generic formula for calculating the number of unique BBs for a BB population of size q. U_i is the number of ways to select a unique BB after picking i BBs. Figure 9 is provided to give an example of how to calculate the expected number of unique BBs if a population of size 4 is generated in a problem having q BBs. The expected value, $E(q)$, is equivalent to Ω_t.

Conjecture of static and temporal operating modes for a BBB at generation t are the following:

$$\textbf{\textit{Static Mode(BBB}}_t\textbf{\textit{)}} = \begin{cases} \text{RED}, & \log(Gain_{a(t)}) = 0 \\ \text{GREEN}, & \log(Gain_{a(t)}) \geq 0 \end{cases} \tag{34}$$

$$\textbf{\textit{Temporal Mode(BBB}}_t\textbf{\textit{)}} = \begin{cases} \text{IDLE}, & Gain_{a(t)} - Gain_{a(t-1)} \quad = 0 \\ \text{ACTIVE}, & Gain_{a(t)} - Gain_{a(t-1)} \quad Otherwise \end{cases} \tag{35}$$

Clearly, the desired mode of operation for any BBB that has not completely solved a problem at generation t is $\mathcal{ACTIVE}$ $\mathcal{GREEN}$, meaning that there are plenty of unique good BBs yet to be found and the BBB has progressed in finding at least one good unique BB after generation $t-1$. Any BBB operating in mode $\mathcal{RED}$ and having not solved the problem is undoubtedly not going to solve the problem. In addition, mode $\mathcal{RED}$ indicates that the BBB should stop executing or dynamically adjust BBB parameters (*i.e.*, including population size, rates of crossover or mutation). No proof is provided for this conjecture for the mode of operations; however, Equations 32, 34, and 35 can be used as a BBB dash board *idiot light* to indicate a change in parameters is required for when BBB progression is stagnant.

2.9 Convergence and/or Stopping Criteria

BBBs much like any other EA have the underlying theory that they do converge over time with a probability of 1 as time goes to infinity [5]. In other words, a single

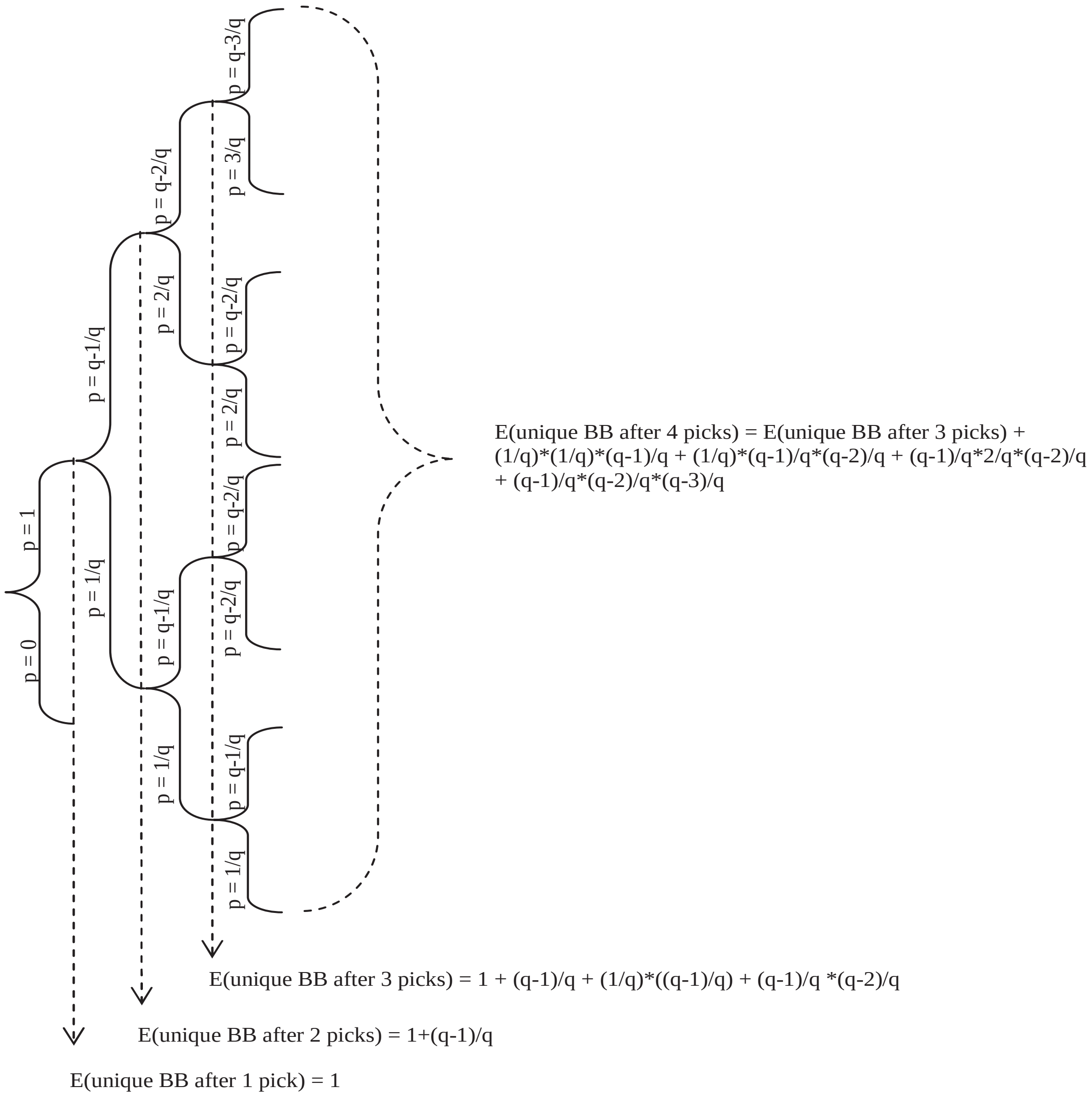

p = q-3/q
p = 3/q
p = q-2/q
p = 2/q
p = q-2/q
p = 2/q
p = q-1/q
p = q-2/q
p = 1
p = 1/q
p = q-2/q
p = 0
p = q-1/q
p = 1/q
p = q-1/q
p = 1/q
E(unique BB after 4 picks) = E(unique BB after 3 picks) + (1/q)*(1/q)*(q-1)/q + (1/q)*(q-1)/q*(q-2)/q + (q-1)/q*2/q*(q-2)/q + (q-1)/q*(q-2)/q*(q-3)/q
E(unique BB after 3 picks) = 1 + (q-1)/q + (1/q)*((q-1)/q) + (q-1)/q *(q-2)/q
E(unique BB after 2 picks) = 1+(q-1)/q
E(unique BB after 1 pick) = 1

Figure 9: This figure illustrates the calculation for the expected number of unique BBs generated by a builder having q number of BBs available $|\mathbb{BB}^*|$.

objective evolutionary algorithm converges to the optimal solution if the algorithm is allowed to run for an eternity. The same has been shown to be true with MOEAs; however, an MOEA converges not to a single optimal solution but to P_{true} solutions or multiple optimal solutions. [31]

Today's state-of-the-art BBBs are based on the same fundamental principles as the past EAs and MOEAs; thus, BBBs must also converge with the probability of 1 when time goes to infinity. [31, 34]

2.10 Building Block Builders

There are two types of BBBs: implicit and explicit. Implicit BBBs evolve the entire chromosome in search of good BBs. Implicit BBBs use common evolutionary operators like crossover and mutation. These implicit BBB are known to have trouble finding good solutions because their operators tend to destroy good BBs when mating solutions [243]. It is for this reason that explicit BBB are better than implicit BBBs. Explicit BBBs are algorithms that explicitly search for good BBs. Upon gathering a group of "so-called" good BBs, these are then juxtapositioned together to construct good complete solutions.

In this chapter, a few BBBs are introduced and a brief description of each is given. The purpose of this discussion is to bring the point out that each BBB has its own good BB definition imbedded within its coding. A few single objective algorithms are discussed: messy Genetic Algorithm (mGA), fast messy Genetic Algorithm (fmGA), and Intelligent Evolutionary Algorithm (IEA). Then, a few MOEAs are discussed including MOMGA, MOMGA-II, MOMGA-IIa, and Intelligent Multi-objective EA (IMOEA). Similarities existing between the MOEAs are highlighted. Finally, a brief discussion of two Probabilistic Model-Building GA (PMBGAs) are discussed: Estimation of Distribution Algorithms (EDA) and Bayesian Optimization Algorithm (BOA).

2.10.1 messy Genetic Algorithm (mGA). The original mGA was designed specifically to solve deceptive problems; problems where the simple Genetic Algorithm (sGA) and steady state Genetic Algorithm (ssGA) get caught in suboptimal trenches in the fitness landscape without hope of climbing out [18, 154]. This brings to light the fact that crossover and mutation may not be flexible enough to find optimal solutions to deceptive problems, causing the need for the mGA.

The mGA consists of the initialization, primordial and juxtaposition phases. The use of partially enumerative initialization builds a population of all combinations of BBs of size $\mathbf{o}$, $\mathbb{BB}^{\mathbf{o}}$. The idea for this is that the optimal solution is guaranteed to exist within the population of these BBs. The problem is using the selection and juxtapositional operator to find it. BBs survive by being selected using tournament selection through the primordial phase. Then, BBs are put together in the juxtapositional phase. This algorithm is designed to rely more heavily on the juxtapositional phase because it begins with a set of BBs that make up at least one set of BBs in $\mathcal{O}^r$. Unfortunately, keeping all the BBs of one size has its limitations because population sizes for higher ordered BBs require an exponential increase in memory. For this reason, the fast messy Genetic Algorithm (fmGA) was developed.

2.10.2 fast messy Genetic Algorithm (fmGA). The fmGA is designed to reduce the complexity of the mGA by replacing the initialization phase and primordial phase with a probabilistic complete initialization (PCI) and primordial phase consisting of selection and BB filtering (BBF). During the PCI and BBF phases, BBs survive by passing a tournament selection operator. Furthermore, PCI and BBF are alternate means to providing the juxtaposition phase with highly fit BBs [93]. When comparing the complexity of the fmGA to that of the mGA, the fmGA is lower. Again, this is a BBB algorithm having much success in solving difficult problems (*e.g.*, see [50–53, 81, 83, 156, 164, 165].)

2.10.3 Intelligent Evolutionary Algorithm (IEA). The IEA is based on an older smaller algorithm called the Intelligent Gene Collector (IGC). The uniqueness of the IGC is in its orthogonal experimental design (OED). IGC has three phases: division phase, conquest phase and combination phase. It is much like the fmGA where the IGC uses a divide-and-conquer approach by first chopping up fully specified solutions into smaller segments. One big difference between the fmGA and the IGC is that the IGC uses contiguous genes and the fmGA can use genes in any sequence. A second major difference is in the division phase of IEA. Where the fmGA randomly deletes bits from chromosomes until all chromosomes are of the same size, designers of the IGC make the assumption that it has a population of good population members from which to extrapolate good BBs. Extrapolation or good BB determination is accomplished in the conquest phase where good gene segments are identified using the IGC. This phase has a huge memory/space advantage over the fmGA because the IGC can only look for contiguous bit segments, where the fmGA looks for all combinations of non-contiguous bit segments. Finally, in the combination phase, the IGC differs from the fmGA's juxtapositional phase in that the IGC works to combine a set of Latin Square combinations of the divided chromosomes together until two children are created; one having the better gene segment from the derived corresponding parents where selection is based on the factor with the smallest main effect difference (MED). Child two is selected from the second best set of factor settings when comparing the combinations of a Latin Square set of factor settings (avoiding a full factorial test set) [102].

2.10.4 Multiobjective messy GA (MOMGA). The original Multiobjective messy GA (MOMGA) is a multiobjective implementation of the mGA. It works in three phases: partially enumerative initialization (PEI), primordial, and the juxtapositional phase [94, 216]. In the primordial phase, partial strings[22] are initialized.

[22] BBs are sometimes referred to as partial string, partial chromosome or a partial solution.

However, it differs from the original mGA in that the MOMGA uses k competitive templates[23] each corresponding to an individual objective function. MOMGA's competitive templates are used to *fill in* missing allele values just before evaluation of a BB. The MOMGA begins with random templates and then evolves templates by replacing old templates with the best complete solution associated with each objective as future templates. Updates to competitive templates are done at the end of each era.

2.10.5 MOMGA-II. The MOMGA-II is the multiobjective version of the fmGA and consists of the following three phases: initialization, BB filtering, and juxtapositional. The MOMGA-II differs from the MOMGA in the initialization and primordial phase. The primordial phase within the MOMGA-II is referred to as the BB filtering (BBF) phase. The initialization phase of the MOMGA-II uses probabilistically complete initialization (PCI) instead of the partially enumerative initialization (PEI) implementation used in the mGA and randomly creates the initial population. The MOMGA-II had the capability to run in both mGA and fmGA mode; however, the PEI phase is changed to PCI for both algorithms. Additionally, MOMGA-II also deviates from the fmGA and MOMGA in calculating a lower required population size. The decrease is largely due to the use of a sample standard deviation population sizing equation.

The MOMGA-II maintains k competitive templates to evaluate BBs. The k competitive templates are replaced by the best complete solutions found with respect to each of the k objectives after the completion of the inner loop. Furthermore, only one of the k competitive templates is randomly chosen for BB evaluation. In addition, the MOMGA-II is implemented as a parallel MOEA having ability to run in generic

[23] A competitive template (CT) is a fully specified chromosome or complete solution. A CT evolves as the EA or MOEA proceeds by the copying of the best found solutions onto the last generation CT.

master-slave, island, and diffusion model mode (see Appendix J on page 398 for a description of each). However, speedup for each model has not been tested [243].

2.10.6 MOMGA-IIa. The MOMGA-IIa is also a multiobjective version of the fmGA and consists of the following three phases: initialization, BB filtering, and juxtapositional. The MOMGA-IIa differs from the MOMGA in the initialization and primordial phase, which is referred to as the BB filtering phase. The initialization phase of the MOMGA-IIa uses probabilistically complete initialization (PCI) instead of the partially enumerative initialization (PEI) implementation used in the mGA and randomly creates the initial population. The MOMGA-IIa can run in single and multiobjective fmGA mode. Additionally, the MOMGA-IIa adopts the calculation sizing equation from the MOMGA-II, decreasing the population size due to the use of a sample standard deviation in the population sizing equation.

MOMGA-IIa BBs are evaluated against competitive templates much in the same way that the MOMGA and MOMGA-II does. However, the MOMGA-IIa maintains three types of competitive templates: $\hat{r}$ regular CTs, $\hat{i}$ inverse CTs, and $\hat{o}$ orthogonal CTs. The added CTs keep a genotype and phenotype diversity as the algorithm proceeds to search. The number of regular CTs are calculated by taking the user specified number of CTs, $\hat{u}$, and multiplying them by the number of objectives, k. In total, the number of CTs maintained by the MOMGA-IIa is $\left(\hat{u} * k + \hat{i} + \hat{o}\right) = \left(\hat{r} + \hat{i} + \hat{o}\right)$. Furthermore, each population member has $\left(\hat{r} + \hat{i} + \hat{o}\right) * k$ fitness values associated with it – corresponding to the k objective functions to optimize, $\hat{r}$ regular competitive templates, $\hat{i}$ inverse templates (this number is equal to $\hat{r}$), and $\hat{o}$ orthogonal templates. Competitive template replacement is done by consulting a competitive template managing system (CTMS). [47]

2.10.7 Intelligent Multiobjective EA (IMOEA). IMOEA incorporates elitism with a capacity of $N_{\mathcal{E}_{max}}$ to maintain diversity and improve the performance of the IMOEA. The IMOEA is similar to the IEA in that a Latin Square of combinations is created to produce children from one IGC operation; however, within the IMOEA, multiple objective values representing each member makes for different selection and mating operators. Moreover, the IMOEA's selection and mating operators result in the increase from two to eight offspring - including two parents, two children, and four by-products. Finally, the IMOEA is a generational *explicit* BB MOEA, with a unique design where it does not specifically hunt for linkages within the chromosome in order to find optimal solutions. IMOEA's pseudocode can be found in Algorithm 38 on page 370.

2.10.8 Estimation of Distribution Algorithms (EDAs). The Estimation of Distribution Algorithms (EDAs) alternatively sample and update a distribution on the search space. Generated individuals are evaluated and the best individuals more heavily impact the updating of the developing distribution. These algorithms are shown to not perform well when a problem has optimal solutions that lie on the edges of the search space [202]. However, regularization heuristics, which calibrate the eigenvalues of this distribution, were shown to successfully overcome such limitations. EDAs can either be variable-based or bit-based. This study is focused on bit-based EDAs because they fit the BBB algorithm model. A few examples of these types of algorithms are the following: Bit-Based Simulated Crossover (BSC) [210], Population-Based Incremental Learning (PBIL) [8], compact Genetic Algorithm, and the Univariate Marginal Distribution Algorithm (UMDA) [153].

2.10.9 Bayesian Optimization Algorithm (BOA). The Bayesian Optimization Algorithm (BOA) evolves a population of complete solutions. BOA begins by generating a random set of complete solutions as the initial population. Updates to the population after initialization come from a selection and variation operator. Selection makes multiple copies of better solutions eliminating the worst ones. Variation consists of constructing a model based on a Bayesian network for the promising solutions that survive the selection process. Sampling the Bayesian network results in a new set of candidate solutions. New solutions are then incorporated into the existing population by eliminating old candidate solutions. The algorithm iterates until the termination criteria are met [182].

This section briefly discussed several of today's state-of-the-art explicit BBBs. Appendix F on page 366 discusses a detailed comparison between these explicit BBBs and Appendix E on page 324 has a detailed discussion of implicit BBBs. Next, the BB hypothesis is discussed. A short discussion about the BBBs and the BB hypothesis is given suggesting that a compromise of efficiency may indeed increase effectiveness in terms of robustness.

2.11 Building Block Hypothesis

The *building block* Hypothesis states that GAs works well when short, low-order, highly-fit schemas recombine to form even more highly fit higher-order schemas. The evident problem with this hypothesis is in deciding what is a good low-ordered highly-fit schema (or BB). Empirically, it has been shown that weighted fitness functions (deception problem) might cause the misclassification of a small BB. In fact, a good BB could be identified as bad depending on the classifier used. According to the original good BB definitions by Zydallis [243], this problem might throw an explicit BB search algorithm into peril. Misclassification of BBs would definitely have a dampening effect on a BBB's ability to solve problems. This leads us to the fact that a BBB classification system *must* be good and overcome the bad classifications discussed earlier. In fact, the dependency of the success of the BBB hinges on the ability for it to identify these good BBs as such.

This dependency brings to light the fact that there must be different BB classification systems built within each BBB that allows for the best BBs to be found over time. It is suggested that there are at least two types of BB classifier systems; L_∞Norm and statistical BB classifiers. The L_∞Norm BB classifier is an adaptive or progressive BB classifier that, over time, can identify good BBs over bad ones by using a maximization function on randomly selected BBs. The statistical BB classifier uses adaptive statistical measures to identify good BBs over time by using either independent or dependent statistical models.

The relevance of this is that BBBs can be classified in this study as either L_∞Norm or statistical BBBs. Within the category of statistical BBBs, two different models are identified; independent or dependent. Table 4 lists single objective BBBs previously discussed and identifies their BB classifying system.

Table 4: Summary of single objective BBBs and associated BB classifier system.

GA	BB classification system
mGA	Adaptive L_∞-Norm
fmGA	Adaptive L_∞-Norm
IEA	Statistical Independent
BOA	Statistical Dependent

With these classifications in mind, it is apparent that BB classification, be it good or bad, is dependent upon which type of BB classifier a BBB is using. Furthermore, BB identification approaches within each BBB may be tied to the classifier a designer wants to use within the BBB, adding another layer of complexity to BBB design for achieving the best possible search results.

It is important to note that the classifier (whatever it may be) within the BBB does *not* change the actual BBs within the solution. The BBs within the solution are problem dependent and never change. It is up to the algorithm to classify or identify these BBs correctly in hopes of being able to put them together in a way to build an optimal solution. Moreover, implicit GA schemata tests have been performed and have shown that the fitness landscape plays an important role for the implicit BBB algorithm [78] to find good BBs. With this said, the problem domain has specific BBs within the optimal solutions that algorithm (implicit and explicit) BB seekers seek. The BBs in the solution for a particular problem do not change; however, the fitness landscape of the problem can make it more difficult to find these BBs. It is up to the builder to choose the optimal BB classifier. Furthermore, there may not be one optimal BB classifier. In fact, just like in racing, depending on the weather on race day, a driver might want to change the type of tire he/she uses on his/her car to optimize performance for a wet or dry surface, it is advantageous to marry the landscape and the classifier in the BBB for an effective search to take place.

2.12 Multiobjective building block hypothesis

The single objective *building block* Hypothesis states that the GA works well when short, low-order, highly-fit schemas recombine to form even more highly fit higher-order schemas. It is a conjecture of the multiobjective building block hypothesis that larger BBs are required for finding extremes and discontinuities along the Pareto front vectors while the short, low-ordered, highly-fit schema can be used to find Pareto front vectors within the interior of a Pareto front set. In most cases, large BB sizes are found to make up vectors found on the extremes of the Pareto front. Although an algorithm may generate low-ordered highly-fit schemas, these must be put together to form a larger-ordered BB to find these *breaks* along the Pareto front.

The P_{known} set contains solutions that evaluate to vectors that are moving toward or onto the true Pareto front. Within each solution making up each vector, BBs exist. Thus, if multiple vectors are moving outward toward the Pareto front, BB sizes can be associated with placement of points on the front. Therefore, the complexity of BB sizes when talking about how well an MOEA might work is high. Thus, it can be said that sizes for BBs within the MOEA field of study have a different meaning than within the EA field of study.

2.13 Good Building Block Classification Approach Definitions

To reiterate, the original good BB definitions 11 on page 23 and 15 on page 39 and the revised good BB definitions 13 on page 27, 14 on page 33, 16 on page 40, & 17 on page 47 are adequate BB definitions for the time that they were developed. However, sufficiency of such definitions should be gauged against how these definitions are used. Each algorithm judges a BB as good or bad according to a different set of rules; therefore, each algorithm inherently has an implicit definition to distinguish between good and bad BBs. For this reason, the following three different definitions are produced to reflect what is thought to be current state-of-the-art BBB good BB definitions.

2.13.1 L_∞-Norm good building block classifier. In this section, the L_∞-Norm Good BB definition is given. Definition 19 reflects how the mGA, fmGA, MOMGA, MOMGA-II and MOMGA-IIa classify good and bad BBs using a memoryless BB classification system (*i.e.*, the classifier does not keep track of previously good BBs.). However, it is a conjecture that these algorithms actually have an *adaptive* good L_∞-Norm BB mechanism where each generation of the algorithm becomes more accurate as to selecting which BBs are the best. The reason for this conjecture is for the competitive templates guiding the search to become closer to optimal solutions (closer to the PF_{true} set). This can mean that first generational good BBs might be tenth generational bad BBs. It all depends on the competitive templates and the BBs being compared to one another.

Definition 19 (Good L_∞-Norm Building Block (MBB)): *Given a set of BBs, $\mathbb{BB}^*$, and a set of complete solutions, $\mathbb{C}$, BBs having the maximum number of objectives recording better fitness values when overlaid onto each of the complete solutions in the set $\mathbb{C}$ are said to be* good *BBs. For example, given two BBs, $\tilde{b}_1$ and $\tilde{b}_2$, one complete solution, $\mathcal{C}_1 \in \mathbb{C}$, and a problem having k objective functions, if $f_\tau(\mathcal{R}(\tilde{b}_1, \mathcal{C}_1))$ is better than $f_\tau(\mathcal{R}(\tilde{b}_2, \mathcal{C}_1))$ in more objectives, then $\tilde{b}_1$ is called a good BB. If $f_\tau(\mathcal{R}(\tilde{b}_2, \mathcal{C}_1))$ is better than $f_\tau(\mathcal{R}(\tilde{b}_1, \mathcal{C}_1))$ in more objectives, then $\tilde{b}_2$ is called a good BB. Otherwise, each BB has a probability of* 0.5 *of being identified as a* good *BB.* □

2.13.2 Statistical Independent Good Building Block Classifier. A statistically independent good BB refers to a BB that meets the requirements of Definition 10 and the mean ranked fitness value of the BB evaluates to a good fitness value over a number of different allelic combinations placed in the unspecified loci. This definition example is the same as what can be found in Definition 11 except for the ranking fitness which can be found in Equation 29 on page 50.

Definition 20 (Good Statistical Independent Building Block (MBB)): *A good single objective BB meets the requirements of Definition 10 and the mean ranked fitness value of the BB evaluates to a good fitness value over a number of different allelic combinations placed in the unspecified loci. The ranked fitness is defined by Equation 29.* □

2.13.3 Good Bayesian Building Block Classifier. Bayesian good BBs refer to BBs that are identified as to have conditional probability dependencies between allele values and a variety of loci. This type of BB is more complex than previous BBs discussed because it heeds the notion that bit linkages are conditionally dependent. The Bayesian BBB has a good BB classifier that incorporates the notion of conditional dependencies between both loci and allele values.

Definition 21 (Good Bayesian Building Block (GBaBB)): *Let the Directed Acyclic Graph G be defined as the set of E, edges, and V, vertices. Let each loci and possible allele value in a solution space represent the vertices in graph G. Let $E_{i,j}$ represents the edge from v_i to v_j indicating a linkage relationship between bit v_i and v_j where $v_h \triangleq \{a, l\}$ such that $a \in \mathcal{A}$ and $l \in \{1 \cdots \ell\}$. A Bayesian Building Block (BaBB) is a set of edges representing an acyclic path of connected bits contained in G. A good BaBB is found when the score of the Bayesian Building*

Block[24] *metric before adding the BaBB in G is worse than the score after the BaBB is included.* □

2.14 Utility of GBB definitions

The identification of GBB definitions or GBB classifiers within state-of-the-art BBBs is important because it identifies fundamental BB classifying techniques that make for successful explicit BBBs. Utility-wise, underlying BB classification systems making up a good explicit BBB can be aggregated or designed together in a manner to synergistically find GBBs that ultimately find optimal solution to a variety of difficult problems.

2.15 No Free Lunch Theorem (NFL) Considerations

The *No Free Lunch (NFL) Theorem* states that for any two search algorithms, over the set of all mathematical problems, each search algorithm will do on average as well as the other. This theorem supports the fact that different algorithms are better at solving different problems [230]. In an attempt to add robustness, it may be better to include every BB classification system within the same algorithm in the BBB design. When doing so, this implies that an algorithm having multiple algorithms might be better at searching. Unfortunately, nothing is free, and this type of implementation suffers from efficiency problems in the end, hence NFL. Still, a thoughtful design with respect to only adding classifiers known to perform well on particular problems may be useful to decision makers needing different problems solved by the same optimization heuristic. Thus, a BBB having different types of BB classifiers should be more robust at solving a variety of problems at a decrease in efficiency.

[24] One such Bayesian Building Block metric is the Bayesian-Dirichlet (BD) metric. The BD assumes that the conditional probabilities follow Bayesian-Dirichlet distribution and makes a number of additional assumptions. Yet another Bayesian Building Block metric is the minimum description length (MDL) metric. BOA uses a two-part MDL metric called the Bayesian information criteria (BIC) which is also used in the extended compact genetic algorithm (ECGA).

2.16 Quality of Building Block Builders

Ultimately, finding the optimal solutions for a problem defines if an algorithm is successful. In answering the questions posed in the beginning of this chapter, the NFL theorem must be consulted as it is stated above.

"How can one determine if one BBB is better than another?" A BBB **A** may be better than BBB **B** at finding solutions for problem **Z**, but there is always a problem **W** where BBB **B** is better than **A**. Thus, BBBs with successful approaches at finding good BBs are similar and in most cases cannot be called *better* than one another. This is shown to be true by many researchers [71]. Finally, BB metrics could be a good way to measure the BB finding performance of a BBB; however, they only show the performance of a builder for a particular problem – not the entire set of problems in the world (simply put, an example does not prove anything).

This is not to say that the total performance cannot be measured for a BBB on a given problem over a finite set of quality metrics - because this can be done. However, there is no BBB that is more robust than another BBB over the totality of all MOPs (*e.g.*, NFL).

2.17 Summary

In this chapter, there is a revision and extension of good BB definitions to include algorithm specific BBB classifiers or approaches. Also recognized is that researchers should marry algorithm BB classifiers with the problem domain landscapes to make a successful search. Also defined is an optimal BB set, which suggests that a BBB seeks to have at least one optimal BB set. Once this optimal set of BBs is found, there is still the problem of putting them together. Techniques like the cut-and-splice operator and sampling are currently used in today's state-of-the-art algorithms. However, one can imagine that the combinatorics for putting all good BBs together may be time prohibitive. Still, if the researcher chooses an explicit

BBB, then the algorithm must put these blocks together either by a cut-and-splice operator or a sampling procedure.

In addition to the BB merge mechanism, a choice of BB classifier must also be done. When referring to the NFL theorem, it can be conclude that without the insertion of domain information, each algorithm independently must perform equally good/bad on a random problem. Finally, this suggests that a multiple classifier design should work better than one that has a single classifier. Next, in Chapter III, MOMGA-II design changes and justification for these enhancements are discussed.

III. MOMGA-IIa Design

This chapter is dedicated to describing the new MOEA resulting from this research. Design changes and justification for these enhancements. These changes add a new active archive that enhance the preservation of good solutions and maintain a distributed collection of good solutions for evolving, a new competitive template management system, and a BB solutions tracing system with a new visual analysis tool. Built upon the scalar fast messy GA (fmGA), the MOMGA-II uses the *competitive template* (CT) for finding good BBs. Described is the design of a *critical* adjustment during CT generation, replacement, and evolution required to unleash the full potential of the MOMGA-series algorithm. The design includes a partitioning of both the phenotype and genotype, allowing for the algorithm to find good *multiobjective* BBs (MBBs) explicitly.

The chapter begins with an introduction describing a brief background on the mechanics of the MOMGA-series algorithm. It shortly moves to describing in detail each MOMGA algorithm in chronological order. Within the detailed description of the latest algorithm, MOMGA-IIa innovative enhancements are described enough to allow for reproduction of each novel mechanism. The chapter ends with a summary of the design discussion.

3.1 Introduction

The latest MOMGA algorithm is MOMGA-IIa. MOMGA-II originates from a single objective mGA extended to handle MOPs [74, 222] (see Section 3.2.1 on page 72). Many different MOEAs were produced when the first MOMGA was developed; however, the MOMGA is the only MOEA explicitly using good BBs to solve problems. Unfortunately, the MOMGA has a population size limitation; as the BB size increases, so does the population size during the Partially Enumerative Initialization (PEI) phase. This renders the MOMGA less useful on large problems.

To overcome this problem, the MOMGA-II, which like the fmGA explicitly uses BBs to find solutions; however, it requires smaller population sizes and has a lower run time complexity when compared to the mGA. MOMGA-II includes many different repair, selection, and crowding mechanisms. Unfortunately, the MOMGA-II is found to be limited when solving large problems due to speciation [46]. This called for the development of a Competitive Template Management System (CTMS) to ensure diversity measures are instantiated within the MOMGA-IIa. The design achieves a better BB search in both the genotype and phenotype domains. Next, the history of the MOMGA-series MOEA is given.

3.2 Structure of multiobjective explicit BB MOEAs

The following is a detailed description of the development and history of the MOMGA-IIa.

3.2.1 Multiobjective messy GA (MOMGA). The multiobjective messy Genetic Algorithm (MOMGA) is an explicit BB-based MOEA created by David Van Veldhuizen and Gary Lamont in 1999 [218]. It is built on the single objective mGA. The authors of the MOMGA took the BB searching mechanism of the mGA and extended it to solve MOPs. At the time of its creation, the MOMGA different from other MOEAs mainly in that it was the only MOEA that explicitly manipulated BBs when searching for solutions.

Like the mGA, MOMGA, consists of three phases: the partial enumeration initialization, primordial, and juxtapositional phases [94]. In the PEI phase, the MOMGA completes a total enumeration of all user-specified BB sizes for its first population. The population is generated through a process of analyzing the specified BB size, generating all of the BBs necessary, and evaluating all the objective functions for each and every population member. Thus, the starting population size is based upon the BB size, chromosome string length, and cardinality of the alphabet. BBs are created *under-specified*, meaning that they are partial strings. BBs can have contiguous and noncontiguous bits, and must have at least one missing allele. Equation 36 defines the size, $\mathcal{N}$, of the population should be according to the variables $\mathcal{A}$, $\mathbf{o}$, and ℓ. Let $\mathcal{N}$ be the population size, $\|\mathcal{A}\|$ the cardinality of the alphabet that is used in the GA (for a binary alphabet, $\|\mathcal{A}\| = 2$), ℓ the length in bits of the chromosome and $\mathbf{o}$ is the BB size.

$$\mathcal{N} = \|\mathcal{A}\|^{\mathbf{o}} \binom{\ell}{\mathbf{o}} \tag{36}$$

The MOMGA makes use of k competitive templates to evaluate BBs. Templates are fully specified chromosomes that transcend modification during a single iteration. These templates allow for a good comparison basis when evaluating population members (partial strings) - in fact, future references to the competitive template may be as a *basis function.* The competitive template is the critical element of the mGA, fmGA, MOMGA, MOMGA-II and MOMGA-IIa; without it, preservation of past good found solutions is not accomplished unless an elitism technique is added.

The initial population of BBs are overlaid onto a randomly selected CT, chosen from the k CTs, just before evaluating the BB's k objective values. The objective values are then stored with the BB. The overlaying process is described in detail within chapter II.

Following the initialization phase, the primordial phase executes. Generations for the primordial phase are dynamically determined based on the proportion[1] of good BBs in the population. The primordial phase process is presented in Algorithm 1 as steps 7 through 13. Binary tournament thresholding[2] selection is used

[1] The primordial phase runs until a certain portion of the BB population is deemed good. Generation of primordial phase includes tournament selection with thresholding to select good BBs for survival.

[2] Selection using thresholding requires that two chromosomes have a certain number of genes in common to be compared. Having this requirement prevents the competition of BBs belonging to

Algorithm 1 MOMGA algorithm

```
 1: procedure MOMGA($\mathcal{N}, \mathcal{G}, f_k(\vec{x})$) ▶ $\mathcal{N}$ members evolved $\mathcal{G}$ gens to solve $f_k(\vec{x})$
 2:     Randomly initialize $k$ Templates
 3:     for j = 1 to o do
 4:         \\\\\\ PEI Phase \\\\\\
 5:         Perform Partially Enumerative Initialization
 6:          Evaluate each BB w.r.t.$k$ Templates
 7:         \\\\\\ Primordial Phase \\\\\\
 8:         for i = 1 to Max Primordial Generations do
 9:             Perform Tournament Thresholding Selection
10:             if Appropriate number of generations accomplished then
11:                 Reduce Population Size
12:             end if
13:         end for
14:         \\\\\\ Juxtapositional Phase \\\\\\
15:         for i = 1 to Max Juxtapositional Generations do
16:             Cut-and-Slice
17:             Evaluate Each Population member's fitness w.r.t.$k$ templates
18:             Perform Tournament Thresholding Selection and Fitness Sharing
19:             $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
20:         end for
21:         Update $k$ templates                ▶ Using best known value in each objective
22:     end for
23: end procedure
```

during this phase along with a population reduction method to enrich the population with good BBs (*i.e.*, bad building blocks are left behind when the BB population is reduced).

Finally, the juxtapositional phase (see steps 14-20 of Algorithm 1) is used to perform recombination. This phase consists of using a cut-and-splice operator alternated with tournament thresholding selection (this is the selection operator or the BB classifier identified in Definition 19 on page 66 as the L_∞Norm BB classifier system). Thresholding in conjunction with selection is used to select similar good BBs over bad ones as the population members grow. Mutation is not used in the

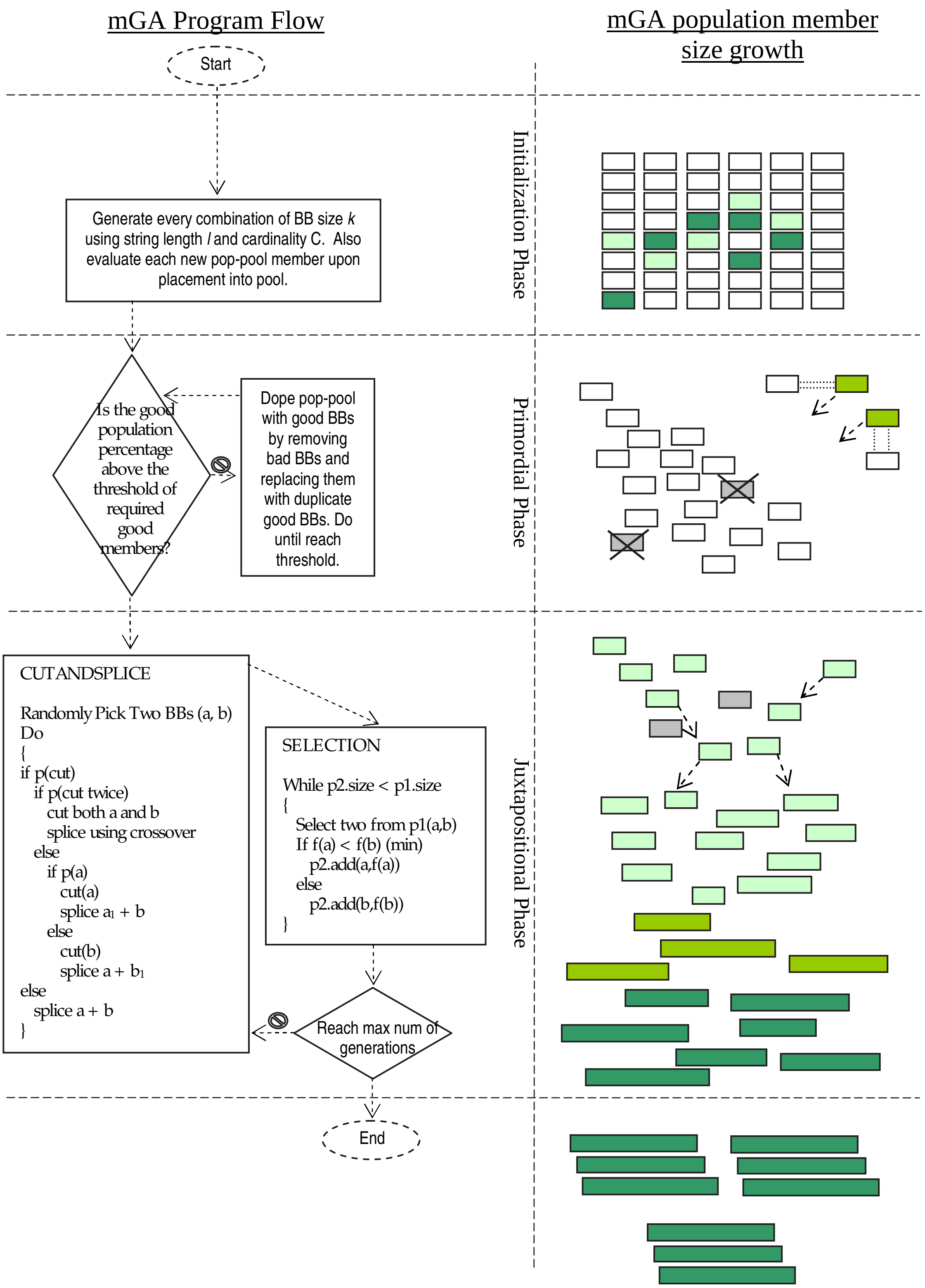
different subfunctions (or hyperplanes). When comparing BBs on two separate hyperplanes, it is called cross-competition.

MOMGA. The cut-and-splice operator is considered to be a crossover operator for variable length strings. The operator is used to build up strings from their under-specified BB size to a fully specified chromosome length. A cut-and-splice operator randomly chooses a point to cut a string, then subsequently splices the string with another cut string. This process is repeated allowing for all population members to grow in length.

Population member string growth is dependent on the splice probability settings of the algorithm. During the intermediate evaluations, competitive templates are used to evaluate the under-specified strings. The templates are fully specified population members that are not modified until the completion of the juxtapositional phase.

Following the juxtapositional phase, the competitive templates are updated with the best individuals found with respect to each objective function and the process repeats itself utilizing the next user-specified BB size. During this phase, chromosomes can become larger than what is required for a single solution. These larger individuals are *over-specified* individuals and they contain an allelic value for a particular gene multiple times. Resolution of over-specified individuals is currently fixed by using the first allele value encountered for a specific locus (gene), when scanning the bit string. Because the MOMGA follows the mGA so closely, Figure 10 is a good diagram for illustrating the program flow and illustrating the population member size growth throughout each phase of the algorithm. Finally, an archive combines the new population with the population found within the archive and all solutions evaluating to dominated vectors are removed.

During all of the selection routines, Pareto dominance-based selection is used. In order to expand the mGA to multiple objective functions, k competitive templates for k objective functions are used [216]. Templates are randomly selected when evaluating a population member so as to avoid convergence to one objective function. In addition, niching is implemented based on Horn's Niched Pareto Ge-


mGA Program Flow
mGA population member size growth
Start
Generate every combination of BB size *k* using string length *l* and cardinality C. Also evaluate each new pop-pool member upon placement into pool.
Initialization Phase
Is the good population percentage above the threshold of required good members?
Dope pop-pool with good BBs by removing bad BBs and replacing them with duplicate good BBs. Do until reach threshold.
Primordial Phase
Juxtapositional Phase
Reach max num of generations
End

```
CUTANDSPLICE

Randomly Pick Two BBs (a, b)
Do
{
if p(cut)
   if p(cut twice)
      cut both a and b
      splice using crossover
   else
      if p(a)
         cut(a)
         splice a1 + b
      else
         cut(b)
         splice a + b1
else
   splice a + b
}
```

```
SELECTION

While p2.size < p1.size
{
   Select two from p1(a,b)
   If f(a) < f(b) (min)
      p2.add(a,f(a))
   else
      p2.add(b,f(b))
}
```

Figure 10: Illustrated in this figure is the program flow of the messy genetic algorithm and the population of BBs and their growth while the algorithm progresses.

netic Algorithm (NPGA) [105, 106]. The MOMGA achieves favorable performance when compared to a limited set of other MOEAs [216]. MOMGA pseudocode can be found in Algorithm 1.

Discussed next is the design and implementation history of the multiobjective version of the fmGA - MOMGA-II. There are population size advantages for having an fmGA-based MOEA, MOMGA-II, instead of the mGA-based MOEA, MOMGA. However, the mGA is guaranteed to have *all* good BBs of a particular size, while the MOMGA-II probabilistically creates a population size of order-$\mathbf{o}$ BBs to overcome noise created by objective functions. With a lower population size, the fmGA is shown, in the literature, to be better than the mGA on selected SOPs. Further, it is shown that the MOMGA-II is better than the MOMGA on select MOPs [244].

3.2.2 Multiobjective Messy GA (MOMGA-II). The MOMGA-II mirrors the fmGA (see Section E.4.6.2 on page 336 for a short discussion of the fmGA) and consists of the following phases: probabilistic complete initialization (PCI), BB filtering (BBF), and juxtapositional. Thus, the MOMGA-II differs from the MOMGA replacing the primordial phase, with the BB filtering phase in the MOMGA-II, and the PEI with PCI phase. The PCI randomly creates the initial population using the size of the population suggested to obtain *good* results for the fmGA is calculated using Equation 37 where ℓ is the total number of genes, ℓ' is the string length, $\mathbf{o}$ is the BB size, $\hat{c}(\alpha)$ is used to determine the acceptable amount of error, β^2 is the signal-to-noise ratio and k is the number of subfunctions [93]. This population sizing equation fixes the mGA's population size problem of having an exponential rising population size as the searched building block size increases (see Equation 36 on page 73). However, a more recent study has shown that this equation is based not on a sampling of the population, but on the entire population. In 2002, Zydallis developed a better population sizing equation for the fmGA and the MOMGA-II using a sample variance calculation. Equations 38, 39, 40, 41, 42 and 43 describe the calculation required for the new population sizing equation in the MOMGA-II. The new equation takes into consideration noise produced by k different objective functions. See [243] for a detailed discussion about the new population sizing equation.

$$\mathcal{N} = \frac{\binom{\ell}{\ell'}}{\binom{\ell-\mathbf{o}}{\ell'-\mathbf{o}}} * 2\hat{c}(\alpha) * \beta^2 * (k-1) * 2^{\mathbf{o}} \tag{37}$$

$$\begin{aligned}
\mu_{diff} &= \mu_{\tilde{b}_1} - \mu_{\tilde{b}_2} \quad \ddagger^{a} && (38)\\
\sigma^2_{\mu_{diff}} &= (\sigma^2_{\tilde{b}_1} + \sigma^2_{\tilde{b}_2}) && (39)\\
z^2(\alpha) &= \frac{\mu^2_{diff}}{\frac{\sigma^2_{\mu_{diff}}}{\mathcal{N}'}},\ where\ \mathcal{N}' = \frac{\mathcal{N}}{k} && (40)\\
\epsilon^2_{\mathcal{N}_i} &= \frac{\sigma^2_{\mathcal{N}_i}}{\sigma^2_M} \quad \dagger^{b} && (41)\\
\epsilon^2_T &= \sum_i^k \epsilon^2_{\mathcal{N}_i} && (42)\\
\mathcal{N} &= 2 * z^2 * \frac{\sigma^2_M}{\sigma^2_{\mu_{diff}}{}^2} * k * (1 - \epsilon^2_T) * |\mathcal{A}|^o && (43)
\end{aligned}$$

[a] ‡ $u_{\tilde{b_j}}$ is the mean fitness for each objective found for BB $\tilde{b_j}$ evaluated after being overlaid into an unspecified number (not specified by Zydallis; suggest 29) different population members.

[b] † σ^2_M is used instead of σ^2_{diff} to indicate that the calculation uses a sample variance - not an entire population variance

The PCI phase creates population members of string size ℓ. The generation of the initial population is different than that of the MOMGA where all possible combinations of order-**o** BBs are created. Algorithm 2 presents the pseudocode for the MOMGA-II's PCI phase in steps 3-5. It is important to note that the MOMGA-II population members are generated randomly where allelic values are randomly selected, as well as the positions (loci) to place these values. Again, the number of population members to create is determined by Equation 43. After the initial population is created, members are evaluated with respect to each of the k objective functions. The MOMGA-II also follows the mGA ($k = 1$), fmGA ($k = 1$), and

MOMGA having k competitive templates for the k objective functions. Furthermore, selection of one of the k CTs to evaluate a BB is done randomly - only one is selected per evaluation. The function of CTs within the MOMGA-II is the same as within the mGA, fmGA and MOMGA. Initially, CTs are generated randomly. Once each BB size is executed, the MOMGA-II updates each of the competitive templates with the best solution found with respect to each objective function.

Algorithm 2 MOMGA-II algorithm

```
 1: procedure MOMGA-II($f_k(\vec{x})$)                          ▶ Solve $f_k(\vec{x})$
 2:     for j = 1 to epoch do
 3:         \\\\\\ PCI Phase \\\\\\
 4:         Perform Probabilistically Complete initialization
 5:         Evaluate each pop member's fitness w.r.t.$k$ templates
 6:         \\\\\\ BBF Phase \\\\\\
 7:         for i = 1 to Max BBF generations do
 8:             if BBF schedule requires cutting at this generation then
 9:                 Perform BBF
10:             else
11:                 Perform Tournament Thresholding Selection
12:             end if
13:         end for
14:         \\\\\\ Juxtapositional Phase \\\\\\
15:         for i = 1 Max Juxtapositional generations do
16:             Cut-and-Splice
17:             Evaluate each population member's fitness w.r.t.$k$ templates
18:             Perform Tournament Thresholding Selection and fitness Sharing
19:             $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
20:         end for
21:         Update $k$ templates          ▶ Using best known value in each objective
22:     end for
23: end procedure
```

The BBF phase (pseudocode shown in Algorithm 2 steps 6-13) reduces the string lengths of the population members generated in the PCI phase by following a user-specified filtering schedule. This BBF schedule specifies the generations to conduct a random deletion of bits and the number of bits to delete from each chromosome. Additionally, the schedule also specifies the number of juxtapositional generations to execute and can be automated by the MOMGA-II. The MOMGA-II uses the filtering schedule where the strings are halved in length with each filtering operation until they are reduced to the specified BB size. As far as *mutation*, the MOMGA-II does not have a mutation operator; however, the BBF operator has been classified as a mutation operator. The reason the BBF operator has been classified as a mutation operator is because it slowly alters the chromosomes within the population by randomly removing bits. To reduce confusion, the BBF operator is called *filtering* - not mutation. The MOMGA-II's BBF operator filters each string with the probability of filtering (p_f) equal to one. The probability of mutation (p_m) for the MOMGA-II is zero.

The filtering process consists of a random deletion of $\hat{d}$ bits from each of the population members. After the first filtering operation, the strings are reduced in length from ℓ bits to $\ell - d_i$ bits, where d_i is the number of bits to remove in filtering step i. Each subsequent filtering operation continues to randomly delete additional bits from all population members. Typically, each filtering operation does not delete the same number of bits. The number of bits to delete during each filtering generation is user specified in the BBF schedule[3]. Filtering is alternated with tournament selection. Tournament selection within the MOMGA-II fosters competition between same hyperplaned BBs, provisioning the next population with the best BBs by selecting the winners from the tournament for advancement into the next population. The reason for using tournament selection is to keep more copies of the best BBs from one generation to the next because it is expected that these BBs generate better solutions. At the conclusion of the BBF phase, the entire population consists of individuals having a specified BB length.

At this point in the algorithm, the population consists of the best BBs found. The end goal of the BBF process is to yield a similar result to the primordial phase

[3] A good BBF schedule has filtering step sizes that are no more than 20% of the string size (ℓ) of the chromosome [155]; however, you must still have BB sizes in the schedule where your BBF phase stops (user specified BB size for each epoch).

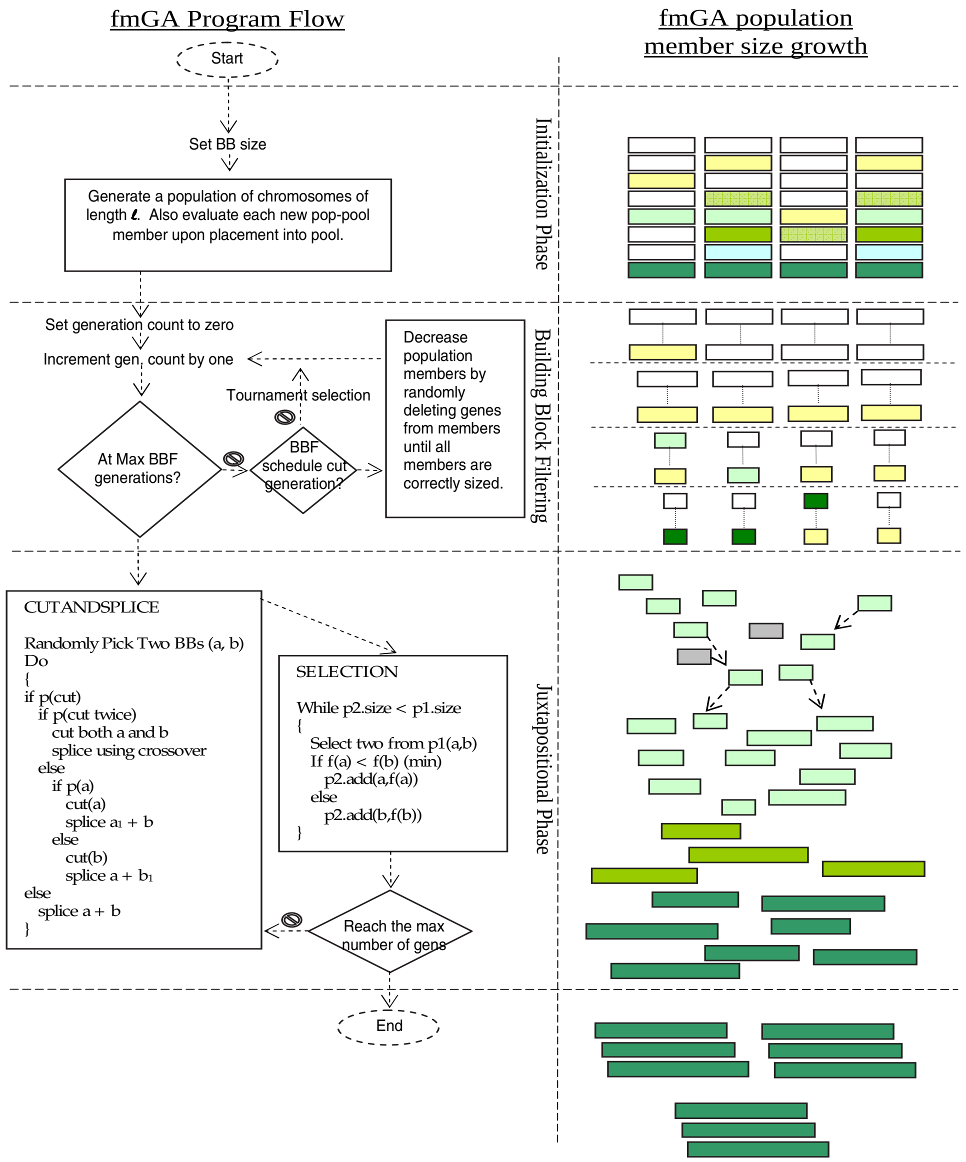
of the MOMGA, a population consisting of a sufficient number of *good* BBs to be recombined in the juxtaposition phase of the algorithm.

The juxtapositional phase proceeds in the same manner as in the MOMGA. Steps 14-20 in Algorithm 2 on page 79 present the pseudocode for the juxtapositional phase. BBs found through the initialization phase and BBF phase are recombined through the use of a cut-and-splice operator alternated with tournament selection with thresholding. Again, the reason for using tournament selection is to be sure that the best is combined with other best BBs within the population. Gradually, population members' length grow toward becoming fully specified – having an allelic value stored in each locus position. The cut-and-splice operator is alternated with tournament selection maintaining a population of the best individuals found.

As in the MOMGA, competitive templates are used to evaluate the fitness values of underspecified population members. In the event of overspecification, where a loci location has multiple allele values, a scan from left-to-right is conducted of the population member and the first value encountered to specify the gene location becomes the allele value for that gene - dropping all right most overspecified allele values. But as the juxtapositional phase advances, population members become less dependent on the competitive templates to fill in missing alleles because of the growth of the individual members.

Following the juxtapositional phase, the BB size is incremented and each τ^{th} competitive template is updated with the best individuals found with respect to the τ^{th} objective function. The MOMGA-II also uses this model of population sizes and program flow, thus it is a good representation of a single BB size for the MOMGA-II.

The MOMGA-II executes all three phases for each BB size and presents solutions and the associated PF vector set generated. Theoretically, the MOMGA-II can be applied to MOPs with any number of decision variables and objective functions - constrained only by the system resources. The MOMGA-II's better performance over the MOMGA is similar to the increased performance of the fmGA over the mGA


fmGA Program Flow
fmGA population member size growth
Start
Set BB size
Generate a population of chromosomes of length ℓ. Also evaluate each new pop-pool member upon placement into pool.
Initialization Phase
Set generation count to zero
Increment gen. count by one
Tournament selection
At Max BBF generations?
BBF schedule cut generation?
Decrease population members by randomly deleting genes from members until all members are correctly sized.
Building Block Filtering
Juxtapositional Phase
Reach the max number of gens
End

```
CUTANDSPLICE

Randomly Pick Two BBs (a, b)
Do
{
if p(cut)
   if p(cut twice)
      cut both a and b
      splice using crossover
   else
      if p(a)
         cut(a)
         splice a₁ + b
      else
         cut(b)
         splice a + b₁
else
   splice a + b
}
```

```
SELECTION

While p2.size < p1.size
{
   Select two from p1(a,b)
   If f(a) < f(b) (min)
      p2.add(a,f(a))
   else
      p2.add(b,f(b))
}
```

Figure 11: Illustrated in this figure is the program flow of the fast messy genetic algorithm and the population of BB and their growth while the algorithm progresses.

and was to be expected. However, improvements to the MOMGA-II are required for it to be able to solve larger problems and those having a higher epistatic level. In fact, the MOMGA-II was shown to not perform well when solving some larger deception problems having a high epistatic level [43]. Discussed next is a detailed outline of the development of the MOMGA-IIa.

3.2.3 Design Modification Rationale. MOEA research has resulted in some solid objectives the multicriteria heuristic designer or architect must consider in designing a new or optimizing a former algorithm. These objectives are referred to as the "PPDPs" of MOEA design. PPDP stands for Preserve, Progress, Diversity and Provide. The PPDPs and description for each are outlined as follows:

Preserve Preservation of solutions that evaluate to non-dominated vectors is required so a strategy can use these former solutions in some manner to find better next generation solutions.

Progress Next generation solutions must evaluation to forward progressing PF vectors - moving toward the PF$_{true}$.

Diversity Maintaining diversity within the phenotype is paramount; however, diversity within the genotype domain is also important. Diversity yields a more extensive non-dominated set of vectors along the Pareto front.

Provide Provide the decision maker with a reasonable number of distributed Pareto front vectors and related Pareto optimal solutions.

These MOEA design fundamentals are found in each MOEA design. Moreover, each essentially compliments the others to meet the end goal for any MOEA, which is to find the complete set of Pareto optimal solutions (P$_{true}$) with respect to the allowable resolution of the algorithm. Built on these design fundamentals, the following design concepts guide MOEA development:

1 Fitness based on dominance ranking (Preserve)
2 **Diversity strategies**: Kernel, Nearest neighbor, Histogram (Diversity)
3 **Uniform/Diverse** Uniformly spaced PF$_{known}$ (Diversity))
4 **Convergence to PF$_{true}$**, PF$_{true}$ (Progress)
5 **Generating non-dominated phenotype points** (Progress)
6 **Archiving** plus **elitism** of chromosome population (Preserve)
7 **Niching or Crowding** on Pareto front (Provide and Diversity))
8 **Visual comparisons** (Provide)
9 Quantitative performance metrics (Provide)
10 **Evolutionary Visualization** (Provide)
11 **Explicit**/Non-Explicit BB manipulation (Expected Progress toward PF$_{true}$)
12 Probabilistic models (Expected Progress toward PF$_{true}$)
13 Non-Pareto vs. **Pareto** approaches (Preserve, Progress, Diversity, and Provide)

The new MOEA structure uses a proven MOEA heuristic. The fmGA is proven to be a good EA and has been shown to perform well for Zydallis as the base line EA from which the MOMGA-II is built. Thus, the fmGA is selected as the base line EA. Although MOMGA-II code was available, due to the amount of structural changes that were planned for the MOMGA-II it was decided that the MOMGA-IIa would not share any code with the MOMGA-II. Furthermore, a re-write of the fmGA to MOMGA-IIa resulted allowing for easier problem integration because the code has a definite separation between algorithm and problem domain code. MOMGA-II design concepts (not code) recognized and needing to be modified to create the MOMGA-IIa are **bolded** above. Modifications are largely due to the limitation of the MOMGA-II to scale well and the lack of space (genotype and phenotype) partitioning within the MOMGA-II [43].

Secondly, the MOMGA-II should be analyzed for usefulness and levied against the MOEA fundamentals listed. In fact, the MOMGA-II is found to meet fundamental MOEA design concepts. However, the question remains, how well are these fundamentals met and can it be done a better way based on algorithmic knowledge? Just patching an algorithm with added mechanisms is often not enough to make the algorithm the best it can be. An algorithm needs to be analyzed for what makes it perform well; then, a suitable modification should be forged, enhancing strengths while either fixing or replacing weak mechanisms. This must include the finding of the BB classifier to identify good BBs identified in Definition 19 on page 66. Analysis can either be accomplished with metrics or reverse engineering techniques. In some cases, a break down of algorithm processes may reveal shortcomings. A list of the MOMGA-II characteristics can be found in Table 5. Next, justification is given for identifying these concepts and associated specifications to be changed within MOMGA-II.

The underlying algorithm is validated to be a good BBB - both in the single and multiobjective field of study [41, 243]. In fact, the underlying EA has the same fundamental GBB classifier, identified in Table 4 on page 64, as a state-of-the-art BBB classifier. Therefore, the BB classifier is not suspect and is not to be changed.

Table 5: MOMGA-II Characteristics

Design Concept	MOMGA-II specifications:
Single objective base algorithm	fmGA
Archive	File Store of previously non-dominated vectors
Elitism	Percentage of good BBs preserved from one generation to next
Repair/Penalty function	MOP specific operators are used for the ALP[a] and MMOKP[b] MOPs.
Niching or Crowding	None
Visual comparisons	Post mortem visualization
Evolutionary process Analysis	None
Explicit/Non-Explicit BB	Explicit
Probabilistic model	None

[a] The Advanced Logistics Problem (ALP) is a real-world problem involving logistics research in resource allocation. See Appendix G section G.8.2 on page 386 for a brief explanation.

[b] The Modified Multiobjective Knapsack Problem (MMOKP) is modeled from the single objective knapsack problem. See Appendix G section G.8.1 on page 385 for a brief explanation.

However, upon analyzing how the MOMGA-II meets major design concepts corresponding to the objectives for an MOEA architect, it becomes clear how to enhance the new MOEA for an improved performance. Based on the design concepts of the MOMGA-II and the MOEA design requirements, the two MOEA design objectives that need focus are the *Preserve* and *Diversity* mechanisms within the MOMGA-II which is why Table 5 bolded the concepts relating to these concepts.

The original fmGA used the competitive template as a means to preserve past best solution, as does the MOMGA-II. It is here where a divergence of architecture lies. In the strictest sense of preserving past-found solutions evaluating to non-dominated vectors to the next generation, the MOMGA-II does not keep the total number of competitive templates required if the algorithm were to perform as its single objective form. This is noted in [216] where Van Veldhuizen suggests the combinatorics for having a template for each PF$_{known}$ vector would be too cost prohibitive. Thus, the number of templates applied equals the number of MOP objectives. Furthermore, the MOMGA-II employs regular elitism. The elitism mechanism manufactures (copies) good results into the population just before these results are cut down to a single BB size. This design goes against the idea of the original design of the algorithm, stifling it from *capturing* a range of overlapping *good BBs* within the same single complete solution and finding *better* BBs from this amassed set of good BBs found within a complete solution. With this conclusion, a design having more than k competitive templates is utilized. The reason for this modification is due to the belief that, within an fmGA-based algorithm, a single complete solution carries more BB information than what can be copied over into the population before filtering. The first enhancement is the adding of more competitive templates to the MO version of the fmGA. This modification comes at the price of space because it takes more computer memory to store each template. The expected benefits outweigh the cost of implementation.

Fundamentally, adding competitive templates is easy - but adding them in an advantageous way can decidedly meet more than one design concept is ideal. Many researchers have illustrated that a diverse elitism approach is a type of niching or crowding [58, 222, 236, 242] and helps keeps a diverse Pareto front resolution. Thus, a group of basis functions having a diversity along the Pareto front in a distributed manner is one way to solve the elitism and crowding/niching problem within the MOMGA-II. In addition, it solves the problem in a way that is fashioned after the original designer's idea for use of competitive templates in the fmGA. Finally, this design meets three MOMGA-II design gaps at once: diversity population strategy, archiving elite, and implicit niching.

A memory-based archive is used to update the basis function mechanism in keeping diverse phenotypic basis functions[4] (CTs). These enhancements adjust the MO fmGA to meet two important MOEA fundamentals: preservation of solutions and keeping diversity along the Pareto front extremes. The avoidance of a mem-

[4] A phenotypic *basis function* refers to the competitive templates used as guides from which the explicit BBB bases its BB search. The term *basis function* is taken to mean that these competitive templates are diverse in the objective space and provide a diversity guide to *base* further good BB searching - it is more related to the genotype locality of the solution when the phenotypic vectors associated with these solution are kept diverse.

ory based archive to preserve good solutions is done in the MOMGA-II; however, eventually elitism is performed post-mortem in an off-line file storing solutions and associated objective values from the previous experiment.

The next modification requires more CPU time than the previous MOMGA-II. Consider for a moment that the algorithm is in search for the absolute *best* BBs. To identify a good BB, an evaluation must be conducted with respect to at least one basis function. MOMGA-II has k basis functions, but it only selects *one* at random to evaluate that particular BB. This results in the possibility of having a good BB, but misses the opportunity to identify it as such because the algorithm did not evaluate each BB against all k competitive templates. The significance of this is high because the algorithm can only expect to create less than half of the number of available unique BBs in the entire BB space. The modification comes in the way that each individual is evaluated against each and every basis function and all values are kept for future reference. Efficiency is important; however, finding the best BBs is the only way the algorithm can find at least one optimal BB set, $\mathcal{O}^*$ (see Definition 18 on page 51); therefore, effectiveness is a priority over efficiency when making changes.

Notice that the specifications for the MOMGA-IIa, presented in Table 6, show the explicit absence of the elitism and repair mechanisms. The repair mechanisms are MOP specific and the elitism is replaced with a competitive template management system. Designs that can incorporate many tools and enhancements into a single mechanism can be more advantageous to algorithm performance than patching an algorithm with historically good mechanisms. The next section describes in detail the MOMGA-IIa design, focusing on the archival store and competitive template management system which is the unique modification that allows for an enhanced performance over the MOMGA-II.

3.3 Multiobjective fast messy GA (MOMGA-IIa)

The MOMGA-IIa is based on the fmGA and consists of four phases: preparation, initialization, BB filtering, and juxtapositional. It differs from the MOMGA's initialization and primordial phases. Where the MOMGA uses the PEI for initialization the MOMGA-IIa uses PCI and where the MOMGA uses the primordial phase the MOMGA-IIa has the BB filtering phase. In addition to these changes, the MOMGA-IIa adds a Preparation Phase that establishes the environment before the MOMGA-IIa begins its search for good BBs and solutions. The MOMGA-IIa differs from both the MOMGA and MOMGA-II by including three types of CTs: regular, inverse, and orthogonal. In addition, MOMGA-IIa's selection operator instantiates the L_∞Norm Good BB definition (see Definition 19 on page 66). Finally, the MOMGA-IIa has a Competitive Template Managing System (CTMS) to evolve (manage) the various types of competitive templates and an evolutionary solution tracing mechanism (ESTM).

The CTMS is made up of a Pareto front structure (PFS) and target vector (TV) slot assignment mechanism that allows for the evolving regular competitive templates to partition the phenotype space; thus, lowering the BB size requirement for the MOMGA-IIa. The inverse and orthogonal competitive templates are mainly used to partition the genotype space. The evolutionary solution tracing mechanism[5] keeps track of the evolutionary process for each particular BB that, once overlaid into a CT, evaluates to a non-dominated vector. The recorded trace allows for a post mortem partial *epistatic* level measurement for a particular MOP.

3.3.1 Preparation Phase. Before the probabilistic complete initialization phase, the MOMGA-IIa prepares the environment for operations. First, it partitions

[5] This evolutionary tracing function tracks by tagging the trail of each solution. Each time a solution is cut down to a particular BB size, selected for tournament selection, and put together with another BB, it is recorded within the tracking system. This is called the Birth-to-PF (BP) tracing mechanism for a solution.

Table 6: MOMGA-IIa Design Specification

Design Concept	MOMGA-IIa specifications:
Single objective base algorithm	fmGA
Archive	PF object to maintain ϵ solutions evaluating to non-dominated vectors ($\mathrm{PF}^{\epsilon}_{known}$)
Elitism	**Implicit**
Repair/Penalty function	Implicit
Niching or Crowding	**Target Vectors**
Visual comparisons	Post mortem visualization
Evolutionary process Analysis	**Birth to PF visualization**
Explicit/Non-Explicit BB	Explicit
Probabilistic model	None

the phenotype domain by adjusting a grid over a 0-1 normalized objective space Figure 12 illustrates how a two dimensional (two objective) grid would be affixed over the objective space. The intersecting lines of the grid designate a point or target vector that partitions the space uniformly with respect to the number of available regular competitive templates, $\hat{ar}$.

Objective space grid resolution is based on the number of regular competitive templates, $\hat{r}$, available to partition the objective space. The number of regular competitive templates is determined by the number of competitive templates specified by the user, $\hat{u}$, multiplied by the number of objectives in the MOP, k. The number of user specified competitive templates are identified within a parameters file - an input file called “CT_config.dat”. The best optimal solutions found w.r.t. each objective are saved in k regular competitive template slots, like the MOMGA and MOMGA-II; however, the rest of the regular competitive template slots become known as available regular competitive templates, $\hat{ar}$. Target vector slots are generated in excess sometimes to capture a good objective space distribution of optimal solutions. Note that a target vector slot maintains both genotype and phenotype vector values for a solution, but the partitioning of target vector slots is done within the phenotype domain. Optimal solutions that evaluate to objective space values that first, once normalized, evaluate closest to a particular target vector and second are furthest away from the origin are assigned to target vector slots. The number of target vector slots, $t\hat{v}s$, is equal to or greater than the number of available regular competitive templates, $\hat{ar}$. Where the number of available regular competitive templates is calculated by taking the number of regular competitive templates, $\hat{r}$, and subtracting the number of objectives, k.

For example, if the user specifies a total of five competitive templates for a two objective MOP. The number of regular competitive templates is ten, the number of available regular competitive templates is eight because two are dedicated to the two worst/best competitive template slots for each objective, and the number of target vector slots available must be equal to or greater than eight.

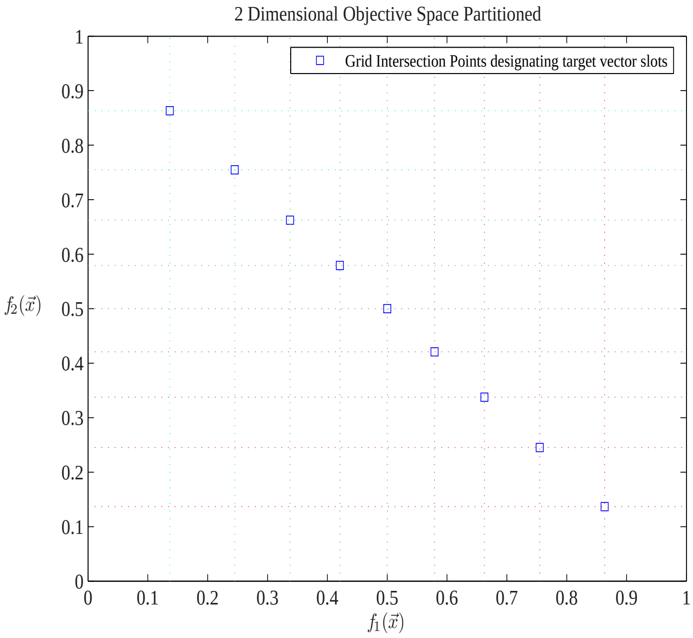


Figure 12: This figure illustrates a two-dimensional grid affixed over a 0-1 normalized objective space. The boxes indicate the cross hairs for target vector direction.

Next, within the preparation phase, the user can indicates the use of a set of inverse competitive templates during a MOMGA-IIa search. An inverse competitive template is simply the mirror image (bit flipping) of each regular competitive template. For example, if regular competitive template one is 1101, the inverse competitive template one is 0010. The number of inverse competitive templates, $\hat{i}$, is the same as regular competitive templates, $\hat{r}$.

Continuing in the preparation phase, an orthogonal bank of $\hat{o}$ chromosomes is created. This bank is to be used later to filter a randomly selected regular competitive template. The number of chromosomes in the bank, $\hat{o}$, is specified by the user within a parameter file, an input file called "in.s". The bank of orthogonal chromosomes is based on the Latin Square; therefore, these chromosomes are not truly orthogonal in the mathematical sense of the word, but this Latin Square competitive template is still referred to as orthogonal throughout this dissertation. Section 3.4.3 describes how this bank of orthogonal chromosomes is created. Table 7 lists each competitive template type used within MOMGA-IIa and how many of each is generated.

Table 7: Competitive Template Specifications for MOMGA-IIa

Competitive Template (CT) type	Number
User Specified CTs	$\hat{u}$
Regular CTs	$\hat{r} = \hat{u} * k$
Available reg CTs	$\hat{ar} = \hat{r} - k$
Target vector slots	$\hat{tvs} \geq \hat{ar}$
Inverse CTs	$\hat{i} = \hat{r}$
Orthogonal CTs	$\hat{o}$

The implementation of target vectors involves fitting the objective space with a grid that has, at the minimum, $\hat{ar}$ intersecting vectors partitioning a normalized 0/1 objective space - where $\hat{ar}$ is the number of *available* regular competitive templates. It is important to distinguish that k templates have been left out of this grid specifically so that the best solution w.r.t. each objective can be copied into the k left over regular competitive template slots. The distinction here between the number of available competitive templates, $\hat{ar}$, and the number of target vector slots, $\hat{tvs}$, is made here. The objective space grid (target vector partitioning) is used to preserve a good phenotype distribution using the PF vector values from evaluating the optimal solutions for next generation BB searching.

Section 3.4.2 describes how the grid is discerned and how many target vector slots ($\hat{tvs}$) are available ($\hat{tvs} \geq \hat{ar}$), where $\hat{ar}$ is calculated as in Table 7. Once the target vector slots are gauged, they are copied to a *mobile* target vector for the application of jitter[6] as the algorithm proceeds. The jitter function randomly moves target vectors within the space a specific amount to guide competitive template selection to different parts of the objective space within a specified normalized objective space regions[7].

3.3.2 Competitive Template Generation. After fitting a grid over a normalized objective space and creating the regular, inverse, orthogonal template space, as well as an orthogonal bank of chromosomes, the algorithm begins by randomly generating all regular competitive templates. Inverse competitive templates are then generated by inverting each regular competitive template. Finally, a set of orthogonal competitive templates is generated by filtering a randomly selected regular competitive template through the bank of orthogonal chromosomes. Table 8 illustrates an example of how a regular competitive template is filtered through a bank of orthogonal chromosomes to generate a set of orthogonal competitive templates. The selected regular competitive template is listed in the left most column. For each row within the bank of orthogonal chromosomes each bit within the selected regular competitive template is $\overline{XOR}$ed[8] with that row's orthogonal chromosome bit in the same loci position to produce that row's orthogonal competitive template (listed in the right most column).

[6] *Jitter* is a term used for the mechanism that applies a random drift from the absolute target vectors calculated in the beginning of the algorithm.

[7] The specified region is bounded in a way to prevent target vector objective space area overlap (*i.e.*, no two target vectors can be mapped into the same region).

[8] $\overline{XOR}$ is the logical complimented exclusive OR function.

Altogether, the total number of competitive templates used by the MOMGA-IIa, $\hat{cts}$, is calculated as $\hat{cts} = \hat{u} * k + \hat{i} + \hat{o}$. Originally, only a set of k regular competitive templates were used during algorithm execution, but it was conjectured that using only k regular competitive templates would result in speciation[9] within this explicit BBB. Design modifications are made to specifically avoid this problem and, to guarantee better exploration of the genotype space, inverse and orthogonal competitive templates are added.

3.3.3 Probabilistic Complete Phase. The MOMGA-IIa, like the MOMGA-II, uses a modified version of PCI where the population is initialized randomly. The size of the population suggested to obtain *good* results for the fmGA is shown in Equation 43 on page 78. The difference between the original MOMGA-IIa's population sizing equation and the new equation is that the variance is modified to use a sample variance, and noise interjected by k objective functions is regarded.

The population-sizing equation calls for limiting the population size; thus, only a subset of combinations of sized $\mathbf{o}$ BBs are generated. In fact, the population is a randomly generated pool of ℓ-$\mathbf{o}$ sized strings. Allelic values are randomly picked as well as the loci. After the generation of the initial population, each population member is evaluated with respect to each of the k objective functions and $\hat{cts}$ com-

[9] Speciation occurs when two similar reproducing chromosomes evolve to become too dissimilar to share genetic information effectively or correctly.

Table 8: Illustrated is a toy example of orthogonal competitive template generation using the orthogonal bank of chromosomes generated in the preparation phase of running MOMGA-IIa. See section 3.4.3 on page 108 for a description of how the orthogonal bank of chromosomes is generated.

Selected regular CT	Orthogonal Bank of Chromosomes	Orthogonal CTs
{0100}	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$

petitive templates. Storage of each fitness value is done in an $\mathcal{N}$ by $k\hat{cts}$ matrix of double precision variables.

A MOMGA-IIa solution evaluation for one individual generates $k * \hat{cts}$ objective values. Having this many objective values per population member impacts the selection mechanism for the MOMGA-IIa. Instead of selecting individuals based on a single fitness value for each objective, the best objective value for each competitive template is used for comparison. This adds $(\hat{u}k + \hat{i} + \hat{o} - k)$ fitness comparisons, not function evaluations, to each tournament selection designed with Definition 19 on page 66. Upon a complete evaluation of a single member, the member is submitted to a global Pareto front structure maintaining solutions evaluating to non-dominated vectors and the target vector slot object. The next section describes this Pareto front structure object.

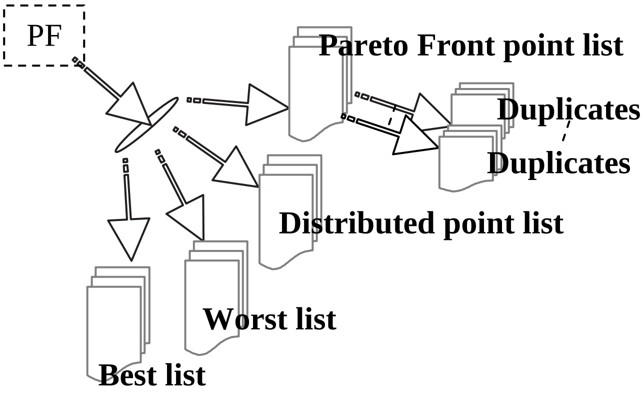


Figure 13: This figure illustrates the Pareto front structure that dynamically holds all the information found by the algorithm. This includes the normalization values, Pareto front vectors, duplicates, and target vector slot assignments. Pointers to linked list objectives are designated with an arrow and linked lists are designated with a multi-documents picture.

3.3.4 Pareto front structure. The Pareto front structure (PFS) object is one of the major differences between the MOMGA-II and the MOMGA-IIa. The PFS is designed to keep track of Pareto front vectors, best/worst solutions for each objective, and target vector assignments. The original MOMGA-II kept the non-

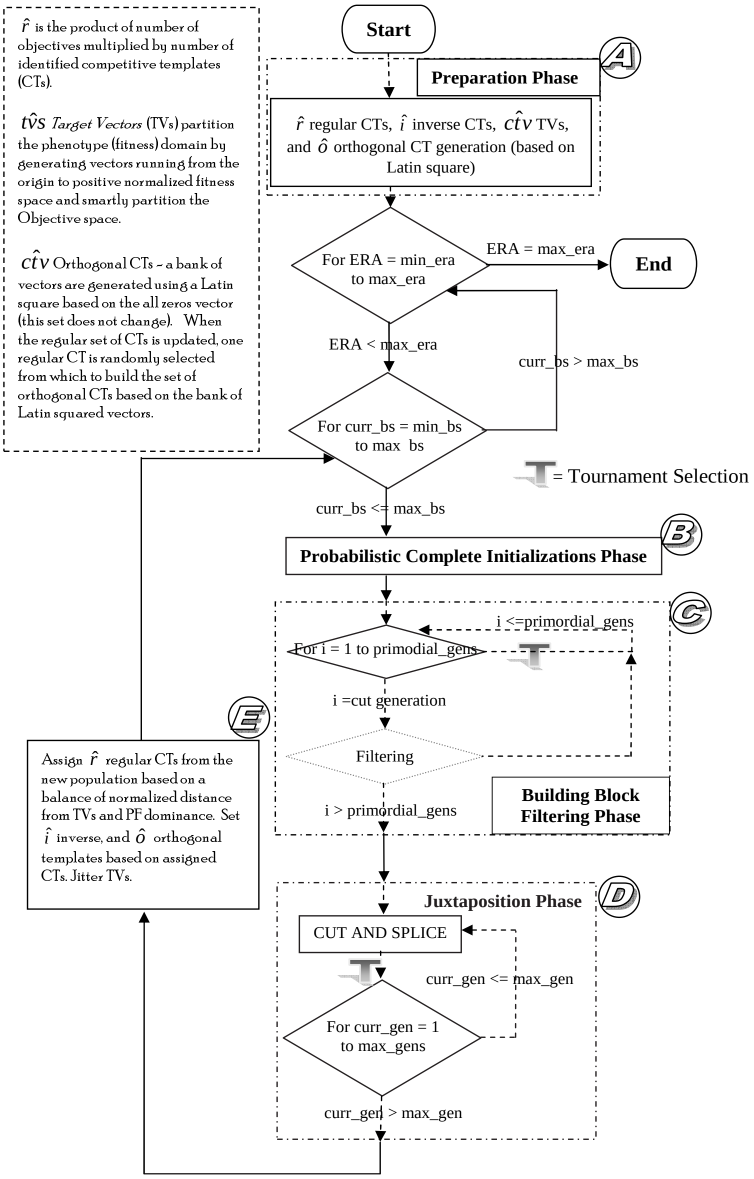

$\hat{r}$ is the product of number of objectives multiplied by number of identified competitive templates (CTs).
$\hat{tvs}$ Target Vectors (TVs) partition the phenotype (fitness) domain by generating vectors running from the origin to positive normalized fitness space and smartly partition the Objective space.
$\hat{ctv}$ Orthogonal CTs – a bank of vectors are generated using a Latin square based on the all zeros vector (this set does not change). When the regular set of CTs is updated, one regular CT is randomly selected from which to build the set of orthogonal CTs based on the bank of Latin squared vectors.
Start
A
Preparation Phase
$\hat{r}$ regular CTs, $\hat{i}$ inverse CTs, $\hat{ctv}$ TVs, and $\hat{o}$ orthogonal CT generation (based on Latin square)
For ERA = min_era to max_era
ERA = max_era
End
ERA < max_era
curr_bs > max_bs
For curr_bs = min_bs to max bs
= Tournament Selection
curr_bs <= max_bs
B
Probabilistic Complete Initializations Phase
C
i <=primordial_gens
For i = 1 to primodial_gens
i =cut generation
E
Filtering
Assign $\hat{r}$ regular CTs from the new population based on a balance of normalized distance from TVs and PF dominance. Set $\hat{i}$ inverse, and $\hat{o}$ orthogonal templates based on assigned CTs. Jitter TVs.
i > primordial_gens
Building Block Filtering Phase
Juxtaposition Phase
D
CUT AND SPLICE
curr_gen <= max_gen
For curr_gen = 1 to max_gens
curr_gen > max_gen

Figure 14: Illustrated in this figure is the program flow of the MOMGA-IIa. Note the placement of each phase and where tournament selection is performed. Additionally, the MOMGA-IIa exploits and partitions in both the phenotype and genotype domains by updating and generating regular, inverse, and orthogonal competitive templates. Section 3.3 on page 88 describes the algorithm and Algorithm 4 on page 100 reflects its pseudocode.

dominated population pool found after each outer look unsorted between consecutive experiments (this was observed when solving the mQAP in [126]). Within each experiment, the MOMGA-II does keep an archive between the inner and outer loop.

In the case where the MOMGA-II used elitism, the Pareto dominance sort (see Algorithm 3 on page 96) is completed before the elite solutions are transferred to the new population. In contrast, the MOMGA-IIa, submits solutions to the PFS after the evaluation of each BB during all phases of algorithm execution. The PFS's job is to keep track of optimal solutions and associated Pareto front vectors. This is MOMGA-IIa's so-called archive of solutions evaluating to non-dominated vectors (optimal known solutions). As new objective bests and worsts are found, these values update the normalizing factor for each objective (see Section 3.4.1 on page 102 for a description of the normalization process). The PFS keeps track of the genotype, phenotype (fitness), evolution trace, and (if unique or copies) duplicates for each solution. In addition, the PFS counts the number of times a particular copy has been found and the size of the BB (partial string) when discovered. Figure 13 illustrates the object storage components of the PFS in computer memory. The MOMGA-II used a different technique for finding solutions evaluating to non-dominated vectors - when able, it loads every single solution into memory before beginning the sort. This technique was not memory efficient and became prohibitively expensive as the number of acquired solutions increases.

Algorithm 3 Kung et al.'s algorithm (1975) [143]

```
procedure KUNG'S DOMINANCE SORT($\mathbb{P}'$, $f_k(\vec{x})$)    ▶ Population $\mathbb{P}'$ evaluated
        with $k$ objective functions: $f_k(\vec{x})$
    Sort population in descending order of importance of $f_1$
    Front($\mathbb{P}'$)
    if ($|\mathbb{P}'| = 1$) then
        Return($\mathbb{P}'$)
    else
        $T = Front(\mathbb{P}'^{(1)}--\mathbb{P}'^{(|\mathbb{P}'|/2)}) \wedge Front(\mathbb{P}'^{(|\mathbb{P}'|/2+1)}--\mathbb{P}'^{(|\mathbb{P}'|)})$
        if (i$^{th}$ solution of B evaluates to a vector that is not dominated by the
            evaluated vector for any solution in T) then
            $M = T \cup \{i\}$                              ▶ Create a merged set M
            Return($M$)
        end if
    end if
end procedure
```

The real gain of the PFS is the added feature to capture good BBs from the PCI, BBF, and juxtapositional phases. During these phases, BBs are created according to a filtering schedule. Thus, BBs of all sizes are generated. The MOMGA-II also created BBs in these phases, but the previous versions did not attempt to capture solutions evaluating to non-dominated vectors within each phase as the MOMGA-IIa does. Storage of solutions are only considered in the juxtapositional phase of the MOMGA-I and MOMGA-II when the MOEA has reduced every population member to a particular BB size (user-specified) before splicing the BBs back together.

3.3.5 Target Vector Slot ($\hat{tvs}$) Assignments. Upon completing solution evaluation, the population member and associated objective values are passed to the Competitive Template Managing System (CTMS) where they are appraised for Pareto front classification and then target vector slot assignment. The MOMGA-IIa has the capability to keep standard solutions evaluating to (weak and strong) non-dominated vectors and non-ϵ-dominated vectors. [34, 148].

Upon processing a solution's evaluated objective values for Pareto dominance or Pareto ϵ dominance, the un-normalized objective values are used. However, when processing a solution's evaluated objective values for target vector slot assignment the normalized objective values are used. Once all processing is complete for the incoming solution, no alteration is done to that solution's associated objective values (un-normalized fitness values are kept). The reason for not storing the normalized fitness values is because the normalizing factors are not stable[10]. This is especially true for the first epoch and BB size. Once the normalizing factors become stable, the same fitness value may map onto the same spot within the 0-1 normalized objective space.

[10] Stable factors are ones that do not change over time. As the search proceeds solutions evaluating to better objective values than found in the last generation change the best/worse normalizing factors for the target vector assignment mechanism. Thus, the normalization factors are unstable in most cases from one generation to the next.

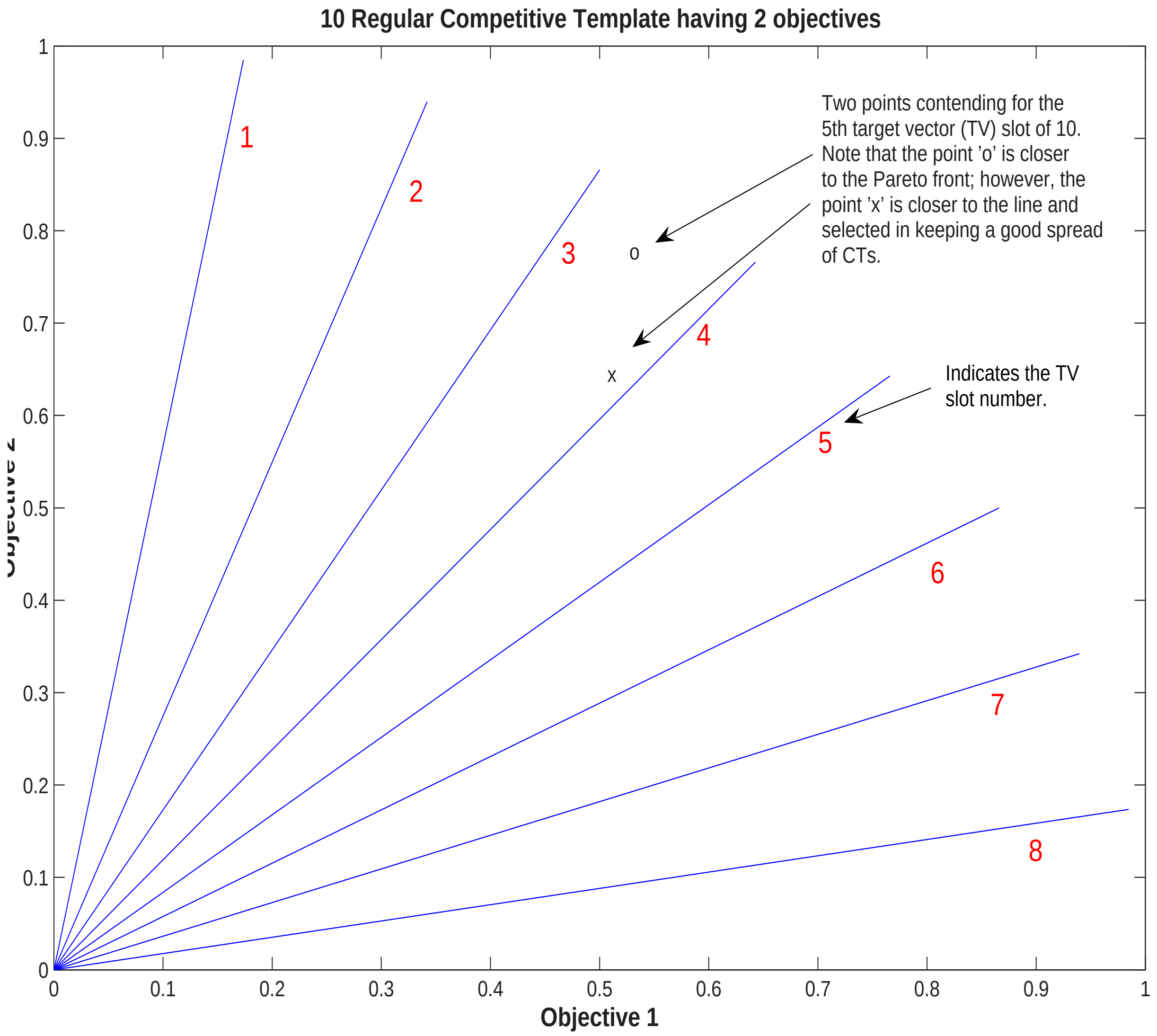


Figure 15: 2D example of the partitioning of the phenotype, objective, space. This figure also illustrates how point selection is accomplished. These target vectors are unrelated to the vectors used for the utility functions R2 and R3 in section 4.3.2.10 on page 138. However, the origin reference point is the worst of each objective value found and can be used when determining the utility with respect to the quality indicators R2 and R3.

When a solution is being processed for target vector assignment, both the assigned vector and the incoming solution's objective values are normalized to the values of the best and worst for each objective found. See Section 3.4.1 on page 102 for a complete description of how the normalizing is accomplished in the MOMGA-IIa. Figure 15 on page 98 illustrates the normalized objective space having eight target vectors. The normalization of the points coming into the region are marked by an x and o; however, if the best and worst fitness values change for the objective values, then the mapping of the same two points may place them into a different region in the 0-1 normalized space. The idea of this technique is to uniformly spread the competitive templates selected for the next generation - thus achieving an indirect niching[11] relative to the basis function for the algorithm.

An example of evaluating an incoming (just evaluated) solution for assignment into a target vector i, $(\vec{tv_i})$[12], is determined by the following short list of rules. If $\vec{tv_i}$ is not filled, the incoming solution and associated PF vector is assigned to slot $\vec{tv_i}$. If $\vec{tv_i}$ is assigned a solution, the incoming solution's objective values are tested to see if it target dominates the currently assigned solution's objective values. If the incoming point is target dominate over the currently assigned vector, the incoming vector is assigned to that target slot. Previously assigned points are only removed when target dominated (see Definition 22 on page 107) by another point, but they are not completely deleted from the PFS if these points are also held as a non-dominated vector or assigned to another target vector slot. Target dominance assignment consists of steps 15 and 29 in Algorithm 4 on page 100. After each individual in the population is evaluated, processed for Pareto dominance, and assessed for target slot assignment, the BB filter phase begins.

3.3.6 Building Block Filtering (BBF) and Juxtapositional Phase.

The BBF and the Juxtapositional phase for the MOMGA-IIa is exactly the same as it

[11] This indirect niching is similar to crowding where the PF vectors guide a local GBB search around a phenotypic partitioned space. This is similar to how the MOEAs OMOEA and PAES maintain diversity by recursively dividing the objective space. See section E.5.9 on page 358 for a discussion of the OMOEA and section E.5.6 on page 355 for a discussion of the PAES.

[12] The symbol, $\vec{tv_i}$, is used to reference a target vector i having coordinates and a place where a solution may be assigned according to the rules given in the paragraph above. Each $\vec{tv_i}$ is a vector having a number of coordinates equivalent to the number of MOP objectives, k, being solved.

Algorithm 4 Detailed MOMGA-IIa

```
procedure MOMGA-IIa algorithm($\mathbb{P}$, $f_k(\vec{x})$)
    \\\\\\Preparation Phase\\\\\\                                            ▶Ⓐ
    Initialize $\hat{r}$ Competitive Templates ($\mathrm{CT}^{\hat{r}}_{1,\cdots,\hat{r}}$)
    †Generate $\hat{tvs}$ Target Vectors ($\vec{tv}_{1,\cdots,\hat{tvs}}$)
    †Generate $\hat{o}$ orthogonal bank of chromosomes ($\mathrm{CT}^{\hat{o}}_{1,\cdots,\hat{o}}$)
    if (Using Inverse Templates) then
        †Generate $\hat{i}$ inverse CTs where $\hat{i} = \hat{r}$ ($\mathrm{CT}^{\hat{i}}_{1,\cdots,\hat{r}}$)
    end if
    for (epoch = 1 to 4) do
        for ($bb_{size} = BB_{min}$ to $BB_{max}$) do
            \\\\\\PCI Phase\\\\\\                                            ▶Ⓑ
            Probabilistic Complete Initialization
            Evaluate Each Member with respect to $\hat{cts}$ CTs
            †Submit each solution and objective values to PFS for evaluation        ▶ Update
                Best/Worst norm factors and $\mathrm{PF}_{known}$ in PFS
            †Assign Target dominant vectors to $\hat{tvs}$ slots
            \\\\\\BBF Phase\\\\\\                                            ▶Ⓒ
            for (h = 1 to Max BBF generations) do
                if (h is a BBF cut generation) then
                    Conduct filtering on all population members
                else
                    Conduct Tournament selection with Thresholding
                end if
            end for
            \\\\\\Juxtapositional Phase\\\\\\                                ▶Ⓓ
            for (j = 1 to Max Juxtapositional Generations) do
                Cut-and-Splice
                Evaluate Each Member with respect to $\hat{cts}$ CTs
                †Submit each solution and objective values to PFS for evaluation    ▶ Update
                    Best/Worst norm factors and $\mathrm{PF}_{known}$ in PFS
                †Assign Target dominant vectors to target vector slots
                Conduct Tournament selection with Thresholding
            end for
        end for
        †Jitter Target Vectors                                               ▶Ⓔ
        †Update $\mathrm{CT}^{\hat{r}}$s                                  ▶ Use slotted TV solutions
        if (Using Inverse Templates) then
            †Update $\hat{i}$ inverse CTs
        end if
        $q \leftarrow rand\{1, \hat{r}\}$        ▶ $q$ is randomly assigned one integer from the set $\{1, 2, \cdots, \hat{r}\}$
        †Filter $\mathrm{CT}^{\hat{r}}_q$ through bank of orthogonal chromosomes ($\mathrm{CT}^{\hat{o}}$)
    end for
end procedure
```

is for the MOMGA-II except that after an evaluation of a population member; that individual is submitted to the Pareto Front Structure for classification and Target Vector Dominance check.

3.3.7 Evolutionary Solution Tracing Mechanism. The Evolutionary Solution Tracing Mechanism (ESTM) is a process that tags special characters onto a solution as it is shaped (evolves) throughout the MOMGA-IIa algorithm. This BB tracing mechanism allows for post-mortem analysis of BB sizes when used within the competitive templates to find solutions evaluating to non-dominated vectors. Solutions found using a particular BB size can gain insight into how an explicit BBB algorithm in conjunction with the multiple CTs solves MOPs. In addition, BB material identified by the ESTM as being significant can be used to gain knowledge about decision variables stability within MOP. This can become useful in finding decision variable resolution issues and significance w.r.t. the objective functions being solved. Details of this mechanism and how it can help researches gain insight into the problem domain can be found in Section 3.4.4.

3.4 MOMGA-IIa Enhancements

The following sections decompose major enhancements found within the MOMGA-IIa. The first section begins with an explanation of how the objective space (phenotype) is normalized using incoming fitness values. This process is important for the assignment of points to target vector slots. Target vector slot building and point assignment is described in detail to include pseudocode for each algorithm used. Included are the algorithms for building the target vector grid and target vector slot assignment. Next, a discussion on the generation of the orthogonal bank of chromosomes follows. Lastly, a detailed discussion of the evolutionary solution tracing mechanism and a justification for using competitive templates that partition the both the genotype and phenotype space. This includes the inverse competitive template justification as well. The genotype and phenotype partitioning discussion includes a description about how closer genotype boundaries facilitates a faster convergence for the MOMGA-IIa.

3.4.1 Normalizing Objective Fitness Values. Solutions, after evaluation, are passed to the CTMS. Steps within Algorithm 2 on page 79 marked with † directly relate to the CTMS. In addition, Figure 14 on page 95 also identifies part of the CTMS marked byⒶ, Ⓔ and the upper left hand box. The CTMS rates a point's Pareto dominance merit and then evaluates it for target vector assignment. The target vector evaluation begins with the normalization of each objective value for that particular solution. The normalization function for MOMGA-IIa is much like any other normalizing function. The CTMS keeps track of all book keeping required for objective normalization. The best, $f_\tau(\vec{x}_{max})$, and worst, $f_\tau(\vec{x}_{min})$, for the τ^{th} objective are stored accordingly. These values are used to evaluate incoming points for the best fit into a target vector slot. Equation 44 describes the normalization function where τ is the objective number and $f_\tau(\vec{x}_{\overline{min}}) = f_\tau(\vec{x}_{min}) + \triangle$ such that $\triangle$ is a predetermined *void-factor* to move the points away from the origin and consequently add separation between target vectors with respect to an incoming point. Figure 16 illustrates how the *void-factor* works. In the corner of the cube, a quarter circle void area of $\frac{1}{5}$ is shown to illustrate how the $\triangle$ actually makes for an infeasible region near the origin. This *void-factor* is added to ensure that a single vector does not target dominate all other vectors when in competition for a target vector slot. Vector selection is described next in Section 3.4.2.

$$Norm(f_\tau(\vec{x})) = \frac{f_\tau(\vec{x}) - f_\tau(\vec{x}_{\overline{min}})}{f_\tau(\vec{x}_{max}) - f_\tau(\vec{x}_{\overline{min}})} \tag{44}$$

Algorithm 5 Partition Objective Space

```
procedure PartitionOS(k, $\hat{r}$, &divisions)
    $k$ is the number of objectives; $\hat{r}$ is the number of regular CTs; $\hat{ar} = \hat{r} - k$ is the number of available regular CTs;
    division[all] = 1; g = 1;h = 1;j = 1; finished = False;
    while (!finished) do    ▶ When enough target vector slots are produced with the imaginary grid, stop
        g=1;
        for (i = 1 to k-1) do    ▶ Count up the current amount of cross points on the grid
            g=g*divisions[i-1];
        end for
        if ($g < \hat{ar}$) then
            divisions[j-1]=h+1;    ▶ Division vector has the number of partitions are in each objective space
            j=j+1;
            if ($j > (k - 1)$) then
                j=1; h=h+1;
            end if
        else
            divisions[j-1]=h+1;    ▶ Add the last division to ensure the partitioning of the space is complete
            finished = True;
        end if
    end while
    Build TV Coordinates(divisions);    ▶ Complexity $\mathbb{O}(k^2)$
end procedure
```

3.4.2 Target Vector (TV) Slots. The number of target vector slots is based on the number of *regular* competitive templates, $\hat{r}$, and the number of optimization objectives. A problem having k objectives has at the minimum $\hat{r} - k$ target vector slots available to partition the objective space. The following two sections describe how to build target vector slots (or coordinates) and the selection rules used for assigning incoming solutions to these target vector slots. It should be noted here that sometimes the generation of more $\hat{tvs}$ than $\hat{r} - k$ is required to keep an even partitioning of the objective space. Algorithm 6 is given as a procedure to count how many $\hat{tvs}$ slots are required for an even partitioning of the objective space while at the same time keeping the number of $\hat{tvs}$ to a minimum. The procedure to count

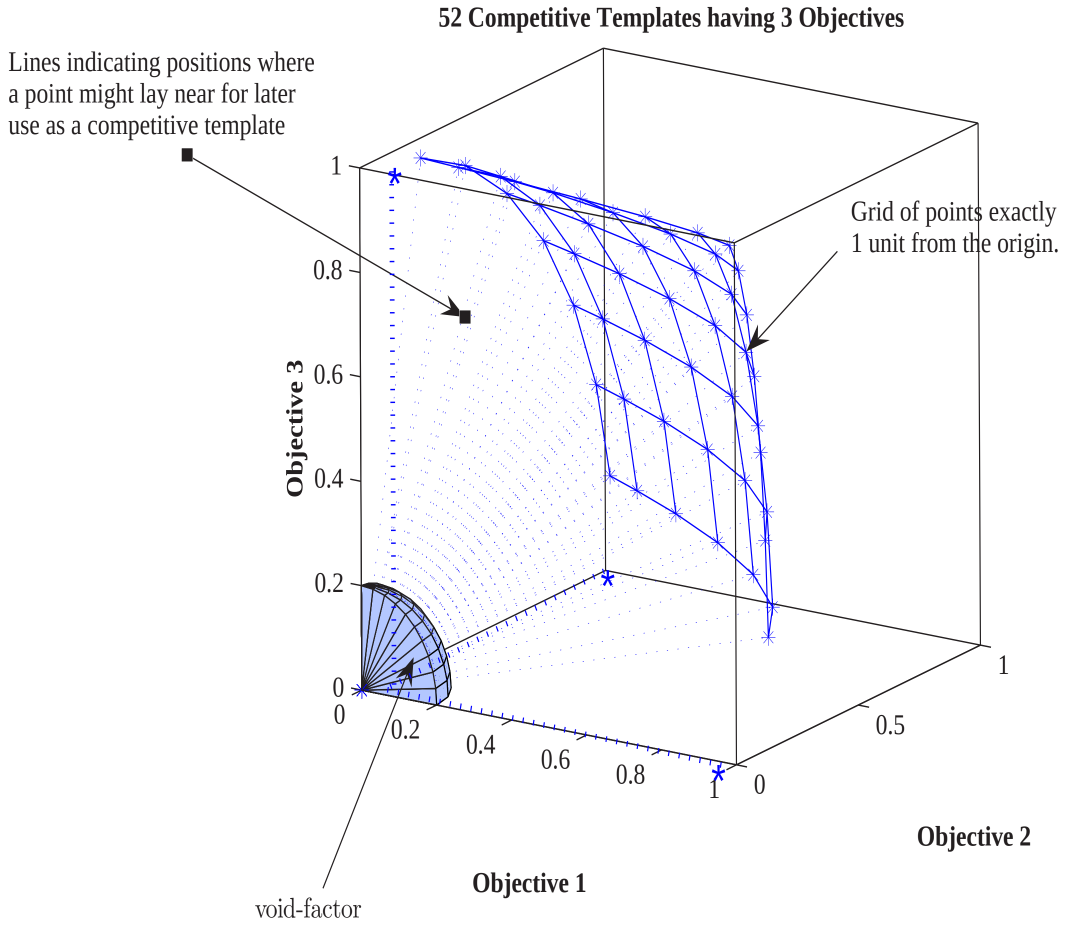


Figure 16: 3D example of the partitioning of the phenotype space.

$t\hat{v}s$ is included in Algorithm 7 on page 118 but because Algorithm 7 also builds the target vector coordinates it is neatly embedded and difficult to find.

Building Distributed Target Vector Slots: Each target vector slot is characterized by a unique target vector. A target vector is an imaginary line in space starting from the origin leading out to another point in space exactly 1 unit away from the origin. Target vector end points not located on the origin may lie only in the ob-

Algorithm 6 Counting $\hat{tvs}$

```
procedure Count $\hat{tvs}$(int $k, \hat{ar}$)
    for ($i = 1$ to $k$) do
        $a_i = 1$;
    end for
    $j = 1$;
    while ($\hat{tvs} < \hat{ar}$) do
        $a_j = a_j + 1$;
        if ($j > k$) then
            $j = 1$;
        end if
        $\hat{tvs} \leftarrow \prod_{i=1}^{k} a_i$
    end while
end procedure
Return($\hat{tvs}$)
```

jective space with coordinates having *non-negative* values. The dimensions of the objective space directly correspond to the number of optimization objectives. Thus, each target vector has $k - 1$ angles defining its direction - where each angle range is $[0, \cdots, \frac{\pi}{2}]$.

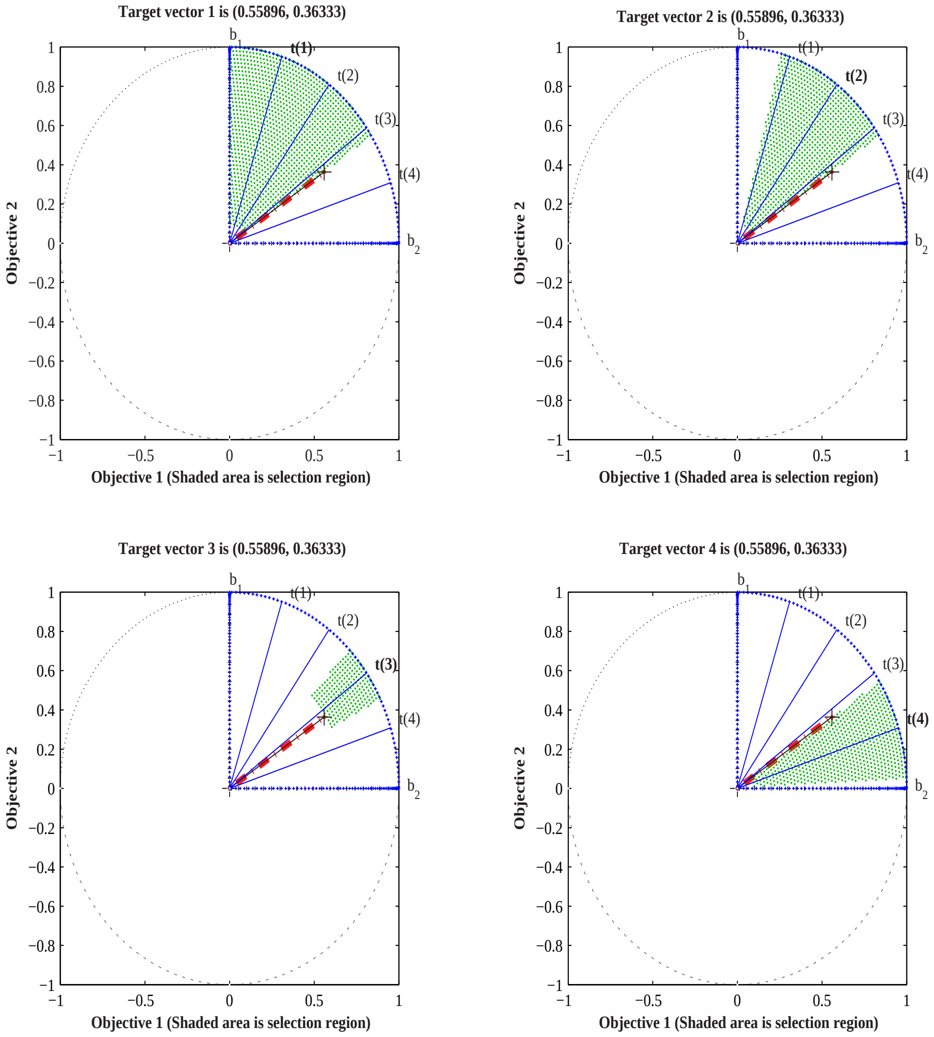
Building a distributed bank of target vector slots that evenly partitions the objective space is challenging - but required if evolving competitive templates are to perform both as elitism and as an implicit niching mechanism. Algorithms 5 and 7 present the pseudcode that can be used to build the target vectors for an MOP. This method generates a partitioning of the objective space using the number of available regular competitive templates, $\hat{ar}$, and the number of optimization objectives, k. The number of target vectors must be equal or greater than the number of available regular competitive templates, $\hat{ar}$. Each set of target vectors for any two problems are the same if the number of objectives and specified regular competitive templates are the same. Figure 15 on page 98 represents a bi-objective problem having ten regular competitive templates but having only eight available target vector slots. The last two regular competitive templates are reserved for solutions evaluating to points found on the objective extremes. Figure 16 on page 104 represents a three objective problem having 52 regular competitive template slots, but only 49 available target vectors. In Figure 16, the target vectors spread (distribution) works for algorithm settings of 46 to 52 regular competitive templates or 43 to 49 available regular competitive templates - when reducing by k objectives.

Selecting Points to place in TV Slots: Incoming solutions are evaluated and selected for target vector slot assignment if the incoming solution's normalized objective values target dominate the currently assigned solution. For clarity, a definition for target vector dominance is given in Definition 22 on page 107.

Definition 22 on page 107 mathematically describes how to determine target point dominance or how to select one point over another point for target vector slot assignment. Equations 45, 46 and Figure 17 on page 107 can be used to explain how target vector slot appointment is accomplished. It is assumed that all vectors represented in Definition 22 are normalized using Equation 44.

Target vector slot point assignment starts with defining half the minimum angular distance between any two target vectors - the angles to choose from are defined as the $\vec{\theta}$ in Algorithm 7. This angle value, $\angle_{min}$, becomes the border between selecting points that are closer to a particular target vector and points that are closer to the Pareto front. All incoming points are normalized to the space of non-dominated vector objective bests and worsts found at that time of slot assignment. This normalizing method is described in Section 3.4.1. Finally, each target vector is checked to see if the incoming point, $\vec{w}$, target dominates the currently assigned point, $\vec{at_i}$, at a particular target vector i. Initially, this caused a single point to be assigned to multiple target vector slots; however, a rule was later added to allow only unique points to a single target vector slot[13].

[13] An incoming solution evaluating to a objective space point is unique in that no two pointers within the target vector assignment object can point to the same solution; however, if duplicates of this point are found later then each duplicate is also considered unique and may be assigned to a different target vector slot if it meets the target dominance requirements.


Target vector 1 is (0.55896, 0.36333)

Target vector 2 is (0.55896, 0.36333)

Target vector 3 is (0.55896, 0.36333)

Target vector 4 is (0.55896, 0.36333)

Figure 17: This figure illustrates the area of dominance if the point, marked by an '+', is assigned to each of the four target vector slots available. In this example, the shaded area represents the target dominate area for incoming vectors, $\vec{w}$, and the '+' represents the target dominated vector, $\vec{at_i}$ in accordance with Definition 22 on page 107.

Definition 22 (Target Vector Dominance): *Vector $\vec{w} = (w_1, \ldots, w_k)$ target vector dominates $\vec{at_i} = (at_{i_1}, \ldots, at_{i_k})$ with respect to target vector slot $\vec{tv_i} = (tv_{i_1}, \ldots, tv_{i_k})$ (denoted by $\vec{at_i} \prec_{t} \vec{w}$) if and only if* ***one*** *of the following two conditions holds:*

1) $\angle \vec{at_i} \leq \angle_{min} \wedge \angle \vec{w} < \angle_{min} \wedge len(\vec{w}) > len(\vec{at_i})$

2) $\angle \vec{at_i} > \angle_{min} \wedge \angle \vec{w} < \angle \vec{at_i}$

with the follow constraints: $\forall j \in \{1, \ldots, k\}, (w_j \geq 0)$

where $\angle_{min} = Minimum(\theta(1), \ldots, \theta(k))$*;*

Equation 45 and 46, and Figure 17 are required to complete this definition and give a visual illustration of its meaning.

$$len(\vec{\xi}) = \sqrt{\sum_{j=1}^{k} \xi_j^2} \tag{45}$$

$$\angle(\overrightarrow{\xi_{t\vec{v}_i}}) = \arccos\left(\frac{\overrightarrow{\xi} \cdot t\vec{v}_i}{|\overrightarrow{\xi}||t\vec{v}_i|}\right) \tag{46}$$

□

3.4.3 Building Orthogonal Arrays. This section has a description of the methodology to build a bank of orthogonal bit arrays. The design is based on a Latin Square design where variables are allowed to have a variety of levels in [232][14]. The orthogonal arrays generated in [232] are constructed for the MOEA, Orthogonal Multiobjective Evolutionary Algorithm for Multiobjective Optimization (OMOEA)[15]. The orthogonal arrays generated in the OMOEA are used for assembling a niche-population. Other approaches for building a good distribution of orthogonal arrays were examined; however, no comparison is made[16]. The Latin Square design, after a slight modification, yields suitable results for the purposes of the MOMGA-IIa. The string size, ℓ, is associated with the string size for a complete solution. The number of orthogonal arrays in the bank is determined by the user-specified param-

[14] See Algorithm 1 in the journal article "Construction of Orthogonal Array" [232]

[15] A description of OMOEA can be found in Appendix E, Section E.5.9 on page 358. An example of a bank of 5 orthogonal 20 *bit* chromosomes (*bit* indicates 2 levels for each variable) can be found in Table 70 on page 404.

[16] These other approaches for building orthogonal arrays include Taguchi arrays, Plackett-Burman arrays, and full factorial. Appendix K on page 403 discusses these methods.

eter, $\hat{o}$. A summary of how the Latin Square function, $L_{\hat{o}}(|\mathcal{A}|^{\ell})$, works can be found in Appendix K, Section K.2 on page 403. The function $L_{\hat{o}}(|\mathcal{A}|^{\ell})$ builds a bank of orthogonal arrays for the MOGMA-IIa and Table 70 in Appendix K on page 404 presents an example used when solving MOP 6 (VL6).

There are two reasons for having a set of orthogonal competitive templates. The first, is to ensure the algorithm has a mechanism that allows for exploration within the genotype domain without regard to the specific decision variables[17]. Solutions assigned to target vector slots are not guaranteed to be well distributed w.r.t. the genotype domain even though their evaluated objective values are w.r.t. the normalized objective space.

Moreover, solutions assigned to target vector slots might have a Hamming code distance of one; meaning they are only one bit apart and concentrated to a specific location within the genotype. Thus, there needs to be both a controlled group that is on the PF$_{known}$ front and a group that explores based on at least one solution evaluating to a non-dominated point on the PF$_{known}$ front. The second reason for having an orthogonal bank of arrays is in that the maximum Hamming code distance between each orthogonal competitive template can bound the maximum BB size required for a BBB search. This new idea in bounding the maximum BB size is discussed in Section 3.5 on page 114. See Figure 18 on page 113 for an example of how the space BB size requirements are reduced. Next, the evolutionary solutions tracing mechanism is discussed in detail.

3.4.4 Evolutionary Solution Tracing Mechanism. The Evolutionary Solution Tracing mechanism (ESTM) tags special characters onto a solution's object in memory as it travels through the MOEA. The major phases each have a label

[17] Although often times the genotype domain is considered to be one in the same with the decision variables, here the distinction is made that a decision variable might be made up of several bits yet the orthogonal design is concerned with the entire set of bits when making an orthogonal array, not the specific decision variables which is made from combining several bits together.

so the postmortem processing can tell how that particular solution evolved. Most importantly, the BB sizes and epistasis are recorded as the solution is tracked. A typical trace might produce the coded line given below.

F 11001111110000000000 0.4698 -0.8115 p 1831 Cs2.p.0(19)(19)E1.BS1

The first character identifies if the solution is the first solution found having the same phenotype merit of being a non-dominated vector. An ***F*** indicates that the point is listed as a first point. A ***D*** indicates that the point is a duplicate of the first. Please note that the evolutionary trace still applies the same for duplicates as it does for solutions found first. The next run of characters, ***11001111110000000000***, indicate the solution's genotype (or decision variable). In the example, solutions have 20 loci positions and $\{0, 1\}$ as the alphabet. This is followed by the solutions evaluated phenotypical (objective) values. The number of phenotype values depends on the number of objectives in the MOP. In this example ***0.469849*** and ***-0.811523*** are the objective space values representing this particular point. The character following the fitness (phenotype) values is a letter identifying if the solution came from a competitive template, ***c***, or a BB, ***p***. The number following this solution type identifier indicates how many times this solution has been found (or observed). Note, other traces also leading to this exact objective space point and genotype solution are not kept - only the fact that the genotype solution has been found several times is recorded. Following the number of times this genotype is found, the evolutionary trace is recorded. A second example presenting a longer evolution is the following:

F 0· · · 0 -0.767 -0.274 p 48 CtB9B8B7B6B5B4ttB3ttu.p.o.1(10)(12)E1.BS3

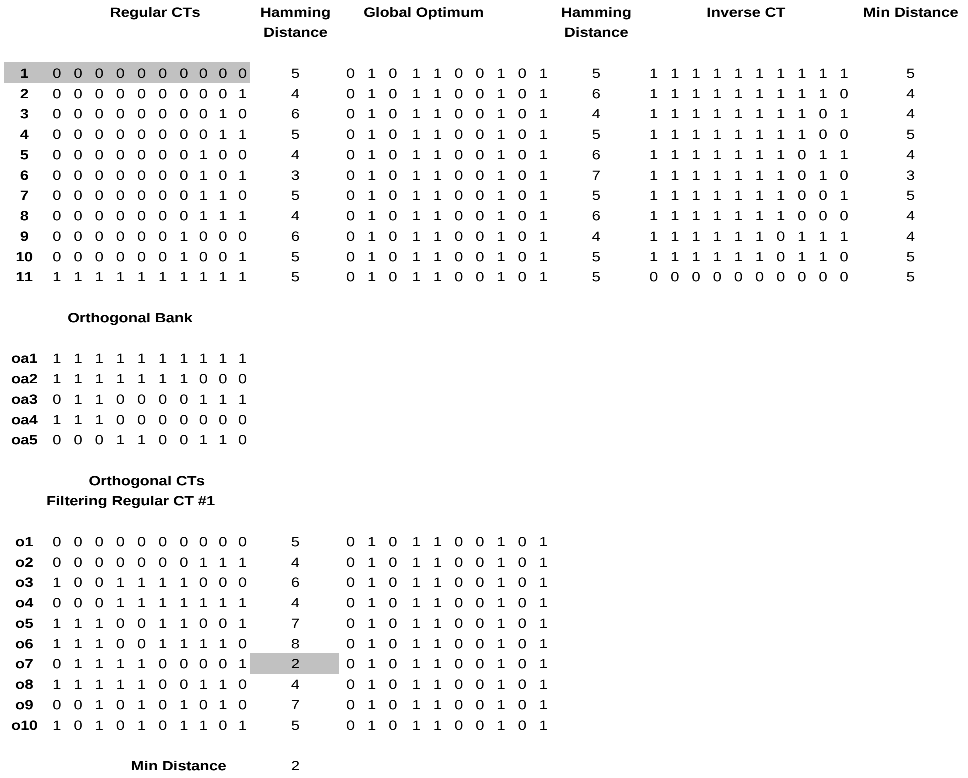
This second example is presented to illustrate different trace lengths for different evolutionary paths taken by a solution; note that most of the genotype solution is left out due to page width constraints. The first example has a short path, whereas the second example has a longer path. The ***C*** represents the PCI phase (Ⓑ in Figure 14 on page 95), or point of inception for this particular solution. Characters after this point in the trace can mean different things. As this BB proceeds through the MOEA search, different characters are added to the trace indicating where the BB traveled. A ***t*** indicates that at least one tournament selection is performed on this solution at that injunction. A ***B#*** indicates that the solutions is reduced to a size # of bits. In the second example, the solution is first filtered down to 9 then 8,7,6,5,4, and finally 3 bits. These filtering operations have a few tournament selection operators, ***t***, mixed within the filtering operations. Finally, if the string is spliced together, a ***u*** appears in the trace. If ***m*** appears, the solution was mutated. At the end of the evolutionary trace, the solution is submitted to the PFS for storage - a "***.***" in the trace indicates the evolutionary process for this solution is complete.

Once inside the PFS, more tags are added to the solution to specify exactly the type and construction of the solution. Each tag from this point on is divided by a delimiter of some kind. A ***p*** indicates a population member and a ***c*** indicates a competitive template. In both the first and second example, the solution is found within the population so a ***p*** appears. Next, if the solution is a population member, the evaluation of the member is overlaid onto a competitive template regardless of size. Clearly, if the solution is not a partial string, the competitive template plays no role in its makeup, yet the competitive template type and number are still recorded. If the competitive template is something type other than a regular template, it is indicated directly after the solution type indicator. An ***o*** indicates an orthogonal template, and ***i*** indicates an inverse template. Following the template type is the template number used for this solution. In the first example, the number ***0*** regular competitive template is used. Note that array indices start at 0 rather than 1 - thus, this is the 1st regular competitive template in the list of regular templates. In the second example, the number ***1*** orthogonal template is used to evaluate the partial solution. This is indicated by the ***o.1*** in the trace. Next, the partial string size is recorded and found within parenthesis. This is the size of the string or BB that is overlaid onto the competitive template. These are ***19*** and ***10*** for examples 1 and 2, respectfully. Following the string size, the space between the extreme loci positions of specified bits is given within parenthesis. These are ***19*** and ***12*** for examples 1 and 2, respectfully. Finally, the tracing mechanism records *when* within the entire MOMGA-IIa evolutionary search this particular solution is found. It is important to note that the algorithm is run ten times for each experiment and the archive stays active the entire time, so solution and associated objective value vector transfer occurs between each experiment. Therefore, the experiment number and the BB stage are recorded. As a check, a partial string should never be smaller than the BB stage recorded at the end of the trace. In the examples, the first solution is found in BB stage ***1*** of experiment ***1*** and the second solution is found in BB stage ***3*** of experiment ***1***.

Traces for each solution are dumped to a points file (containing the first solution found for each member) and a duplicates file (containing the first and all duplicates for each member). In addition to these two files, there are files used for storing the competitive templates and the slotted target vector solutions after each BB stage. For the post mortem analysis, the point and duplicate files are used.

The post mortem analysis is important because it is desirable to know the sizes of BBs and *epistasis* used to make resulting solutions. Figures 95, 96, 97, 98, 99, 100, 101, 102, and 103 in Appendix D on page 312 illustrate the epistasis found in Test Suite MOPs and deception problems of size 60. Because of the newness of this type of graphical analysis (recording the BB sizes used in finding solutions evaluating to non-dominated vectors), the only metric that can say something about results is the level of *epistasis* a problem may contain. There is another BB analysis done in [125]; however, this analysis assumed much away, did not trace solution evolution, or waited to the end of a generation to use the same sized BBs to create each solution.

Traces are run on many different problems, each having different characteristics. One example is presented in Figure 19 on page 120. Insight can be drawn from these illustrations relating BB sizes to Pareto front vectors for problems having different characteristics. Each problem describes BB size w.r.t. PF vectors within



	Regular CTs	Hamming Distance	Global Optimum	Hamming Distance	Inverse CT	Min Distance
1	0 0 0 0 0 0 0 0 0 0	5	0 1 0 1 1 0 0 1 0 1	5	1 1 1 1 1 1 1 1 1 1	5
2	0 0 0 0 0 0 0 0 0 1	4	0 1 0 1 1 0 0 1 0 1	6	1 1 1 1 1 1 1 1 1 0	4
3	0 0 0 0 0 0 0 0 1 0	6	0 1 0 1 1 0 0 1 0 1	4	1 1 1 1 1 1 1 1 0 1	4
4	0 0 0 0 0 0 0 0 1 1	5	0 1 0 1 1 0 0 1 0 1	5	1 1 1 1 1 1 1 1 0 0	5
5	0 0 0 0 0 0 0 1 0 0	4	0 1 0 1 1 0 0 1 0 1	6	1 1 1 1 1 1 1 0 1 1	4
6	0 0 0 0 0 0 0 1 0 1	3	0 1 0 1 1 0 0 1 0 1	7	1 1 1 1 1 1 1 0 1 0	3
7	0 0 0 0 0 0 0 1 1 0	5	0 1 0 1 1 0 0 1 0 1	5	1 1 1 1 1 1 1 0 0 1	5
8	0 0 0 0 0 0 0 1 1 1	4	0 1 0 1 1 0 0 1 0 1	6	1 1 1 1 1 1 1 0 0 0	4
9	0 0 0 0 0 0 1 0 0 0	6	0 1 0 1 1 0 0 1 0 1	4	1 1 1 1 1 1 0 1 1 1	4
10	0 0 0 0 0 0 1 0 0 1	5	0 1 0 1 1 0 0 1 0 1	5	1 1 1 1 1 1 0 1 1 0	5
11	1 1 1 1 1 1 1 1 1 1	5	0 1 0 1 1 0 0 1 0 1	5	0 0 0 0 0 0 0 0 0 0	5

	Orthogonal Bank
oa1	1 1 1 1 1 1 1 1 1 1
oa2	1 1 1 1 1 1 1 0 0 0
oa3	0 1 1 0 0 0 0 1 1 1
oa4	1 1 1 0 0 0 0 0 0 0
oa5	0 0 0 1 1 0 0 1 1 0

	Orthogonal CTs Filtering Regular CT #1		
o1	0 0 0 0 0 0 0 0 0 0	5	0 1 0 1 1 0 0 1 0 1
o2	0 0 0 0 0 0 0 1 1 1	4	0 1 0 1 1 0 0 1 0 1
o3	1 0 0 1 1 1 1 0 0 0	6	0 1 0 1 1 0 0 1 0 1
o4	0 0 0 1 1 1 1 1 1 1	4	0 1 0 1 1 0 0 1 0 1
o5	1 1 1 0 0 1 1 0 0 1	7	0 1 0 1 1 0 0 1 0 1
o6	1 1 1 0 0 1 1 1 1 0	8	0 1 0 1 1 0 0 1 0 1
o7	0 1 1 1 1 0 0 0 0 1	2	0 1 0 1 1 0 0 1 0 1
o8	1 1 1 1 1 0 0 1 1 0	4	0 1 0 1 1 0 0 1 0 1
o9	0 0 1 0 1 0 1 0 1 0	7	0 1 0 1 1 0 0 1 0 1
o10	1 0 1 0 1 0 1 1 0 1	5	0 1 0 1 1 0 0 1 0 1
	Min Distance	2	

Figure 18: This figure illustrates an example of how the inverse or orthogonal competitive templates make a difference in lowering the BB size required to move from a competitive template to a global optimum. The Hamming code distance between the competitive template and global optimal solution represents how large a BB would have to be to get to the global optimal solution when using a particular competitive template. The upper matrices illustrate that by using 11 regular CTs, the minimum BB size required to get to the global optimal is 3 bits and the maximum BB size required is 6; however, when adding inverse CTs the minimum BB size stays the same at 3 bits, but the maximum BB size drops to 5 bits. The bottom matrices indicate that by filtering a selected regular CT through 5 orthogonal CTs, the minimum BB size drops to 2 bits. This is a simple example of BB size requirement reduction by using carefully generated CTs. Unfortunately, adding orthogonal templates does not guarantee such a reduction each time, but the adding of inverse CTs does.

the respective MOP's result and analysis section if the EST is used when solving that MOP. Note that along the back wall of the bottom plot is the epistatic level for each PF vector found. The lower the epistatic level, the more difficult that solution is to find for an implicit BBB. Next, genotypic space partitioning is discussed related to BB theory and BBBs.

3.5 Advantages in Partitioning the Genotype Space

BB theory is based on the assumption that if a BBB finds at least one optimal set of BBs, then the builder has a better probability to put these BBs together to find all optimal solutions - be they single or multiobjective. However, if a BBB has an operator that destroys good BBs, the builder is limiting the possibility for finding an optimal set of BBs. Furthermore, considered in the set of destructive BB operators, like crossover, is the inverse operator. Within the design of the MOMGA-IIa there is an inverse operator used for competitive template generation. Moreover, the mechanism for orthogonal competitive template design can also be considered a partial inverse operator. This being said, it must be for good reason that these operators are included in a state-of-the-art BBB, like the MOMGA-IIa.

There are two reasons for implementing these operators in the MOMGA-IIa. The first goes back to the reason for the original design of the mGA as an algorithm developed to defeat deception. The population sizing equation for the mGA identifies the number of BBs of size **o** that are needed to construct every possible solution - including all optimal solutions. Now, as **o** increases, so does the population size for the mGA. Looking forward to the new population sizing equation used in the MOMGA-II and MOMGA-IIa, the BB size is still a factor in the size. Thus, by reducing the highest BB size required for finding the optimal set of BBs, the population size can also be reduced while keeping the same probability for finding optimal solutions.

The second reason, as was mentioned in Section 3.4.3, is that there is a need for a balance of exploration and exploitation within any evolutionary algorithm. The orthogonal and inverse operators are good operators for partitioning the space and allowing exploration to occur within the MOMGA-IIa.

$$\|\mathcal{A}\|^{\mathbf{o}} * \binom{\ell}{\mathbf{o}} \tag{47}$$

To show the reason why a reduction of the required BB size results from using inverse and orthogonal templates, an example is used. First, suppose there is a gain of a competitive template (possibly a local minimum) having a Hamming code distance $\ell - 1$ away from the global optimal solution. Based on the optimal BB set definition, the optimal BB set is the BB that places the solution on the global optimal solution using a single $\ell - 1$ BB. One might be able to find several smaller BBs that may place the combination of these blocks on the same global optimal solution; however, it is less likely that someone may first find many smaller BBs and then put these blocks together in such a way to discover the global optimum. Unfortunately, it would require a single BB of size $\ell - 1$ to find this global optimal. Yet, if an inverse competitive template is used, the inverse is only a Hamming code distance of 1 away from the solution. Unfortunately, this is the best case example. The worse case example is when the regular competitive template is a Hamming code distance of $\frac{\ell}{2}$ away from the global solution. Inverting this competitive template makes no difference in the distance from the global solution (remaining $\frac{\ell}{2}$ Hamming code distance). So, the advantage to having an inverse competitive template is that one halves the required BB size. In other words, the builder only needs to search for BB sizes 1 to $\frac{\ell}{2}$ – bringing the genotype space boundaries closer. The orthogonal competitive templates are also designed to bring the boundaries of optimal solutions closer; however, because there are not a full factorial number of orthogonal arrays in use, it is not possible to give an exact BB size reduction for the number of orthogonal templates. Moreover, for some global solutions, orthogonal templates may reduce greatly the required BB size while other, orthogonal templates may not reduce this required size at all. Furthermore, the inverted templates are the only guaranteed reduction template.

A final advantage to using inverse templates comes from possibly figuring out how much emphasis should be given to certain good BBs that are embedded within the template. This is called *triangulation.* Triangulation works like this. First a regular competitive template, $\mathcal{C}$, is inverted, $\bar{\mathcal{C}}$. As the algorithm proceeds, good BBs for both the $\mathcal{C}$ and $\bar{\mathcal{C}}$ are collected. Making the assumption that $\mathcal{C}$ is definitely found to be adaptively good in the phenotype, BBs - representing good schema - found for $\bar{\mathcal{C}}$ and also having commonality in $\mathcal{C}$ are said to be uniformly good.

Current understanding of what to do with uniformly good BBs is not recognized; however, duplication of these uniformly good schema within the population may help to keep continuously good BBs in a healthy supply.

3.6 Advantages in Partitioning the Phenotype Space

While the advantages are many for partitioning the genotype space, there are several advantages to keeping a partitioned phenotype space, as well. First, the original thought behind keeping CTs distributed across the phenotype space is that it allows for an even search across the Pareto front – this is called *implicit niching.* In addition, when the PF_{known} vectors are presented to the decision maker, a shorter distributed list (possibly made of the competitive templates) can be given as a smaller list[18] of, well objective space distributed, solutions from which to choose. Even if the full list is given to the decision maker, the portioning of the phenotype space helps make the distribution along the Pareto front evenly spaced. The Orthogonal Multiobjective Evolutionary Algorithm I and II (OMOEA) are two other MOEAs in which the entire structure of the algorithm is based on keeping a distributed phenotype search – it is called niching within these MOEAs. These MOEAs are discussed in Appendix E on page 324. The MOMGA-IIa is not as rigorous at keeping a partitioned phenotype space search as the OMOEA because it is willing to select a search vector off the Pareto front (dominated) that is closer to the target vector guide to keep a good phenotype distribution of competitive templates.

One final advantage of the MOMGA-IIa for a decision maker is the fact that the algorithm can be made to keep solutions that evaluate to be Pareto epsilon

[18] The smaller list of optimal solutions represents a subset of the entire list of optimal solutions.

non-dominated vectors. Definition 6 on page 12 describes the decision criteria for keeping these epsilon non-dominated vectors. Figure 20 presents a Gaussian ridged distribution that could have been calculated from a band of vectors that are Pareto epsilon non-dominated, giving the decision maker (DM) more PF points to choose from and a weight on possible points that might be close to the mean of the band. Next, the MOMGA-IIa's serial and parallel implementation is discussed.

3.7 MOMGA-IIa Extras

As presented earlier, the MOMGA-IIa has many enhancements from the original MOMGA-II. In addition to these enhancements, there are a few extras that also follow this design. First, the MOMGA-IIa can operate in MOMGA-II mode by setting the MOMGA-IIa flag to *false* and adding only one competitive template in the *CT_config.dat* file. MOMGA-IIa can operate in single objective mode. This operating mode is a modified fmGA when the MOMGAII-a flag is set to *true* and a traditional fmGA when the MOMGAII-a flag is set to *false*. In addition to the innovative design, the software is written in a way that can be useful. New MOPs can be *Plugged* into the algorithm without integrating the new code directly into the algorithm. Finally, Matlab code is written to assist in the post mortem visualization of the BB trace.

3.8 Parallel versus Serial Implementation

One of the first design decisions in rebuilding an algorithm is deciding if the data structure representation is adequate for a projected application. The original MOMGA is directly modified from the mGA's C code. Zydallis took the MOMGA code and integrated the fmGA to make the MOMGA-II; keeping a bit-wise representation for each chromosome. The MOMGA-II cut down on storage space requirements for the population sizes using a bit-wise chromosome. The MOMGA-IIa is designed from the fmGA used in the PSP problem [41]. In the old fmGA code,

Algorithm 7 Build TV Coordinates

```
procedure BUILD TV COORDINATES(float &divisions)
    $\hat{tvs} = 1$
    for ($i = 1$ to $k$) do                              ▶ Each obj. func. is parted using $\frac{\pi}{2}$radians
        $\theta[i-1] = \frac{PI}{2*(divisions[i-1]+1)}$;
        $\hat{tvs} = \hat{tvs} * divisions[i-1]$;
    end for
    for ($i = 1$ to $\hat{tvs}$) do                      ▶ Assign unique parts for each obj. to each TV
        for ($j = 1$ to $k-1$) do
            $Tvector[i][j] = 1$
        end for
    end for
    for ($i = 1$ to $\hat{tvs}$) do
        $h = i - 1$; $j = k - 1$;
        while ($h < i$) do
            if  ($i = 1$)  then
                $Tvector[i-1][j-1] = 0 + 1$
            else
                $Tvector[i-1][j-1] = Tvector[i-2][j-1] + 1$
                for ($g = 1$ to $j-1$) do
                    $Tvector[i-1][g-1] = Tvector[i-2][g-1]$
                end for
            end if
            if ($Tvector[i-1][j-1] > divisions[j-1]$) then
                $Tvector[i-1][j-1] = 1$; $j = j + 1$;
            else
                $h = h + 1$
            end if
        end while
    end for
    for ($i = 0$ to $k-1$) do
        for ($j = 0$ to $\hat{tvs}$) do
            $TVangles[i][j] = Tvector[i][j] * \theta[j]$
        end for
    end for
    for ($i = 0$ to $\hat{tvs} - 1$) do                  ▶ Create Cartesian coordinates for the tv vectors
        for ($h = 0$ to $k-2$) do
            for ($g = 0$ to $h-1$) do
                $TVcc[i][g] = TVcc[i][g] * \cos(TVangles[i][g])$;
                $TVcc[i][h+1] = \sin(TVangles[i][h])$;
            end for
        end for
    end for
end procedure
```

Algorithm 8 Orthogonal Arrays

```
procedure ORTHOGONAL ARRAYS($\hat{o}, Q, \ell$)   ▶ $\hat{o}$: Number of vectors, Q: Range of Levels, $\ell$: String Length
    Q is always set to 2 for a binary alphabet; $Q_j = \hat{o}$
    zeros($oa[\hat{o}][\ell]$)
    $J = \lceil \frac{log(\ell * Q - \ell + 1}{log(Q)} \rceil$
    for ($\rho = 1$ to $J$) do
        $j = \frac{Q^{\rho-1}-1}{Q-1} + 1$
        for ($i = 1$ to $Q_j$) do
            if (($i - 1 > \hat{o}$) OR ($j - 1 > \ell * \hat{o}$)) then
                Skip
            else
                $oa[i-1][j-1] = \lfloor \frac{i-1}{Q^{J-\rho}} \rfloor$ modulo $Q$
            end if
        end for
    end for
    for ($\rho = 2$ to $J$) do
        $j = \frac{Q^{\rho-1}}{Q-1} + 1$
        for ($s = 1$ to $j - 1$) do
            for ($\tilde{t} = 1$ to $Q - 1$) do
                for ($g = 0$ to $\hat{o} - 1$) do
                    if (($g > \hat{o}$) OR ($j + ((s - 1 * (Q - 1)) + \tilde{t} - 1) > \ell$)) then
                        Skip
                    else
                        $oa[g][j + ((s-1) * (Q-1)) + \tilde{t} - 1] = (\tilde{t} * (oa[g][s-1]) +$
                            $oa[g][j-1])$ modulo $Q$
                    end if
                end for
            end for
        end for
    end for
    Return(oa)
end procedure
```

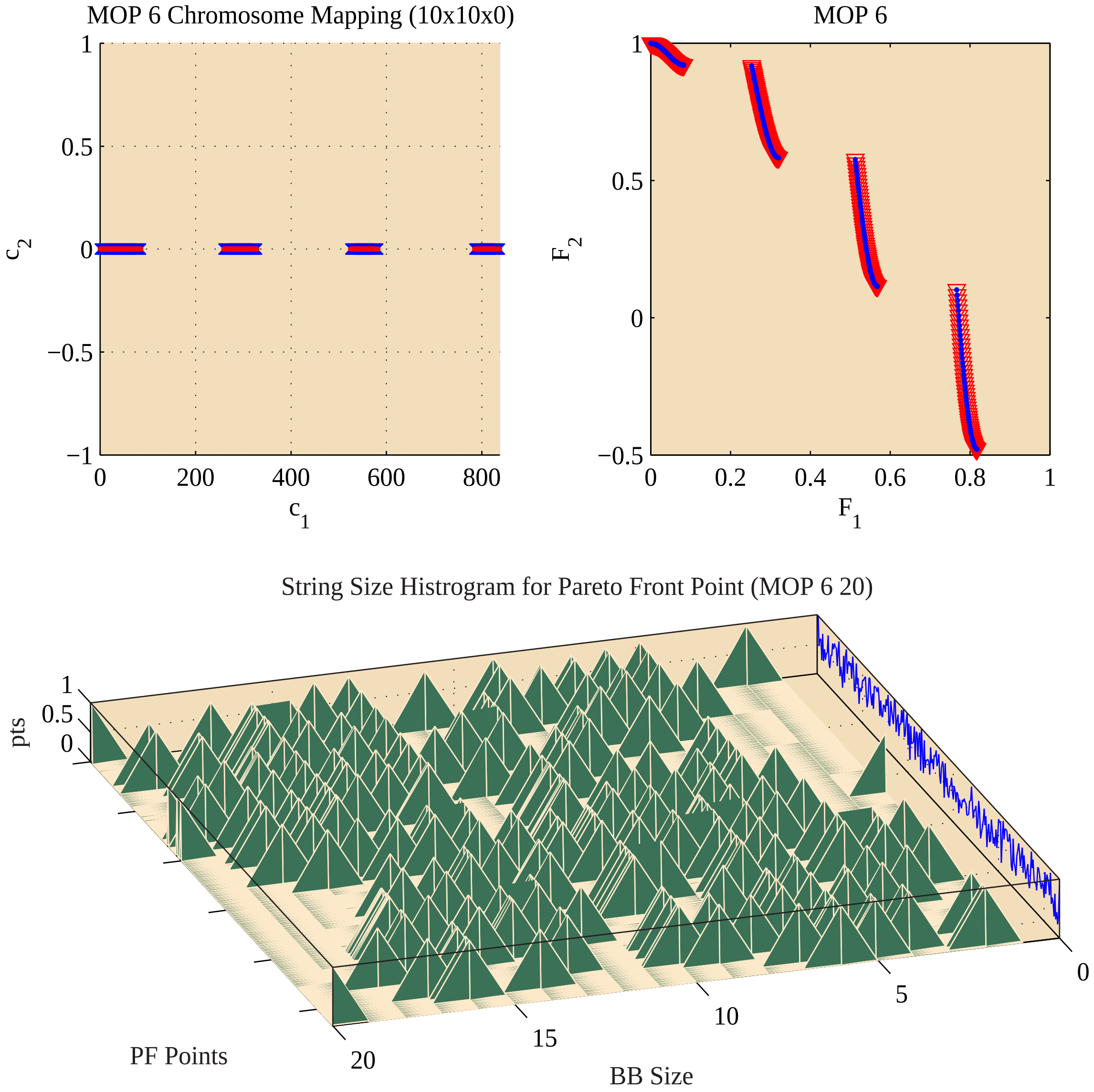


Figure 19: This figure illustrates an example of a short analysis for MOP 6 (VL 6) in [34]. The top left plot presents the genotype domain where the PF$_{true}$ vectors lie. The top right plot shows the MOMGA-IIa finding the solutions evaluating to non-dominated vectors. The bottom contour illustrates the BB sizes used in finding these points. The PF points axis sequentially lists (in order) the PF vectors from upper left to bottom right of the top plot. In other words, the left most PF vector on the bottom plot maps the best solution found for F_2 and the right most PF point on the bottom plot maps directly to the best solution for F_1. Notice that the BBs sizes are generally under size 10. The overall string size is 20 and it is noticeable that the extremes are illustrating a larger BB size for finding solutions evaluating to non-dominated vectors. Empty triangles (although difficult to see) indicate PF$_{true}$ vectors, while filled in circles represent PF$_{known}$ found by MOMGA-IIa.

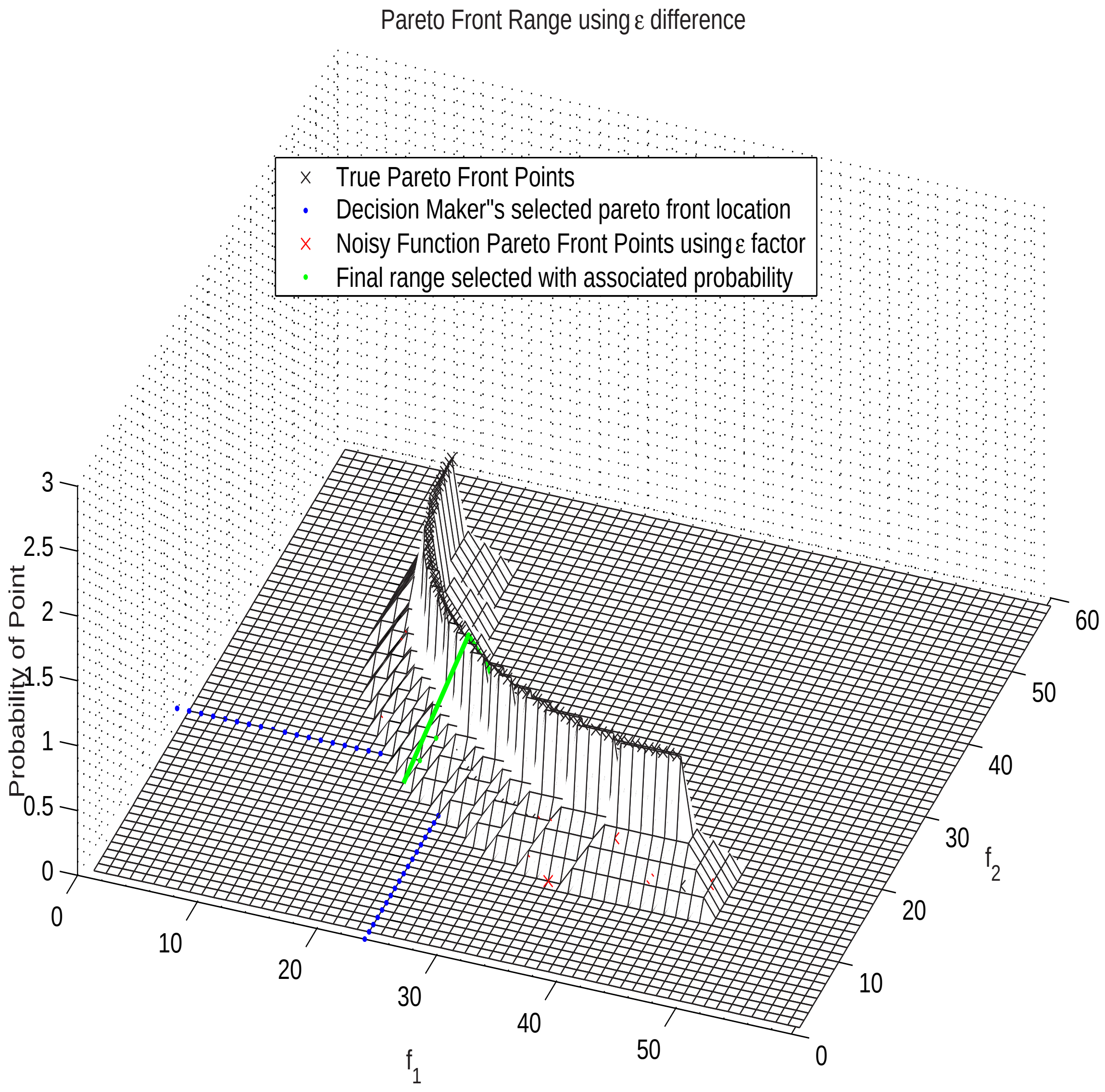


Figure 20: Graphic illustrating how a noisy fitness function might cause the appearance of a distribution for the decision maker (DM) when selecting solutions that evaluate to vectors that are epsilon non-dominated.

bit representation is done using strings of characters called alleles, and bit position representation is given in the form of a vector of integers. The step away from code shared by the MOMGA and MOMGA-II is indeed a big one; however, the idea to go from the old MOMGA-II code to the MOMGA-IIa was calculated as being overwhelming due to the projected enhancements. Therefore, a slim lined fmGA having a different chromosome representation (described above) is used as the base EA to build the MOMGA-IIa.

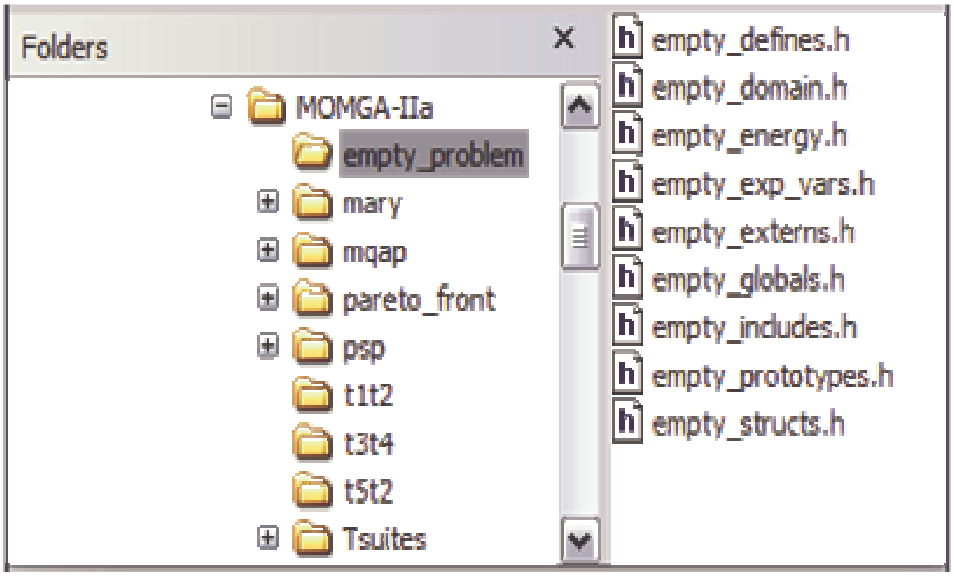


Figure 21: Graphic illustrating a layered design approach for the MOMGA-IIa to allow easy MOP integration and removal. The base folder, "MOMGA-IIa", holds the algorithm code. Three parameter files and two header files need editing when changing between MOPs. Subfolders hold MOP files including all global variables, external variables, special structures, and functions (including local searches) specific to that MOP. Illustrated in this figure is a list of files that are required for each and every MOP, even if that file remains empty.

3.9 Decomposing the MOMGA-IIa into a parallel implementation (low level design

Parallelization of a new MOEAs can be difficult depending on the parallel model implemented. Commonly, parallel implementations are married with the hardware architectures intended to saddle the algorithm. Clearly, old MOEA parallel models can be used for parallelizing the MOMGA-IIa. These models include, but are not limited to, the Island model, Master-slave model, and diffusion model. Appendix J on page 398 reviews these models for completeness. These are well known and have been used by many researchers in the past [31, 53, 223, 244]. A relatively new parallel MOEA model is called the Hierarchical Fair Competition (HFC) Parallel Model. This model is based upon a change in algorithm structure the HFC used to combat an EA's premature convergence [108, 110]. The HFC structure is based on an assembly line structure (see Figure 22 on page 123 for an illustration). It is thought that, by using this model, the small BBs are connected together to form larger, more tightly linked BBs as the algorithm proceeds where, finally, end solutions gain the full benefit of intermediate levels of BBs. This model is interestingly similar to the MOMGA-IIa's juxtapositional *phase*.

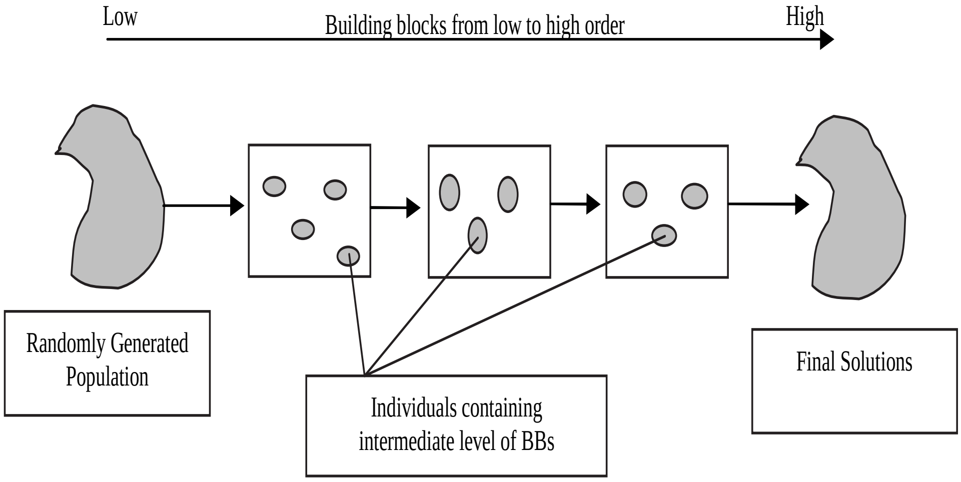

Low
Building blocks from low to high order
High
Randomly Generated Population
Individuals containing intermediate level of BBs
Final Solutions

Figure 22: The assembly line structure of the continuing EA model – how the model is used as the HFC Model [189].

The parallel version of this model has multiple subpopulations organized in a hierarchy, where each subpopulation can only accommodate individuals within a specified range of fitness values (phenotype partitioning). Each subpopulation has an admission buffer that determines member admission - allowing for only qualified candidates to be collected from other subpopulations. Each subpopulation maintains only a range of fitness values assigned to individuals. Individuals can evolve out of a subpopulation to a subpopulation with a better fitness range, but never to a subpopulation of lower fitness ranges.

Another way to look at this type of model within a multicriteria problem domain would be the following: suppose you have a population of solutions each evaluating to two or more values. Each population member is then ranked according to non-dominance - similar to the ranking that occurs within the NSGA-II (see Section E.5.4 on page 350). All population members having the same dominance rank are assigned to the same subpopulation for further development using appropriate MO EVOPs. As population members evolve in the subpopulations they tend to get better fitness values and move *up* into subpopulations holding members having a better dominance ranking. This is called migration or population member exchange. Each subpopulation has assigned an export threshold that is like an escape velocity where, after meeting this threshold, the member can migrate to a new subpopulation. Migration only occurs in one direction. The number of levels in the hierarchy (number of subpopulations) can be fixed (predefined up front) or adaptively as needed.

A new MOEA called the HEMO is constructed and tested using the HFC model inside components of the PESA, SPEA, and extending the NSGA-II's controlled elitism. It is expected that HEMO is to perform better on multi-modal real-world problems where pre-convergence is known to hinder the PESA, SPEA and the NSGA-II [109].

Paralleling the MOMGA-IIa can be done in a similar manner, where the main difference would lie in ranges and thresholds of the subpopulations. The parallel version of HFC uses a vertical approach to partition subpopulations, but by changing the partitioning to a lateral[19] approach, the phenotype space can be subdivided to

[19] Lateral in this case might be in the form of a wedge (2 dimensions) or cone (3 dimensions) coming from the origin where the origin is the two lowest values found during the entire search process.

have only fitness values that fall within a fitness range of values associated with the target vector guides described earlier in this chapter. If these two models are used together, there would be vertical and lateral partitions of the phenotype space leading to a grid of populations and many different exchange thresholds for members to be able to cross to escape into another subpopulation. Although neither of these models are implemented within the MOMGA-IIa, either the vertical, lateral or both are viable parallel models.

3.10 Summary

In conclusion, the MOMGA-IIa is much like the MOMGA and MOMGA-II in that it is an explicit BBB. However, the MOMGA-IIa gets a massive design reconstruction of the competitive template generation, updating, and evolving system. Additionally, the archive that is updated in all phases helps trap good BBs before they become disrupted within the BBF phase. The MOMGA-IIa's new design has shown to compete well with the mBOA and MOMGA-II when solving several deception problems and the mQAP. Redesign of an algorithm is sometimes an uncertain venture - the no free lunch theorem always applies. As many researchers know, it is much easier to add proven mechanisms from the single or multi-objective optimization field into incomplete algorithm for added performance because it takes much of the risk away from achieving desired results. This path is not taken when designing the MOMGA-IIa. This chapter described the development of the MOMGA from its single objective origins to its latest state-of-the-art form. Descriptions include justification and details for each new mechanism. The CTMS, PFS, and BB tracing mechanism are all described in detail. Enhancements to this algorithm are accomplished with two objectives in mind. The first objective is to allow for the MOMGA-X to compete well with other explicit BBBs. Many new mechanisms to solve the limitations of the MOMGA-II are added and, in the next chapter, validated to work well. The second objective is to design the algorithm in such a way that allows for a BB researcher to identify how the BB sizes are related to solving each problem. This is successful, and insight is drawn from each problem solved by the MOMGA-IIa.

Discussed in this chapter are the design details for the MOMGA-IIa. Justification for each enhancement is given as well as a detailed description of how the implementation is accomplished. In the next four chapters, these enhancements are shown to improve performance of the MOMGA-series MOEA in both effectiveness and efficiency on selected MOPs. In addition to the performance analysis, experimental design and problem set characteristics are also discussed.

IV. MOEA Design of Experiments

Measuring efficiency[1] and effectiveness[2] of MOEAs as applied to MOPs is no simple task. The purpose of this chapter is to justify and discuss the MOEA metric and MOPs used for testing the MOMGA-IIa. This chapter clearly applies a statistically correct method and standardized metrics to determine how well the MOMGA-IIa performs in comparison to at least the previous implementation (MOMGA-II). Also, where possible, comparisons to other MOEAs are made. In addition to experimental-finding comparisons, MOEA metric validation is addressed, as well as MOMGA-IIa BB size findings from the BB tracing mechanism via visualization techniques.

4.1 Design of Experiments

There are several approaches used in conducting a scientific method of experimentations for MOEAs. In the past, seven metrics were identified for an accurate, reliable, consistent and non-arbitrary representation of MOEA performance on an MOP. These seven metrics can be found in section 4.3.2 on page 133. In addition, a set of MOPs were also selected as a *benchmark* suite to test an MOEA's performance over a range of MOPs having different characteristics [31–33, 39, 55, 76, 128, 129, 132, 196, 208, 220, 235]. The test suite of MOPs is described in section 4.7 on page 4.7. In this section, the following plan of attack is formulated:

1. State experimental Goal
2. Make hypothesis and/or prediction
3. Choose metrics to measure aspects of each MOEA
4. Design and Execute Experiment (gather data) "test"

[1] Efficiency is the measuring of computational time (or wall clock time given all things equal - computer hardware related) to obtain solutions. [34]

[2] Effectiveness measures the accuracy and convergence of obtained solutions. Researchers often include robustness, scalability, and ease of use qualities into the effectiveness; however, this effort mainly focuses on measuring the accuracy and convergence of MOEA found PF vectors to the set of vectors in PF$_{true}$. [34]

5. Analyze results
6. Draw Conclusions and Conjectures
7. Report results

This approach is consistent with outlines presented by Barr [9], Jackson [111], and Jain [112]. The scientific method as a more generic approach has four steps: observation[3], hypothesis, predict using hypothesis, and testing [10, 140, 229]. These steps are implicitly followed.

4.2 Focus for each experiment

The main focus of this chapter is the comparison of MOEA performances over a variety (class-wise) of MOPs. Both effectiveness and efficiency are studied where possible; however, in some instances, effectiveness is the only performance measured. The two algorithms under test are MOMGA-II and MOMGA-IIa. In some cases, test data from other algorithms illustrates an idea or conclusion. Algorithm settings are kept the same for most experiments (see Appendix A on page 272 for parameter settings of each MOEA for each experiment). The main difference separating the algorithms is the competitive template management system and the active archive (see section 3.3.4 on page 94). Also note that the analysis does not attempt to say that one MOEA is better than another across each and every MOP (*i.e.*, respect NFL theorem). It simply recognizes an MOEA's performance on a particular MOP.

4.3 MOEA Metrics

Similar to previous studies when comparing MOEAs [222,244], seven standardized metrics identified by Van Veldhuizen and Zydallis are used. Many researchers have studied MOEA metrics [32–34, 39, 55, 76, 128, 132, 196, 208, 220, 235] and a summary of good metrics that can be used to statistically compare MOEA is found

[3] Observation in this case is used as a foundation to design or model to the problem from the real-world.

in [34] on page 155. In addition and at the same time, Zydallis recognized that the metrics identified by [34] were a good start to comparing MOEAs and allowed for a statistical comparison. Today, researchers add a satisfiability[4] tied to the metrics used when comparing two MOEAs. Knowes recommends two complementary approaches for comparing MOEA results [129]. It is suggested that an empirical attainment function and set of dominance-compliant quality indicators be used to evaluate and compare the approximation sets from multiple runs for two or more stochastic MOEA optimizers. The attainment function is discussed but not used; however, the rigorous dominance-compliant quality indicators suggested are heeded. Ziztler also supports this suggestion of having certain quality indicators and advertises that it is advantageous to have metrics that are both *complete* and *compatible* to make good comparisons between MOEA results [240]. These three additional metrics are the following: ϵ indicator and the R2 and R3 utility indicators. Thus, in addition to the seven standardized metrics used by Van Veldhuizen and Zydallis, three more metrics are added to obtain a *complete* and *compatible* comparison for stochastic multiobjective algorithms [129,239]. There are many other metrics within the MOEA field; however, the ones chosen for use are thought to be adequate for this testing. Appendix I on page 395 addresses other metrics that can be used for analyzing two MOEAs.

Definition 23 (Approximate Set): *Let $\hat{\mathcal{A}} \subseteq \Omega$ be a set of all objective vectors. $\hat{\mathcal{A}}$ is called an approximation set if any evaluated individual of $\hat{\mathcal{A}}$ evaluates to a vector that does not weakly dominate (see Table 9 on page 133 for the definition of weak domination) any other objective vector produced by evaluating individuals in $\hat{\mathcal{A}}$. The set of all approximation sets is denoted as Ω.* □

[4] A formula is called satisfiable if it takes at least one true value in some interpretation.

Definition 24 (Quality indicator (metric)): *A h-ary quality indicator is a function* $\mathcal{I}_i : \Omega^{\dot{h}} \mapsto \mathbb{R}$*, which assigns each vector* $(\hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2, \cdots, \hat{\mathcal{A}}_{\dot{h}})$ *of* $\dot{h}$ *approximation sets a real value* $\mathcal{I}_i(\hat{\mathcal{A}}_1, \cdots, \hat{\mathcal{A}}_{\dot{h}})$*.* □

The rest of this section describes the complete set of dominance relationships, attainment function, quality indicators, the ten MOEA metrics, and the Kruskal Wallis/Mann-Whitney Test. The following definitions are used to prepare the reader for some of the terminology used within this chapter. First, the approximate set is defined in Definition 23. The approximate set is a set of solutions (PF$_{known}$) found by an MOEA when solving any MOP. The approximate set represents a set of solutions and the associated objective values that are evaluated by a quality indicator and/or metrics used to determine how good the MOEA is at solving that MOP under test. The second and third definitions describe these quality indicators (Definition 24) and a comparison method using the quality indicators (Definition 25). Finally, compatibility and completeness is defined in Definition 26 on page 131. To be compatible and complete a comparison, $\ddot{\mathcal{C}}$, must, when operating on the approximate sets A and B; first, be able to describe or measure the difference between approximation sets (*i.e.*, $\ddot{\mathcal{C}}$(A,B) $\Rrightarrow$ A ▶ B) and, second, if given a difference be able to indicate the order within the comparison (*i.e.*, A ▶ B $\Rrightarrow$ $\ddot{\mathcal{C}}$(A,B)).

Definition 25 (Comparison method): *Let* $A, B \in \Omega$ *be two approximate sets,* $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2, \cdots, \mathcal{I}_j)$ *a combination of quality indicators, and* $\boldsymbol{E}$:$\mathbb{R}^j$x$\mathbb{R}^j \mapsto \{false, true\}$ *a Boolean function which takes two real vectors of length j as arguments. If all indicators in* $\mathcal{I}$*, the comparison method* $\ddot{\mathcal{C}}_{\mathcal{I},\boldsymbol{E}}$ *defined by* $\mathcal{I}$ *and* $\boldsymbol{E}$ *is a Boolean function* *of the form*

$$\begin{aligned} \ddot{\mathcal{C}}_{\mathcal{I},\boldsymbol{E}} &= \boldsymbol{E}(\mathcal{I}(A), \mathcal{I}(B)) \qquad (48) \\ &= \boldsymbol{E}(\mathcal{I}(A_1, \cdots, A_{\acute{i}}), \mathcal{I}(B_1, \cdots, B_{\acute{j}})) \\ &= \boldsymbol{E}(\quad \{\mathcal{I}_{\mathbf{1}}(A_1, \cdots, A_{\acute{i}}), \cdots, \mathcal{I}_{\mathbf{j}}(A_1, \cdots, A_{\acute{i}})\}, \\ &\qquad \left\{\mathcal{I}_{\mathbf{1}}(B_1, \cdots, B_{\acute{j}}), \cdots, \mathcal{I}_{\mathbf{j}}(B_1, \cdots, B_{\acute{j}})\right\} \quad) \end{aligned}$$

□

4.3.1 Dominance Relations. To begin, see Table 9 on page 133 for the definition of new dominance symbols used within this section. Let there exists the MOP having k objective functions: $\mathbf{F}^1 = \{f_1(\vec{x}), \cdots, f_k(\vec{x})\}$. $\mathbf{F}^1$ is to be minimized. Assume also that each objective function assigns every solution $\vec{x}$ in the search space Ω a real value $z_i = f_i(x)$ reflecting the merit according to the i^{th} criteria for a particular solution $\vec{x}$. Thus, every $\vec{x} \in \Omega$ is mapped to a vector $\vec{z} = \{z_1, \cdots, z_k\} \in \Omega$. Accordingly, the approximation set or PF$_{known}$, is defined in Definition 23 on page 129.

Ideally, selection of quality indicators that are both compatible and complete is something that MOEA statisticians strive for when selecting metrics for comparing MOP optimization heuristics; however, Zitzler proved that one metric cannot possibly have this quality [239].

Definition 26 (Compatibility and Completeness): *Let ▶ be a binary relation on approximation sets. The comparison method $\ddot{\mathcal{C}}_{\mathcal{I},\boldsymbol{E}}$ is denoted as ▶-compatible if either for $A, B \in \Omega$*

$$\ddot{\mathcal{C}}_{\mathcal{I},\boldsymbol{E}}{:}A \blacktriangleright B$$

or for any $A, B \in \Omega$

$$\ddot{\mathcal{C}}_{\mathcal{I},\boldsymbol{E}}{:}B \blacktriangleright A$$

The comparison method $\ddot{\mathcal{C}}_{\mathcal{I},E}$ *is denoted as* $\blacktriangleright$*-complete if either for any* $A, B \in \hat{\Omega}$

$$A \blacktriangleright B{:}\ddot{\mathcal{C}}_{\mathcal{I},E}$$

or for any $A, B \in \Omega$

$$B \blacktriangleright A{:}\ddot{\mathcal{C}}_{\mathcal{I},E}$$

Thus, if a set of indicators, $\mathcal{I}$*, operating on two approximation sets, A and B, are said to be compatibility and completeness then the following is true:*

$$\ddot{\mathcal{C}}_{\mathcal{I},E}{:}A \blacktriangleright B \Leftrightarrow A \blacktriangleright B{:}\ddot{\mathcal{C}}_{\mathcal{I},E} \;\; \textit{and} \;\; \ddot{\mathcal{C}}_{\mathcal{I},E}{:}B \blacktriangleright A \Leftrightarrow B \blacktriangleright A{:}\ddot{\mathcal{C}}_{\mathcal{I},E}$$

□

Furthermore, he showed the best one can possibly have is compatibly without any completeness (*i.e.*, $\succ\succ$-compatibility without any completeness, or $\ntriangleright$-compatibility in combination with $\triangleright$-completeness.). That means either a strong statements can be made (the evaluated individuals of A strongly dominates the evaluated individuals of B) for only a few pairs of evaluated members of A $\triangleright$ B; or weaker statements can be made (the evaluated individuals of A are not worse than the evaluated individuals of B (*i.e.*, A $\succeq$B or A $\parallel$B) for all pairs A $\succ$B [240]). See Table 9 on page 133 for definitions of symbols described within this paragraph.

Knowles presented a similar idea in his EMO 2005 tutorial on the performance assessment of stochastic multiobjective optimizers. Knowles recommends having quality indicators that are dominance compliant, $\triangleright$, to guarantee that one algorithm's results are at least better than another before calling that algorithm itself better. This test is presented within Table 9 and called the *Better* relation. It is from Knowles' research that the quality indicators are chosen to compare MOMGA-II and MOMGA-IIa. Indicators are chosen to reduce the dimension of approximation

Table 9: Dominance relations on objective vectors and approximation sets when working with compatibility and completeness. For a graphic illustration of how these relations are used, please refer to Figure 23 on page 134.

Relation	Objective Vectors	
Strictly Dominates	$f_\tau(\vec{x}^1) \succ\succ f_\tau(\vec{x}^2)$	$\forall_{i \in \mathcal{F}}$, ($f_\tau(\vec{x}^1)$ is better than $f_\tau(\vec{x}^2)$)
Dominates	$f_\tau(\vec{x}^1) \succ f_\tau(\vec{x}^2)$	$f_\tau(\vec{x}^1)$ is not worse than $f_\tau(\vec{x}^2)$ in all objs and better in at least one objective
Weakly Dominates	$f_\tau(\vec{x}^1) \succeq f_\tau(\vec{x}^2)$	$f_\tau(\vec{x}^1)$ is not worse than $f_\tau(\vec{x}^2)$ in all objs
Incomparable	$f_\tau(\vec{x}^1) \parallel f_\tau(\vec{x}^2)$	neither $f_\tau(\vec{x}^1)$ weakly dominates $f_\tau(\vec{x}^2)$ nor $f_\tau(\vec{x}^2)$ weakly dominates $f_\tau(\vec{x}^1)$

	Approximation Sets	
Strictly Dominates	$A \succ\succ B$	every $a^2 \in B$ is strictly dominated by at least one $x^1 \in A$
Dominates	$A \succ B$	every $x^2 \in B$ is dominated by at least one $x^1 \in A$
Better[a]	$A \triangleright B$	every $a^2 \in B$ is weakly dominated by at least one $x^1 \in A$ and $A \neq B$
Weakly Dominates	$A \succeq B$	every $x^2 \in B$ is weakly dominated by at least one $x^1 \in A$
Incomparable	$A \parallel B$	neither A weakly dominates B nor B weakly dominates A

[a] Indicates that the indicator is dominance compliant and the left side results w.r.t. the indicator is better than the right side

sets while respecting the dominance compliance. Next, the chosen quality indicators are given, as well as the reason for these indicators.

The hypervolume indicator, described in Section 4.3.2.8, can be used as the basis of a dominance compliant comparison [129, 239]. In fact, given the results of two algorithms, E and F, it is shown that all cases where E is better than F are detected by this indicator while respecting the dominance compliance. Also, suggested indicators are the epsilon indicator, R_2, and R_3 indicators, described in Sections 4.3.2.9 and 4.3.2.10. Each indicator is based on different preference information; therefore, using them all provides a range comparisons rather than just one.

4.3.2 Quality Indicators. The ten selected metrics/indicators used are discussed this section. Appendix I on page 395 presents other metrics that can be used when comparing MOEAs; however, these are selected to not be used due to measurement overlap. It should be noted here, there is also a convergence metric presented within the additional metrics appendix (Appendix I on page 395); however, convergence within this analysis means that the MOEA found each and every optimal PF_{true} vector available to be found using the limited word length of variables in the MOP.

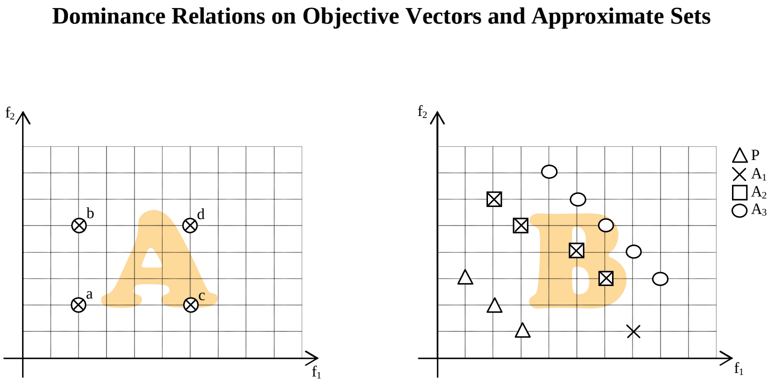

Dominance Relations on Objective Vectors and Approximate Sets

Figure 23: Graphic examples of minimization dominance relations on objective vectors and approximation sets. From the symbols defined in Table 9 and Figure A the following objective vector relationships hold: a $\succ$ b, a $\succ$ c, a $\succ$ d, b $\succ$ d, c $\succ$ d, a $\succ\succ$ d, a $\succeq$ a, a $\succeq$ b, a $\succeq$ c, a $\succeq$ d, b $\succeq$ b, b $\succeq$ d, c $\succeq$ c, c $\succeq$ d, d $\succeq$ d, and b $||$ c. Also from the symbols defined in Table 9 and Figure B the following approximation set dominance relationships hold for algorithm results A_1, A_2, and A_3 having a PF_{true} of P: $A_1 \succ A_3$, $A_2 \succ A_3$, $A_1 \succ\succ A_3$, $A_1 \succeq A_1$, $A_1 \succeq A_2$, $A_1 \succeq A_3$, $A_2 \succeq A_2$, $A_2 \succeq A_2$, $A_3 \succeq A_3$, $A_1 \triangleright A_2$, $A_1 \triangleright A_3$, and $A_2 \triangleright A_3$.

4.3.2.1 Error Ratio (ER): The Error Ratio (ER) metric reports the number of vectors in PF_{known} that are not members of PF_{true}. This metric requires that the researcher knows PF_{true}. Mathematically, this metric is represented in Equation 49:

$$\mathbf{ER} \triangleq \frac{\sum_{i=1}^{|\mathrm{PF}_{known}|} e_i}{|\mathrm{PF}_{known}|} \tag{49}$$

where e_i is a zero when the i^{th} PF$_{known}$ vector is an element of PF$_{true}$ or e_i is one if the i^{th} vector is not an element of PF$_{true}$. [34]

So when $ER = 0$, the PF$_{known}$ is the same as PF$_{true}$; but when $ER = 1$, this indicates that none of the points in PF$_{known}$ are in PF$_{true}$. A lower ER is better.

4.3.2.2 Generational Distance (GD): The Generational Distance (GD) reports how far, on average, PF$_{known}$ is from PF$_{true}$ [34, 217]. This metric requires that the PF$_{true}$ be known. It is mathematically defined in Equation 50.

$$GD \triangleq \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{|\mathrm{PF}_{known}|} \tag{50}$$

where $|\mathrm{PF}_{known}|$ is the number of vectors in PF$_{known}$, $p = 2$, and d_i is the Euclidean phenotypic distance between each member, i, of PF$_{known}$ and the closest member in PF$_{true}$ to that member, i. When $GD = 0$, PF$_{known}$ = PF$_{true}$.

4.3.2.3 Hyperarea and Ratio (HA,HR): The hyperarea (hypervolume) and hyperarea ratio metric is the area of coverage of PF$_{known}$ with respect to the objective space [34, 241] for a two-objective MOP. This equates to the summation of all the rectangular areas, bounded by some reference point and $(f_1(\vec{x}), f_2(\vec{x}))$. Mathematically, this is described in Equation 51:

$$\mathbf{HA} \triangleq \left\{ \bigcup_i area_i | vec_i \in \mathrm{PF}_{known} \right\} \tag{51}$$

where vec_i is a non-dominated vector in PF$_{known}$ and $area_i$ is the area between the origin and vector, vec_i. It is important to note that if PF$_{known}$ is not convex, the results may be misleading [217]. It is assumed that the reference point for the hyperarea is the minimum value for each objective. Note, the Hypervolume (HV) and hyperarea measurements are similar, except the HV can be used with dimension above two.

Mathematically, the hyperarea ratio metric definition is shown in Equation 52:

$$\mathbf{HR} \triangleq \frac{HA_1}{HA_2} \tag{52}$$

where HA_1 is the PF$_{known}$ hyperarea and HA_2 is the hyperarea of PF$_{true}$. Implementation of the hyperarea metric is considered only for maximization MOPs because the MOMGA-II and MOMGA-IIa only solve maximization MOPS[5]. Thus, **HR** values less than one indicate a found Pareto front that is not as good as the true Pareto front. When HR equals one, then PF$_{known}$ = PF$_{true}$. Finally, this metric requires that the researcher knows PF$_{true}$.

4.3.2.4 Spacing (S): The spacing (S) metric numerically describes the spread of the vectors in PF$_{known}$ [34]. This metric measures the distance variance of neighboring vectors in PF$_{known}$. Equations 53 and 54 define this metric.

$$\mathbf{S} \triangleq \sqrt{\frac{1}{|\mathrm{PF}_{known}| - 1} \sum_{i=1}^{|\mathrm{PF}_{known}|} (\bar{\mathbf{d}} - \mathbf{d_i})^2} \tag{53}$$

and

$$\mathbf{d_i} = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|) \tag{54}$$

where $i, j = 1 \ldots, |\mathrm{PF}_{known}|$, $\bar{\mathbf{d}}$ is the mean of all $\mathbf{d_i}$. When $S = 0$, all members are spaced evenly apart. Note that this becomes important in the deception problems where all Pareto front vectors are equally spaced. This metric does *not* require the researcher to know PF$_{true}$.

[5] All problems coded for the MOMGA-II and MOMGA-IIa within this work are considered maximization problem. If there is an objective function that requires minimization, the negative of the objective function values is used (*i.e.*, if $f_i(\vec{x})$ is a minimization objective then $\hat{f}_i(\vec{x})$ is substituted for $f_i(\vec{x})$ where $\hat{f}_i(\vec{x}) = f_i(\vec{x})$.)

4.3.2.5 Overall Non-dominated Vector Generation (ONVG): The Overall Non-dominated Vector Generation (ONVG) measures the total number of non-dominated vectors found during MOEA execution and is defined as:

$$\mathbf{ONVG} \triangleq |\mathrm{PF}_{known}| \tag{55}$$

4.3.2.6 Overall Non-dominated Vector Generation Ratio (ONVGR): Overall Non-dominated Vector Generation Ratio (ONVGR) measures the ratio of the total number of non-dominated vectors found PF_{known} during MOEA execution to the number of vectors found in PF_{true}. Coello [34] defines this metric as shown in Equation 56:

$$\mathbf{ONVGR} \triangleq \frac{|\mathrm{PF}_{known}|}{|\mathrm{PF}_{true}|} \tag{56}$$

When $ONVGR = 1$,this states only that the same number of points have been found in both PF_{true} and PF_{known}. It does not infer that $\mathrm{PF}_{true} = \mathrm{PF}_{known}$. This metric requires that the researcher knows PF_{true}.

4.3.2.7 Maximum Pareto Front error (ME). The Maximum Pareto Front error (ME) measures how well a set of vectors compares to another. More specifically, it measures the largest minimum distance between each vector in PF_{known} and the corresponding closest vector in PF_{true}. Equation 57 presents this metric mathematically.

$$\mathbf{ME} \triangleq max_j \left\{ \left\{ min_i \left(\sum_{k=1}^{m} |f_k^i(\vec{x}) - f_k^j(\vec{x})|^p \right)^{\frac{1}{p}} \right\} \right\} \tag{57}$$

where $i = \{1, \cdots, |\mathrm{PF}_{known_1}|\}$ and $j = \{1, \cdots, |\mathrm{PF}_{known_2}|\}$ index vectors in PF_{known} and PF_{true} respectively. A resultant of 0 indicates $\mathrm{PF}_{known} \subseteq \mathrm{PF}_{true}$. Any other resultant value indicates that at least one vector of PF_{known} is not in PF_{true}.

Next, the four newer quality indicators are discussed.

4.3.2.8 Hypervolume (HV). The hypervolume indicator is defined as the area of coverage of PF_{known} with respect to the objective space [34, 241] for a two-objective MOP. This equates to the summation of all the rectangular areas, bounded by some reference point and $(f_1(\vec{x}), f_2(\vec{x}))$. Mathematically, this is described in Equation 58:

$$\mathbf{HV} \triangleq \left\{ \bigcup_i vol_i | vec_i \in \mathrm{PF}_{known} \right\} \tag{58}$$

This indicator is the same as the hyperarea metric discussed above, but this indicator does go beyond two dimensions and substitutes vol_i for the $area_i$ in Equation 58. Figure 24.A illustrates how the hyperarea is calculated for a minimization MOP from two approximation sets, A and B. All MOPs are translated to maximization MOPs if not already defined as such; therefore, the areas are summed up from the bottom left.

4.3.2.9 ϵ-indicator. Given two approximate sets, A and B, this ϵ-indicator measures the smallest amount, ϵ, that must be used to translate the set, A, so that every point in B is covered. Figure 24.B illustrates how far A must move to cover B.

4.3.2.10 R_R Indicators. The final two indicators used are the R_2 and R_3 utility indicators [98]. Note, there is a third indicator, R_1, in this same class; however it is not utilized within this analysis (see Appendix I, Section I.1 on page 395 for more information on the R_1 indicator). The utility, $u(A, \tilde{\lambda})$, of the approximation set A, on scalarizing vector, $\tilde{\lambda}$, is the minimum distance of a point in the set, A, from the reference point. Equations 59 and 60 mathematically define these two indicators.

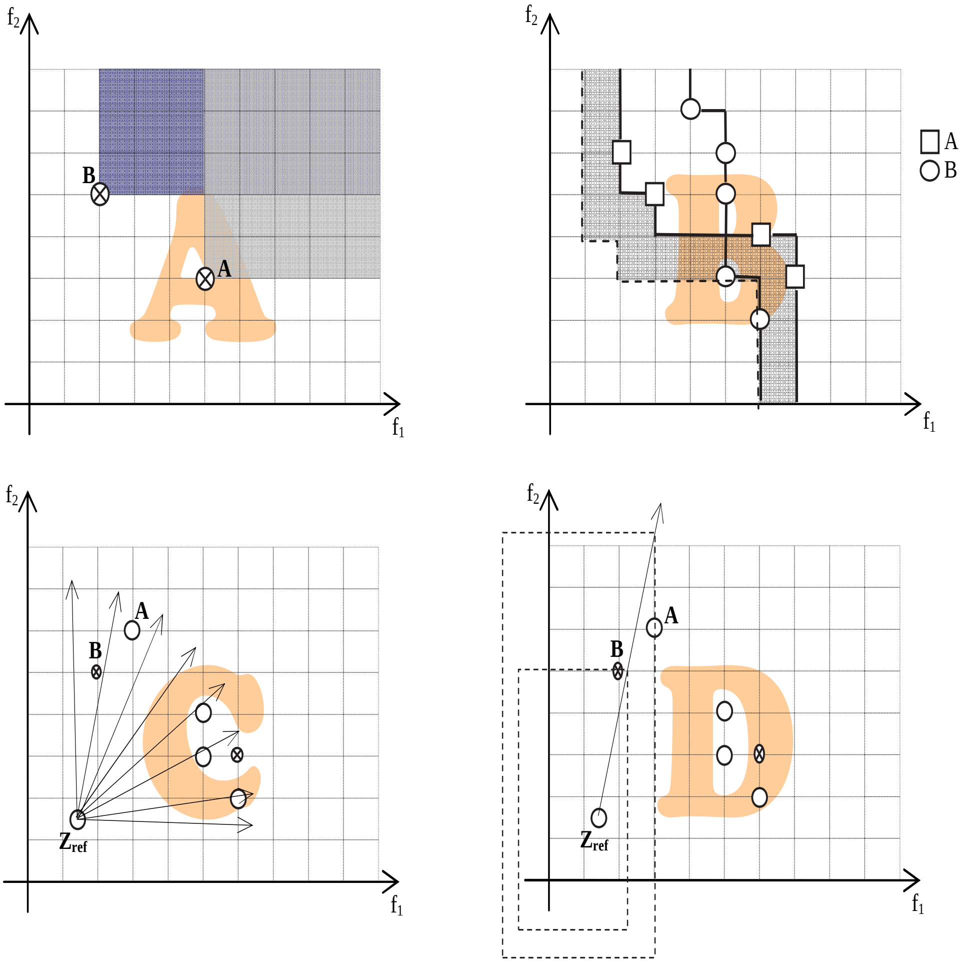


Figure 24: Illustrated in this figure are the hypervolume, R_2, R_3 and ϵ indicators as [129] described in the presentation at EMO 2005. Graphic "A" indicates a minimization MOP and an example of how the hypervolume is calculated. Graphic "B" shaded area indicates how the epsilon indicator calculates how far Pareto front A must move in each objective to cover Pareto front B (*i.e.*, How far must the vectors resulting from evaluating individuals of A be moved to dominate the vectors resulting from evaluating individuals of B in all objectives). Finally, graphics "C" and "D" illustrate how the utility functions of R_2 and R_3 are rendered. Graphic "C" illustrates how the vectors are evenly spread out from the worst reference point to the best reference point. Graphic "D" illustrates the difference is calculated with respect to each vector.

$$\mathcal{I}_{R2} = \frac{\sum_{\tilde{\lambda}\in A} u(\tilde{\lambda}, B) - u(\tilde{\lambda}, A)}{|\tilde{\lambda}|} \qquad (59)$$

$$\mathcal{I}_{R3} = \frac{\sum_{\tilde{\lambda}\in A} [u(\tilde{\lambda}, B) - u(\tilde{\lambda}, A)]/u(\tilde{\lambda}, B)}{|\tilde{\lambda}|} \qquad (60)$$

Graphically, R_2 and R_3 are illustrated in Figure 24.C and 24.D. These utility functions, u, require a reference point and a user-specified number of scalarizing vectors, $\tilde{\lambda}$. Vectors are uniformly distributed across the objective space. The distance of the point (in each set) that is closest to the reference point is measured and the differences in these distances are added up. In order to obtain an indicator from these two indicators, the set, B, is replaced with a reference set containing the true Pareto front points, $\mathtt{R}$. These indicator functions then effectively measure the difference in the mean distance of the attainment surfaces A and $\mathtt{R}$ from a user-defined reference point.

Table 10 lists the seven MOEA metrics and three indicators used when comparing MOEAs. The table indicates whether each metric/indicator requires PF$_{true}$ and explicitly compares results from one generation to another. Discussed next is a more natural metric requiring no calculation because it is a visual comparison between two approximation sets.

Table 10: Summary of the ten MOEA Metrics/Indicators used in comparing MOEAs.

	Metric Name	PF$_{true}$ required?	Generational Metric?
1	Error Ratio (ER)	Yes	No
2	Generational Distance (GD)	Yes	Yes
3	Hyperarea Ratio (HR)	Yes	No
4	Spacing (S)	No	No
5	Overall Non-dominated Vector Generation (ONVG)	No	Yes
6	ONVG Ratio (ONVGR)	Yes	Yes
7	Max PF error	Yes	No
$\mathcal{I}1$	ϵ indicator	No	No
$\mathcal{I}2$	Utility R$_2$ indicator	Yes/No	No
$\mathcal{I}3$	Utility R$_3$ indicator	Yes/No	No

4.3.3 Visualization. Visualization is considered to be one of the elementary ways to distinguish the difference between two approximation sets. When using this technique the researcher visually looks at the graphically representation of the

PF$_{known}$ and PF$_{true}$ set and determined if the results are good/bad or indifferent. For example, graphical analysis can help the researcher determine the cardinality of the set, total number of disjoint fronts, and structure of the front. Advantages of this method are many and can be concluded faster than statistically analyzing the approximation sets.

Many MOEA researchers recognize that visualization of MOEA results provides an easy mechanism to see the general MOEA performance when compared to another reference set. A more detailed analysis using other metrics, like the ones selected within this chapter, is necessary to statistically compare the performance of multiple MOEAs. One disadvantage to using visualization techniques is that as the dimensionality increases so does the ability to visually see difference between approximation sets. Since three dimensions is typically the maximum that one can easily visualize (3D is also difficult to recognize differences), it is suggested that visual analysis is limited to three objectives.

This concludes the study of MOEA metrics and selection; next, a discussion of non-parametric statistics is presented as a way to compare these chosen metrics when applied to the approximation sets found by the two MOEAs under test.

4.3.4 Attainment Function.

The attainment function approach gathers statistic data on generational results from a MOEA. The data provides detailed information about how and where the performance difference occurs between two MOEAs. Since the concern of this dissertation is focused on MOEA performance in terms of efficiency and effectiveness and not in where the performance differences lie, the attainment function is not considered within this document. However, it would be a mistake not to mention such a function. Next, the non-parametric statistical test used to compare metrics is discussed.

4.3.5 Non-Parametric Statistics (Analysis of Variance).

Required for comparisons of MOEA performance is a non-parametric inference test because the distribution of the population for metric results is unknown. Furthermore, if the population for metric results does turn out to be a normal distribution, other methods may be more accurate at deciding if there is a difference; however, the non-parametric statistics are inaccurate only in errors on the side of caution. Thus, there is no error made if the nonparametric statistic concludes that there is a difference between the two algorithms.

There are many statistical tests that can be used when comparing if two or more algorithms are different (better or worse) from one another. One of the most common non-parametric tests is the Wilcoxon Rank-Sum (Mann-Whitney) test for two independent samples. The more general form of this test is called the Kruskal-Wallis Statistic where h independent samples can be compared. This statistical test is performed on each comparable metric using gathered experimental data. The next section discusses how to apply this nonparametric statistic.

The Kruskal-Wallis H test (KWtest) is the main statistical method used in the determination if two samples are from the same population. An alternative to the one-way independent-samples Analysis of Variance (ANOVA) is the Kruskal Wallis Test. This test is primarily used when no knowledge of the type of distribution is known; however, it can be shown that the sampling distribution of H is nearly a chi-squared[6] distribution with $h-1$ degrees of freedom, given that $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_h$ sum to at least 5 [206]. In all KWtests accomplished, both the Chi-squared-statistic and the F-statistic[7] are evaluated. The definition of the Kruskal-Wallis H Test is the following:

$$H = \frac{12}{\mathcal{N}(\mathcal{N}-1)} \sum_{i=1}^{h} \frac{\tilde{R}_i^2}{\mathcal{N}_j} - 3(\mathcal{N}+1) \tag{61}$$

[6] Under the null hypothesis that the positive and negative values are equally likely, the test statistic follows the chi-square distribution with $h-1$ degree of freedom.

[7] The F test assumes known population variances of approximately normal distribution and the population variances are homogeneous.

- Given

  h sample sizes $\mathcal{N}_1, \mathcal{N}_2, \ldots \mathcal{N}_h \therefore \mathcal{N} = \sum_{i=1}^{h} \mathcal{N}_i$

  h samples are ranked together according to size, therefore the ranks are $\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_h$

Upon calculation of H using Equation 61, this value, H, is treated as though it were a value of chi-square sampling distribution with the degrees of freedom (df) $= h - 1$. This nonparametric method for analysis of variance is used for a one-way classification, or one-factor experiments, and generalizations can be made [206].

4.4 Other Metrics

Certainly other metrics can be used because there are 30 experimental tests. The mean and standard deviation is collected on each metric for each set of results found by both MOEAs. The central limit theorem allows for the assumption that, after 30 measurements are taken, a normal distribution for a particular measurement can be assumed (see footnote 16 on page 31). Once a normal distribution is assumed either a student-t or z test may be performed to compare results. These tests were not accomplished within this chapter because the KWtest errs on the side of caution and can be used in their place.

4.5 Applying the Statistical Methods

The statistical method used to determine if two metrics are different than one another is the following: 30 experiments are performed (using different random seeds for each), each metric is applied to the final results found by each MOEA, then a non-parametric test is applied to the data from each MOEA for each metric. Finally, the different metrics are listed as being better or worse for each MOEA and a decision is made about if the MOEA is better or worse according to how many metrics statistically record a difference between the algorithms. Clearly, the metric determining how many true Pareto front points found by an algorithm outweighs

Table 11: Table summarizing Test Suite MOPs and associated characteristics.

	Genotype							**Phenotype**				
Function	Connected	Disconnected	Symmetric	Scalable	Solution Type(s)	Objectives	Side Constraints	Geometry	Connected	Disconnected	Concave	Convex
VL1:MOP 1	X		X		1R	2	0	Curve	X			X
VL2:MOP 2	X		X		2R	2	0	Curve	X		X	
VL3:MOP 3		X			2R	2	2	Curves		X		
VL4:MOP 4		X	X	X	$\hat{n}$R	2	0	Curve		X	X	
VL6:MOP 6		X			2R	2	0	Curves		X		
VLC4:Tanaka		X			2R	2	2+2S	Curves		X		
DTLZ3			X		3R	3		Surface		X	X	

other metrics. However, the final decision on each MOP for data gathered from each MOEA test is different and must be analyzed separately.

The PF$_{true}$ set is obtained by running an exhaustive search on a selected chromosome size used for experiments for each test suite MOP. Chromosome sizes, ℓ, were all equal to or less than 30, $\ell \leq 30$, due to the computational time to exhaustively search a search space having more then 30 bits. In some cases, careful ranging of variables was heeded to ensure the PF$_{true}$ could be obtained by both the exhaustive search and the MOEAs. Metrics used the PF$_{true}$ set found by the exhaustive search to compare generational MOEA results using the ten metrics described earlier. Metric calculations are embedded within both MOEAs except for the three extra metrics added from research findings by Knowles [129]. These metrics were run separately on results, using performance metric code from Zitzer [129]. Efficiency was also tested using wall clock time to completely solve the MOP. An MOEA solved the MOP when PF$_{known}$ = PF$_{true}$. At the time that this occurred during a search, this time is recorded as the convergence time. The convergence time is compared for the efficiency metric results.

4.5.1 Reading Table Results. Metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa are listed in several tables within this chapter. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different. Upon an indication of different, $\notin$, the researcher must go back to the visualization of the statistical data to determine which is indeed better. Unless otherwise specified, it is found that $\notin$ indicates that the MOMGA-IIa performs better than MOMGA-II.

4.6 Experimental Hypothesis

Each experiment begins with the hypothesis that the MOMGA-IIa outperforms the MOMGA-II effectiveness-wise. As for the efficiency of the algorithms, it is thought that the MOMGA-IIa must be less efficient. However, in certain circumstances, the MOMGA-IIa becomes more efficient than the MOMGA-II. Each MOP has a section describing and analyzing results from the experiments.

4.7 MOEA Test Suites

This section describes the test MOPs used to test the validity that the MOMGA-IIa is indeed a better algorithm than the MOMGA-II. This is the hypothesis. Table 11 lists the names and characteristics of the MOPs picked for evaluation. First, it is expected that effectiveness should increase and it is desired, but not totally expected, that efficiency (faster convergence) is also observed. Table 12 lists all MOP test suite functions and associated characteristics and constraints.

The MOMGA-II and MOMGA-IIa are tested and compared against each selected test suite MOP to illustrate if there is a statistical difference between the two algorithms on MOPs of different characteristics. The following sections describe each test suite MOP experiment and results. Although algorithm development is accomplished on Cluster 1, 2 and 3, only Cluster 1 is used for MOP test suite experiments (see Table 13 on page 147 for hardware configurations of these Clusters).

Table 12: MOEA Test Suite Functions

MOP	Definition	Constraints
VL1:MOP 1 P_{true} connected P_{true} convex	$F = (f_1(x), f)2(x))$, where $f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$-10^5 \leq x \leq 10^5$
VL2:MOP 2 P_{true} connected P_{true} concave, number of decision vars	$F = (f_1(\vec{x}), f2(\vec{x}))$, where $f_1(x) = 1 - exp\left(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2\right)$ $f_1(x) = 1 - exp\left(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2\right)$	$-4 \leq x_i \leq 4, i = 1,2,3$
VL3:MOP 3 P_{true} disconnected P_{true} disconnected (2 Pareto curves)	Max(F), where $F = (f_1(x,y), f_2(x,y))$ $f_1(x,y) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2]$ $f_2(x,y) = -[(x+3)^2 + (y+1)^2]$	$-\pi \leq x, y \leq \pi$ $A_1 = 0.5sin(1) - 2cos(1) + sin(2) - 1.5cos(2)$ $A_2 = 1.5sin(1) - cos(1) + 2sin(2) - 0.5cos(2)$ $B_1 = 0.5sin(x) - 2cos(x) + sin(y) - 1.5cos(y)$ $B_2 = 1.5sin(x) - cos(x) + 2sin(y) - 0.5cos(y)$

VL4:MOP 4
P_{true} disconnected
P_{true} disconnected
(3 Pareto curves) number of decision var scalable
	$F = (f_1(\vec{x}), f_2(\vec{x}))$
	$f_1(x) = \sum_{i=1}^{n-1}\left(-10e^{-0.2*\sqrt{(x_i^2 + x_{i+1}^2)}}\right)$
	$f_2(x) = \sum_{i=1}^{n}\left(\lvert x_i\rvert^a + 5sin(x_i)^b\right)$
		$-5 \leq x_i \leq 5, i = 1,2,3$
		a = 0.8,
		b=3

VL6:MOP 6 P_{true} disconnected P_{true} disconnected (4 Pareto curves) number of Pareto curves scalable	$F = (f_1(x,y), f_2(x,y)), where$ $f_1(x) = x,$ $f_2(x) = (1+10y) * [1 - \left(\frac{x}{1+10y}\right)^\alpha -$ $\frac{xsin(2\pi qx)}{1+10y}]$	$0 \leq x,\ y \leq 1,$ $q = 4,$ $\alpha = 2$
VLC4:Tanaka	$F = (f_1(x,y), f_2(x,y))$, where $f_1(x,y) = x,$ $f_2(x,y) = y$	$0 < x, y \leq \pi,$ $0 \geq -(x^2) - (y^2) + 1 + (acos(atan(\frac{x}{y})))$ $a = 0.1$ $b = 16$
DLTZ3	$Min(F)$ $f_1(x) = (1 + g(x_M))cos(\frac{x_1\pi}{2})cos(\frac{x_2\pi}{2})\cdots$ $\cdots cos(x_{M-2}\frac{\pi}{2})cos(x_{M-1}\frac{\pi}{2})$ $f_2(x) = (1 + g(x_M))cos(\frac{x_1\pi}{2})cos(\frac{x_2\pi}{2})\cdots$ $\cdots cos(x_{M-2}\frac{\pi}{2})sin(x_{M-1}\frac{\pi}{2})$ $f_3(x) = (1 + g(x_M))cos(\frac{x_1\pi}{2})cos(\frac{x_2\pi}{2})\cdots$ $\cdots sin(x_{M-1}\frac{\pi}{2})$ $\vdots \quad \vdots$ $\vdots \quad \vdots$ $f_{M-1}(x) = (1 + g(x_M))cos(x_1\frac{\pi}{2})$ $f_M(x) = (1 + g(x_M))sin(x_1\frac{\pi}{2})$ $g(x_M) = 100[\lvert x_M\rvert + \sum_{x_i \in x_M}(x_i - 0.5)^2 -$ $\cdots cos(20\pi(x_i - 0.5))]$	$0 \leq x_i \leq 1, for\ i = 1, 2, \cdots, n$

Both efficiency and effectiveness of both algorithms are studied. Each MOEA is run 30 times for good statistical analysis.

Each metric is analyzed separately and compared in both a regular statistical analysis and a Kruskal Wallis test. It is assumed that the Kruskal Wallis test is a better test to compare metrics for each algorithm because it is a non-parametric test. Zydallis and Van Veldhuizen both used seven statistical measures in determining algorithm performance comparison to another. Here, ten different metrics are used. The three added metrics are the R2, R3, and epsilon indicators [129]. Fonseca, Knowles, Thiele and Zitzler assumed that the three metrics along with the HR metric (part of the Zydallis and Van Veldhuizen set) are the only metrics necessary to determine if one algorithm is better than another. Each metric is talked about separately and conclusions are drawn based on separate metric statistical findings.

Table 13: System Configuration

	Cluster 1 (TAHOE)	**Cluster 2 (ASPEN)**
OS	Fedora Core 2	Redhat Linux 9.0
Processors	Dual Opteron 2.2 ghz	Athlon XP 3000+ 2.1ghz
Cache(L1 I,D/L2)	(64,64/1024)KB	(64,64/512)KB
Backplane	Gb Ethernet	Fast Ethernet
RAM	4 GByte	1 GByte
Switching	Crossbar Switch	Crossbar Switch
Disk I/O	RAID 5	RAID 5
Memory type	Distributed	Distributed
Node Specifics	48 node,2 CPUS/node	48 node,2 CPUS/node
	Cluster 3 (Polywells)	
OS	Redhat Linux 7.3	
Processors	Athlon XP 2800+ 2.0ghz	
Cache(L1 I,D/L2)	(64,64/512)KB	
Backplane	Gb Ethernet	
RAM	1 GByte	
Switching	Crossbar Switch	
Disk I/O	RAID 5	
Memory type	Distributed	
Node Specifics	16 node,1 CPU/node	

MOP 1 (VL1): MOP 1 is published by Van Veldhuizen and Lamont in 1998 [221] as an *easy* test suite MOP. It is also known as Schaffer's first (unconstrained) two-objective function. The characteristics of this MOP can be found in Table 11 on page 144. Settings for the MOMGA-II and MOMGA-IIa are kept the same with the exception of the use of more competitive templates for the MOMGA-IIa. Table 44 in Appendix A on page 272 lists all settings used for both MOEAs when solving MOP 1. As a general note, all parameter settings, including chromosome size indicating solution resolution, for each statistical experiment conducted between the MOMGA-II and MOMGA-IIa can be found in Appendix A on page 272.

Results for this MOP are not surprising in that both MOEAs solve this problem easily because this is the easiest of all test suite MOPs [34]. Figure 25 on page 149 illustrates the phenotypic results for these experiments. Note that the MOMGA-II has an unequal spacing of results, while the MOMGA-IIa has a good distribution and an equal spacing of results. This is not to say that MOMGA-II cannot solve this problem totally and have the equal spacing like what is found by MOMGA-IIa; but, when limited with the population sizing, BB search sizes, BB filter settings provided, and primordial and juxtapositional generations these results are observed.

Effectiveness: Visual results of the statistical analysis for the 30 runs on MOP 1 for each algorithm are illustrated in Figure 26. It is difficult to make a call on if one algorithm performs better than another according to this figure. The only two metrics glaringly different visually are the ONVG and ONVGR. The ONVG is based on $|\mathrm{PF}_{known}|$ and the ONVGR is the ratio of the $|\mathrm{PF}_{known}|$ to $|\mathrm{PF}_{true}|$ found. Moreover, the ONVGR may be a good metric to rely on for the analysis of algorithm goodness because it reflects what is illustrated when looking at the visual graphic in Figure 25 where MOMGA-II does not cover the Pareto front true points completely. However, this single metric is not on the qualified list of metrics

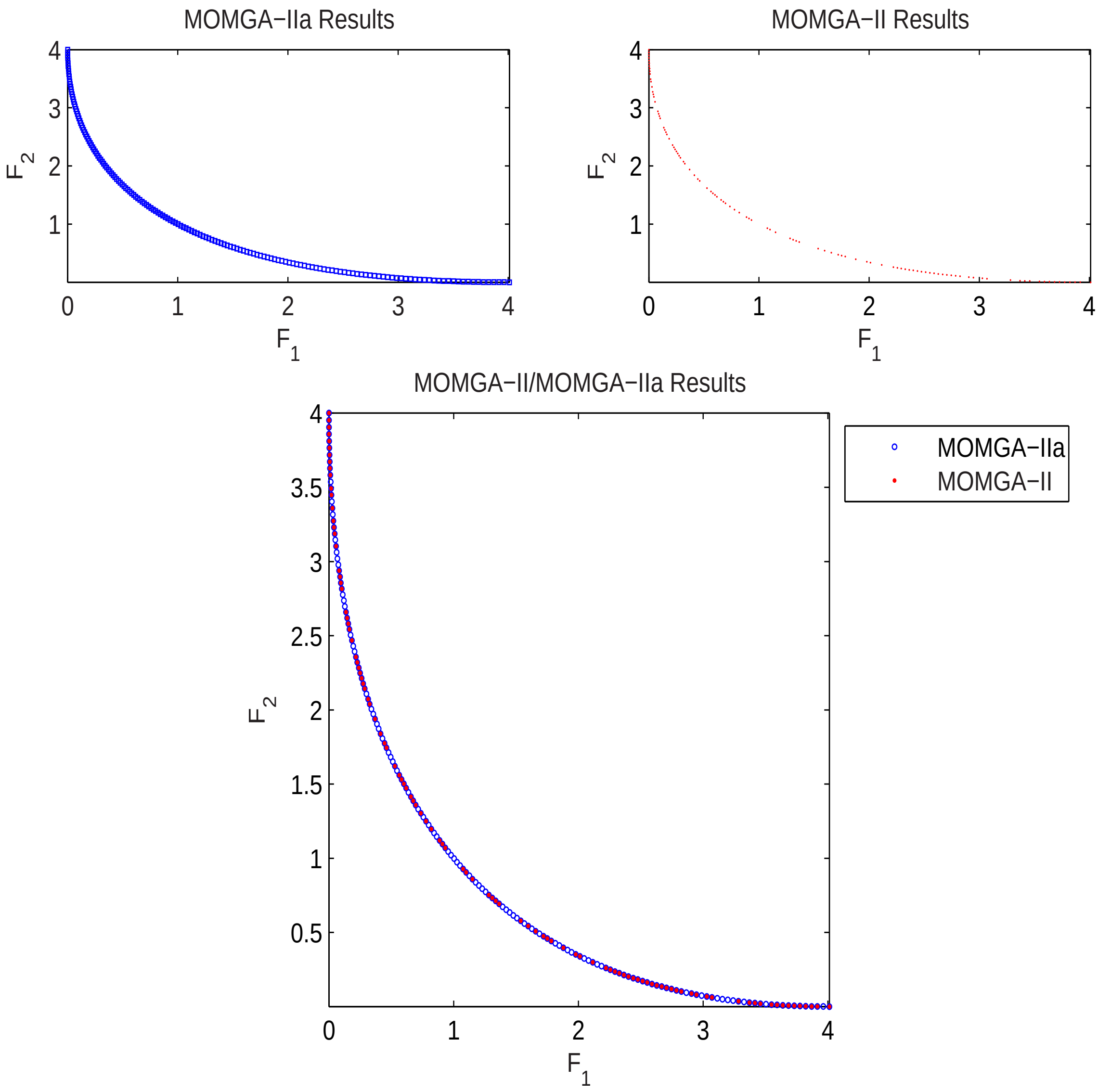


Figure 25: This figure illustrates typical results from running MOMGA-IIa (top left) and MOMGA-II (top right) using the same parameter setting except the added archive and number of competitive templates for MOP 1 (VL1). The bottom figure shows one graph overlaid on the other. Open circles represent optimal solutions evaluating to PF$_{true}$ vectors not found by MOMGA-II. The size differences between the upper and lower graphs are to allow for a better visual of missing PF vectors. These figures are created using Matlab's subplot function call and the bottom plot is considerably larger to allow for a closer look at the comparison of results for each MOEA.

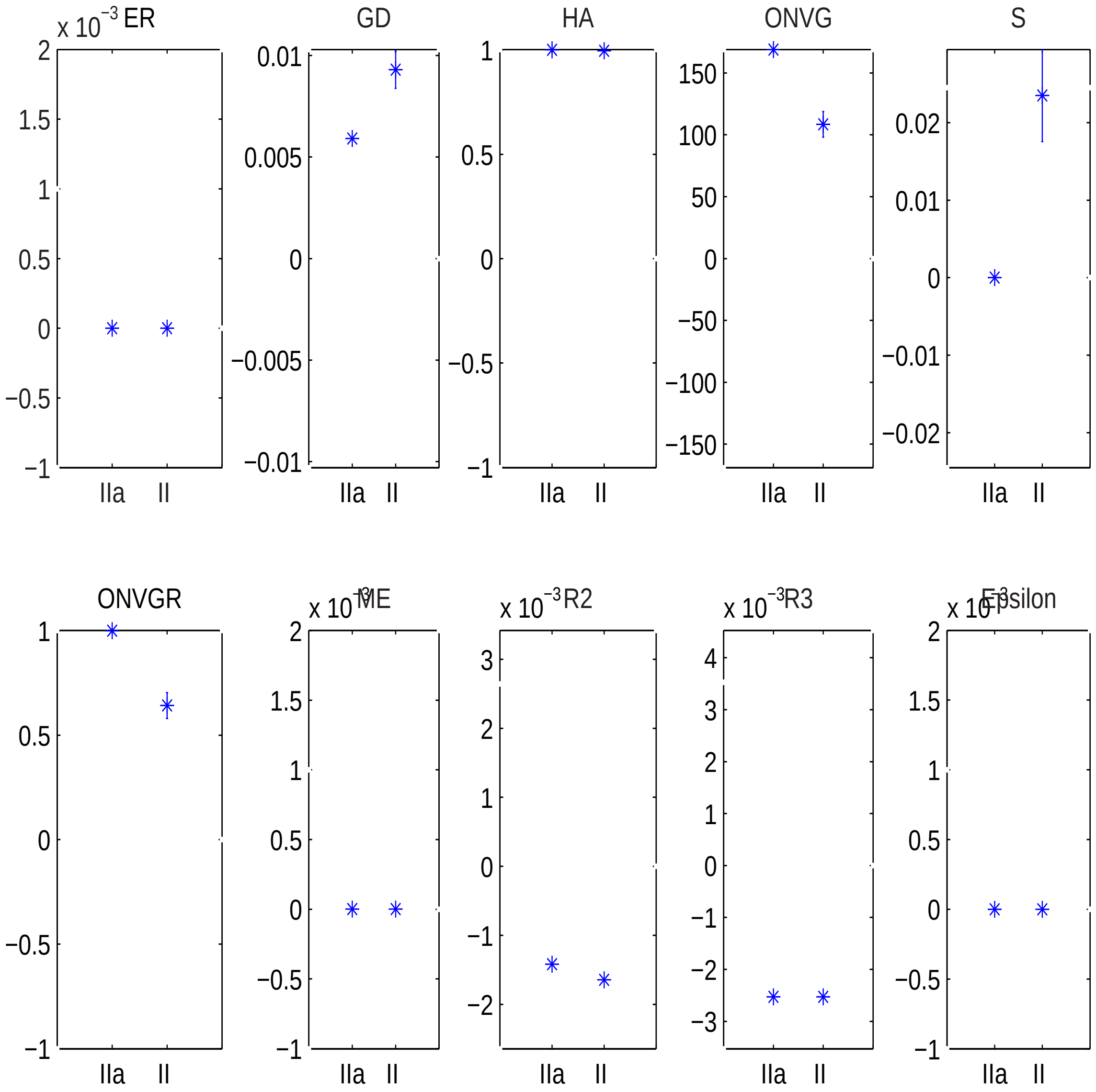


Figure 26: Illustrated are the statistical results of the MOMGA-II versus MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL1. Results indicate visually that these MOEAs perform similarly over all metrics except ONVG and ONVGR where the MOMGA-IIa found more PF_{true} vectors than MOMGA-II on MOP 1. Results for each MOEA and all metric have error bars on the mean, but in most cases the variance is too small to visualize on the plots.

to determine if one algorithm is better than another according to Fonsea, Knowles, Thiele and Zitzler [129].

Table 14: MOP 1 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\in$
F	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\in$

Kruskal Wallis results for each metric are listed in Table 14. It is concluded that too many of these metrics report that the algorithms are the same. Therefore, there is not enough evidence to say that one algorithm is better than another.

Efficiency: The final conclusion is that these two algorithms are not that much different in solving this problem. In fact, MOMGA-IIa takes ∼51 seconds to completely solve the problem while the MOMGA-II takes ∼26 seconds to come close to solving this problem with these algorithm settings.

Table 21 on page 166 lists MOEA timing for this MOP. This evaluation of algorithm timing is another point of evaluation for these two algorithms. The results of MOMGA-IIa at or just before the time MOMGA-II completes are conclusively not worse (or better) than the MOMGA-II. For MOP 1, MOMGA-IIa is more efficient than MOMGA-II because at the time the MOMGA-IIa finds all PF$_{true}$, its data is recorded as more effective for every metric except ER and ME, R3 and ϵ than the MOMGA-II's. Lastly, MOP 1 MOMGA-IIa BB size visualizations are presented in Figure 95 in Appendix D on page 313. Next, the results for test MOP 2 are discussed.

MOP 2 (VL2): This MOP is similar to MOP 1, but the input has three real variables (scalable) and the phenotype has a convex Pareto front curve vice MOP 1 has a single variable and concave Pareto front. MOP 2 is considered to be more difficult to solve than MOP 1. Parameter settings used for MOMGA-II and MOMGA-IIa when solving MOP 2 are listed in Appendix A in Table 45 on page 273.

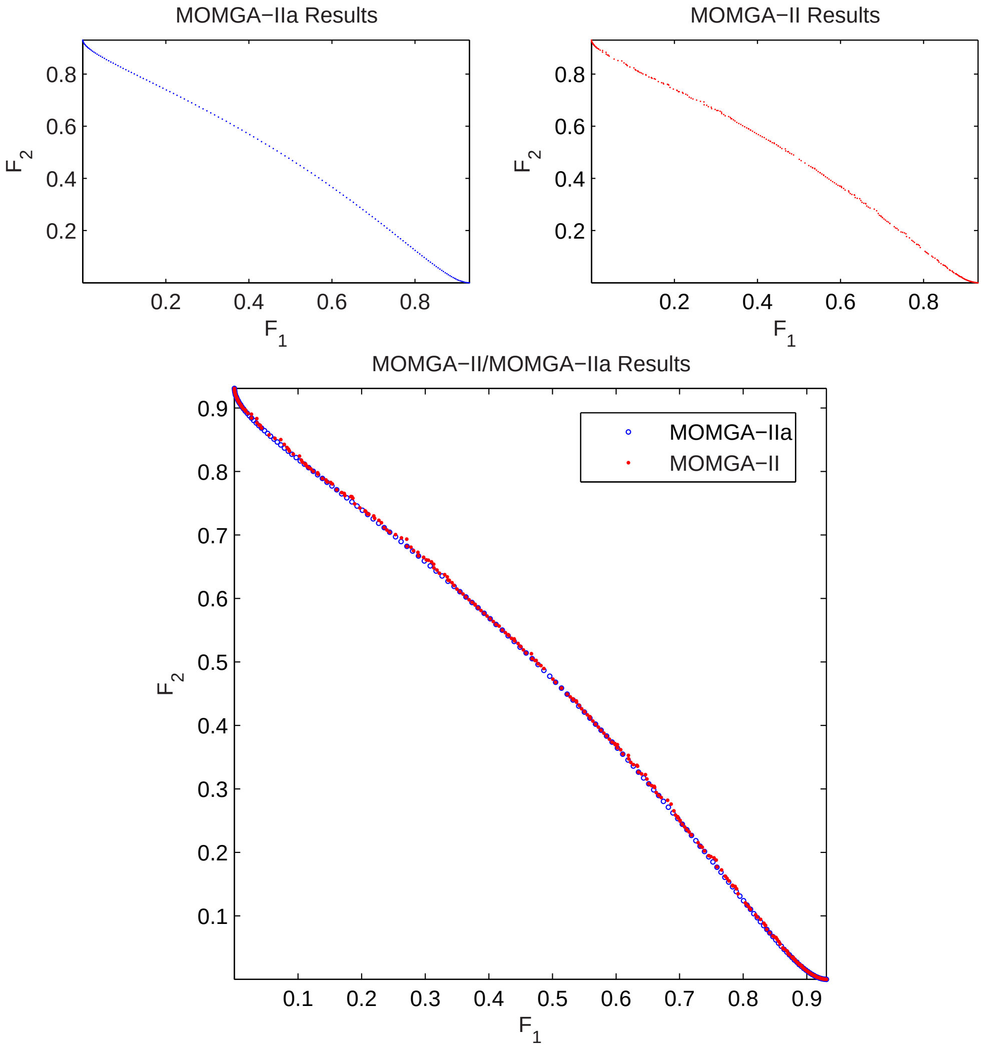


Figure 27: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa using the same parameter setting except the added archive and number of competitive templates on MOP 2.

Effectiveness: Figure 27 illustrates the results found for a typical MOMGA-IIa and MOMGA-II run on MOP 2. Statistical analysis of 30 runs for both algorithms can be found in Figure 76 in Appendix B on page 292. Once again, metrics ONVG and ONVGR stand out as being different. KWtest, listed in Table 15 on page 153, also reveals that in many of the metrics present there is a difference in algorithm results. In fact, metrics that are tested as different according to the KWtest are GD, HR, ONVG, ONVGR, R2, and R3.

Efficiency: Run times for both algorithms for this particular MOP are dissimilar. MOMGA-II puts up a ∼14 minute run time where as MOMGA-IIa clocks in at ∼320 minutes. These run times seem somewhat unbalanced because the algorithm must evaluate each individual more times according to the number of competitive templates added to the algorithm. In this particular case, MOMGA-IIa is not more efficient than MOMGA-II; however, it does find all the solutions evaluating to all PF$_{true}$ vectors upon completion. The final efficiency comparison is to evaluate the results of MOMGA-IIa at ∼14 minutes, when MOMGA-II completed. MOMGA-II and MOMGA-IIa performs, efficiency wise, similarly on this particular MOP. Table 21 on page 166 lists MOEA timing for this MOP.

Table 15: MOP 2 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\notin$	$\in$
F	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\notin$	$\in$

It is important to note for this MOP that, when using the same setting minus the modifications for MOMGA-IIa, MOMGA-IIa finds every solution evaluating to each and every PF$_{true}$ vector upon completion in every experiment and MOMGA-II does not. Finally, it is the conclusion that MOMGA-IIa does become more effective on MOP 2, but at the cost in overall efficiency. Lastly, MOP 2 MOMGA-IIa BB size visualizations are presented in Figure 96 in Appendix D on page 314. Next, MOP 3 results are discussed for both MOEAs.

MOP 3 (VL3): The third MOP visited in this investigation is Poloni's MOP (VL3). This is a maximization problem having two objectives with disconnected areas on the PF$_{true}$. Figure 28 illustrates results for both algorithms. Parameter settings used for the MOMGA-II and MOMGA-IIa when solving MOP 3 are found listed in Table 46 on page 274.

Effectiveness: Statistical results conclude that there is a difference in the MOEA results for this experiment. By visual observation (see Figure 77 on page 293) it is concluded that ER, ONVG, and ONVGR are different in favor of MOMGA-IIa. KWtest concludes that R2, R3 and ϵ indicator results are from the same population and are the same (see Table 16 on page 154). Otherwise, all other metrics are found to be different. Student-t testing also supports these results (see Figure 89 in Appendix B on page 305 for box plots of the applied student-t test).

Efficiency: MOMGA-II's runs took on average ∼3 hours to converge, while the MOMGA-IIa took ∼40 minutes to converge on all solutions that evaluate to each and every PF$_{true}$ vector. This indicates that the MOMGA-IIa actually is more effective and efficient at solving this particular MOP. Table 21 on page 166 lists MOEA timing for this MOP.

Table 16: MOP 3 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$
F	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$

In conclusion, the MOMGA-IIa is more effective and efficient when solving MOP 3 using the specified settings. Although many metrics appear to be the same, PF$_{true}$ tallies are quite different for these two algorithms, indicating that the

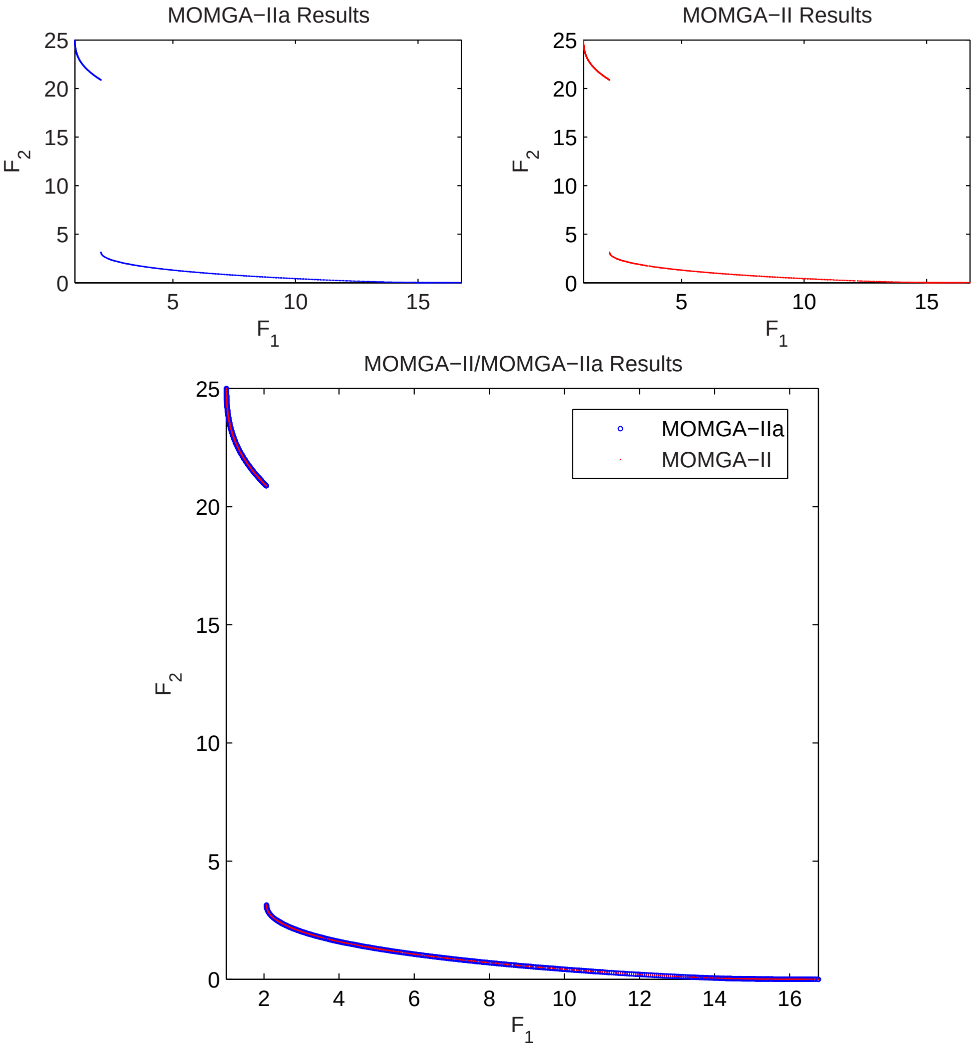


Figure 28: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa on MOP 3 using the same parameter setting except the added archive and number of competitive templates.

MOMGA-IIa is better. Lastly, MOP 3 MOMGA-IIa BB size visualizations are presented in Figure 97 in Appendix D on page 315. Next, MOP 4 (VL4) is discussed.

MOP 4 (VL4): Kursawe defined this bi-objective MOP to have several disconnected and non-symmetric areas in the phenotype space [31]. This is another scalable variable MOP. Parameter settings used for MOMGA-II and MOMGA-IIa when solving MOP 4 are listed in Table 47 on page 275. Typical resultant Pareto fronts from MOMGA-IIa and MOMGA-II are illustrated in Figure 29. Notice within the figure that MOMGA-II solutions evaluate to vectors that are not quite on the PF$_{known}$ vectors found by MOMGA-IIa. In addition, the MOMGA-II results are unevenly distributed, where MOMGA-IIa found all solutions evaluating to each and every vector in PF$_{true}$. Lastly, MOP 4 MOMGA-IIa BB size visualizations are presented in Figure 98 in Appendix D on page 316.

Effectiveness: Using visual statistical inferences from Figure 78 on page 294 GD, S, and R3 are quite similar. Table 17 lists the KWtest results where all metrics are found to not be from the same population; thus, are different. This concludes that final results for MOMGA-IIa are better than that of MOMGA-II.

Efficiency: Efficiency for MOMGA-IIa when solving this MOP is not as good as previous tests. In fact, the difference in overall run time is 11 times longer when comparing a complete MOMGA-IIa run to a MOMGA-II run. Fortunately, sampling the results of the MOMGA-IIa at the time that the MOMGA-II completes, it is found that the MOMGA-IIa results are still more effective.

After analyzing the algorithms at the time of completion of the MOMGA-II, it can be said that for MOP 4 (VL4), MOMGA-IIa is both more effective and efficient. Next, results from experiments for the MOMGA-IIa and MOMGA-II when solving MOP 6 are discussed.

MOP 6 (VL6): MOP 6, from Coello, Van Veldhuizen, and Lamont [34], is constructed using Deb's methodology using single-objective functions having de-

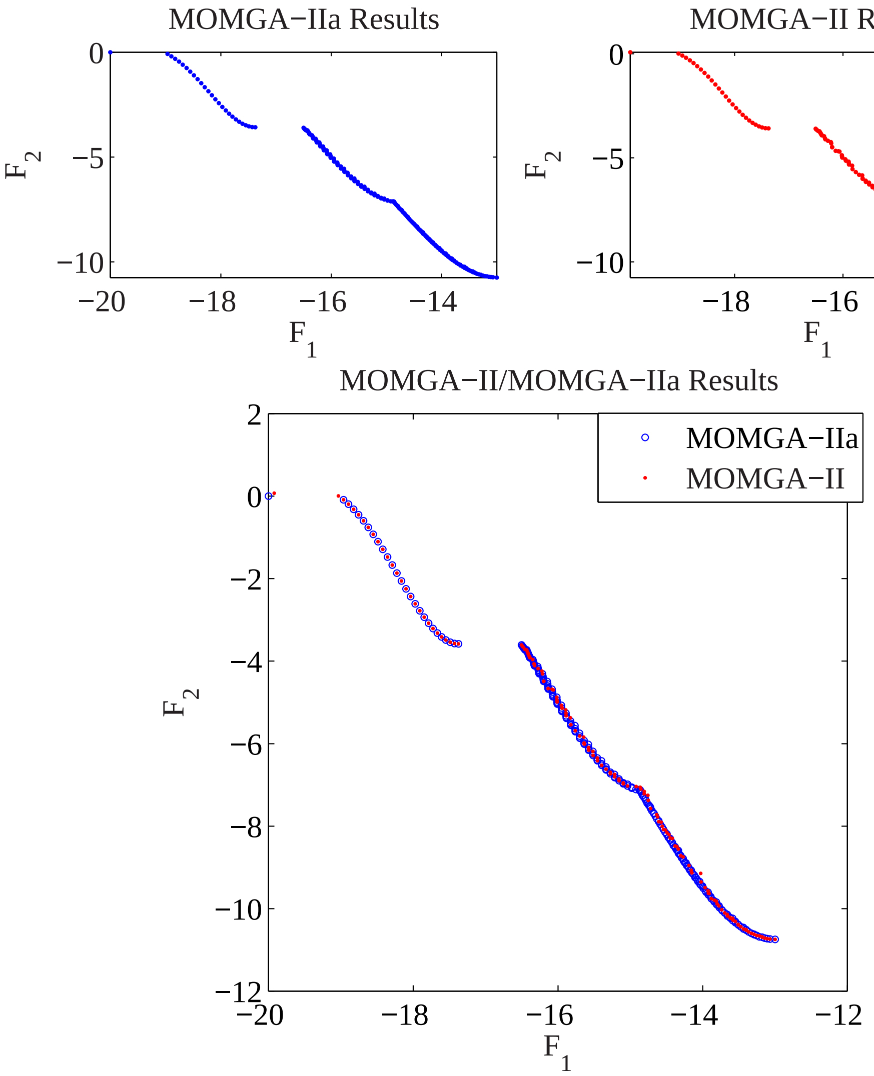


Figure 29: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa using the same parameter setting except the added archive and number of competitive templates on MOP 4.

sired characteristics [54, 61]. This MOP has four disconnected Pareto curves in the objective space and two variables for optimizing. Figure 30 illustrates a typical set of results from the MOMGA-IIa and MOMGA-II solving this MOP. Parameter settings used for the MOMGA-II and MOMGA-IIa when solving MOP 6 are listed in Table 48 on page 276.

Effectiveness: Visual analysis of statistical results of 30 experiments conclude that effectiveness results are *no* different (see Figure 79 on page 295). KWtest indicates similar results (see Table 18).

Efficiency: Table 21 on page 166 lists MOEA timing for this MOP. Both MOEAs solve this problem to completion ($\mathrm{PF}_{known} = \mathrm{PF}_{true}$); however, MOMGA-IIa converges before MOMGA-II in many cases. Although the effectiveness of both algorithms is the *same*, the MOMGA-IIa has an average running time that is more efficient than MOMGA-II, and the KWtest concluded that these run times are indeed not from the same population (different). Lastly, MOP 6 MOMGA-IIa BB size visualizations are presented in Figure 99 in Appendix D on page 317. Next, a MOP having side-constraints is discussed.

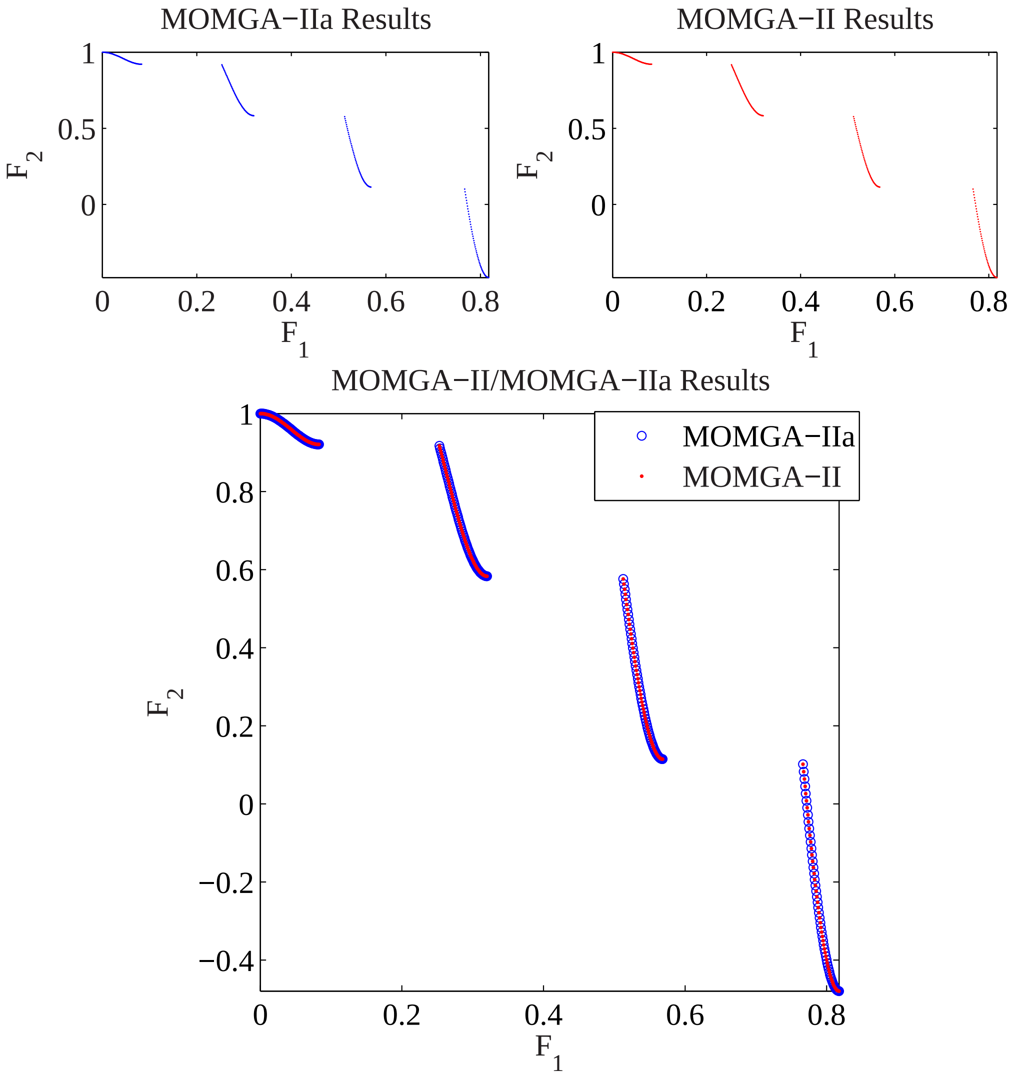
Tanaka (VLC4): The Tanaka MOP, also found as MOP-C4 (VLC4) within [34] is a side-constrained MOP having two objective functions with two non-linear constraints. Figure 31 illustrates an exhaustive search over the original Tanaka MOP. The Pareto front is darkened on the minimized edges of the objective space. Figure 32 illustrate the comparison of results found by both the MOMGA-IIa and MOMGA-II. Finally, Table 21 on page 166 indicates that the MOMGA-IIa finds these solutions at

Table 17: MOP 4 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
F	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$

Table 18: MOP 6 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$
F	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$



MOMGA−IIa Results

MOMGA−II Results

MOMGA−II/MOMGA−IIa Results

Figure 30: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa using the same parameter setting except the added archive and number of competitive templates on MOP 6.

a more efficient rate. Parameter settings used for the MOMGA-II and MOMGA-IIa when solving VLC4 can be found listed in Table 49 on page 277.

Effectiveness: Visual analysis of statistical data represented in Figure 81 on page 297 indicates that these two MOEAs perform similarly in effectiveness. Finally, KWtest indicates (results listed in Table 19) that these algorithms are similar

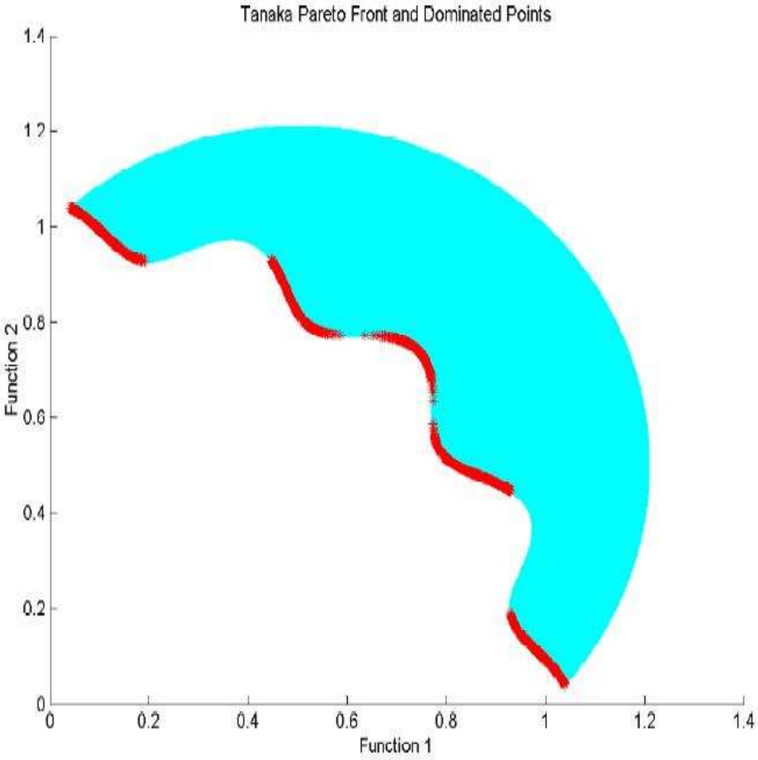


Figure 31: This figure illustrates search space and the Pareto front found for the Tanaka MOP. Darker areas represent the Pareto front.

only for R2, R3, and ϵ indicator metrics. In conclusion, the seven metrics used in previous AFIT MOEA studies actually indicate that these algorithms are different, but recently added MOEA metrics indicate that they are not different. By studying the data, MOMGA-IIa always finds all solutions evaluating to each and every PF$_{true}$ vector and, on average, MOMGA-II finds them, as well. Being that there is no one single metric that measures the overall effectiveness of a solution set when compared to another, it is concluded for the Tanaka MOP that these MOEAs perform similarly.

Table 19: Tanaka metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$
F	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$

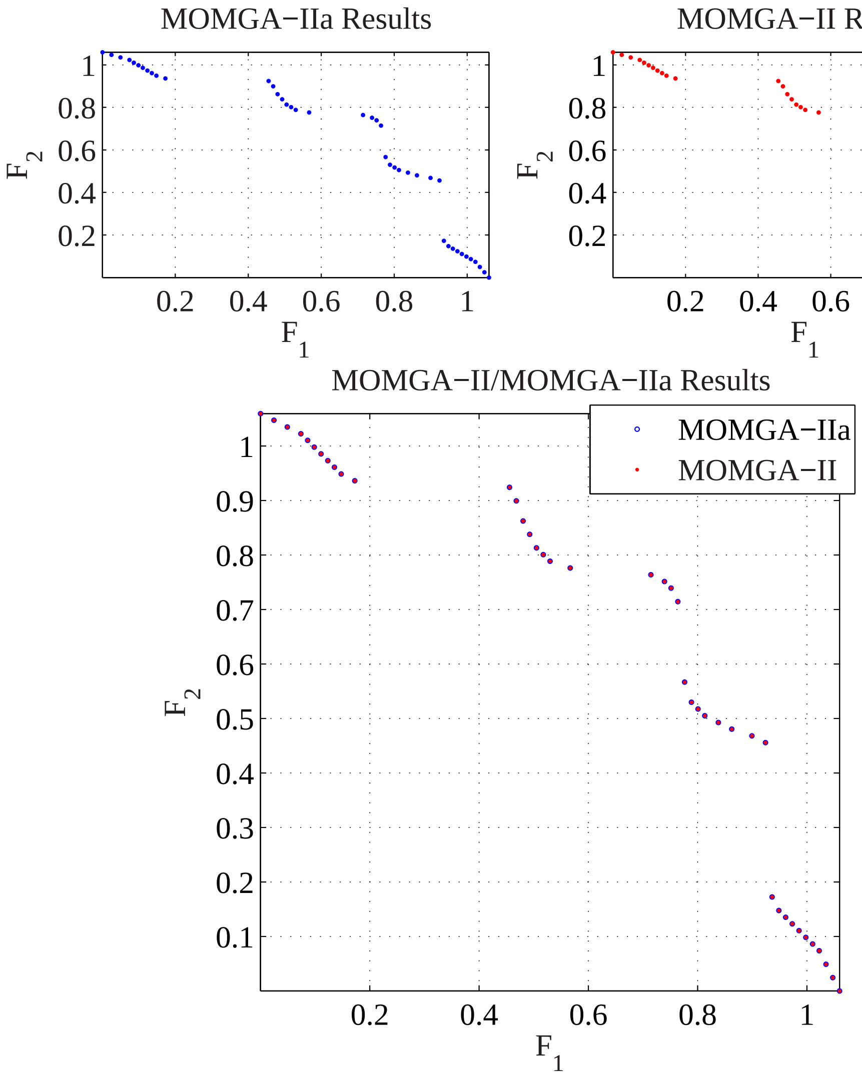


Figure 32: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa for MOP VLC4 (Tanaka) using the same parameter setting except the added archive and number of competitive templates.

Efficiency: In conclusion, results for this MOP are closely similar but the KWtest indicates that MOMGA-IIa is more efficient at finding all PF$_{true}$ points than MOMGA-II. Lastly, Tanaka MOMGA-IIa BB size visualizations are presented in Figure 100 in Appendix D on page 318. Next a three dimensional MOP, DTLZ3, is discussed. Variations of this test function can be found here [62].

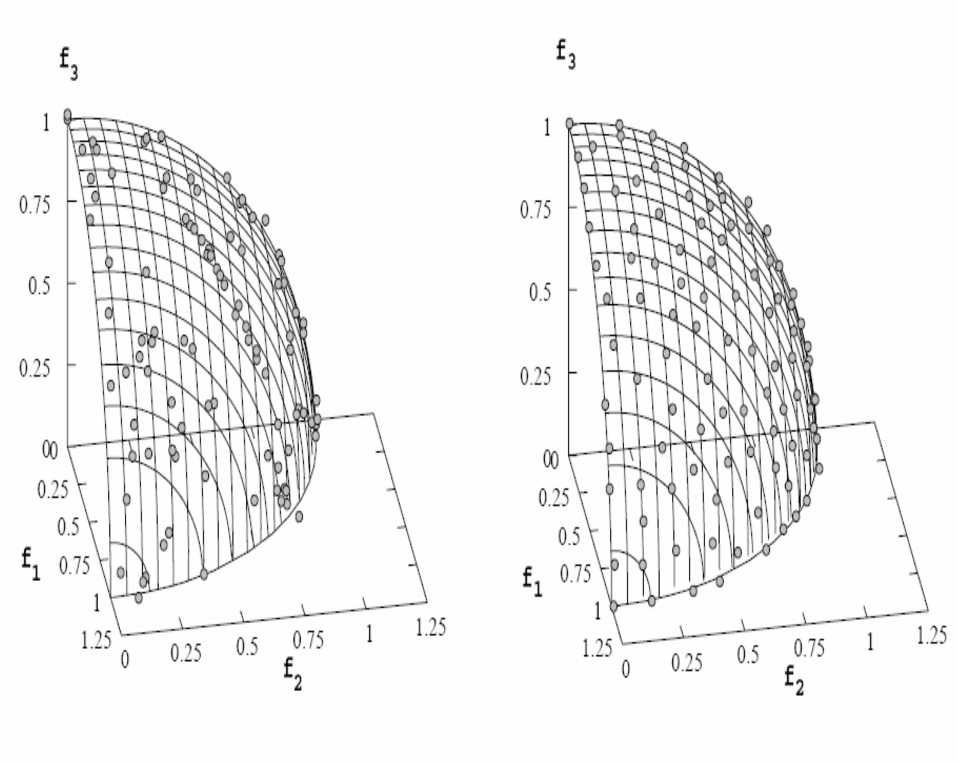


Figure 33: This figure illustrates PF vectors representing the evaluation of found solutions when solving DTLZ3 using MOEAs NSGA-II and SPEA 2. The graph on the left illustrates PF$_{known}$ vectors found by the NSGA-II and the graph on the right are results found by the SPEA 2. [62]

DTLZ3: DTLZx ($x = 1 \cdots 9$) is designed by Deb et al. [62] to meet the following five criteria:

- Easy to construct.
- Scalable to have any number of decision variables.
- Scalable to have any number of objectives.
- The resulting Pareto-optimal front (continuous or discrete) must be easy to comprehend, and its exact shape and location should be exactly known. The corresponding decision variable values should also be easy to find.

- Test problems should introduce controllable hindrance to converge to the true Pareto optimal front and also to find a widely distributed set of Pareto-optimal solutions.

There are many of these test problems that are created with these criteria including $DTLZ1, \cdots, DTLZ9$. DLTZ3 has three objectives and the PF$_{true}$ set is a mesh quarter sphere with radius 1. Figures 33 and 34 both illustrate the true Pareto front and PF vectors of evaluated solutions found by two implicit MOEAs: NSGA-II and SPEA 2.

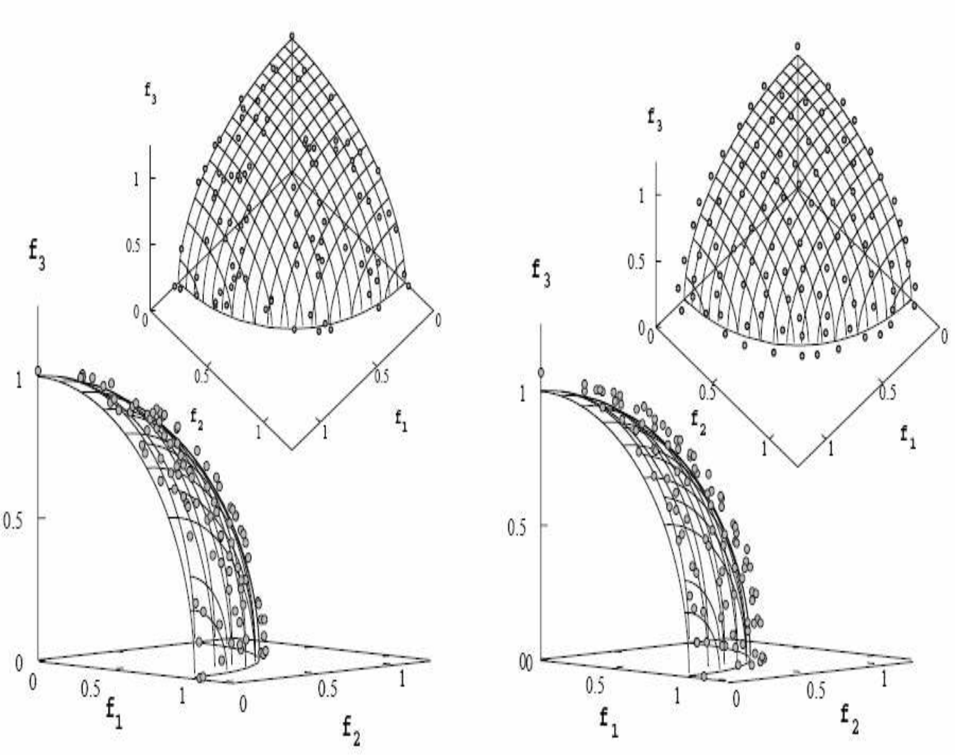


Figure 34: This figure illustrates the PF$_{known}$ vectors representing the evaluation of found solutions when solving DTLZ3 using MOEAs NSGA-II and SPEA 2. The graph on the left illustrates PF$_{known}$ vectors found by the NSGA-II and the graph on the right are results found by the SPEA 2. [62]

Figure 35 illustrates the comparison of solutions evaluating to PF_{known} vectors found by both MOMGA-IIa and MOMGA-II. Notice the difference in the spread and symmetry found by the explicit BBBs (MOMGA-II(a)) and the implicit BBBs

(NSGA-II and SPEA 2). Parameter settings used for MOMGA-II and MOMGA-IIa when solving DTLZ3 can be found listed in Table 50 on page 278. MOMGA-IIa BB size findings for each dimension of DTLZ3 can be found in Figure 101 in Appendix D on page 319 .

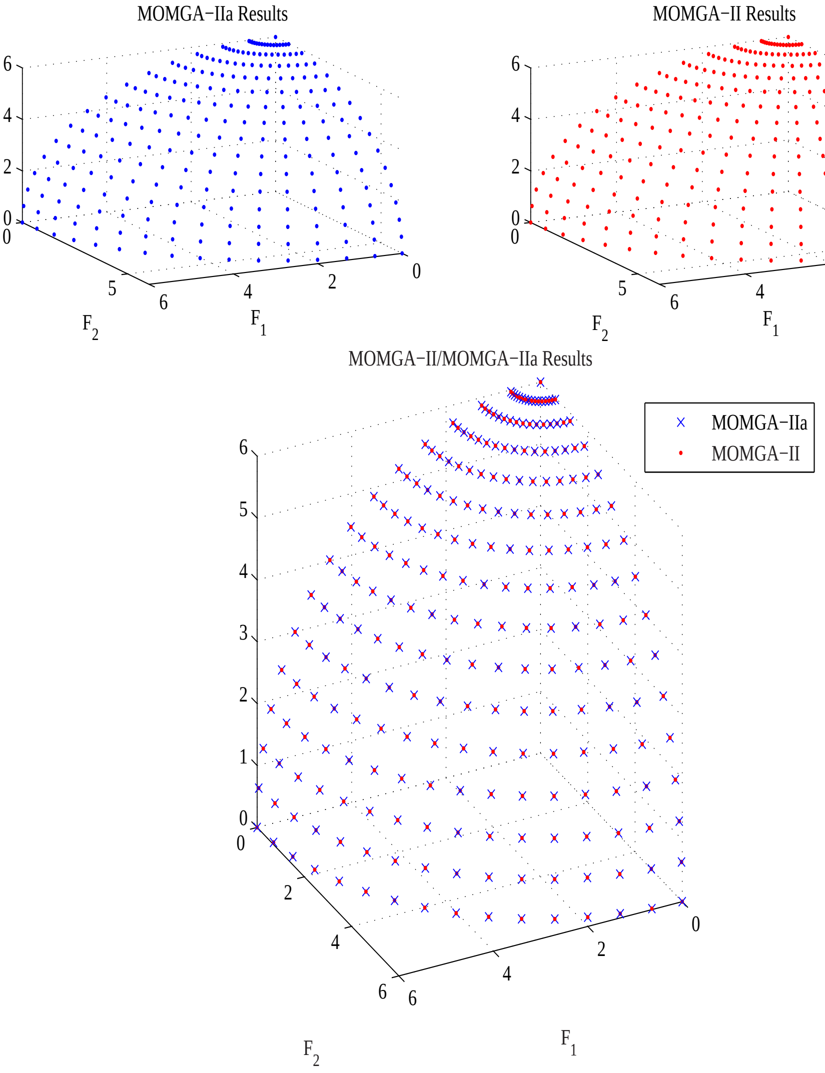
Effectiveness: Visual analysis of statistical data represented in Figure 80 on page 296 indicates that these two MOEAs perform similarly in effectiveness. Finally, KWtest indicates (results listed in Table 20 on page 164) that these algorithms are similar for all but the GD, ONVG, S, and ONVGR metrics. Therefore, it can be concluded that these MOEAs perform similarly in effectiveness for this particular MOP. However, it should be noted that MOMGA-II only solved the problem completely 17 times out of 30 experiments, while MOMGA-IIa solved it completely for all experiments.

Table 20: DTLZ3 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa.

	Metrics									
KWtest	ER	GD	HV	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$	$\in$
F	$\in$	$\in$	$\notin$	$\notin$	$\notin$	$\notin$	$\in$	$\in$	$\in$	$\in$

Efficiency: Lastly, the efficiency of the MOEAs is checked. Although the MOMGA-II did not solve the MOP for each experiment, an extra loop is added to the MOMGA-II to allow it to solve this problem to it entirety over 30 experiments. This modification is used only to gain insight to how much time would it take the MOMGA-II to solve this MOP completely. Listed in Table 21 are the results from extending the algorithm's run time until its PF$_{known}$ vectors, resulting from the evaluation of found solutions, converged onto every PF$_{true}$ vectors. In conclusion, the MOMGA-II is more efficient on this MOP than the MOMGA-IIa.

Covered within this section is the evaluation of effectiveness and efficiency for each MOEA under evaluation and test suite MOP considered. Although many more test MOPs exist [63, 219, 239], these are considered a good representation of


MOMGA−IIa Results

MOMGA−II Results

MOMGA−II/MOMGA−IIa Results

Figure 35: This figure illustrates typical results from running MOMGA-II and MOMGA-IIa using the same parameter settings except the added archive and number of competitive templates.

the complete set available to test MOEAs. The next section concludes this chapter with a summary of the findings. Table 22 includes a summary of all findings from experiments and the statistical analysis.

4.8 Limitations and Rationale for Applications

Several studies attempt to measure the goodness of certain multiobjective approximation solution sets. In fact, there have been several presentations given suggesting a few *carefully crafted* indicators that should yield results that present results that measure one approximation set against another. However, as it is shown in the chapter, there is no one discrete set of metrics that can describe totally the merit of

Table 21: Summary of timing for MOEAs solving test suite MOPs

MOP	MOMGA-II	MOMGA-IIa	KWtest
VL1	25.77 ± 7.63	51.20 ± 24.83	$\notin$
VL2	832.06 ± 6.036	19396.85 ± 8089.18	$\notin$
VL3	10712.54 ± 96.53	2433.71 ± 1593.42	$\notin$
VL4	148.38 ± 0.84	1673.18 ± 481.00	$\notin$
VL6	95.72 ± 53.11	74.371875 ± 38.71	$\notin$
VLC4 (Tanaka)	2374.25 ± 647.30	1153.83 ± 389.17	$\notin$
DTLZ3	250.51 ± 54.69	387.82 ± 141.54	$\notin$

Table 22: Summary of Effectiveness and Efficiency results for the MOMGA-II and MOMGA-IIa run on the test suite MOPs identifying favorable metrics for MOMGA-IIa (ER, GD, HR, ONVG, S,ONVGR, ME, R2, R3, and epsilon) indicating the difference in MOEAs.

MOP	Effectiveness	Efficiency	Favorable Metrics
VL1	none	none	
VL2	IIa	none	GD, HR, ONVG, ONVGR, R2 and R3
VL3	IIa	IIa	ER, GD, HR, ONVG, S,ONVGR, and ME
VL4	IIa	IIa	ER, GD, HR, ONVG, S, ONVGR, ME, R2, R3, and ϵ
VL6	none	IIa	
VLC4 (Tanaka)	none	IIa	
DTLZ3	~IIa	II	HV, ONVG, S, ONVGR

one approximation set to another. This is stated within a proof by Zitzter in [239]. The ten metrics (indicators) selected within this document are done so for the purpose of a better measurements of the approximation sets found by the two MOEAs under test. It is thought that these measurements, when used and analyzed together, capture the merit of each approximation set with respect to the PF$_{true}$ reference set.

Given that test suites themselves have limitations, there is a need for evaluating an MOEA's performance in real-world applications having higher complexities. Table 83 on page 421 lists examples of each problem to which an application might be mapped. Table 84 in Appendix M on page 421 specifies the type of fitness landscapes expected for each problem. Finally, Table 11 on page 144 lists test suite MOPs used to round off the good subset of MOP characteristics which might be faced in any real-world application.

4.9 Summary

Although it is difficult to definitively determine if one MOEA is better than another, an attempt is made using state-of-the-art metrics on a subset of test suite MOPs. The best MOEA metrics known to MOEA researchers are used within this study; however, it is not to say these are the only metrics available. Numerous metrics are available, but these are selected as good metrics to use in comparing these two MOEAs. The two MOEAs under test are the MOMGA-IIa and the MOMGA-II. Fortunately, these algorithms are similar, thus most of the algorithm settings are kept constant between the two. Design changes improve the effectiveness of the MOMGA-IIa on some MOPs; however, this comes at a cost in efficiency. Table 22 on page 166 lists the MOPs and the corresponding better MOEA. It can be surmised from this Table 22, MOMGA-IIa is at least as good, if not better, at solving all the MOPs tested (in terms of time to convergence). For VL2, VL3, VL4, and DTLZ3, MOMGA-IIa is more effective, although not totally more efficient. For VL3, VL4, and VL6, MOMGA-IIa is more efficient. In cases where MOMGA-II did not completely solve the problem using the parameter settings the MOMGA-II's runs were extended to let the MOMGA-II run time extend past the MOMGA-IIa's convergence time for a second effectiveness check.

Considering the problem size of these MOPs are somewhat small (< 30), the following chapters more thoroughly test these MOEAs. Beginning with deception problems ranging in sizes from 30 to 90, MOEA scalability is tested. Next, the following real-world problems are tested: the protein structure prediction problem, multiple quadratic assignment problem and m-ary symbol set design problem.

To conclude, it is apparent that MOMGA-IIa is no worse and better in most cases in effectiveness for the test MOPs studied. Furthermore, it is expected that the explicit BB MOEA must perform better on the range of MOPs that the MOMGA-II has shown to perform well on because these are indeed similarly structured algorithms. Finally, it is a conjecture that the MOMGA-IIa scales well and will perform better on the upcoming deception problems and real-world applications.

V. Deception Problems

Deception problems are among the hardest problems to solve using ordinary genetic algorithms. Designed to simulate a high degree of *epistasis*[1], deception problems imitate extremely difficult real-world problems [188]. Studies show that Bayesian optimization and explicit BB manipulation algorithms, like the fmGA, solve these problem faster [122]. This chapter compares the results acquired from MOMGA-IIa, MOMGA-II, mBOA, and the non-dominated sorting genetic algorithm-II (NSGA-II) when applied to three different deception problems. The three deceptive problems studied are: interleaved minimal deceptive problem, interleaved 5-bit trap function, and interleaved 6-bit bipolar function [122]. The *unmodified* MOMGA-II explicitly learns BB linkages, a requirement if an algorithm is to solve these hard deception problems. Results using MOMGA-IIa are exceptional when compared to the implicit BBB results of both the NSGA-II and good when compared to another explicit BBB, the mBOA.

5.1 Introduction

Algorithms that solve problems by realizing good BBs are useful in solving extremely difficult problems: Protein Structure Prediction [41], 0/1 Modified Knapsack [243], Multiple Objective Quadratic Assignment Problem [42, 125], Digital Amplitude-Phase Keying Signal Sets with M-ary Alphabets and many academic problems [34, 46]. As discussed in Chapter III, MOMGA-IIa originated as a single objective mGA. It evolved from being a single objective mGA into a multiobjective mGA called the MOMGA [74]. Many different Multiobjective Evolutionary Algorithms (MOEAs) were produced during this time period; however, the MOMGA is

[1] Evolutionary Computation researchers use the term *epistasis* when referring to any kind of strong interaction among genes, not just masking effects. According to `http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/Q99_E.htm`, epistasis is the interaction between different genes in a chromosome. It is the extent to which the contribution to fitness of one gene depends on the values of other genes.

the only MOEA explicitly using good BBs to solve problems. The MOMGA has a population size limitation: as the BB size increases so does the population size during the PEI phase. This renders MOMGA less useful on large problems. To overcome this problem, MOMGA-II, based on the single objective fmGA, is designed. The fmGA is similar to the mGA in that it specifically uses BBs to find solutions; however, it has a reasonable population size and lower run time complexity (see Table 23) when compared to the mGA. MOMGA-II includes many different repair, selection, and crowding mechanisms. Unfortunately, the MOMGA-II is found to be limited when solving large deception problems [46]. This called for the development of basis function diversity measures in the MOMGA-IIa which are designed for smart BB searching in both the geno- and pheno-type domains.

Table 23: Complexity Estimates for serial EAs and MOEAs

	Single Objective Algorithm				MOEAs		
Phase	*sGA*[b]	*ssGA*	*mGA*	*fmGA*	*NSGAII*	*mBOA*[c]	*IIa*[d]
Initial	$O(\ell^{\mathcal{N}})$	$O(\ell^{\mathcal{N}})$	$O(\ell^{\mathbf{o}})$	$O(\ell)$			
Recomb	$O(t\mathcal{N}q)$	$O(t)$					
Primordial	$O(\emptyset)$		$O(\emptyset)$	$O(\ell^2)$[e]			
Juxtapos			$O(\ell \log \ell)$	$O(\ell \log \ell)$			
Overall	$O(\ell^{\mathcal{N}})$	$O(\ell^{\mathcal{N}})$	$O(\ell^{\mathbf{o}})$	$O(\ell^2)$	$O(k\mathcal{N}^3)$	$O(\mathcal{N}^{3.5})$	$O(ket\mathcal{N}^2)$

[b] q is group size for tournament selection

[c] This complexity is problem specific and in this case has been taken from the spin glass problem. [181]

[d] e = number of eras

[e] Building Block Filtering

The next section discusses mBOA and NSGA-II. The mBOA and NSGA-II have been used to solve these three multiobjective problems (MOPs) in previous research [46, 122]. The interleaved minimal deceptive problem, interleaved 5-bit trap function, and interleaved 6-bit bipolar function [122] are described in detail in Section 5.2. Next, experimental design, resources, parameter settings, and algorithm efficiency are discussed briefly in Section 5.3. Finally, in the results section, the mBOA, NSGA-II, MOMGA-II and MOMGA-IIa results are compared and analyzed.

5.1.1 Non-dominated Sorting Algorithm-II. The NSGA-II is an implicit BB MOEA based on the original design of NSGA (see Section E.5.4 on page 350 for a description of the NSGA). NSGA-II builds a population of compete individuals, ranks and sorts each individual according to non-domination level, applies EVolutionary OPerations (EVOPs) to create a new pool of offspring, and then combines the parents and offspring before partitioning the new combined pool into fronts. The NSGA-II then conducts niching by adding a crowding distance to each member. It uses this crowding distance in its selection operator to keep a diverse front by making sure each member stays a crowding distance apart. This keeps the population diverse and helps the algorithm to explore the fitness landscape. For a more detailed summary of the NSGA-II see Section E.5.4 on page 350.

5.1.2 Multiobjective Bayesian Optimization Algorithm (mBOA). The mBOA was also used to solve these MOPs in previous research [121]. mBOA is identical to the single objective Bayesian Optimization Algorithm (BOA) [121] minus the selection procedure. The BOA's selection procedure is replaced by the non-dominated sorting and selection mechanism of NSGA-II. The BOA generates a child population of size $\mathcal{N}'$ from a parent population. The child and parent population is then merged and the combined population is Pareto ranked. Based on the Pareto ranking and crowding distance function, a new population is created from which BOA builds a new probabilistic model to generate children again.

5.2 Deception Problems

In 1987, Goldberg's research group introduced deception to test the abilities of current genetic algorithms to solve high epistatic leveled problem [88]. They designed problems having specific difficulties which genetic algorithms might face in problem solving. These *deception* problems are often challenging to optimize and involve some degree of deception – resulting in conflicting objectives (*e.g.*, the k-arm bandit competitions between hyperplanes [228]). Later, Whitley [228] proved deceptive attractors must have complementary bit patterns to the global optimum pattern in order to be either fully deceptive or consistently deceptive problems. He then defines a deceptive problem at least one more relevant lower order hyperplane competitor that guides a genetic search away from the global winner. Imagine a hill climbing search algorithm starting anywhere except with the bit configuration *111x*. The hill climbing algorithm always finds the suboptimal fitness of 9 as a solution. This example illustrates how a competitor hyperplane might guide a GA away from the optimal solution. Furthermore, it is every GA engineer's desire to build an algorithm that finds proper linkages within a problem, overcoming this type of deception.

We evaluate the following five objective functions in this chapter:

1. T1 - Interleaved minimal deception problem
2. T2 - Complement of T1
3. T3 - Interleaved 5-bit trap function
4. T4 - Complement of T3
5. T5 - Interleaved 6-bit bipolar deception function

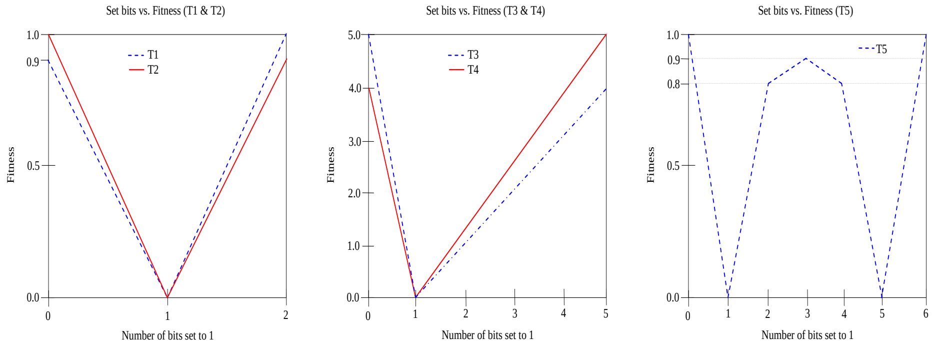
These test functions are difficult in four respects: deception, loose linkage, multimodality, and combinatorially - having a large search space [59, 60, 92]. Sections 5.2.1 through 5.2.3 included a detailed discussion of these deceptive problems.

In addition to solving these five test functions, difficulty is added by combining these functions together to make three multiobjective problems. By aggregating these test functions together, the order of deception is increased because the functions are paired in a manner that adds a relevant lower order hyperplane competitor to guide a genetic search away from the global winner. The following is the list of the MOPs investigated in this chapter:

1. MOP 1: T1 and T2 (T1T2)

2. MOP 2: T3 and T4 (T3T4)
3. MOP 3: T2 and T5 (T2T5)

Note, do not get confused with MOP 1, 2 and 3 previously defined. Throughout this chapter, MOP 1 refers to T1T2, MOP 2 refers to T3T4 and MOP 3 refers to T2T5.



a) T1 and T2 b) T3 and T4 c) T5 and T2

Figure 36: These figures illustrate the fitness landscape for each function. Subfigure *a* illustrates deception problems T1 and T2, subfigure *b* illustrates deception problems T3 and T4, and subfigure *c* illustrates deception problems T5 and T2

5.2.1 Interleaved Minimal Deceptive Problem (T1 & T2). The interleaved minimal deceptive problems are designed to test an algorithm's ability to discover loosely linked bits by dividing the string into two halves and coupling one bit from each half. Figure 37 illustrates how the bits are correlated. Bits having the same pattern are rewarded, while alternating couplets are not. Additionally, Figure 36.a illustrates the bit couplet fitness for T1 and T2.

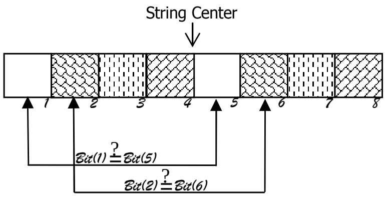


Figure 37: This figure illustrates bit linkage in an eight-bit solution. Notice that every i^{th} and $(\frac{\ell}{2}+i)^{th}$ bit is linked such that $i \leq \frac{\ell}{2}$, ℓ=length of string, and the 1^{st} bit is bit 1.

5.2.2 Interleaved 5-bit trap function (T3 & T4). The interleaved 5-bit trap function is devised to test an algorithm's ability to find loose linkages having non-consecutive bits. Bits in problems T3 and T4 both have correlated bits with a distance of $\frac{\ell}{5}$ from one another. Figure 39 illustrates how the bits in groups of five are coupled. Additionally, Figure 36.b graphically illustrates how the fitness behavior varies according to the number of bits that are set in the described 5-bit linkage pattern. Notice that T3 in Figure 36.b is similar to the classical deception problem example illustrated in Figure 38.

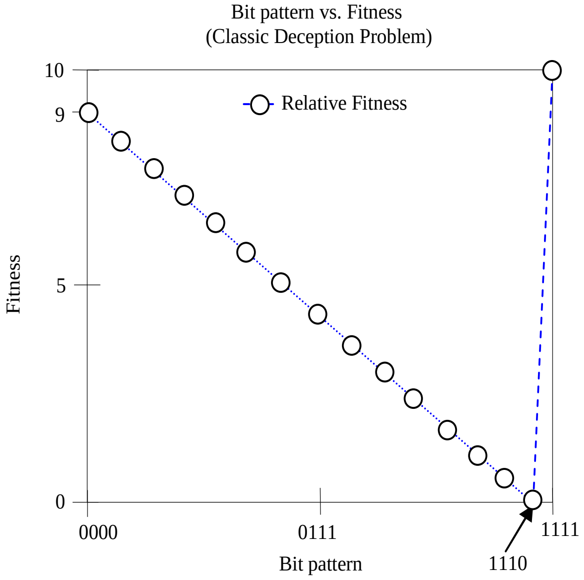


Figure 38: This figure illustrates a classical deception problem.

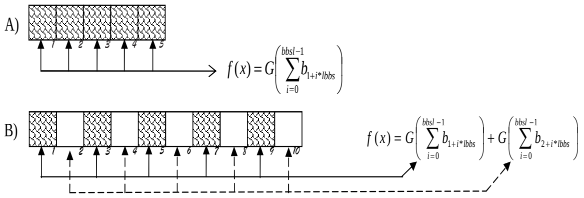

A)
1
2
3
4
5
$f(x)=G\left(\sum_{i=0}^{bbsl-1} b_{1+i*lbbs}\right)$
B)
1
2
3
4
5
6
7
8
9
10
$f(x)=G\left(\sum_{i=0}^{bbsl-1} b_{1+i*lbbs}\right)+G\left(\sum_{i=0}^{bbsl-1} b_{2+i*lbbs}\right)$

Figure 39: This figure illustrates bit linkage in a 5- and 10-bit solution. In the figure, the fitness function for each set of bits is shown to the right with arrows indicating which bits contribute to each term of the fitness summation.

5.2.3 Interleaved 6-bit Bipolar Function (T5). The interleaved 6-bit Bipolar function is constructed as a loose linkage problem having correlated bits in variable placement in the string (see Figure 40). The first three bits and the last three bits are correlated, then the 4^{th}, 5^{th}, 6^{th}, $(\ell - 4)^{th}$, $(\ell - 5)^{th}$, and $(\ell - 6)^{th}$ and so on until the middle 6 bits of the string are left. Graphically, the fitness function for T5 is illustrated in Figure 36.c.

5.3 Experiments

The experiment for all MOMGA-II MOPs were run simultaneously on the two computational clusters (ASPEN and Polywells) listed in Table 13 in Chapter IV. The MOMGA-IIa ran in serial on one computational cluster (TAHOE). The MOMGA-II is given 30 to 50 experiments to solve the three MOPs while MOMGA-IIa is run for ten experiments or less. The MOMGA-II is given more experimental loops in an attempt to allow for it to find all Pareto front vectors for the larger deception problems. Unfortunately, even with the extra experimental loops, MOMGA-II still did not find all Pareto front vectors.

5.3.1 Resources. Table 13 in Chapter IV lists the resources used for these experiments. Each MOMGA-II experiment took approximately one week to complete including the time to process all data for presentation. Each MOMGA-IIa experiment took approximately two days. This includes the time jobs sat idle in the

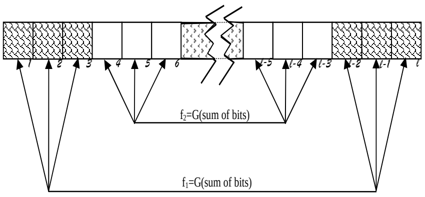


Figure 40: This figure illustrates 6-bit bipolar linkage in an ℓ bit solution. The bit string is broken in the middle to enhance the idea that string can be of any size as long it is a multiple of 6.

scheduler queue and the post mortem collection of data for analysis. NSGA-II and mBOA run times were not found in previous publications [122].

Table 24: Summary of Era parameters settings for each experiment using MOMGA-II(MOMGA-IIa)

Experiment	Start ERA	End ERA	Runs	CTs	Inverse CTs	Orthogonal CTs
MOP 1(30)	1	10(4)	30(3)	4(18)	0(18)	0(41)
MOP 1(60)	1	10(4)	30(9)	4(18)	0(18)	0(45)
MOP 1(90)	1	10(4)	30(10)	4(18)	0(14)	0(9)
MOP 1(120)	1	10(4)	30(10)	4(18)	0(14)	0(9)
MOP 2(30)	1	10(4)	30(10)	4(18)	0(18)	0(41)
MOP 2(60)	1	8(4)	50(10)	4(18)	0(18)	0(45)
MOP 2(90)	1	6(4)	50(10)	4(18)	0(18)	0(10)
MOP 2(120)	1	4(4)	30(8)	4(14)	0(14)	0(49)
MOP 3(30)	1	10(4)	10(1)	4(14)	0(14)	0(41)
MOP 3(60)	1	12(4)	10(1)	4(14)	0(14)	0(41)

5.3.2 Parameter Settings. The MOMGA-II(a) has many parameters for proper program execution. BB sizes must be determined, elitism percentages, cut probability, splice probability, mutate probability, population sizing variable, n_a, era generations, and BB filtering schedule. The program is run x times and Pareto front members are collected from each solution set. Table 24 indicates the number of times the program is run on each MOP. In some cases, the experiments were terminated early if all Pareto front vectors were found. All MOMGA-II(a) experiments are executed using a multiple objective fmGA. Unless otherwise specified, Tables 25

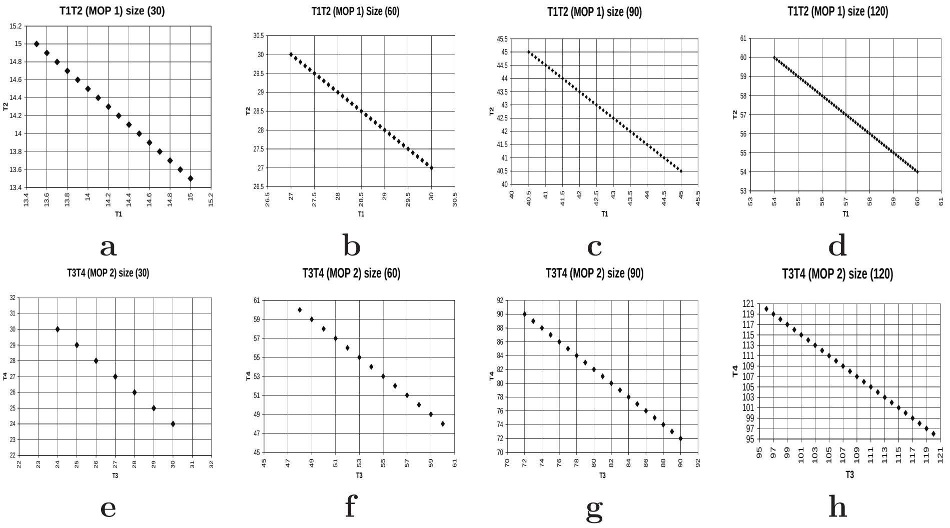


Figure 41: This figure illustrates the Pareto front findings for the T1 versus T2 and T3 versus T4 experiment using a string length of 30, 60, 90, and 120 bits.

Table 25: Summary of static parameters set for each experiment regardless of MOP and problem size. MOMGA-II(MOMGA-IIa)

Static parameter settings for each MOP			
Parameter	Setting	Parameter	Setting
Maximization	1(n/a)	Thresholding	0(n/a)
mGA(0)/fmGA(1)	1(n/a)	Shuffle Number	2(2)
		Tiebreaking	0(n/a)
Overflow	2.0(2.0)	Reduced initial pop	0(n/a)
Elitism %	25(0)	Extra pop members	0(n/a)
Prob cut	0.02(0.02)	Stop criteria factor	1.00(n/a)
Prob splice	1.0(1.0)	Partition file	0(n/a)
Prob mutation		Plotting file	0(n/a)
allelic	0.0(n/a)	Pop record file	0(n/a)
genic	0.0(n/a)	Copies	5 1 1(n/a)
Inverse Template	n/a(Y)	CT Guesses	n/a(1)

and 24 list all the parameters used for the experiments conducted in this chapter. The n_a values for the MOMGA-II can be found in [46] while n_a values were set to 500 for the MOMGA-IIa. Table 25 lists the constant parameter settings, while Table 24 identifies parameters that were varied for each MOP. In the cases where the programs were run for less than 30 times, this is because the MOMGA-II(a) found the optimal Pareto front before each run completed.

BB size selection for these problems is tricky because the identification of the length of one particular linkage (say five bits long) may not be enough to transform solutions out of the search basis provided by the resident competitive templates. This is prominent for MOMGA-II when solving the MOP 2 of size 90 where the competitive templates used constrict the search into a space where the small BB sizes cannot overcome the competitive template basis. Normally, a BB size can be selected upon knowing the length of these linkages; however, in MOP 2, there are multiple linkages of five bits long each magnifying the difficulty and requiring a larger BB size to allow the MOMGA-II to find more PF$_{true}$ points. It should be noted that when the BB sizes increase, population size is also increased, as well as run time for the algorithm to complete. MOMGA-II's limitation was found in MOP 2 with a size greater than or equal to 90. This limitation is overcome by MOMGA-IIa by using specially chosen competitive templates for search after each BB generation.

5.3.3 Efficiency Finding Pareto-front points.

Reduction of relative execution time for MOMGA-IIa is achieved by keeping the Pareto front points (including duplicates) in memory. The MOMGA-IIa benefits from a creatively designed structure maintained as a dynamic linked list object which holds all Pareto front members. Solutions evaluating to dominated vectors are deleted from the structure and from memory including all duplicates for that particular point. As the program runs, the Pareto front point listings can become long. This is a disadvantage; however, elitism selection is O(PF_{known}) run time as all PF vectors, PF_{known}, are readily listed. In addition to this enhancement, epsilon dominance, crowding techniques and dominance linkage can be instantiated. The MOMGA-IIa also has the ability to trace BB evolution. This enhanced feature comes at a cost in space (memory or disk) and efficiency. All experiments in this chapter are run with an active trace feature to allow for post mortem analysis of Pareto front point conception.

5.4 Results and Analysis

The MOMGA-IIa results are superior. In every case, the MOMGA-IIa has either found more Pareto front vectors or more unique solutions than all other algorithms tested on these three MOPs. As expected from the last investigation [46], this investigation found MOP 2 to be the most difficult to solve. Difficulty comes in the form of time to find all Pareto front members.

The following sections describe, in detail, the experiment results for each MOP. Notice that the Pareto front for MOP 1 and MOP 2 is linear. This is due to the linear slopes of the individual objective functions making up each of these MOPs. It is also worth noting that each point on the MOP 1 and MOP 2 Pareto front may have many unique strings (solutions) equating to that particular Pareto front point.

Table 26: Summary of Results for all experiments. Included in this table are the number of optimal Pareto front points, number of unique strings making up these optimal points, and the number of points each algorithm (MOMGA-IIa (M-IIa), MOMGA-II (M-II), mBOA, and NSGA-II) have found in each category.

MOP 1

T1 versus T2 - Sizes: (30/60/90/120)
Unique Strings: $\{(2^{15}/2^{30}/2^{45}/2^{60})\}$ Pareto front strings: (16/31/46/61)

Algorithm	Unique Strings Found				Pareto front pts found			
	30	*60*	*90*	*120*	*30*	*60*	*90*	*120*
M-IIa	**32768**	**300776**	**57661**	**32876**	**16**	**31**	**46**	**61**
M-II	596	21364	28	138	**16**	**31**	16	56
mBOA	224		591		**16**		**46**	
NSGA-II	7		5		6		3	

MOP 2

T3 versus T4 - Sizes: (30/60/90/120)
Unique strings: $\{(2^{6}/2^{12}/2^{18}/2^{24})\}$ Pareto front strings: (7/13/19/25)

Algorithm	Unique Strings Found				Pareto front pts found			
	30	*60*	*90*	*120*	*30*	*60*	*90*	*120*
M-IIa	**64**	**565**	**1280**	**3594**	**7**	**13**	**19**	**25**
M-II	32	98	54	31	**7**	12	6	14
mBOA	30	102	327		**7**	**13**	**19**	
NSGA-II	0	1	0		0	1	0	

MOP 3

T5 versus T2 - Sizes: (30/90)
Unique strings: (1/1) Pareto front strings: (1/1)

Algorithm	Unique Strings Found		Pareto front pts found	
	30	*90*	*30*	*90*
M-IIa	**1**	**1**	**1**	**1**
M-II	**1**	**1**	**1**	**1**
mBOA	**1**	**1**	**1**	**1**
NSGA-II	0	0	0	0

This phenomenon is illustrated in Table 26 where both the number of Pareto front points and unique strings are listed.

5.4.1 T1 versus T2 (MOP 1). Figures 41a-d reflect the results of the MOP 1 of sizes 30, 60, 90, and 120. Diamonds indicate the Pareto front vectors found. Notice that these points are linear and discrete. For problem sizes 30, 60, 90, and 120, there are 16, 31, 46, and 61 total Pareto front points. Not only does

MOMGA-IIa find the same or more true Pareto front points for MOP 1 than every other algorithm tested, it also finds more unique strings making up each of these Pareto front points. Table 26 numerically shows in bold that MOMGA-IIa is the front runner in this experiment. In the problem size 30, the MOMGA-IIa has actually found all unique strings corresponding to each of the 16 Pareto front vectors.

5.4.2 T3 versus T4 (MOP 2). Figures 41e-h illustrate the results of the MOP 2 experiment. Similarly, diamonds indicate the Pareto front vectors found. Notice that these points are linear and discrete. For the problem sizes 30, 60, 90, and 120, there are 7, 13, 19, and 25 total Pareto front points in the entire search space. Not only does MOMGA-IIa find the same or more true Pareto front points for MOP 1 than every other algorithm tested, it also finds more unique strings making up these Pareto front points. Specifically, MOMGA-IIa overcomes the problem size issue and finds more Pareto front points than its predecessor, MOMGA-II. Table 26 numerically shows in bold that MOMGA-IIa is the front runner in this experiment. In addition to finding all the Pareto front vectors for MOP 2 size 30, the MOMGA-IIa finds all corresponding unique strings.

5.4.3 T5 versus T2 (MOP 3). No figures in this chapter reflect the result of MOP 3 for it is rather uninteresting. See [46] for an example. There is only one point on the Pareto front. All algorithms but the NSGA-II found the one and only Pareto front point and the only unique string representing this PF vector.

5.4.4 Non-statistical Comparison. MOMGA-IIa results are excellent. The MOMGA-IIa finds all Pareto front members in every MOP and at every size. Additionally, the MOMGA-IIa also finds more unique strings than any other algorithm tested. Clearly, MOMGA-IIa outperforms MOMGA-II and is able to scale up to solve larger deception problems by having a pool of better competitive templates.

5.4.5 Statistically Comparing MOMGA-II versus MOMGA-IIa. For a statistical comparison between MOMGA-II and MOMGA-IIa, there must be some parameter consistency in order to have a somewhat fair comparison. These settings are given in Appendix A in Tables A.9, A.10, A.11, A.12, A.13, and A.14 on page 272 corresponding to MOP 1:30, MOP 1:60, MOP 1:90, MOP 2:30, MOP 2:60, and MOP 2:90. Also, statistical results for the ten metrics (ER,GD,HR,ONVG,S,ONVGR,ME,R2,R3,ϵ) used previously can be found in Figures 82, 83, 84, 85, 86, and 87 in Appendix B on page 290:

Metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa are listed in several following tables. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different. Upon an indication of different, $\notin$, the researcher must go back to the visualization of the statistical data to determine which is indeed better. Unless otherwise specified, it is found that $\notin$ indicates that the MOMGA-IIa performs better than MOMGA-II.

MOP 1, (30, 60, 90) (T1T2:30:60:90): The deception problem T1T2 was introduced earlier in this chapter. This particular MOP is a difficult test problem; however, it has certain qualities that make it easier to solve than T3T4. Evaluated is the effectiveness and efficiency of the two MOEA: MOMGA-II and MOMGA-IIa. The 30, 60, and 90 are evaluated. 30 experimental runs on each MOP are accomplished, and the mean and standard deviations (error bars) are given in the specified figures above.

Effectiveness: Visually, the statistical results for the T1T2:30 MOP look very much the same. The KWtest for this MOP also concludes that both MOEA findings are the same (see Table 27). The increase from size 30 to 60 changes the results a bit. MOMGA-II findings begin to drop off and becoming degraded due to not finding all the Pareto front points. The characteristics of the Pareto front make for metrics like ER, GD, S, and ME rather useless. Once on the Pareto front, there is really no progression; it becomes just a test of finding more points along that line. Therefore,

Table 27: MOP 1 (T1T2:30) metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$
F	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$	$\in$

Table 28: MOP 1 (T1T2:60) metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$
F	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$

evaluation of these MOEAs should rely more heavily upon the other six metrics that are not as deceived by this type of Pareto front. In the problem of size 60, the KWtest reveals that the other size metrics do indicate a difference between algorithms (see Table 28). It is concluded that MOEA effectiveness is different and in favor of the MOMGA-IIa for MOP 1 of size 60. Finally, similar results can be found in Table 29 for MOP 1 of size 90. In fact, the visual comparisons really illustrate a huge difference between these two algorithms for these larger sized deception problems. Next, the efficiency of these problems is discussed.

Efficiency: Efficiency results for each size of this MOP can be found in Table 30. It is apparent that the MOMGA-IIa is more efficient than the MOMGA-II

Table 29: MOP 1 (T1T2:90) metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$
F	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$

for MOP 1 size 30; however, this changes for size 60 and 90. With these larger sized MOPs, the effectiveness cannot be ignored. For these larger sized MOPs, the MOMGA-II does not completely solve the problem in the allotted time; thus, reported is that for MOP 1 (60) the MOMGA-IIa is more efficient and for the MOP 1 (90) it is unknown which algorithm is more efficient. Listed in Table 31 are the results stated. Next, all sizes of MOP 2 are discussed and conclusions are drawn.

Table 30: Summary of timing for MOEAs solving T1T2 and T3T4 MOPs

MOP	MOMGA-II	MOMGA-IIa	KWtest
T1T2:30	617.48 ± 164.87	8.01 ± 4.79	$\not\in$
T1T2:60	290.58 ± 0.75	286.90 ± 59.50	$\in$
T1T2:90	584.94 ± 2.07	2386.53 ± 469.80	$\not\in$
T3T4:30	655.43 ± 111.51	255.03 ± 463.09	$\not\in$
T3T4:60	333.58 ± 3.05	2980.26 ± 3092.08	$\not\in$
T3T4:90	741.78 ± 190.91	41047.72 ± 678.09	$\not\in$

Table 31: Summary of Effectiveness and Efficiency results for the MOMGA-II and MOMGA-IIa run on the T1T2/T3T4 MOPs identifying favorable metrics (ER, GD, HR, ONVG, S,ONVGR, ME, R2, R3, and epsilon) indicating the difference in MOEAs.

MOP	Effectiveness	Efficiency	Favorable Metrics
T1T2:30	none	IIa	
T1T2:60	IIa	IIa	GD, HR, ONVG, ONVGR, R2, R3, and ϵ
T1T2:90	IIa	none	GD, HR, ONVG, ONVGR, R2, R3, and ϵ
T1T2:30	IIa	IIa	GD, HR, ONVG, ONVGR, R2, R3, and ϵ
T1T2:60	IIa	none	GD, HR, ONVG, ONVGR, R2, R3, and ϵ
T1T2:90	IIa	none	GD, HR, ONVG, ONVGR, R2, R3, and ϵ

MOP 2, (30, 60, 90) (T3T4:30:60:90): The deception problem T3T4 is introduced earlier in this chapter. This particular MOP is the most difficult test problem utilized. Evaluated is the effectiveness and efficiency of the two MOEA: MOMGA-II and MOMGA-IIa. The 30, 60, and 90 are evaluated. 30 experimental

Table 32: MOP 2 (T3T4:30:60:90) metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. $\in$ indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$
F	$\in$	$\notin$	$\notin$	$\notin$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	$\notin$

runs on each MOP are accomplished and the mean and standard deviations (error bars) are given in the specified figures above.

Effectiveness: Visually, the statistical results for T3T4 of sizes 30, 60 and 90 are different. Starting with T3T4 size 30, there is enough of a difference to call the effectiveness to be better for the MOMGA-IIa. This difference increasingly becomes greater as the string size increases. The KWtest for all sizes for this MOP also concludes that both MOEA findings are different (not from the same population). Table 32 lists results for all T3T4 experiments. Characteristics of the T3T4 Pareto front make metrics like ER, GD, S, and ME rather useless. Once on the Pareto front, there is really no progression; it becomes just a test of finding more points along that line. Therefore, evaluation of these MOEAs should rely more heavily upon the other six metrics that are not as deceived by this type of Pareto front.

It is concluded that MOEA effectiveness is different and in favor of the MOMGA-IIa for all sizes of MOP 2. In fact, the visual comparisons really illustrate a huge difference between these two algorithms for these larger sized deception problems. Next, the efficiency of these problems is discussed.

Efficiency: Efficiency results for each size of this MOP can be found in Table 30. It is apparent that the MOMGA-IIa is more efficient than the MOMGA-II for MOP 1 size 30; however, this changes for size 60 and 90. It is apparently that because the MOMGA-II does not solve this MOP no matter the number of increased loops; it is rather difficult to describe the efficiency for it to solve T3T4:60 and T3T4:90. That being said, efficiency for T3T4:60 and T3T4:90 is not addressed; however, it can be said that after 50 experiments using the settings identified in Appendix A on page 272, the MOMGA-II could not solve this problem. A suggestion is to either increase the population size or BB search sizes.

Listed in Table 31 on page 183 are the results found by comparing the MOMGA-II versus MOMGA-IIa. Next, conclusions are drawn and suggestions are made about future work in this area.

5.5 Evolutionary Trace for BB Visualization

Interest in why these deception problems are difficult exists among MOEA researchers. In addition, a long standing conjecture that larger BB are required to find Pareto front points along the extremes of the front [245] is unproven. The new innovative tracing visualization technique designed in this research can be consulted for possible answers to these conjectures or inquiries by researchers.

T1T2: Two evolutionary traces for solving T1T2 of size 30 and 60 are presented within Figures 42 and 43. Deduced by these graphics is that the phenotype is linear and equally spaced. The genotype has patterns; however, it is nevertheless not held to a small area. In fact, it looks well distributed. Finally, the BB sizes for both T1T2:30 and T1T2:60 show a more dominant amount of smaller BBs used to find solutions evaluating to PF vectors across the entire Pareto front, while larger BBs are used to find solutions evaluating to PF vectors more centered along the Pareto front.

The fact that the required BB sizes are evenly distributed along the Pareto front illustrates how this problem is easier to solve than a deception problem like T3T4 and mQAP. BB sizes arbitrarily distributed along the Pareto front indicate a more difficult problem to solve because the problem has a larger variety of BBs sizes making up the good BBs required. Next, a T3T4 evolutionary trace is discussed.

T3T4: Two evolutionary traces for solving T3T4 of size 30 and 60 are presented within Figures 44 and 45. It is shown again that the phenotype Pareto front

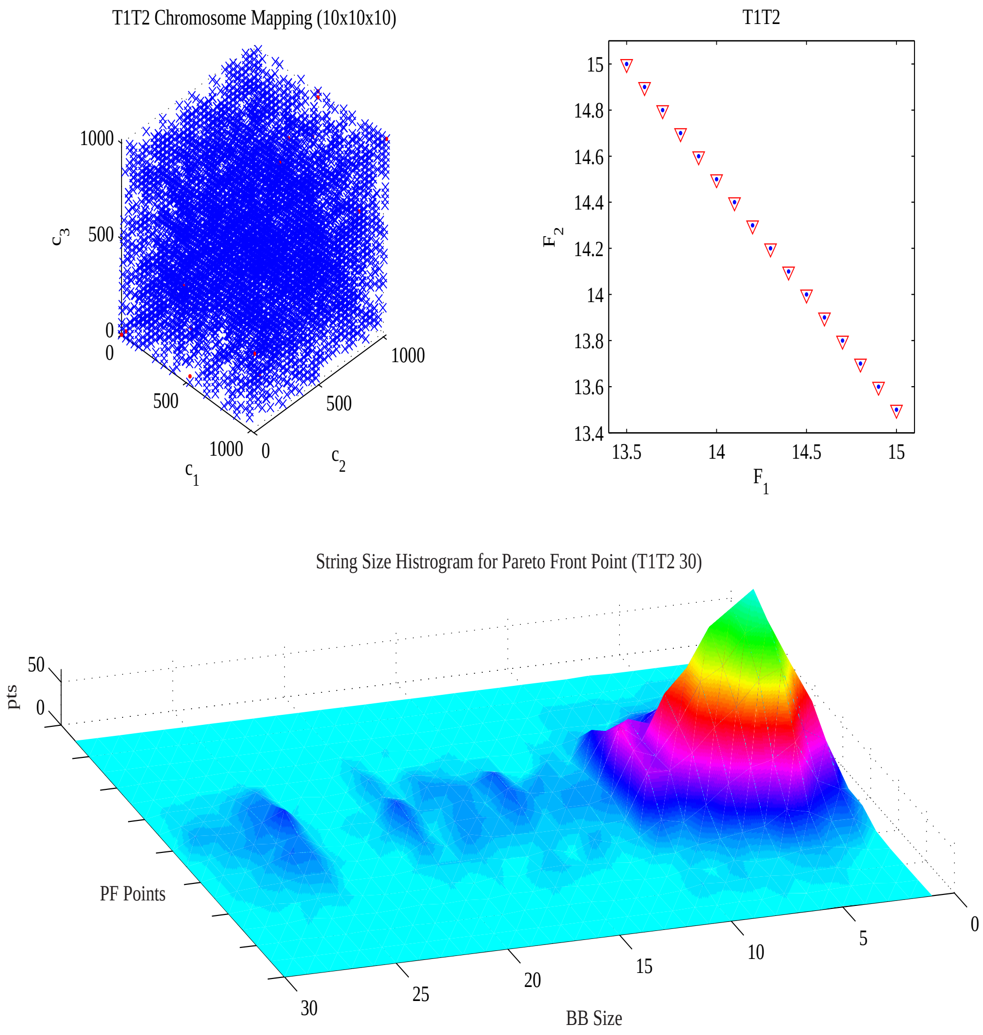


Figure 42: Illustrated within this figures is genotypic, phenotypic and the associated BB sizes found for each PF$_{known}$ vector for T1T2 30. The lower figure represents tallies for the size of BBs found for each PF vector. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. Notice that more solutions evaluating to PF vectors found are within the middle of the Pareto front than on the extremes. This is no coincidence – the number of solutions evaluating to PF vectors that can represent the inner part of the Pareto front are greater than the extremes.

is linear and equally spaced. The genotype has patterns; however, it is nevertheless not held to a small area. In fact, it looks well distributed. Finally, the BB sizes for

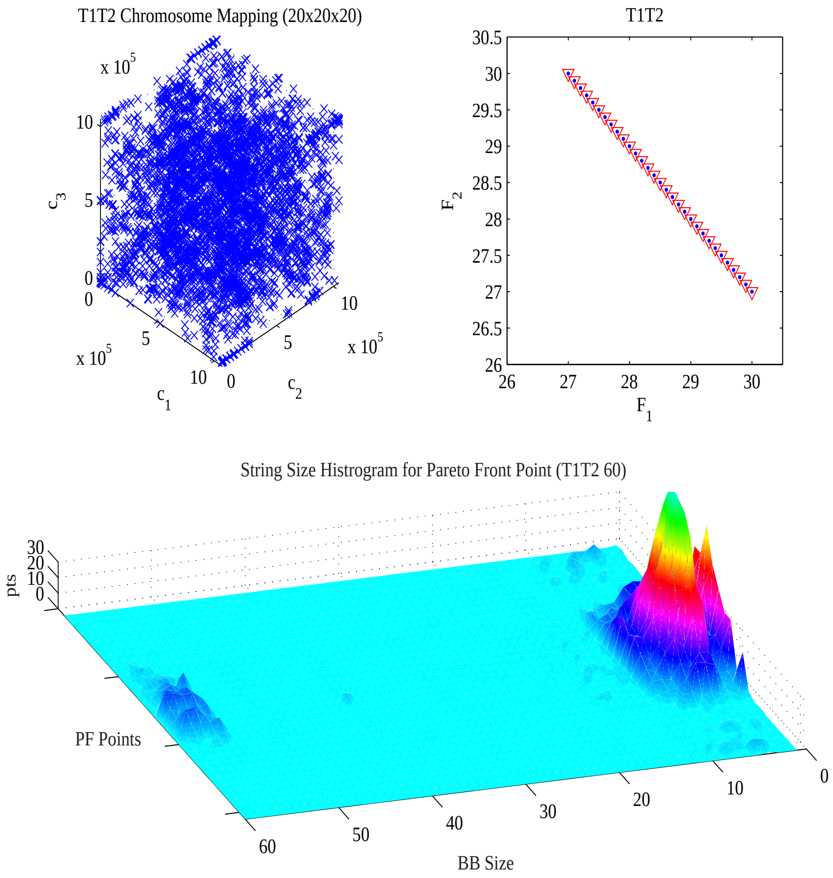


Figure 43: Illustrated within this figures is genotypic, phenotypic and the associated BB sizes found for each PF$_{known}$ vector for T1T2 60. The lower figure represents tallies for the size of BBs found for each PF vector. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. Notice that more solutions evaluating to Pareto front vectors within the middle of the Pareto front than on the extremes. This is no coincidence – the number of solutions evaluating to PF vectors can represent the inner part of the Pareto front are greater than the extremes. Notice that number of middle sized BBs[3] increase from T1T2 30 to T1T2 60.

both T3T4:30 and T3T4:60 show a variety of BBs sizes required to find solutions that evaluate to PF vectors across the entire Pareto front. In fact, there seems to be an area where the required BBs are longer. It is near the function 2 extremes where the longer BBs are required to find duplicates on the Pareto front. This is different than T1T2, where the found BB sizes are more uniform along the Pareto front.

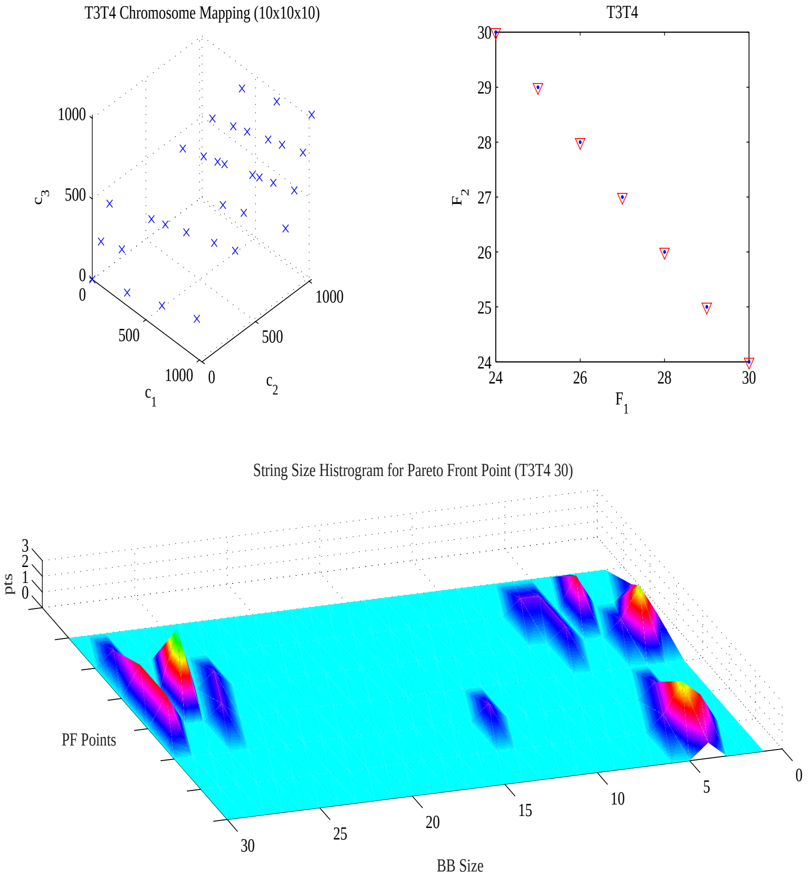


Figure 44: Illustrated within this figures is genotypic, phenotypic and the associated BB sizes found for each PF_{known} vector for T3T4 30. The lower figure represents tallies for the size of BBs found for each PF vector. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. Notice that more solutions evaluating to PF vectors are found to be within the middle of the Pareto front than on the extremes. This is no coincidence – the number of solutions evaluating to PF vectors representing the inner part of the Pareto front are greater than the extremes. For an example of how the PF vectors are partially ordered along the PF Points axis on the lower plot see Figures 104 and 105 on pages 322 and 323 respectfully.

The evolutionary trace reveals why the T3T4 is more difficult a problem, requiring variable sizes of BBs in various places along the Pareto front. It is apparent

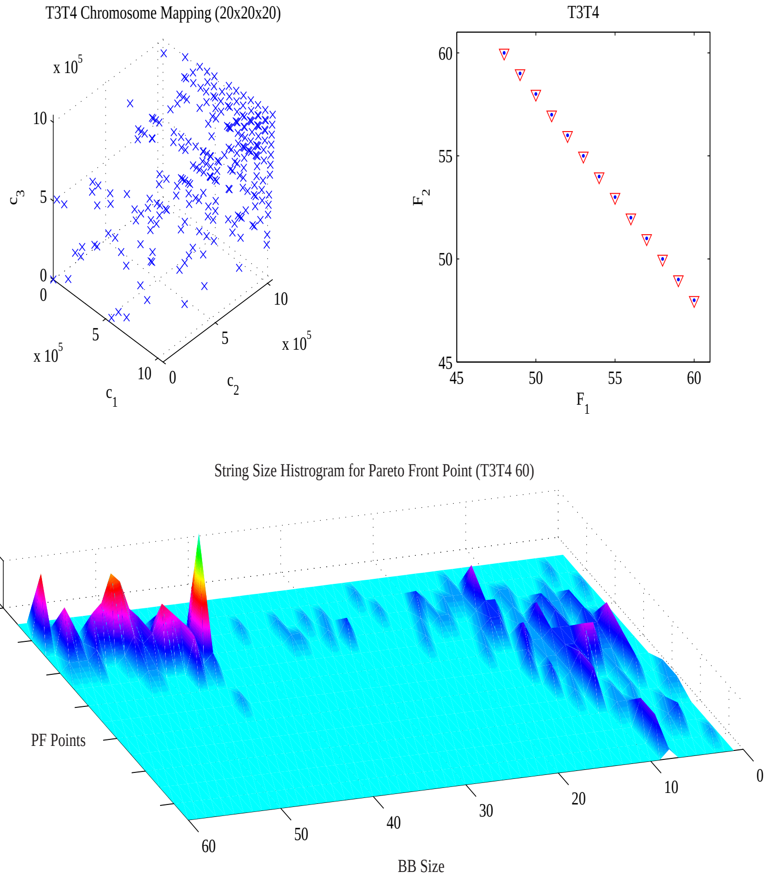


Figure 45: Illustrated within this figures is genotypic, phenotypic and the associated BB sizes found for each PF_{known} vector for T3T4 60. The lower figure represents tallies for the size of BBs found for each PF vector. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. Notice that more solutions evaluating to PF vectors are found within the middle of the Pareto front than on the ends. This is no coincidence – the number of solutions evaluating to PF vectors that represent the inner part of the Pareto front are greater than the extremes. Notice that number of middle sized BBs are increased from T3T4 30 to T3T4 60.

that in this particular type of problem, the MOMGA-II has difficulty in finding the correct BBs to solve the problem. It is believed that having multiple competitive templates allows for the MOMGA-IIa to overcome this limitation.

5.6 Summary

In conclusion, this experiment illustrates an explicit BB genetic algorithm's ability to solve deception problems. MOMGA-IIa's capabilities to explicitly find and use good multiobjective BBs (MOBBs) is illustrated. MOMGA-IIa can find the loosely linked bits in deceptive problems using its manipulation of BBs and has been shown to scale when multiple *good* competitive templates are selected. The MOMGA-IIa implicitly solves problems by identifying good MOBBs when iteratively selecting *good* competitive templates that partition both the geno- and pheno-type domains. Important aspects of this algorithm are the BB size, BB schedule and competitive template numbers (regular, inverse, and orthogonal). An important conjecture learned from this experimentation is the fact that, as the search space increases, it is important to increase the number of competitive templates - thus limiting the largest required BB size and ultimately limiting the population size generated by the MOMGA-IIa.

VI. Application: Protein Structure Prediction

This chapter discusses the analysis of replacing a highly computational fitness function with a neural network. The applied algorithm is MOMGA-IIa and the highly computational fitness function to replace is the energy calculation used in solving the Protein Structure Prediction problem (PSP) which is a Grand Challenge problem [35, 138]. Previous studies have shown the PSP's fitness function to be a major hindrance in finding good solutions, forcing the development of creative parallel computing programs (including an MOfmGA) [41]. Solving this problem involves finding a methodology that consistently and correctly determines the geometrical conformation of any fully folded protein without regard to the folding process. However, one must study the entire complexity of the problem to admire this *Gordian knot*[1]. For details of this problem and the definition of the fitness function see [18, 50–53, 65, 82, 83, 115–119, 156, 158–160, 164, 165].

In addition to the neural network fitness function emulation, it is also interesting to see how well a BBB, like MOMGA-IIa, solves this particular problem in single and multiobjective mode operation. Finally, a BB size study is completed on the multiobjective case study and efficiency gains are compared with parallel farming model findings.

6.1 Protein Structure Predication (PSP) problem

Protein structure prediction has been previously addressed using various computer modeling methods. For example, Chemistry at HARvard Molecular mechanics (CHARMm) version 22 has been used at the Air Force Institute of Technology to model protein potential energy when searching for good protein structures. Applying CHARMm is computationally expensive; therefore, an alternative to CHARMm

[1] A knot tied by Gordius, king of Phrygia, held to be capable of being untied only by the future ruler of Asia, and cut by Alexander the Great with his sword [72]

is needed to expedite search results. In this chapter, results of modeling CHARMm with a multilayered perceptron neural network conclude that this is a viable speedup tool, but effectiveness must be more carefully addressed. In building a neural network to emulate the CHARMm, many parameter settings are studied. One such parameter is the number of generations to train the neural network. Under and over training of the neural network using test data is a concern. In this study, special attention has been paid to the training of the neural network. Finally, the accuracy with which a neural network can mimic CHARMm and the time savings realized when using a neural network in place of CHARMm (effectiveness and efficiency) are investigated.

6.2 Force Field Approximation Using Artificial Neural Networks

Protein modeling is a major problem in bioinformatics studies. In this chapter, the focus is on reducing the computation time spent evaluating a protein's potential energy given a particular conformation. The idea is not to entirely replace the potential energy function with a neural network. The neural network is to work as a fast search parser. It is to identify good conformations quickly - steering the genetic algorithm through massive numbers of solutions while identifying a small subset of these for further analysis using the more computationally expensive, yet more accurate, fitness function.

The Air Force Institute of Technology (AFIT) has had interest in protein structure prediction since 1990. AFIT began by designing a protein potential energy function. AFIT's protein energy evaluation software is based on the Chemistry at HARvard Molecular mechanics (CHARMm) version 22 [17, 19]. To evaluate a protein's potential energy, the program requires that the protein be specified by dihedral angles. A dihedral angle is the angle formed by four atoms connected in a chain-like manner. The dihedral angle itself is developed when the two planes (formed by the first three and last three atoms in the chain) make an angle. Figure 46 illustrates a dihedral angle. This particular dihedral angle is called ψ angle.

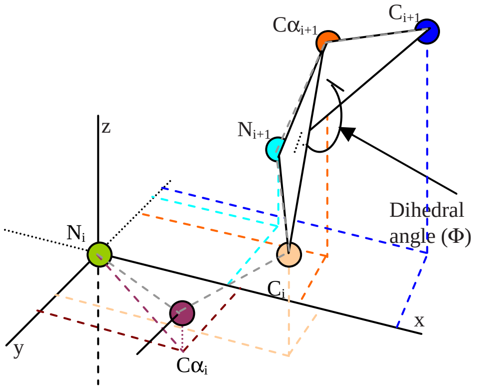


Figure 46: This figure illustrates the ψ angle in a small section of backbone atoms within a protein. This angle representation is the same at each C-N-C_α-C instance in any protein. To uniquely identify angles having the same symbol, angles are indexed according to how many times they appear in a particular protein. This method ensures that every angle is uniquely specified.

The entire process for specifying a protein is more involved than just identifying dihedral angles of a protein. Assumptions and domain information are also included in the specification process to make the search space more reasonable [41]. After the protein is specified with dihedral angles, each atom is locatable in a three-dimensional space allowing for a potential energy program, like CHARMm, to calculate energy between each atom. It takes approximately $6.8ms$ to calculate the potential energy for a single conformation on an Intel PIII $800Mhz$ machine. It would take more than a lifetime for us to expect to find the absolute lowest energy conformation for even a tiny protein (by calculating the energy for each conformation after discretizing all angles to 1024 bit degrees [41]). The protein used in this study is the Met-Enkephalin (MET). The fully specified MET has 24 adjustable dihedral angles. For each set of values for these 24 input angles, there is an associated potential energy value. As a graphical example of the energy calculation - the following is given:

- Energy Calculation

  Figure 47 illustrates numerous positions bonded atoms might have in a protein.

In addition, it plots how these configurations influence the potential energy calculation. The balls represent atoms and graphed curved lines (on the right of each position) identifying the interacting variable affecting the potential energy for that conformation. Each of these conformations occurs between each bonded atom within a protein. Each of these six functions make up the energy function used to calculate the *fitness* of a particular conformation.

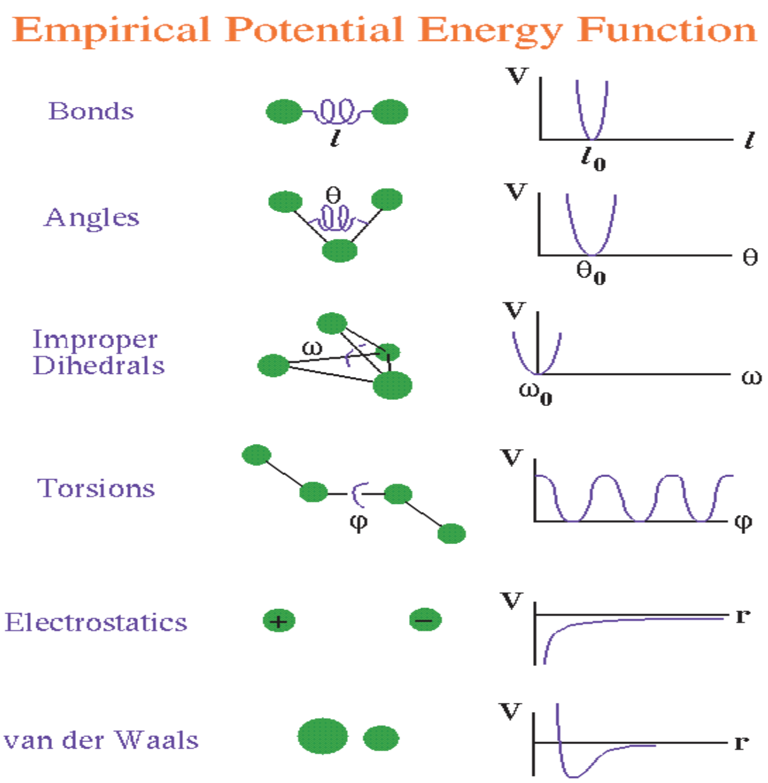


Figure 47: Graphical description of energy functions and how they are translated from physical atom-bond relationships to Potential Energy Functions.

First, other approximations methods are described. Then, the PSP fitness function is described. Following the PSP fitness function and other PSP type energy function descriptions, Section 6.5 on page 200 explains how the neural network input data is generated using the MOMGA-IIa in single objective mode. Section 6.6 on page 203 then describes the experiments conducted in finding the best parameter setting for the Multilayered Perceptron Neural Network (MLPNN) and Radial

Basis Function Neural Network (RBFNN). Results and analysis of using an ANN to replace CHARMM are then discussed. The chapter then discusses a single and multiobjective experiment using the MOMGA-IIa to analysis the problems BB sizes and results are related to other methods used within the field. Finally, conclusions are then drawn based on results.

6.3 Other Approximation Methods

Within this chapter, an artificial neural network is selected to replace the highly computational CHARMm fitness function used for solving the PSP problem. There are other mathematical models which can be used to approximate the relationship between a set of inputs (independent variables) and outputs (responses or evaluation resultants). These models can be called behavior models in some instances; however, their main purpose is not just to mimic behavior but to reduce the evaluation time for the exact computation (by a factor of 2-10). A few of these other approximation models are the following: response surface models, Kriging model, and variable complexity modeling [70].

Response Surface Models: This model approximates using simple algebraic functions (lower order polynomials). This model is effective for linear, quadratic, cubic, and quartic type fitness functions. Chaotic fitness landscapes cannot benefit from this approximation model because the maximum order model is quartic (4^{th}). Furthermore, the data points (response) are not guaranteed to be on the response surface. Closeness of the response relies heavily on the regression technique used in matching the input/output data with the model. [70]

Kriging model: The Kriging model attempts to account for a global and local response. The model's mathematical form is $y(x) = f(x) + Z(x)$ where f(x) is a function representing a *global* response of the design space and Z(x) is a function that represents the *localized* response of the design space at that particular global location. Creation of an accurate Kriging model can be computationally expensive. [70]

Variable complexity model: A variable complexity model has different fidelity code: high and low. The low fidelity code is less accurate, but has a shorter run time and the higher fidelity code is more accurate but has a higher run time. The high fidelity code might even be the exact code or function. This actually might work with the PSP problem because some of the fitness functions can be easily removed from the function call. However, this model was not used because some research already had been accomplished using this type of approximation model by Steven Michaud in 2000 [163]. [70]

6.4 Evolutionary Computation (EC) techniques applied to PSP problem

In 1991, AFIT launched an effort to solve the PSP problem using EC techniques. The first EC techniques applied were the simple GA (sGA) and messy GA (mGA) [154]. Later, binary and real valued hybrid GAs were tested. Next, Linkage learning and Immunological Computation algorithms were used. Finally, the fmGA and generic Multiobjective fmGA(MOfmGA) were found to be better than the other GAs at finding good protein conformations [41]. In addition to applying these different EC techniques to solving this problem, most techniques were also implemented in serial, parallel, single and multiobjective. This effort uses the MOMGA-IIa in single objective mode. In Chapter III on page 71, a detailed description of MOMGA-IIa is given.

Search algorithms rely solely upon the ability to be able recognize good solutions. For the PSP problem, this recognition comes in the form of a fitness model, energy function or fitness function. A solution's objective values found to be nondominated by one model/function may be found to be weaker by others. This is why it is extremely important to have the most suitable fitness function or model for the problem. This suitability is particularly difficult to achieve for the PSP problem. Many factors are involved in choosing a suitable energy function. Potential energy [120], quantum mechanical energy, chemistry of the protein [6, 86, 87, 192, 224], empirical force fields energy, energy surface with the largest statistical weight [86] and entropy [169] are just a few of the fitness function ingredients that may be used.

Essentially, the selected CHARMm energy function, E_i, sums the internal terms or bonded atom energy and external terms or non-bonded atom energy of a particular protein in a specific conformation.

$$E_{total} = \sum_{(i,j)(connect)} E(bonded) + \sum_{(i,j,h,\ldots,g)(!connect)} E(non - bonded) \tag{62}$$

Bonded energy is the sum of bond stretching, bond rotation, bond bending, improper torsion and hydrogen bonding energy reduction between each connected or bonded atom.

$$E_{stretching} = \sum_{(i,i+1)} K_b(b - b_0)^2 \tag{63}$$

where K_b is the force constant determining the strength of the bond, b is the actual bond length and b_0 is the ideal bond length. The bending energy is similar to that of the stretching energy where K_θ is the force constant, θ is the actual measured angle, and θ_0 is the ideal angle. This is primarily a penalty function and represented in Equation 64.

$$E_{bending} = \sum_{angles(i,j,h)} K_\theta(\theta - \theta_0)^2 \tag{64}$$

The third term in the bonded energy calculation representing a reduction in the Van Der Waals term for the interaction between the hydrogen atom and the acceptor atom is Equation 65 [20] .

$$E_{hydrogen} = \sum_i \left(\frac{A'}{r^i_{AD}} - \frac{B'}{r^i_{AD}} \right) \cos^{\mathcal{N}}(\theta_{A-H-D}) * \cos^{\mathcal{N}}(\theta_{AA-A-H}) \tag{65}$$

The fourth term in the bonded energy calculation representing the torsion angle potential function which models the presence of steric barriers between atoms separated by three covalent bonds (1,4 pairs) is shown in Equation 66.

$$E_{torsion} = \sum_{(i,j,h)\in D} K_{\theta}(1 - cos(g\phi)) \tag{66}$$

$$E_{Improper-torsion} = \sum_{\omega} K_{\omega}(\omega - \omega_0)^2 \tag{67}$$

Equations 63, 64, 65, 67 and 66 make up the energy for bonded atoms:

$$E_{bonded} = E_{torsion} + E_{bending} + E_{hydrogen} + E_{stretching} + E_{Improper-torsion} \tag{68}$$

The final terms for the calculation of energy are the non-bonded related terms, electrostatics, water-water interaction and Van-Der-Waals. These terms may be combined into the following sum:

$$E_{lennard-jones} = \sum_{(i,j)\in\mathcal{N}} \left[\left(\frac{A_{ij}}{r_{ij}}\right)^{12} - \left(\frac{B_{ij}}{r_{ij}}\right)^{6} \right] \tag{69}$$

Constants A and B are interaction energy using atom-type properties. D is the effective dielectric function for the medium and r is the distance between two atoms having charges q_i and q_k.

$$E_{electrostatics} = \sum_{(i,j)\in\mathcal{N}} \left[\frac{q_i q_j}{D r_{ij}} \right] \tag{70}$$

$$E_{water-water1} = \sum_i K_i(r_i - r_{i0})^2 \tag{71}$$

$$E_{water-water2} = \sum_i K_i(\theta_i - \theta_{i0})^2 \tag{72}$$

Contributions of water-water constraints of distance and dihedral angles are shown in Equations 71 and 72, respectively. Furthermore, the entire contribution of non-bonded energy is given by equation 73.

$$E_{non-bonded} = E_{lennard-jones} + E_{electrostatics} + E_{water-water1} + E_{water-water2} \tag{73}$$

The CHARMm energy function is computationally expensive. For example, the search landscape with this model for the polypeptide [Met]-Enkephalin has been experimentally generated using parallel random search [163] over an immense number of dihedral angles. In this case, the PSP landscape structure is an extremely dense set of points at relatively high energy levels over a wide energy band versus the range of possible dihedral angles. Few points are close to the minimized energy levels. These points are reflected at the bottom of narrow phenotype energy *wells* with cascading sides. These optimal points are difficult to find even with integrated local search techniques.

Within Table 33, a comparison of CHARMm, AFIT CHARMm, Amber, ECEPP, and Optimized Potentials for Liquid Simulations (OPLS) is illustrated. Notice that CHARMm covers each one of the possible energy equations; however, AFIT's CHARMm has reduced this function due to the insignificance of these other forces. AFIT's version of CHARMm was used in our investigations because it has been found to be a valid model [147].

In addition to these energy models many other models, have been used in other studies. The Random Energy Model (REM) was applied to the PSP problem by Bryngelson and Wolynes [22]. This energy model was originally used in spin glass theory [67]. Other such fitness function models have been applied to the PSP problem using enthalpy [169], conformational entropy, hydrophobic/hydrophilic [184], and distance matrix models employing Frobenius norm of differences, Hoeffding inequality keeping corrected distances for fitness function terms [184], and ring closure on local conformations [87]. Moreover, all these models have the same theme in trying to define the properties a protein has when folded. Currently, there is no single model that has prevailed and thus, the search for the perfect energy model continues.

Table 33: Comparison of common energy functions used in solving the PSP problem.

	Eq# ⟶	63	64	66	67	69	70	65	71	72
Acronym	**Name**									
CHARMm	Chemistry at Harvard using Molecular Mechanics	✓	✓	✓	✓	✓	✓	✓	✓	✓
CHARMm at AFIT [64, 120, 190]		✓	✓	✓	✓	✓	✓			
Amber	Assisted Model Building with Energy Refinement	✓	✓	✓		✓	✓	✓		
ECEPP/3	Empirical Conformational Energy Program for Peptides			✓		✓	✓	✓		
OPLS	Optimized Potentials for Liquid Sim.			✓		✓	✓			

6.5 Generating Data

It is important to have a balanced set of good and bad solutions to train the neural network in hopes of getting a system to approximate the energy function well. Unfortunately, ensuring that a researcher has a group of data representative of both good and bad solutions is challenging because the PSP problem has a sparse distribution of good conformations in the search space. So, the use of the MOMGA-IIa is required to allow for a fast accumulation of good solutions. If a random set were taken, the data set might have one good solution for every $100,000$ solutions evaluated.

The first data set is formed by letting MOMGA-IIa generate conformations (protein structures) for evaluation. After evaluation of each conformation, the angles and associated energy values are written (24 angles plus the fitness value calculated by CHARMm) to a file. As expected, this data set is not a realistic representation of the entire data set because the MOMGA-IIa finds good solutions and discards bad ones. Figure 48 illustrates a histogram of the records found using the GA.

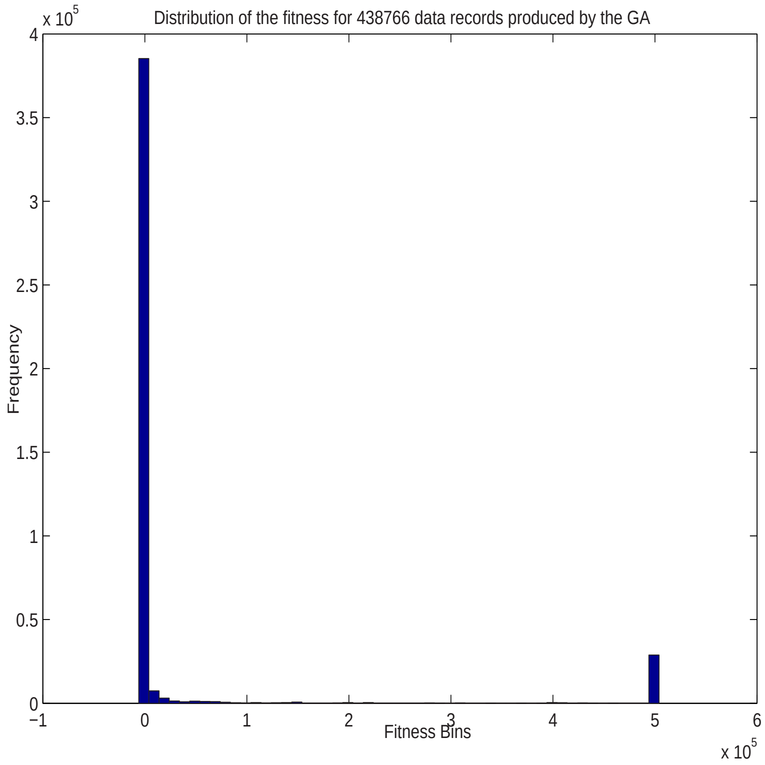


Figure 48: Data records generated by the genetic algorithm and then categorized by fitness value. The highest frequency of records is reportedly near or at zero. Because the fitness function is a minimization function, this figure is illustrating how MOMGA-IIa only acquires and populates itself with good solutions.

In order to get a better representation of the record set, the GA is allowed only to randomly evaluate conformations. This resulted in a realistic data set, having mostly bad solutions. Figure 49 illustrates the energy distribution of data records. At this point, there is a record set larger than what Matlab can load into memory. Over 43,000 data records are from the MOMGA-IIa produced data set and over 4,000,000 were from the random selection data set.

After gathering the training and test data, it is important to select useful data in training and testing possible neural network configurations. Several preliminary classification tests using a RBFNN were run on randomly selected data records from

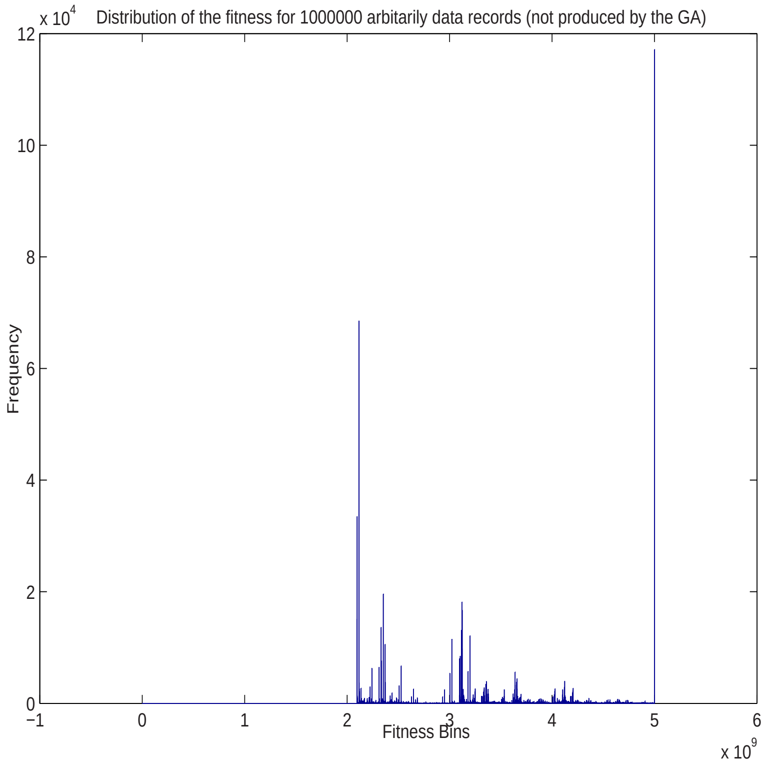


Figure 49: Data records generated randomly and then categorized by fitness value. The highest frequency of records is reportedly near or higher than 5×10^9. This illustrates what the actual search space is like.

each set yielding extremely bad mean squared error results (+50). In order to work toward having a system with a suitable mean squared error, it was then determined that this experiment should concentrate on data records that are localized to a certain angular area (protein structure wise). Therefore, records closest to the root mean squared difference from the average dihedral angle values should be ranked and the top $1,000$ records were selected. In addition, the top 200 records were used for parameter determination to reduce run time during experimentation. See Figure 50 for an illustration of the fitness distribution of these 200 selected records.

Finally, the data is normalized to zero mean unit variance. This included normalizing the output data. Originally, the log base 10 of the output data was taken to keep the variance of the energy values from making the normalized output

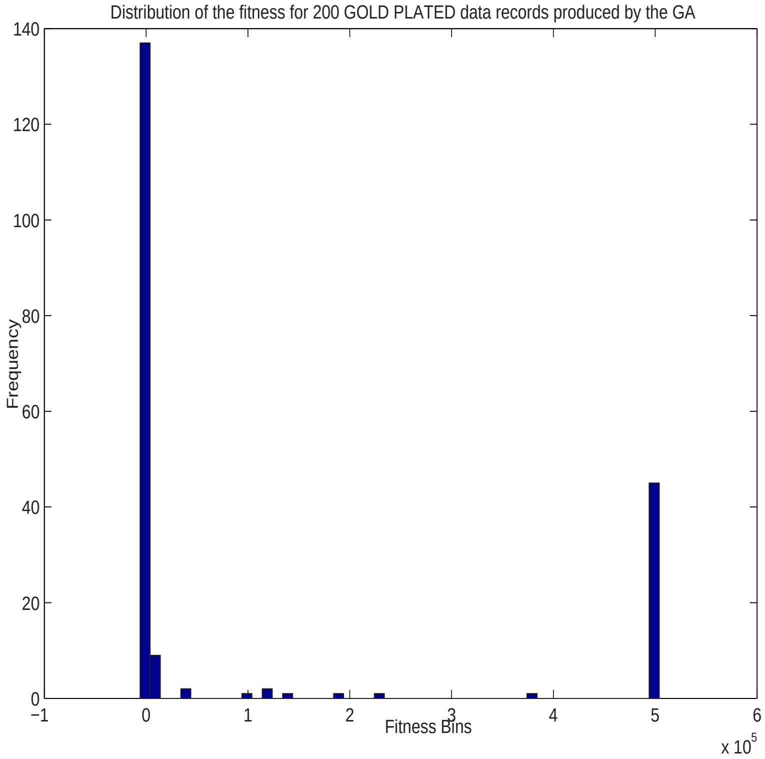


Figure 50: 200 Data records categorized by fitness value. The highest frequency of records is reportedly near or higher than 5×10^9. This illustrates the distribution of our working data set.

data small; however, after taking only records from the GA produced output, this was not necessary. Therefore, the output data was also normalized to zero mean and unit variance without first taking the log base 10 before normalization.

6.6 Experimentation

Two neural networks are studied to replace the potential energy program. The goal is to gain efficiency and maintain a system that has 99% correct classification. The first neural network is a Radial Basis Function Neural Network (RBFNN) and the second is a Multilayered Perceptron Neural Network (MLPNN). Both networks have 24 inputs to 1 output. In the case of the MLPNN, many experiments were run to select the best parameters for optimized results. According to [14], an attempt to get an estimate for the number of neurons required for 99% effectiveness can be made.

$$N_{min} \cong \frac{W}{\epsilon} = \frac{24 * Neurons + Neurons + 1}{0.01} \tag{74}$$

Equation 74 is the relationship between the total number of weights (W), the number of data records (N_{min}), and the fraction of correctly classified fitness values ($1 - \epsilon$) [14]. This equation is mainly for discrete output values or categorized outputs; therefore, it can only be applied loosely to this problem. When applying the equation to this experiment having $1,000$ data records coupled with a 99% correct classification requirement, this conclusion is that each neural network should have only one neuron. Being that this number neurons is not realistic to use in a neural network, it is decided to used 3, 4, 10 and 100 neurons in each network for this experiment. Using Equation 74, the theoretical fraction of correctly classified fitness values to be 90, 88, 74, and 0 percent ,respectively, can be calculated.

6.7 Parameter Determination

Results of testing for the best training parameter can be found in Figure 51 on page 205. Training parameter **traingdx** is selected as training method for these data records. Illustrated in Figure 51 on page 205 is the best half of the parameters – containing the results of parameters (*a*-*f*).

The results of the experiment showed that both the hyperbolic tangent sigmoid transfer function (*tansig*) and Log-sigmoid transfer function (*logsig*) were both good choices for this data set; however, *tansig* was the overall better choice having a lower means squared error and variance.

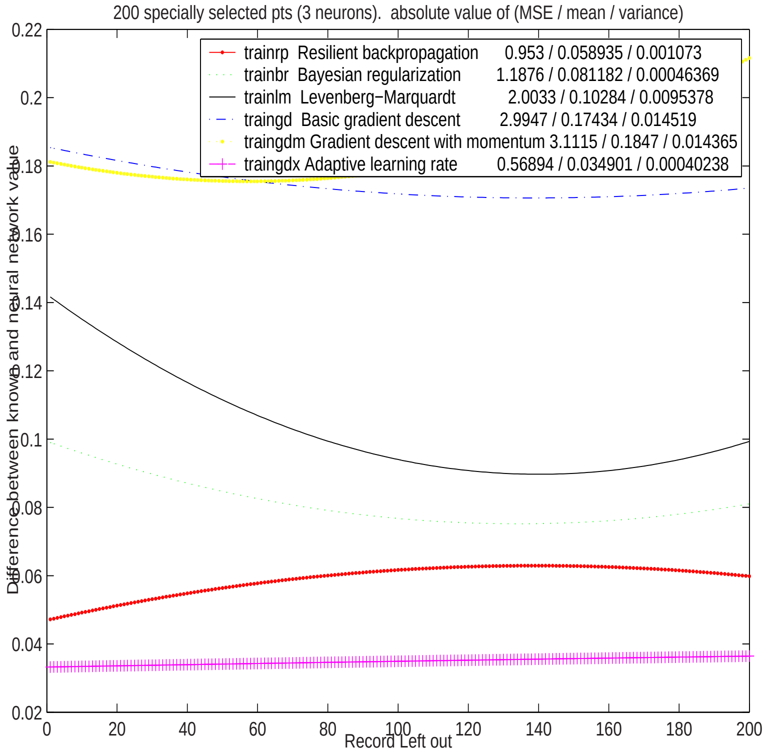


Figure 51: Experiments testing the training parameter for 200 data records and 500 epochs. The x-axis is the record number left out in the leave one out, train, test-routine.

6.8 Approximating the Force Field Function with ANN

This section primarily discusses the force field function being approximated with an ANN. Advantages and disadvantages are covered, as well as the description of analyzing what kind of ANN to choose for application.

6.8.1 Training Generations for a MLPNN. There are many benefits to using multilayered ANNs in emulating a high run time fitness function. First, multilayered ANNs have nonlinearity problem solving capabilities. The artificial neurons can be programmed to be either linear or nonlinear depending on what type of problem is being solved. In some cases, one can have a mixture of neuron types. Secondly, multilayered ANNs have a popular paradigm of learning, called supervised learning, where a training set teaches the multilayered ANN by modifying the synaptic weights making up the connections of the neural network. A third advantage to using neural networks is adaptability and their ability to cope well with noisy data. multilayered ANNs, by design, have a built-in capability to adapt their synaptic weights to changes. Finally, multilayered ANN have an ability to respond quickly (fast calculation) to a changing environment. Studies on systems producing high amounts of data quickly show that a neural network can process this data in a timely manner – keeping up with the high influx of data. One example of this is studies on communication network congestion control that found the structure of a neural network is such that it can calculate (base on inputs) quickly and determine an operating mode or classify the pattern just as fast. [29, 180]

Drawbacks for using NNs come in the form of *training* the network and determination of the size and type of the network to use. Fortunately, studies have been accomplished where estimates are derived of how many neurons should be used in a neural network. These estimates are dependant on the required accuracy of the network, restricted to two layer networks, and are normally designed to work for a particular type of problem (usually integer based) [14]. Unfortunately, determining the right amount of training is still required for the best results. Over-training, or over-fitting, a NN is just a harmful as under-training. Figure 120 on page 427 illustrates an example of what happens when a researcher over-trains a NN. People in the pattern classification field using NNs stop the training when a validation set reaches its first minimum [69]. Finally, the data used to train the NN must be somewhat accurate. It is true that a NN works well with noisy data; however, if the data set used to train the NN is not even close to the actual pattern sought to imitate, then the trained NN is ill-trained and is not expected to perform well.

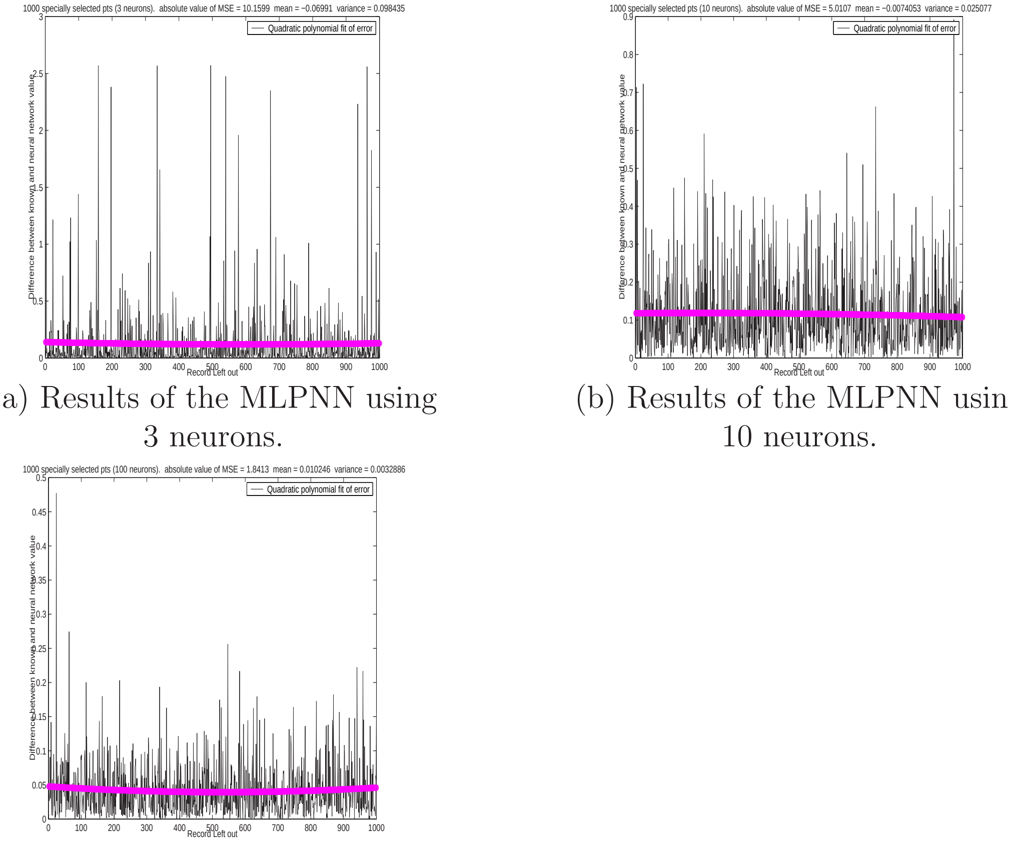
In this section, tests are conducted with a three layered MLPNN. The first two layers have 25 neurons each and the third layer has a single linear neuron. According to the results listed in Table 87 on page 424, selection of the number of generations to train the two-layered 25-neuron neural network should be between 300 to 500 generations. Notice that the table is not completely filled with values. These test results have not completed or need to be tested further to pin-point the best number of generations to train the network.

6.8.2 Results and Analysis. A successful analysis of any added feature solving a problem must include both a comparison between effectiveness (indicating that the new algorithm now has either better precision or accuracy) and efficiency (indicating that the new algorithm now completes sooner than it did before). Mechanisms that excel under both criteria are a positive change and should be adopted. The following sections evaluate how the neural network fairs in both the effectiveness and efficiency categories when replacing the CHARMm energy fitness function in the MOMGA-IIa. It should be noted again that the plan was not to replace CHARMm completely, for the algorithm must still use CHARMm to search in areas identified by the neural network as having possible good conformations. In addition, in some cases it may be acceptable to accept a lower effectiveness from the neural network in order to speed up the search process.

Effectiveness: The first experiment discussed is for the RBFNN implementation. The RBFNN software tuned the ANN for the user [66] automatically, making parameter selection null and void. The RBFNN found the best effectiveness results overall with a mean squared error of 0.54. See Table 34 on page 207 for the summary of results. Notice also that the RBFNN has the lowest variance.

Table 34: Summary of Effectiveness results

Neurons	Training	Transform	MSE	MEAN (μ)	Var (σ)
25	**RBF**	**RBF**	**0.54**	**0.0171**	**0.000001**
3	traingdx	tansig	2.10	0.0461	0.0228
3	trainrp	tansig	10.16	0.0699	0.0984
4	trainrp	tansig	10.27	0.0559	0.1024
10	trainrp	tansig	5.01	0.0074	0.0251
100	trainrp	tansig	1.84	0.0102	0.0033
100	traingdx	tansig	1.52	0.0396	0.0007



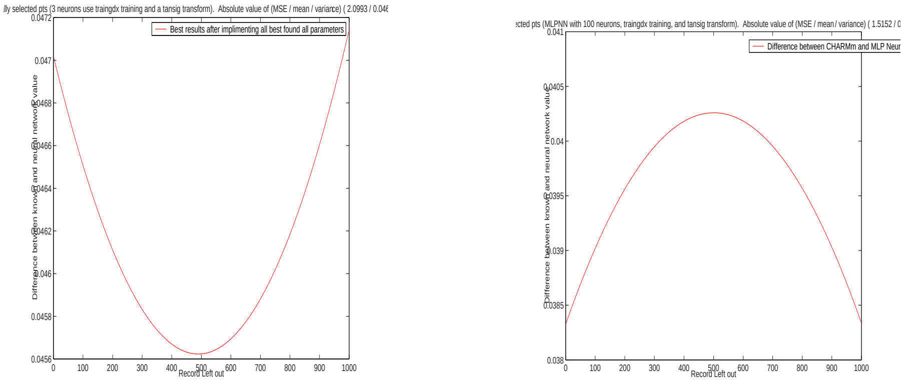
(a) Results of the MLPNN using 3 neurons.

(b) Results of the MLPNN using 10 neurons.

(c) Results of the MLPNN using 100 neurons.

Figure 52: Results of the Multi Layered Perceptron Neural Network using 3, 10, and 100 neurons, 2000 training generations, and 1000 “gold plated” data points. Figures a and b illustrate the use of a low number of neurons in the neural network and also results in a high mean squared error. Figure c illustrates the results of a neural network having 100 neurons and a lower (approximately 1.8) mean squared error. Notice the larger data set (1000) utilized in this experiment and the larger number of training generations (2000). This experiment illustrates a more realistic experiment than the earlier ones having only 200 data records.

It is also prudent to test the validity of Equation 74 on page 204 from [14]. Therefore, several ANN having different ranges of neurons are tested to find the best configuration and check the neuron formula for accuracy. Figures 52.a, 52.b, and 52.c each have 3, 10 and 100 neurons respectively. The four-neurons graph is not shown because it is similar to the three-neurons graph. Each also trained the network using 1,000 gold plated data records, 500 epochs, hyperbolic tangent sigmoid transfer function and used a resilient training parameter. It is interesting that in



(a) Results of the MLPNN using 3 neurons.

(b) Results of the MLPNN using 100 neurons.

Figure 53: Results of the Multi Layered Perceptron Neural Network using 3 and 100 neurons, trangdx, tansig, 2000 epochs, and 1000 good points.

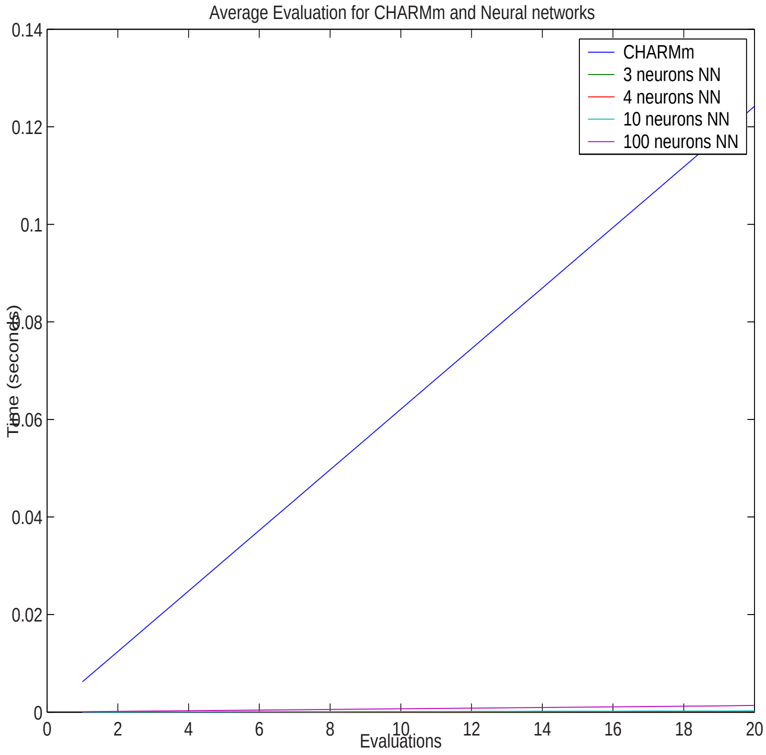


Figure 54: Run time after 20 evaluations using each configuration in Table 35.

Figure 52.c, it shows that the neural network having 100 neurons outperforms the three neuron case contradicting Equation 74. Results from each of these experiments can be found in Table 34 on page 207.

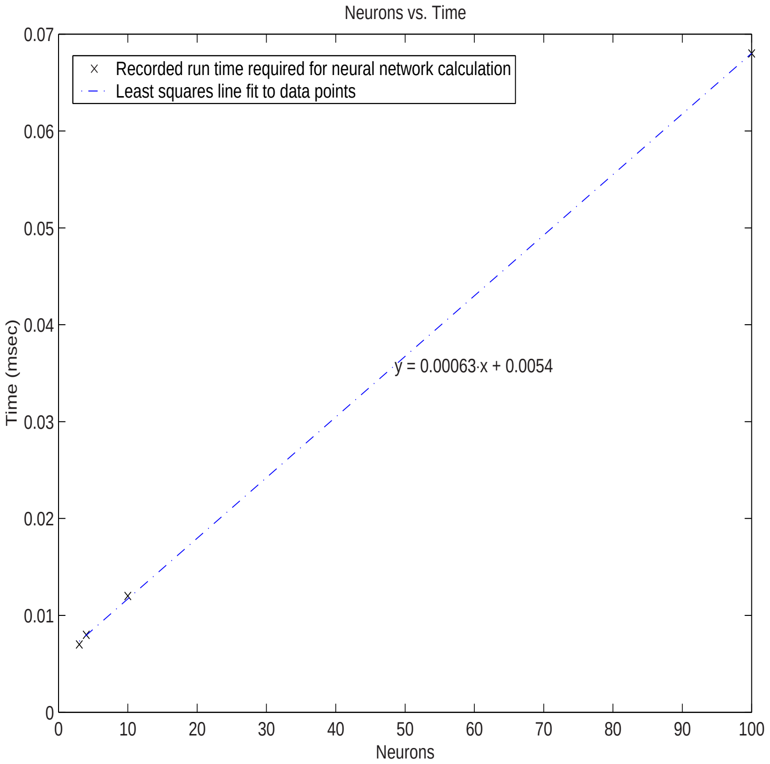


Figure 55: Least squares line fit to the data points showing the increase in time as the number of neurons are added.

The final MLPNN experiment tested an aggregate configuration of validated parameter settings found in previous experiments. Figure 53.a illustrates the results of this experiment. While it was able to perform better than previous tests, it failed to outperform RBFNN. One final test was conducted to compare MLPNN using 100 neurons with RBFNN; even in this configuration MLPNN was unable to outperform RBFNN. Figure 53.b illustrates results of such a test.

Table 35: Average time for evaluations on PIII 800Mhz (Efficiency results)

Application	msec
CHARMm	6.211
1 Layer 3 Neurons	0.007
1 Layer 4 Neurons	0.008
1 Layer 10 Neurons	0.012
1 Layer 100 Neurons	0.068

Efficiency: Efficiency measurements were taken from run times of the original MOMGA-IIa using the CHARMm code versus a mach-up neural network code in place of the CHARMm function. Table 35 on page 210 summarizes the results. The neural networks have been shown to provide an increase in efficiency over the CHARMm code. Figure 54 on page 209 graphically illustrates this speedup. Notice the 25 neuron RBFNN is not listed. It can be approximated to take 0.021ms, which is easily more efficient than the CHARMm code and better than speedup found using a parallel farming model [53]. Additionally, Figure 55 on page 210 is provided to illustrate the increase in calculation time spent as the number of neurons is added to the neural network.

6.9 ANN Study Summary

The results of this experiment are inconclusive in that a suitable approximation to the CHARMm fitness function may have been found; however, more testing needs to be completed to ensure that the difference between outputs of the neural network is acceptable compared to the CHARMm. Large differences in the approximation function may mislead an EA into areas of the search space that do not have good solutions; therefore, effectiveness must be accurate. Future testing needs to show that the neural network can identify areas in the search space where, once evaluated, good protein conformations are confirmed by empirical methods.

6.10 Single and multiobjective PSP experimentation

An interesting case study of solving the PSP problem with a fmGA and MOfmGA can be found in [41]. Interesting enough, there is also a study of what sized BBs are required to find the best solutions; however, this study is done by varying the important cut BB size within the algorithm – it is not a direct study of what sized BBs make up these good solutions (which consequently is done for the first time in this PhD thesis). The following two sections describe the single and multiobjective results found solving these problems using MOMGA-IIa.

Single objective experiments are consistent with previous findings using a fmGA [41]. This is without using any additional features for seeding the population with good individuals or limiting the range of variables by implementing a feasibility or range mapping for each dihedral angle. Thus, the single objective implementation of MOMGA-IIa is similar to running just the fmGA with no frills.

The multiobjective experiment is feature experiment in this section. The fitness function is decomposed into two meaningful subsets: Physics (Objective 1) and Chemistry (Objective 2). The Physics subset represents the non-bonded energy functions of the CHARMm fitness model. This objective targets the coulomb interactions and steric anatomy of the protein are kept correct. Additionally, it can be understood that objective 1 focuses on keeping good topology of the protein. Whereas, the Chemistry subset represents the bonded energy functions of the CHARMm fitness model due to the *fixed* model characteristics. This objective helps keep the classical description of Chemistry for a protein correct. Moreover, dihedral angle, bond lengths, and bond angle energies are optimized with this objective. For a complete breakdown of the objective functions into energy functions see [41].

6.10.1 Evolutionary Trace for BB Visualization. One important result to dissolving the fitness function into a MOP is that it allows for an evolutionary trace to be performed for BB visualization and insight to needed BB sizes for finding Pareto front vectors.

Met-Enkephalin:

The first evolutionary trace is accomplished on the Met-Enkephalin protein. Figure 56 presents these results. One detail to notice is that genotype only presents visualization for 150 bits of data. This is due to the limitation of Matlab's binary to decimal function requiring less than 53 bits for evaluation. The graphic illustrates that there are regions on the Pareto front requiring similar BB sizes. It is also important to note that the Pareto front point closest to the *corner* of the Pareto front requires a large BB, this particular position on the Pareto front point must represent an edge where slight changes in the genotype cause significant changes in the phenotype. Finally, BB sizes between 80 and 200 are absent indicating that either extremely large BB sizes or BB sizes less than 80 are required to find good solutions to this MOP.

Polyalanine$_{14}$*:*

The second BB visualization is for the Polyalanine$_{14}$ protein. Figure 57 represents the results of this evolutionary trace and BB size analysis. Results for this particular protein are similar to the MET-Enkephalin BB size graph in that there are *patches* or regions along the Pareto front that require larger BB sizes. In addition, points found close to the *corner* of the Pareto front require these large BB sizes indicating this *edge* referred to above. Again, there seems to be a lack of BB sizes between 100 and 500. This could be an indicator that BB sizes for this MOP are either very large or less than $\frac{1}{5}$ of a full chromosome length.

Previous BB size experiment for Polyalanine$_{14}$*:*

Previous experiments attempting to find the best BB sizes for a particular protein are presented in Figure 58 on page 216 [41]. BB sizes of 30-32 yielded the best results for POLY. Although this BB size is specific for POLY, it should apply to other proteins having an alpha helix structure. Additionally, BB size 30-32 yielded the best overall fitness value found during all of the BB testing of -140 kcal, which is in the neighborhood of the accepted CHARMm fitness for this protein.

These previous results are not different than the experiments conducted because BB sizes larger than 40 were not tested in the past experiments. Furthermore, if previous experiments were to increase the BB size to the sizes found by the method purposed, the computation time would be costly. In fact, those researchers may be

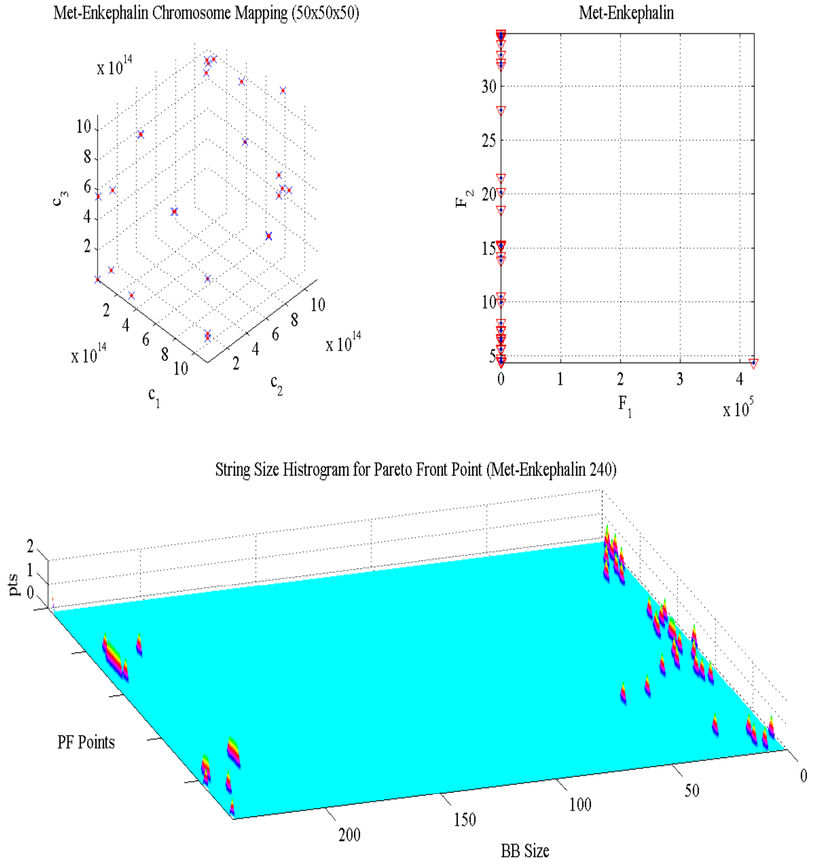


Figure 56: Illustrated within this figure is genotypic, phenotypic and the associated BB sizes found for each PF$_{known}$ solution for MET as an MOP. The lower figure represents tallies for the size of BBs found for each solution. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. These are interesting results because BB sizes are variable throughout the Pareto front with areas concentrated with larger BB size requirements.

still waiting on the results of experiments having larger BB sizes because the population size for the fmGA would increase exponentially when the BB size is increased - as previously discussed in Chapter III on page 71. However, if BB sizes larger than 100 are used, it is quite possible the results would indicate a lower BB size is more effective because this is indicated by the BB size analysis done here.

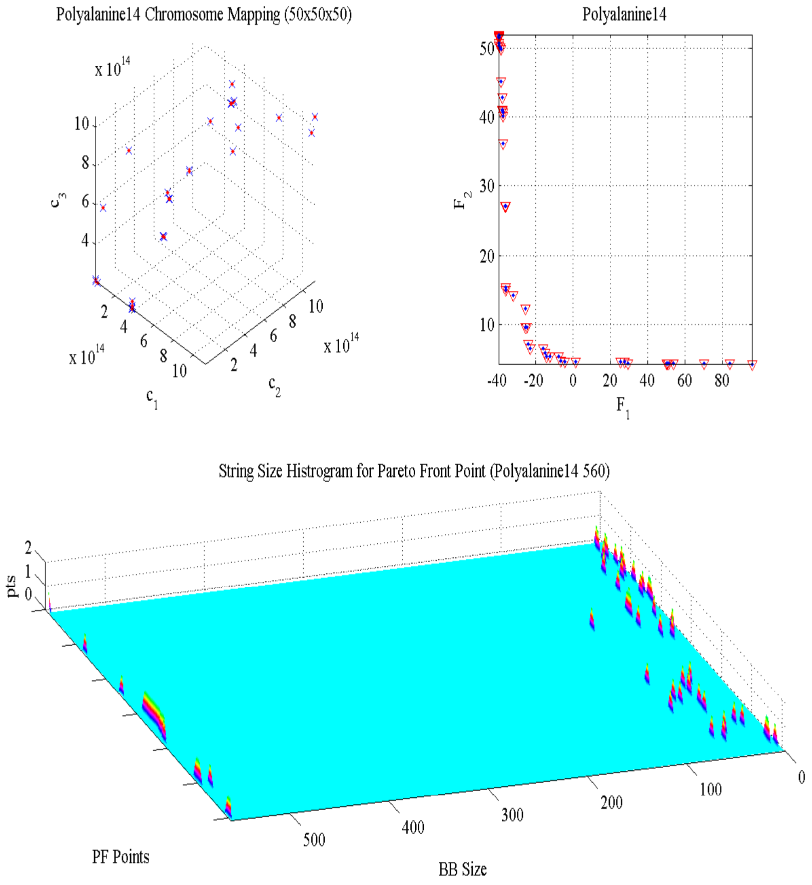


Figure 57: Illustrated within this figure is genotypic, phenotypic and the associated BB sizes found for each PF$_{known}$ solution for Polyalanine$_{14}$ as an MOP. The lower figure represents tallies for the size of BBs found for each solution. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. These are interesting results because BB sizes are variable throughout the Pareto front with areas concentrated with larger BB size requirements [41].

6.11 Summary

This chapter studied the advantages and disadvantages of approximating a computationally expensive fitness function with an artificial neural network. Of the neural networks studied, it is decided that the RBFNN is the best choice among the approximation method tested; however, other approximation methods not tested in

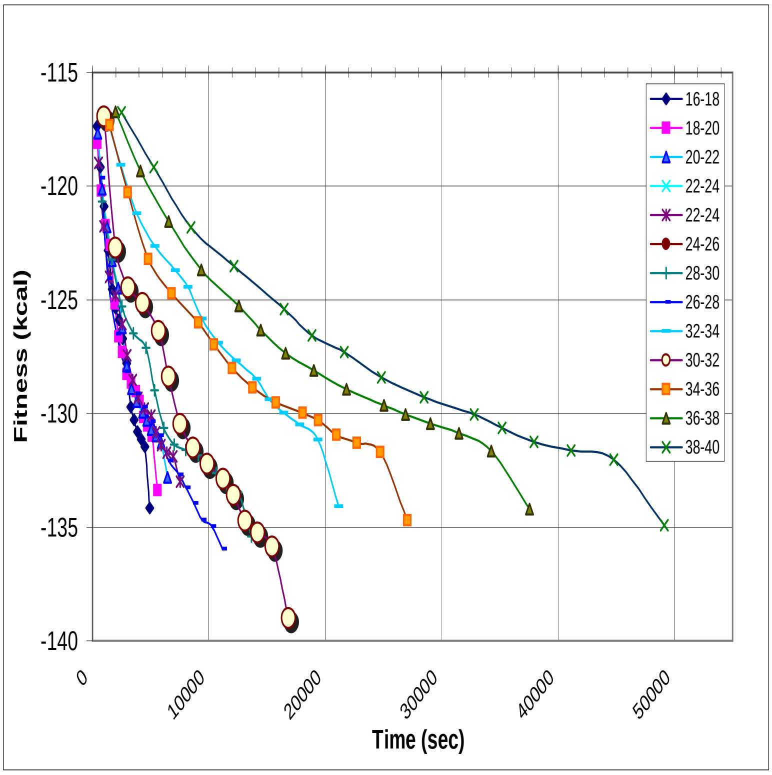


Figure 58: Time versus BB Test plot of BB sizes and associated best fitness found from each experiment.

this chapter may be a better choice. Certainly, it is important to reduce the calculation cost a function that must be evaluated over and over again within the algorithm; however, at the same time, an evolutionary computation researcher must be aware that approximating the fitness function may actually mislead the algorithm into regions that have no optimal solutions to find. Finally, an evolutionary trace to study the BB sizes required to find good solutions for the PSP once transformed into an MOP is also given. Findings from the trace are similar to those conducted earlier using a different technique. The advantages of the new technique are accentuated by the fact that all BB sizes can be watched during one experiment where the old technique required many experiments to be run for evaluation of BB size effectiveness. The next chapter continues the study of the MOMGA-IIa as an effective explicit

BB MOEA by solving the multiple quadratic assignment problem and conducting evolutionary traces on one MOP for a BB size visual analysis.

VII. Application: Organic Air Vehicle Flight Formation Problem

Intelligence gathering is a crucial part of military operations. Today's urban warfare has defined a new battleground for conducting military operations, making the identification of military targets on the battlefield more difficult. Further, sensitivity toward destroying non-military targets is extreme – normally followed by negative public relations. Incidents like the accidental bombing of the Chinese embassy in Belgrade [73], the wedding party bombing [36] in Kabul, Afghanistan, and finally the bombing of the command and control center in Iraq that was filled with civilians [96] must be avoided in the future. These incidents indicate a need for small spy Organic Air Vehicles acting as a tool for validation of military targets within an urban environment.

7.1 Urban Environment

One of the most adverse environments for targeting and maneuvering is urban territory. For soldiers, this environment is hostile and dangerous to traverse when attempting to gather intelligence. Sniper fire, ambushes, booby traps, and guerrilla warfare are only a few of the obstacles that are significantly more easily hidden in a city environment. Small flying objects have a better chance of survival and going unnoticed than a soldier infiltrating a city. Furthermore, the New York Times has reported the following with regards two US services:

> United States Army and Marine Corps doctrine recommends isolating and bypassing urban areas when possible, because cities are difficult, dangerous and costly to fight in. [27]

7.2 Interrogation of Indoor Facilities

Interrogation of indoor facilities may be required for intelligence gathering or target clarification. There are many problems associated with a procedure having Organic Air Vehicles (OAVs) gathering intelligence from a possible military target. In partitioning the problem, three major areas of concern can be seen: 1) the aeronautical problem associated with designing vehicles capable of transporting equipment and flying in a controlled stealthy manner, 2) designing radar, and 3) components on the unmanned aircraft to be able to find the target and learning information leakage points.

7.3 Organic Air Vehicle (OAV) designs

Design of OAVs to be capable of identifying a building's information leakage points and then exploiting those points is a complicated task. Stationary vehicles or vehicles with a hovering or slow loitering capability are similar to a helicopter OAV class of spy vehicles. Figure 59 is an illustration of this class of OAV – called the Interrogator UAV[1] (IUAV) in this document. The helicopter in the figure is a vehicle currently being used as a camera OAV.

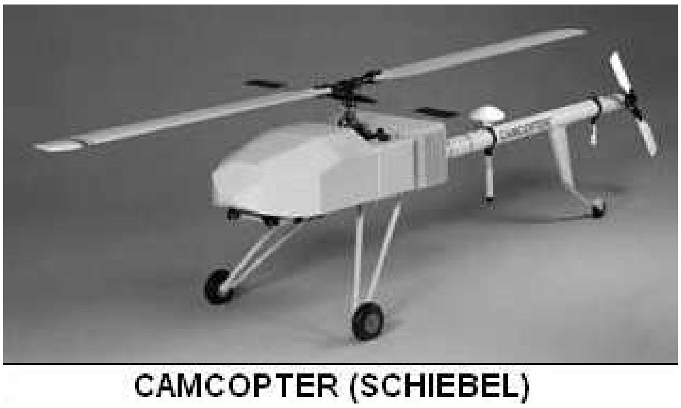


Figure 59: This is an example of a helicopter OAV currently being used. AFRL/SNRR has a helicopter OAV for radar testing purposes; however, the actual OAV is not provided.

7.4 Building Interrogation and Exploiting Techniques

For an OAV to interrogate a military target, the OAV must travel to the target or structure before gathering information. The assumption is made that the OAV

[1] OAV and Unmanned Aerial Vehicle (UAV) are synonymously used throughout this chapter

makes the trip to the target via a Transport OAV (TUAV). The OAVs are assumed to have autonomous control with preprogrammed objectives to interrogate the specified target and extract information of possible intelligence value. Once dropped from the transport, the OAVs fly to the predetermined target location and begin negotiating the structure searching for leak points. Communication between OAVs is vital to the success of the mission. In addition to control and coordination communication, target leakage information should also be aggregated to help identify the best positioning of the OAVs. Algorithm 9 specifies the sequence of events.

Algorithm 9 Place OAVs on Target

```
 1: procedure OAVsOnTarget(Number,Target)               ▶ OAVs on Target Events
 2:     Transport OAVs to target
 3:                               ▶ MOEA optimizes OAV flight formation positioning
 4:     OAVs Map leakage points of target
 5:                      ▶ Positional and leak intensity are reported to a control OAV
 6:     Leak points are prioritized according to some rule set.
 7:     Position OAVs at least cost position to relay information
 8:                           ▶ MOEA optimizes spy OAV positioning around building
 9:     OAVs loiter at these high leakage points and relay information back
10: end procedure
```

This chapter focuses on using an MOEA for the optimizing of OAV flight formation positioning to reduce power consumption when OAVs communicate during the flight to the target. This is an important step in putting OAVs on target for military operations using OAVs for reconnaissance (see step 2 of Algorithm 9). This approximated real-world problem is formally called the OAV flight formation problem OAVFFP). Fortunately, a modified version of this optimization maps directly to the multiobjective quadratic assignment problem. However, first the single objective quadratic assignment problem is discussed.

7.5 Quadratic Assignment Problem (QAP)

The QAP was originally designed to model a plant location problem [26]. Mapping the OAV problem into a QAP is accomplished with replacement. By inserting

OAVs for plants, flight formation positions for plant locations, and communication traffic for supply flow, the OAVs problem is mapped directly onto the QAP. The mQAP is similar to the scalar QAP[2], with the exception of having multiple types of flows (communications) coming from each object (OAV).

The plant location problem is quoted as follows:

> A fixed number plants, $\mathcal{N}$, are placed at a fixed number of locations, $\mathcal{N}$ (one and only one at each location). Each plant has a specified supply flow between one another. The goal of the QAP is to find the best placement of the plants at locations such that the product of the distances and flows are minimized. (see Equation 75).
>
> $$min \left\{ C(\pi) =_{\pi \in P(\mathcal{N})}^{min} \sum_{i=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{N}} a_{ij} b_{\pi_i \pi_j} \right\} \tag{75}$$
>
> where $\mathcal{N}$ is the number of plants/locations, a_{ij} is the distance between location i and location j, b_{ij} is the flow from plant i to plant j, and π_i gives the location of object i in permutation $\pi \in P(\mathcal{N})$ where $P(\mathcal{N})$ is the QAP search space, which is the set of all permutations of $\{1, 2, \ldots, \mathcal{N}\}$ [134]. This problem is not only NP-hard and NP-hard to approximate, but is almost intractable. Optimal solutions for $\mathcal{N} \geq 20$ cannot be found within a reasonable time frame [26,179].

Once OAVs are substituted for plants, flight formation locations are substituted for plant locations, and multiple lines of communication traffic to-and-from each OAV is substituted for multiple supply flows the OAVFFP is mapped to mQAP. For example, the OAVs may use one communication channel for passing reconnaissance information, another channel for target information, and yet another channel for OAV received radar signatures. The end goal is to minimize all the communication flows between OAVs. The mQAP[3] is defined mathematically in Equations 76 and 77.

$$min\{C(\pi)\} = \{C^1(\pi), C^2(\pi), \ldots, C^k(\pi)\} \tag{76}$$

[2] See `http://www.seas.upenn.edu/qaplib/` for more info about the QAP.

[3] See `http://dbk.ch.umist.ac.uk/knowles/mQAP/` for more info about the mQAP.

$$C^g(\pi) =_{\pi \in P(n)}^{min} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b^h_{\pi_i \pi_j} \; such \; that \; h \in \{1..k\} \tag{77}$$

where n is the number of objects/locations, a_{ij} is the distance between location i and location j, b^h_{ij} is the g^{th} flow from object i to object j, π_i gives the location of object i in permutation $\pi \in P(n)$, and 'min' means to obtain the Pareto front [134].

Many algorithm approaches have been used on the QAP. QAP researchers can only optimally solve for problems that have less than 20 plants/locations. Furthermore, problems having 15 plants/locations are extremely difficult [26]. When feasible, optimal solutions are found using branch and bound methods [26,97]. However, since many real-world QAPs have more than 20 plants/locations, other methods need to be employed in order to find a good solution in a reasonable amount of time. The use of stochastic local searches and ant colony optimization have been explored and found to perform well when compared to some of the best heuristics available for the QAP and mQAP [79,151,175]. Evolutionary algorithms have also been applied [107, 125, 162]. Additionally, several researchers have compared the performance of different search methods [161,212].

7.6 Algorithms Solving mQAP

Five different approaches are applied to mQAP. The first two are stochastic population based explicit BBB called MOMGA-II and MOMGA-IIa. The third is a local search method created by Knowles and Corne in [134]. The fourth approach is simply an exhaustive search algorithm and the fifth is another local search method developed by Luis Paquete in 2004 [178].

Other algorithms have been used to solve the QAP [23, 150] but only a few have been applied to mQAP [125]. mQAP test instances developed in [134] are used to test the five algorithms. Table 36 lists all MOP instances solved except

Table 36: List of mQAP MOPs numbered and listed according to size and number of objectives. There are *real like* (#rl) and *uniform* (#uni) instances. The size of each problem is indicated by the two digit number following the *KC* (KC##). The number of objectives for each problem is indicated by the number preceding the *fl* (#fl). Each column lists the sizes of MOPs used: 10, 20, and 30. The shaded area of the table is identifying the MOPs with three objectives - others have two objectives.

(MOP #)	Name (size 10)	(#)	Name (size 20)	(#)	Name (size 30)
1	KC10-2fl-1rl	9	KC20-2fl-1rl	17	KC30-2fl-1r1
2	KC10-2fl-1uni	10	KC20-2fl-1uni	18	KC30-3fl-1rl
3	KC10-2fl-2rl	11	KC20-2fl-2rl	19	KC30-3fl-1uni
4	KC10-2fl-2uni	12	KC20-2fl-2uni	20	KC30-3fl-2rl
5	KC10-2fl-3rl	13	KC20-2fl-3rl	21	KC30-3fl-2uni
6	KC10-2fl-3uni	14	KC20-2fl-3uni	22	KC30-3fl-3rl
7	KC10-2fl-4rl	15	KC20-2fl-4rl	23	KC30-3fl-3uni
8	KC10-2fl-5rl	16	KC20-2fl-5rl		

one. The last test MOP comes from a test instance generated by Paquete having 25 locations/plants and 2 flow.

MOMGA-II: The MOMGA-II is a stochastic population based explicit building block multiobjective evolutionary algorithm that is discussed in detail in Chapter III in section 3.2.2 on page 77.

MOMGA-IIa: The MOMGA-IIa (extended MOMGA-II) is a stochastic population based explicit building block multiobjective evolutionary algorithm that is discussed in detail in Chapter III in section 3.3 on page 88.

Local search approach: The Local Search (LS) method employed for the mQAP is where positions of facilities (or objects) are switched (called 2-opt) [133]. The new positioning is kept if the new configuration yields a lower fitness value. This search method works well for solving the QAP [209]; however, the mQAP makes employing a strict LS approach difficult for the deceptive hyperplanes that accompany multiobjective problems. Specifically, a researcher is faced with how to initialize the LS method. Knowles and Corne concluded that the starting points would randomly be selected out of a basin of attraction [134, 209]. After a new point is picked, the LS method is applied for a specified number of generations. This is the technique used by [134] finding most, if not all, Pareto front (PF) vectors. The solutions and Pareto front vectors for the larger problems have not been published, making comparisons difficult.

Complete iterative approach: The complete iterative approach is an exhaustively deterministic approach that can be accomplished on MOPs 1-8 (see Table 36). The number of solutions that must be evaluated is calculated by Equation 78, where $\mathcal{N}$ is the number of facilities and $\mathcal{N}'$ is the number of locations. Consequently, for the mQAP, $\mathcal{N} = \mathcal{N}'$.

$$x \approx \frac{n!}{(n-k)!} \tag{78}$$

Function calculations for each MOP are $k*10!$, $k*20!$, and $k*30!$ or $k*3628800$, $\approx k*2.43e18$ and $\approx k*2.65e32$. These numbers are not to be confused with the search space size. For each search space solution, k calculations must occur.

Paquete's Local search approach: Paquete uses two stochastic search techniques. The first technique is a component-wise ordering of objective value vectors of neighboring PF vectors along the PF front and the second is based on using different scalars for the objective function vectors. Paquete takes advantage of the matrix constructs reducing the problem, thus, reducing the amount of search required to find these good solutions.

7.7 Design of Experiments

Experiments for MOMGA-II were conducted on Clusters 2 and 3 listed in Table 13 on page 147. Experiments for MOMGA-IIa were done on Cluster 1 in the same table. The MOMGA-II was run 10 times in parallel and the data was then processed incrementally so as to show better solutions gradually being found. The MOMGA-IIa ran 10 experiments in serial and kept one pool of PF vectors and associated decision variables at all times. The MOMGA-II was run using BB sizes 1 through (10, 10, and 10) while the MOMGA-IIa was run using BB sizes 1 through (10, 15, and 20) for each MOP sized (10, 20, and 30). These experiments are run to determine how well each algorithm can solve the MOPs. This research group's hypothesis is that MOMGA-II's CT generation and evolution mechanism limited the exploration and BB finding ability of the algorithm, while the MOMGA-IIa now has the enhancement required to overcome this limitation.

7.8 MOEA and LS Settings

This section lists settings used for each search heuristic. Settings for the MOMGA-II and MOMGA-IIa are kept the same over the same sized mQAP.

7.8.1 MOMGA-II and MOMGA-IIa. For mQAP problems of size 10, 20, and 30 Tables 59, 60, and 61 on pages 287, 288 and 289 list are setting used for experimentation results listed in Tables 37 and 38. [43, 125].

7.8.2 Local Search. Knowles and Corne [133] collected results by running 1000 local searches from each of 100 (for two-objective instances) or 105 (three-objective instances) different vectors, thus giving them ≈ 200000 records.

7.9 Effects of Feasibility Function on the Solution Space

The main obstacle to implementing the mQAP into a binary string is in the decoding of a chromosome for evaluation. Each chromosome has a number of variables each having $SLICE$[4] bits. A conversion from a binary to an integer number is accomplished for each set of $SLICE$ bits. Thus, each SLICE of bits must be able to represent at least the same number of facilities or OAVs.

All of the mQAP experimentation done use a 10-bit $SLICE$ and, although this might seem to be in excess, this number of bits is used for good reason. The

[4] A $SLICE$ of bits is the number of bits representing one variable within the chromosome.

reason is to even out the probability each solution arises in the solution space. The lack of uniformity in solution representation within the search space is caused by the feasibility function implicitly defined with the decoding function. For each variable, $SLICE$ bits are decoded making a set of $\mathcal{N}$ integer numbers. The feasibility function checks the set of $\mathcal{N}$ numbers for duplicates. When a set of duplicate numbers is located, the feasibility function rank orders them from left-to-right as highest-to-lowest – but maintaining the overall rank of this set of numbers with respect to all other numbers in the set. The setup of this function in this manner has two implications: 1) some sequence appears more than others, 2) the number of bits for one $SLICE$ becomes a factor in making the probability of the appearance of each solution uniform. In fact, when using this feasibility function, as the number of $SLICE$ bits rise, the more uniformly distributed the solution representation becomes. However, the disadvantage to increasing the number of bits for $SLICE$ comes at an increase in search space size. In fact, for each added $SLICE$ bit, there is a $2 * \mathcal{N}$ factor increase in the search space. Next is a short description and graphical illustration of how the probability of each solution appears within the search space.

Effects of $SLICE$ on Solution Representation: Using the left-to-right sort feasibility function makes for some sequences to be represented more often than others. In fact, the sequence $\{1, 2, 3, \cdots, \mathcal{N}\}$ can be represented many more ways than any other sequence, thus, automatically hindering the search algorithm before it even begins the search for optimal solutions. The reason for this phenomenon is because any repeated set of numbers is automatically sequenced from left-to-right, but kept in the same overall position, hence $\{1, 1, 3, 4, \cdots, \mathcal{N}\}$ and $\{2, 2, 2, 4, \cdots, \mathcal{N}\}$ both evaluated back to $\{1, 2, 3, \cdots, \mathcal{N}\}$. Hence, a reduction of repeated values with respect to the entire possible solution space is sought. One good example of using extra $SLICE$ bits to more uniformly distribute the representation of each solution in the search space is given in a 4 OAV mQAP. $SLICE$ bits are 2, 3, 4 and 5, making the chromosome length for each bit setting 8, 12, 16, and 20 bits. This translates to a total search space of 2^8, 2^{12}, 2^{16}, and 2^{20}. However, the decoded genotype domain search space still remains the same at $P_{\mathcal{N}}^{\mathcal{N}}$.

Figure 60 illustrates the point discussed above where a greater number of bits used to represent a variable evens out the representability of each solution within the search space. Next, a few local search techniques are discussed that are used in conjunction with the MOMGA-IIa in an attempt to compete well with the local search technique used by Luis Paquete in 2004 [176–178].

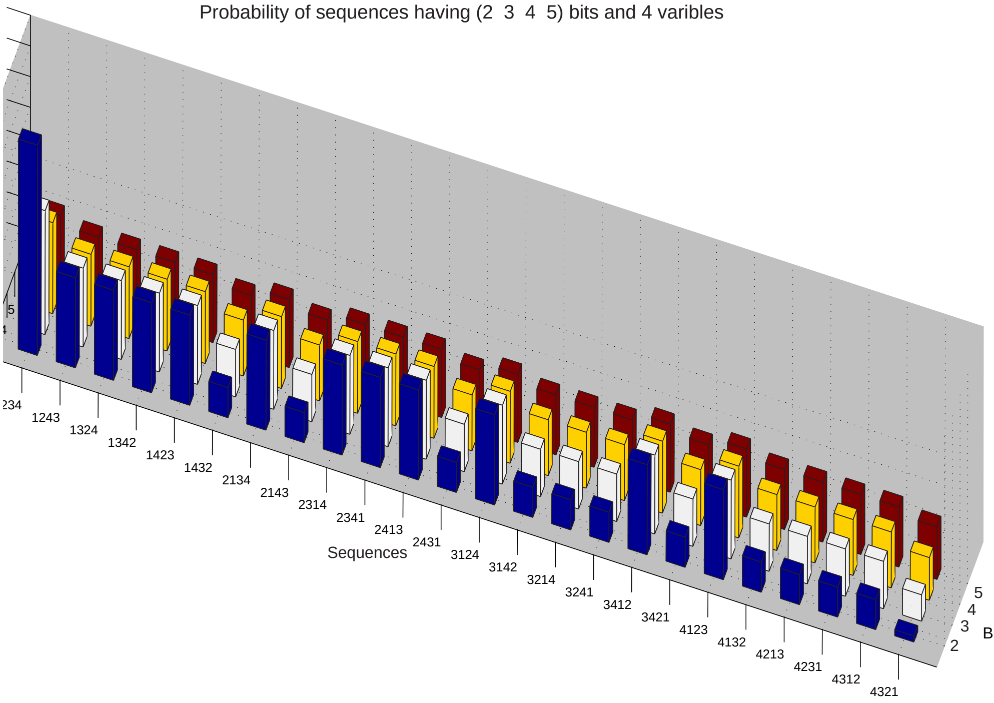


Figure 60: Illustrated are the probability of solution representation when using a different number of bits to represent each solution. It is assumed that a binary search algorithm is used and the left-to-right feasibility sort function (discussed in the text) is embedded within the search algorithm to decode genotypic solutions.

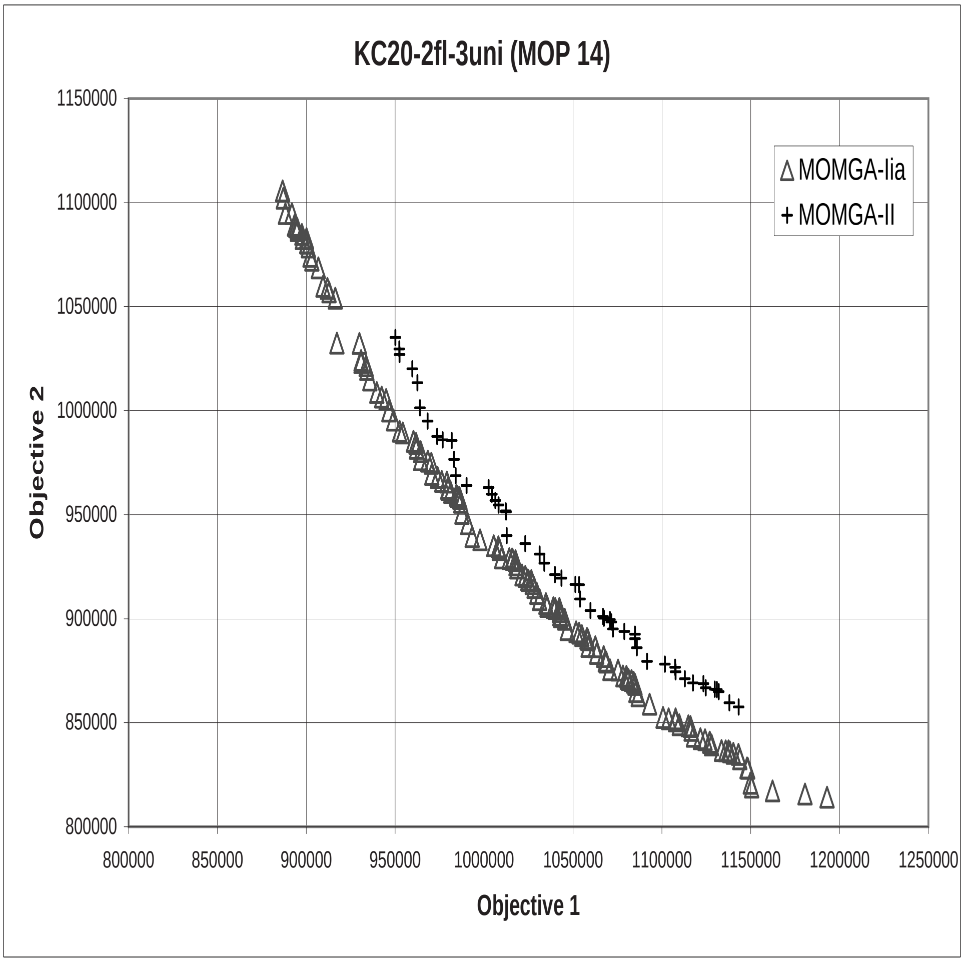


Figure 61: Results for MOP 14 illustrating that the MOMGA-IIa's CT generator is superior at finding good BBs.

7.10 Results and Analysis

Overall the results of MOMGA-IIa are good. Tables 37 and 38 list results of the Air Force application experiment. In the MOP of size less than 20, MOMGA-IIa found all optimal solutions evaluating to all Pareto front vectors in PF$_{true}$ in a short amount of time (under 16 minutes in some cases). MOMGA-II did not. Additionally, in MOPs of size 20, MOMGA-IIa solutions evaluating to Pareto front vectors that dominated MOMGA-II's in every case (illustrated by Figure 61). Finally, in MOPs of size 30, MOMGA-IIa found more PF$_{known}$ vectors than MOMGA-II. In addition, all solutions found by MOMGA-IIa were found to evaluate to vectors that dominated evaluated solutions found by MOMGA-II's in all except for the MOP 20 case where ten MOMGA-II solutions evaluated to PF vectors that were non-dominated when compared to MOMGA-IIa's PF$_{known}$. The reader should also note that MOPs of size 20 and 30 all took several days to solve. As far as the results for the LS method, these results are good in quantity, but without the actual data to compare, no claim that either algorithm LS or MOMGA-IIa is better at solving these MOPs. In conclusion, the reason for the dominance of MOMGA-IIa over MOMGA-II is due to the CT generation and selection mechanism. In addition, MOMGA-II's limited number of CTs might be causing it to destroy some good BBs. Lastly, the CT selection mechanism for MOMGA-IIa allows for better multiobjective BBs to be found - thus, MOMGA-IIa is a better BBB. This phenomenon is reflected in the results of each MOEA for each MOP.

Table 37: Summary of results for all mQAP of size 10. Included in this table are the number of optimal Pareto front points (when known), and the number of PF points found by each algorithm {MOMGA-IIa (M-IIa), MOMGA-II (M-II), and Local Search (LS)}. The diameter (dia) and entropy (ent) is calculated for M-II's and M-IIa's vectors/solutions.

MOEA	**mQAP Number**, Size 10, (Deterministic PF True vectors) (True PF Pts Found/Total PF Pts Found)=(|PF$_{true}$ = PF$_{known}$|)/|PF$_{known}$|							
	1(58)	**2**(13)	**3**(15)	**4**(1)	**5**(55)	**6**(130)	**7**(53)	**8**(49)
LS	58/58	13/13	15/15	1/1	55/55	130/130	53/53	49/49
M-II	57/59	11/12	11/17	0/3	50/53	122/122	25/34	36/45
M-IIa†	58/58	13/13	15/15	1/1	55/55	130/130	53/53	49/49
†Time(mins)	21.5	62.3	29.8	10.9	45.5	68.1	45.4	15.8

7.11 Problem Domain Information Based Heuristics

A second local search technique was accomplished by Luis Paquete in 2004. Problem sizes of $\mathcal{N}$=25, 50, and 75 are studied by Paquete. He produced 105 test problems using the mQAP generator created by Knowles and Corne [131]. For each mQAP size, Paquete generated five problems for each of the following correlation

Table 38: Summary of results for all mQAP of size 20 and 30. Included in this table are the number of optimal Pareto front points (when known), and the number of PF points found by each algorithm {MOMGA-IIa (M-IIa), MOMGA-II (M-II), and Local Search (LS)}. u indicates that it is unknown how many PF_{known} vectors a particular algorithm found when compared to the PF_{best} set found by all MOEAs considered. In addition, diameter (dia) and entropy (ent) is calculated for M-II's and M-IIa's vectors/solutions.

	mQAP Number, Size 20, (Best PF_{known} vectors found)							
MOEA	(Best PF Pts Found/Total PF Pts Found)=$(\vert\text{PF}_{best} = \text{PF}_{known}\vert)/\vert\text{PF}_{known}\vert$							
	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
LS	u/541	u/80	u/842	u/19	u/1587	u/178	u/1217	u/966
M-II	0/17	0/24	0/12	0/5	0/29	0/51	0/28	0/17
(dia/ent)	11.6/0.43	11.4/0.48	11.01/0.45	7.2/0.25	12.1/0.54	12.3/0.55	10.39/0.43	11.76/0.46
M-IIa†	36/36	33/33	31/31	7/7	63/63	139/139	48/48	44/44
(dia/ent)	13.0/0.58	13.7/0.69	11.17/0.47	3.67/0.16	14.1/0.73	15.5/0.88	12.76/0.60	11.37/0.50
†Time	9.8	8.3	8.3	8.3	8.8	8.3	8.3	1.7
†days								
mQAP Number, Size 30								
	17	**18**	**19**	**20**	**21**	**22**	**23**	
LS	-	u/1329	u/705	u/1924	u/168	u/1909	u/1257	
M-II	n/a	0/507	0/552	10/552	0/104	0/795	0/755	
(dia/ent)	-	24.1/0.79	20.1/0.50	24.3/0.78	22.3/0.64	21.2/0.57	20.4/0.56	
M-IIa†	40/40	507/507	552/552	542/552	104/104	795/795	755/755	
(dia/ent)	17.2/0.42	23.9/0.80	23.2/0.76	23.1/0.74	21.9/0.59	24.0/8.11	25.1/0.90	
†days								
†Time	8	∼8	8	8	8	8	8	

settings[5]: ±0.75, ±0.50, ±0.25, and 0.00. Only one test problem is selected for evaluation as a comparison against the latest MOMGA-IIa. Paquete does not discuss run time associated with finding solutions; however, the quality of solutions is good for the selected mQAP test problem: $\mathcal{N}$=25, -0.75, instance 1. The selected problem instance can be found on his web site[6], as well as the solutions and associated PF vectors. PF vectors found by Paquete on this selected problem are shown to be much better than those found by MOMGA-IIa. In fact, two different local search techniques are added to MOMGA-IIa, making a memtic algorithm, in an attempt to

[5] The mQAP generator created by Knowles and Corne [131] has correlation settings to allow for mQAP test MOP matrices to be correlated (+) or uncorrelated (-).

[6] `http://www.intellektik.informatik.tu-darmstadt.de/~lpaquete/`

best Paquete' PF$_{known}$; however, the local search devised by Paquete prevails. Figure 62 illustrates how the PF vectors found by the local search designed by Paquete dominate the entire PF$_{known}$ front found by MOMGA-IIa.

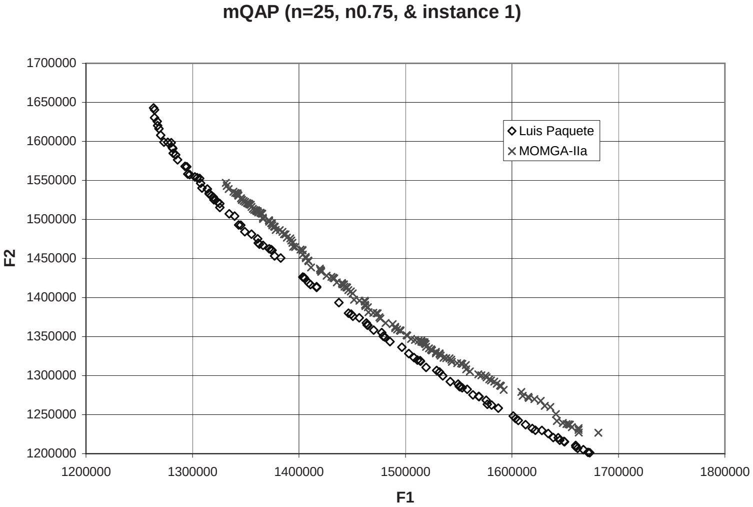


Figure 62: This graph illustrates MOMGA-IIa results versus Luis Paquete's local search results for the first instance of $\mathcal{N}$=25 and -0.75 problem.

It is apparent that MOMGA-IIa does not solve every problem better than all other search techniques. In fact, the local search optimization heuristic applied to an NPC problem outperforms the applied MOEA. Although, using Paquete's local search technique in conjunction with the MOEA might make the total search more efficient, this is a good example where the MOEA does not necessarily find the best solutions and the application of problem domain information can boost performance substantially.

7.12 Statistically Comparing MOMGA-II vs. MOMGA-IIa mQAP 1, (10) (KC10-2fl-1rl)

For a statistical comparison between MOMGA-II and MOMGA-IIa, there must be parameter constants in order to have a somewhat fair comparison. These settings are given in Appendix A in Table A.8 on page 279 corresponding to MOP mQAP 1.

Also, statistical results for the ten metrics (ER, GD, HR, ONVG, S, ONVGR, ME, R2, R3, ϵ) used previously can be found in Figures 88 in Appendix B on page 304:

Effectiveness: Visual analysis of the mean and standard deviations for the results of 30 experiments for both the MOMGA-II and MOMGA-IIa present differences between metrics ER, HA, ONVG, S, ONVGR, and ME. Significant visual test differences are illustrated by metrics HA and ONVGR. KWtest results in finding that each and every metric is different in favor of MOMGA-IIa (see Table 39).

It can be concluded from the data, both using visual analysis and the KWtest results that the MOMGA-IIa is more effective at solving the MOP mQAP 1 with the settings in Table A.8 on page 279.

Efficiency: Efficiency results for this experiment are listed in Table 40. There is a noticeable difference in the wall clock time taken by the MOMGA-IIa when compared with MOMGA-II. If the effectiveness of the algorithms is over looked (MOMGA-II did not solve the problem), then it can be concluded that MOMGA-II is more efficient than MOMGA-IIa. However, further tests conclude that MOMGA-II still does not solve this MOP even when given more experiment loops (enough to double the run time listed in Table 40).

Therefore, it can be concluded that MOMGA-IIa is both more effective and efficient on the mQAP 1 problem. Next, the evolutionary trace visualization is discussed.

Table 39: mQAP 1 metric results using KWtest comparison between algorithms MOMGA-II and MOMGA-IIa. The symbol, $\in$, indicates that groups (experimental runs) are the same and $\notin$ indicates that the groups are different.

	Metrics									
KWtest	ER	GD	HR	ONVG	S	ONVGR	ME	R2	R3	ϵ
χ^2	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$
F	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$	$\notin$

Table 40: Summary of timing for MOEAs solving MOP mQAP 1

MOP	MOMGA-II	MOMGA-IIa	KWtest
mQAP 1	1238.43 ± 7.91	3810.93 ± 1851.13	$\not\in$

7.13 Evolutionary Trace for BB Visualization

One evolutionary trace for solving mQAP 1 of size 100 is presented in Figure 63. The Pareto front is shown to be convex. The genotype representation of the PF vectors found have definitive patterns, but exact locations within the genotype evaluating to vectors in PF_{true} do not represent the only genotype pattern that can evaluate to each vector in PF_{true}. There are many genotype patterns that can map to the same PF_{true} vector (many to one) due to solution representation (discussed above). The many to one phenomena is mainly due to the feasibility function (left-to-right sort) used in decoding the chromosome to be evaluated by the fitness function.

BB sizes for mQAP 1 are variable across the Pareto front. This is similar to the T3T4 problem discussed in Chapter V where the BB sizes are variable and spread out throughout the entire Pareto front. This type of BB size requirement is indicative of a more difficult problem (possibly having a higher *epistasis*). Furthermore, this type of MOP is more difficult to solve for MOMGA-II while MOMGA-IIa statistically is shown to be more effective and efficient at solving an MOP having these characteristics.

7.14 Summary

In summary, MOMGA-IIa statistically performs better than MOMGA-II. However, it is less effective than Paquete's LS method using domain information. In general, it is apparent that mQAPs are difficult problems to solve when the sizes run over 20 facilities. This is confirmed both by algorithm performance and BB size analysis. However, the new MOMGA-IIa does indeed show improvement in both effectiveness and efficiency over MOMGA-II. The reason for the improvement

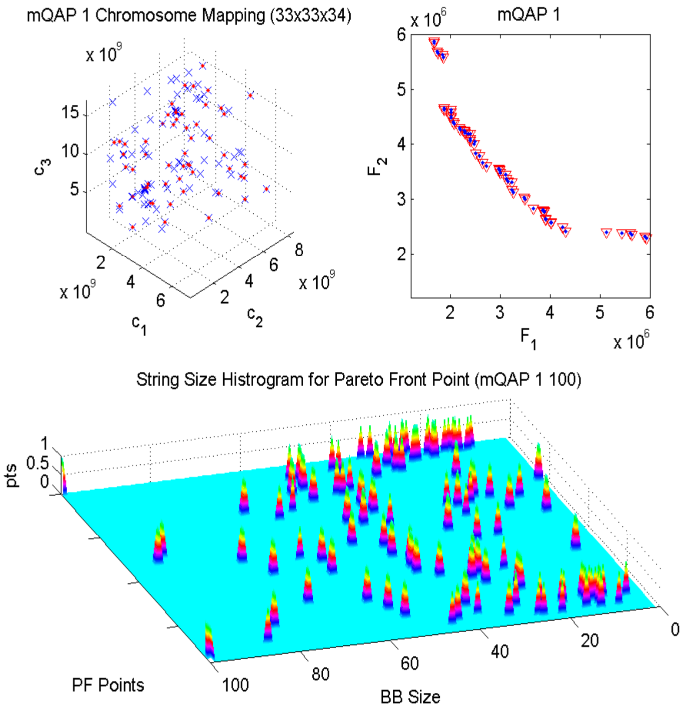


Figure 63: Illustrated within this figure is genotypic, phenotypic and the associated BB sizes found for each PF$_{known}$ vector for mQAP 1. The lower figure represents tallies for the size of BBs found for each PF vector. Pareto front vectors are ordered from the best function 2 to the best function 1 and partially ordered along the *PF Points* axis of the plot. Note that the BB sizes are variable across the Pareto front, indicating that this is a more difficult problem for MOMGA-series algorithm to solve.

must be caused by the combination of the added active archive and the additional competitive templates partitioning of the genotype and phenotype space.

This chapter analyzed the use of MOMGA-IIa on a futuristic military application (approximate real-world problem) of optimizing OAV flight formation positioning to reduce power consumption when OAVs communicate during the flight to the target. MOMGA-IIa is good at finding flight formations in a short period of time, and it is statistically shown that MOMGA-IIa performs better than MOMGA-II. However, visual tests reveal that MOMGA-IIa may still not be as good as some of the local search techniques using domain information to reduce the search space.

However, there was no timing data associated with the local search/ant colony optimizations results found by Luis Paquete, so a fair comparison on efficiency between these two approaches cannot be made. This concludes this chapter on the application of MOMGA-IIa on the OAVFFP. Next, MOMGA-IIa is applied to a third military application in designing a digital amplitude-phase keying symbol set with m-ary Alphabets. Usage of MOMGA-IIa as both a single and multiobjective evolutionary algorithm, like in the PSP problem, is conducted.

VIII. Application: Design of Digital Amplitude-Phase Keying Symbol Sets

Optimal digital symbol set constellation design is important in digital communications for Quadrature Amplitude Modulation (QAM). Optimization realizes a reduced probability of bit error (P_b) while keeping the same bandwidth and power for transmitting the signal. Constellation shapes currently used in QAM include rectangle, triangle, hexagonal, and concentric circles. In this study, two dimensional (4, 8,..., and 256)-ary constellations at specific normalized signal-to-noise ratios $\left(\frac{Eb}{No}\right)$ having lower P_b are sought. Various models are designed to provide an MOEA with a near exact model to utilize as a fitness function. MOEA-found solutions are tested for merit using a Monte Carlo simulator. Comparisons of $\frac{Eb}{No}$ versus P_b between the rectangular constellation and new designs are illustrated. New designs are shown to be different and comparable to the standard constellations used today.

8.1 Introduction

Bandwidth efficient modulation techniques using bounded bandwidth are sought in the digital communications field. Symbol set design is a step towards minimizing the probability of bit error (P_b) at a specific normalized signal-to-noise ratio $\left(\frac{Eb}{No}\right)$ [213]. Optimal constellations with lower P_b at specified $\frac{E_b}{N_o}$ have numerous applications in digital communications. Lowering the inter-symbol interference reduces P_b, but normally this comes at cost of increased signal power or decreased noise interjection. Unfortunately, typical links have distortion elements in channel filters and amplifier nonlinearities that cannot be eliminated or, in some cases, reduced. Therefore, optimizing symbol set constellations is absolutely necessary for lowering this inter-symbol interference.

The ability for a Multiobjective Evolutionary Algorithm (MOEA), more specifically the multiobjective fast messy GA (MOMGA-II) to optimize symbol set design for a decreased P_b at a certain $\left(\frac{Eb}{No}\right)$ is studied. The combinatorics of this problem call for a stochastic search algorithm that can be used in optimizing both single [93, 243] and multiobjective [47] problems because the models generated to capture this symbol set design problem can be constructed as both single and multi-criteria problems.

First, the algorithm of choice is justified, as well as a detailed discussion of the algorithm domain. This discussion is followed by an overview of the problem domain. The problem domain section includes a detailed description of five models constructed for this optimization problem. Next, algorithm parameters are discussed. Then, the *best* MOMGA-IIa constellations and standard rectangular constellations are presented. Comparisons with existing rectangular constellations are drawn where able. Finally, an analysis and future work discussion concludes.

8.2 Problem Domain Considerations

Five different models have been constructed for this investigation in search of the best balance between lower computation complexity and model correctness; however, results from only one model are presented. When encoded, the problem seems simple. Symbols are placed in a unit circle. Each symbol has a location that is identified by an amplitude (distance from the origin) and a phase angle (angle from positive x-axis counter clockwise around the unit circle). This equates to two variables for each symbol, which is easy to encode in our binary string; however, the evaluation of a constellation makes this problem orders of magnitude more difficult. Each constellation has a different P_b associated with each Energy per Bit (Eb) to the Spectral Noise Density (No) or $\frac{Eb}{No}$ tested, causing the problem to become multiobjective. Figure 66.a illustrates a typical $\frac{Eb}{No}$ versus P_b graph that is used to determine if a MOMGA-IIa 2 bit constellation is better than the standard rectangular constellation. The Pareto front that forms naturally is concave down.

Even though the encoding of the problem may seem to alleviate some of this problem's difficulty, this is just a mirage. Every solution can be rotated around the circle one radius step[1] and maintain itself as the same solution because it evaluates to the same fitness. This results in each solution having $\frac{2^{SLICE}}{2\cdot\pi}$ different configurations.

An algorithm with the capability to capture the high *epistasis* caused by multiple constellations having the same fitness, solve single and multi-objective problems, and converge on good solutions for problems having high complexity (see Table 13 in Chapter IV for System Configuration[2] of each configuration) is required. MOMGA-IIa is selected because it possesses these capabilities.

MOMGA-IIa is a linkage learning algorithm that uses explicit BBs to solve difficult problems. MOMGA-IIa is different than most other genetic algorithms because it can be used to solve single objective problems (SOP) as well as multiobjective problems, and it specifically searches for good BBs from which to generate new solutions. Moreover, in previous studies, this algorithm solved hard single objective [41] problems better than both the simple and messy genetic algorithms [154]; it also solved many multiobjective problems better than the Vector Evaluated Genetic Algorithms (VEGA), Non-dominated Sorting Genetic Algorithm (NSGA), NSGA-II, and multiobjective Bayesian Optimization Algorithm (BOA) [47, 243]. See Chapter III on page 71 for a complete description of the MOMGA-IIa. Next, the problem domain is described in detail.

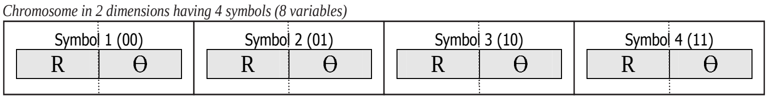


Figure 64: Illustrating a genotype solution (chromosome) for a two-bit symbol set. Each smaller box would contain a number of bits according the precision required for each variable.

[1] A single radius step has the resolution, $(\frac{2\cdot\pi}{2^{SLICE}})$, limited by the encoding a symbol's radius into a binary number - where $SLICE$ is the number of bits allowed to represent the radius variable.

[2] To illustrate how complexity plays a role, using the analytical mode, it would take a computer in Cluster 1 (listed in Table 13 in Chapter IV) $0.0159 * 2^{2560}$ seconds to check every combination and $33.93 * 2^{2560}$ seconds using the simulation model. In either case, it would take more than a lifetime to check every combination for configuration 7 (see Table 41).

8.3 Problem Domain Analysis

The domain for this problem is complex, consisting of reducing the overall P_b for a constellation over a range of $\frac{Eb}{No}$s. Constellations are built by arranging $2^{Signal\ bits}$ symbols[3] inside a unit circle[4] in a way that optimizes P_b versus $\frac{Eb}{No}$. Symbol location is described by $\hat{d}$ variables where $\hat{d}$ is the number of dimensions[5]; furthermore, $\hat{d} = 2$ for this investigation. Each variable used to describe the location of a symbol has *SLICE* bits of resolution.

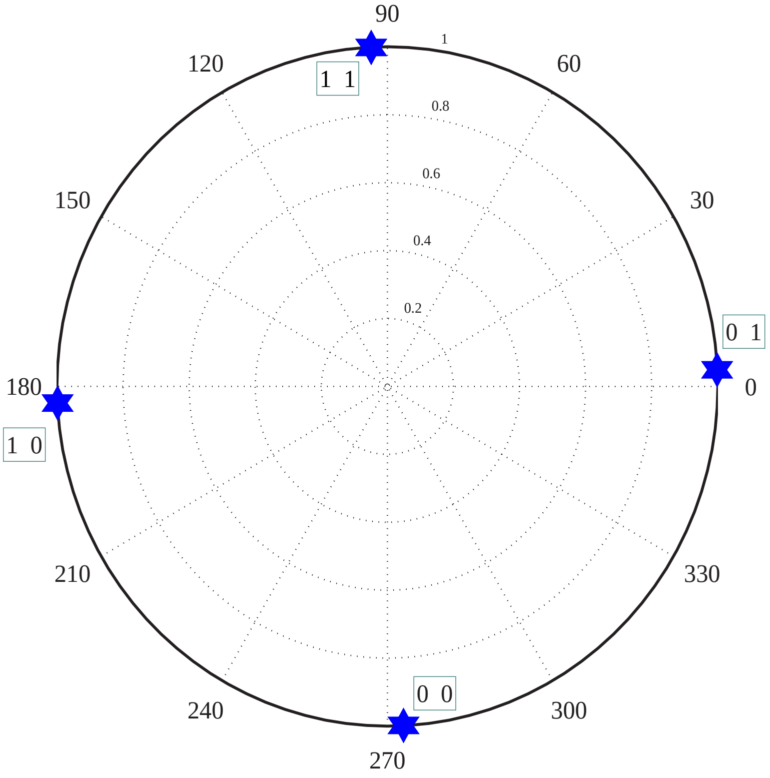


Figure 65: Illustrating two-bit example of constellation and associated symbols. Symbol positions are identified with a six pointed star. Symbol bit patterns are identified right beside each position.

[3] *Signal bits* (S_{bits}) are the number of bits that are used to make up each symbol.

[4] The unit circle is used in this investigation; however, if the number of dimensions changes, so does the dimension of the unit circle (*i.e.*, unit sphere when $\hat{d} = 3$).

[5] Applications for higher dimensional symbol set design is needed, but when ($\hat{d} > 2$), this problem becomes extremely difficult to solve.

Table 41: Configuration Specifications

Config	S_{bits}	# of symbols	String Length	Slice	Search Space
1	2	4	80	10	2^{80}
2	3	8	160	10	2^{160}
3	4	16	320	10	2^{320}
4	5	32	320	5	2^{320}
5	6	64	640	5	2^{640}
6	7	128	1280	5	2^{1280}
7	8	256	2560	5	2^{2560}

Table 41 lists the S_{bits}, String length, $SLICE$, and Complexity associated with each configuration. The String length is the $SLICE$ multiplied by the number of symbols, $2^{S_{bits}}$, and the dimensions, $\hat{d}$, in a constellation. Finally, the Complexity is the size of the search space. Notice, that the $SLICE$ is reduced when going from three to four S_{bits}. This is done to reduce the complexity of larger symbol set configurations; however, this is achieved at the cost of variable resolution.

8.3.1 Fitness function or model designs. Previous researchers either use different coding[6] methods or symbol positioning to get better P_b at a particular $\frac{E_b}{N_o}$ [28, 113, 213]. In the models, the MOMGA-IIa optimizes both of these at the same time by assigning each symbol its bit-wise representation while assigning it a location in space. However, the challenge was found not to be what to optimize or how to represent the problem, but what model (fitness function) best represented the Monte Carlo simulation used to check constellations for goodness.

Normally, new constellations would be tested using a Monte Carlo simulator, where a random stream of symbols is encoded into a signal, s, using the amplitude and phase of symbols identified in a designed constellation. Next, noise is added to the generated signal, $(s + No)$, to simulate the transmission process. The amount of noise added to the signal is related to the $\frac{Eb}{No}$ under test. Then, the signal is decoded

[6] Different coding methods can include n-ary or hybrid gray encoding.

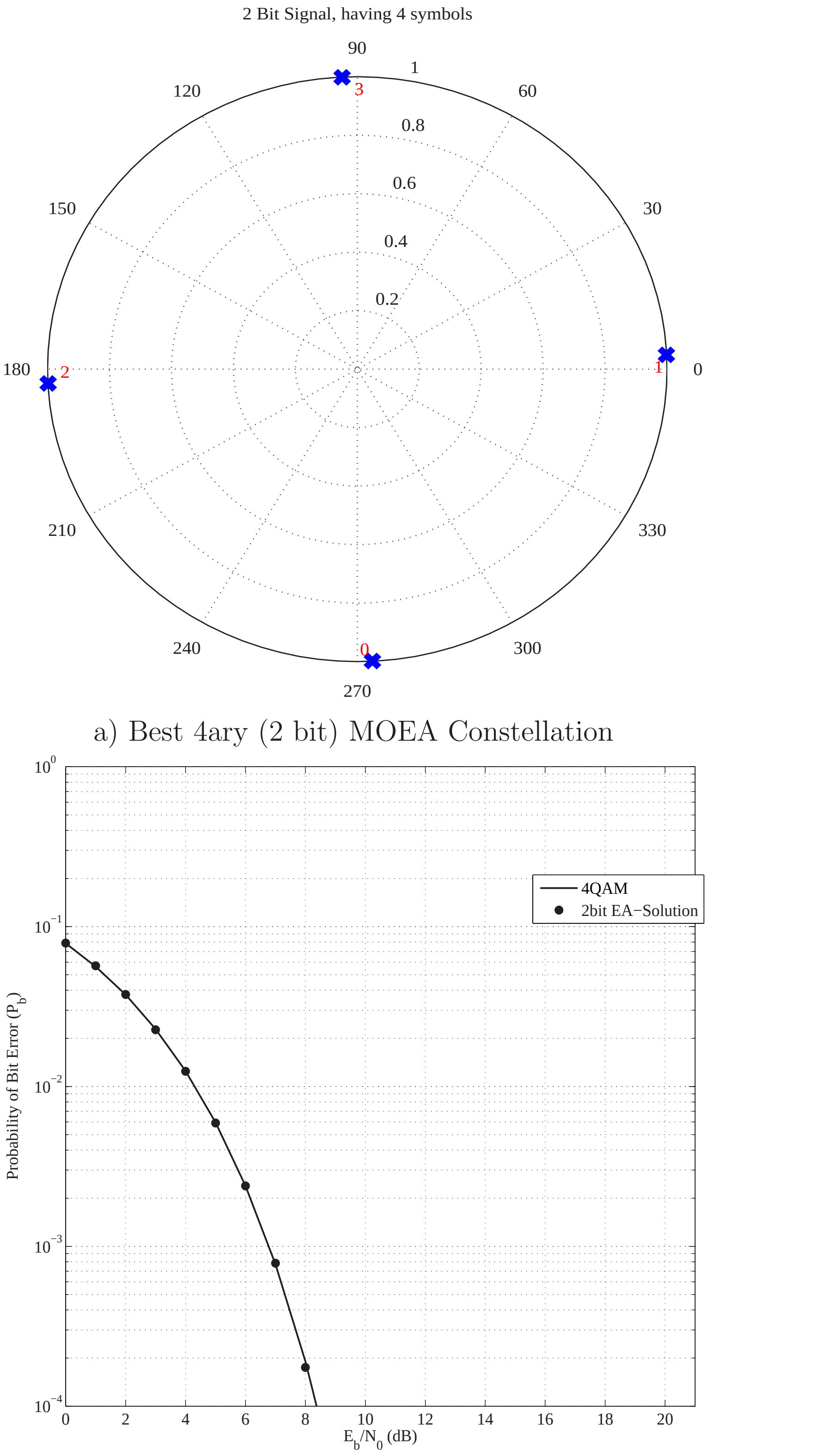
and a reconstructed symbol stream is generated from the noisy signal. Finally, the number of bit errors is calculated by comparing the reconstructed symbol stream to the originally transmitted symbol stream. This test is re-run until enough data is collected to assign a P_b rate to that particular constellation at the $\frac{Eb}{No}$ under test. As discussed earlier, this kind of simulation takes too long to evaluate[7]. Therefore, four different analytical models are designed in search of the optimal balance between computational time and model correctness. This marks the simulation model replacement with a less computational approximation model to reduce fitness function computational time. Five models are described in the next section (Section 8.3.2 on page 241). The first four designs are analytical approximates for this digital system and the fifth design is the Monte Carlo simulation.

Each model uses symbols placed inside a unit circle for amplitude and phase characteristics for each symbol [68]. Each symbol bit pattern is defined to be in one and only one place within the genome. Placement is in binary order, $\{00, 01, 10, 11\}$, and each symbol has $\hat{d} = 2$ degrees of freedom to define its location in the space. Figure 64 illustrates the genome solution for a two bit symbol set and Figure 65 illustrates an example of a two bit decoded constellation.

8.3.2 Four Analytical models and One Simulation.

The analytical models are simple and fast compared to the simulation model, for they are designed for minimal number of calculations while having a close approximation to the real model. These fitness functions have many pre-calculated measures to help speed compute time. For all of these models, each symbol is defined in binary fashion and placed into a bit-matrix like the one found in Equation 79. Furthermore, each symbol's Hamming code distance from one another is calculated and placed into a separate $\hat{d} \times \hat{d}$ matrix called $\mathcal{H}$. A three bit $\mathcal{H}$ matrix is presented in Equation 80. The

[7] One complete simulation (fitness evaluations) of a constellation takes 33.93 seconds on Cluster 1 in Table 13 in Chapter IV.



a) Best 4ary (2 bit) MOEA Constellation

b) 4ary MOEA solution compares well to the 4ary rectangle constellation.

Figure 66: Presented is the best constellations found for 2 bit symbol set.

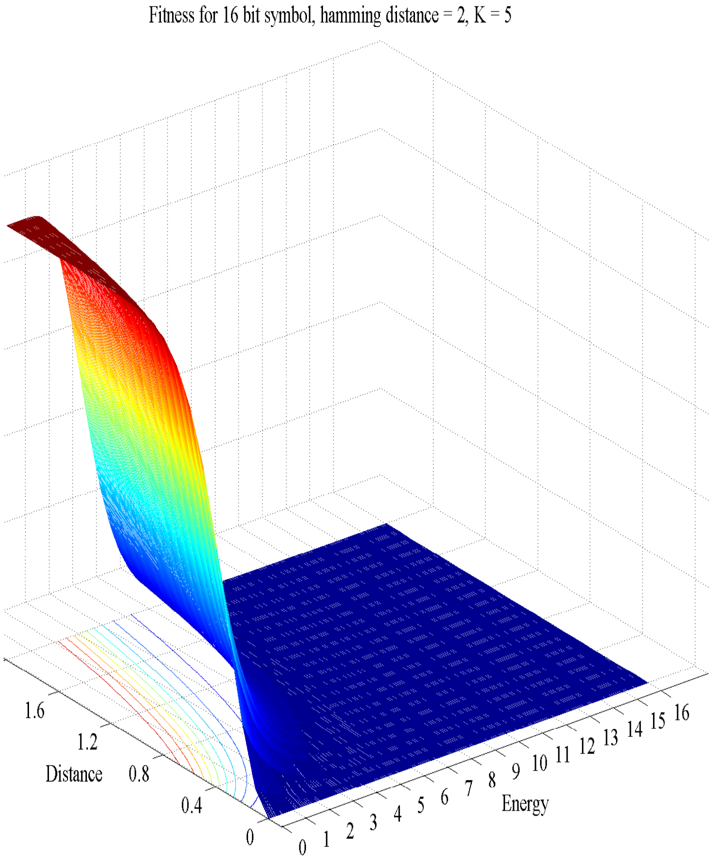


Figure 67: Illustration of the fitness landscape of two symbols having a Hamming code distance of 2, K = 5, symbol distances = [0,2] and constellation energy = [0,16]. A higher fitness value is better than a lower (more negative) value.

Hamming code distance matrix keeps track of the amount of bit errors made when reconstructing the noisy signal and an incorrect symbol is selected.

A complete constellation fitness computation for each and every fitness function begins with a Polar-to-Cartesian coordinate conversion on the entire symbol set. Then, a distance calculation on each symbol, i, to each other symbol, j, follows. Symbol distance calculations are stored in a distance matrix, $\mathbf{D}_{i,j}$. Once the calculations listed are finished, each model's remaining calculations differ slightly.

$$Symbols = \begin{pmatrix} s_1 & = & 0 & 0 & 0 \\ s_2 & = & 0 & 0 & 1 \\ & & \vdots & \vdots & \vdots \\ s_8 & = & 1 & 1 & 1 \end{pmatrix} \tag{79}$$

$$\mathcal{H}_{i,j} = \begin{pmatrix} 0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 \\ 1 & 0 & 2 & 1 & 2 & 1 & 3 & 2 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 3 & 2 & 2 & 1 & 2 & 1 & 1 & 0 \end{pmatrix} \tag{80}$$

1. ***Brute force model***:

   The first model is called the brute force model because it takes into account the intuitive approach about how to represent this problem using only a high level of understanding of the problem domain. A maximization constellation fitness function is sought; therefore, a negative exponential decay of the distance is used to emphasize that it is better to have symbols further apart. The exponential decay is modified using a constant K value to increase or decrease the rate of decay depending on the number of symbols in the constellation. Finally, to account for the bit error increasing when encountering a high Hamming code distance between symbols, the exponentially decaying distance is multiplied by the Hamming code distance, found in the $\mathcal{H}$ matrix, of the two symbols. The resulting equation can be found in Equation 81 on page 244.

$$f_s(\vec{x}) = -\sum_{i=0}^{\mathcal{N}} \sum_{j=i}^{\mathcal{N}} \mathcal{H}_{i,j} * exp(-K * \mathbf{D}_{i,j}^2) \tag{81}$$

This model worked well for finding solutions that competed with the standard rectangular constellations. However, the thought was that by accounting for symbol energy an improvement would be seen over this fitness function. This resulted in Equation 82. The new fitness function describes the same fitness calculation as before, but the symbol set energy is used as a damping (dividing) factor – where energy is the sum of the squared radii for each symbol. An example of the landscape between two symbols using this new fitness function is illustrated in Figure 67.

$$f_s(\vec{x}) = -\frac{\sum_{i=0}^{\mathcal{N}} \sum_{j=i}^{\mathcal{N}} \mathcal{H}_{i,j} * exp(-K * \mathbf{D}_{i,j}^2)}{exp(-\sqrt{energy})} \tag{82}$$

Results of the modified brute force model were not as good as expected, resulting in development of the Volumes and False Alarm Rate models.

2. ***Volumes model***:

The volumes model is based on the principle that the probability of bit error landscape changes with respect to each symbol. Therefore, an k-objective model is designed to account for each new landscape that occurs when inspecting the transmission and reconstruction of each symbol with respect to every other symbol. A Hamming code distance scaled Gaussian distribution is placed on each symbol (*i.e.*, the higher the Hamming code distance, the larger the Gaussian) to represent each symbol's error footprint. Next, a grid is placed on the unit circle and at each grid position the maximum value found with respect to all Gaussian distributions are summed for a fitness value. Equations 83 and 84 represent the calculations used for this model where $\mathcal{S}$ contains the symbols in the entire constellation and $\mathcal{G}$ contains the values in the set $[-1, -0.95, \cdots, 0.95, 1]$ when using the step resolution of $\frac{1}{20}$ for this problem.

$$\mathcal{P}_{err}(i,j,h) = \left\{ \frac{exp\left(-\frac{\sum_{w=1}^{\hat{d}}(g_{i,j}^{(w)} - s_h^{(w)})^2}{2*\sigma^2}\right)}{(2\pi\sigma)} \right\} \tag{83}$$

$$\forall_{v \in S} \left\{ f_v = \sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{G}} Max\left(\forall_{h \in \mathbf{s}} \mathcal{H}_{v,h} \cdot \mathcal{P}_{err}(i,j,h)\right) \right\} \tag{84}$$

$$s.t.\ \mathcal{H}_{v,v} = 1\ and\ (\sqrt{i^2 + j^2} \leq 1)$$

Unfortunately, this model also did not prove to be a good representation of the Monte Carlo simulation. Moreover, hundreds of solutions evaluating to PF vectors along the Pareto front were found, so determination of good solutions still required a simulation evaluation for each.

3. ***False Alarm Rate model***:

The false alarm rate model is designed with the premise that each symbol has the same probability of selection, but the error associated with selection changes. Furthermore, the error found using a minimum selection formula should be used as the fitness of a constellation. This model is similar to the volumes problem in that it is an k-objective problem, un-scaled Gaussian distributions are placed on each symbol, a grid is placed on the unit circle, and Equation 83 can be used to calculate the probability of error when symbol v is transmitted and symbol h is reconstructed at grid position (i,j), $\mathcal{P}_{err}^{(v)}(i,j,h)$. The main differences are the fact that the Gaussian distributions are not scaled and the problem uses the minimum of the sum of the probability of error multiplied by the Hamming code distance as the selection criteria at each grid point.

$$f(v) = \sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{G}} Min \left(\sum_{h \in S} \mathcal{H}(v,h) \cdot \mathcal{P}_{err}(i,j,h) \right) \tag{85}$$
$$s.t.\ \mathcal{H}_{v,v} = 0\ and\ (\sqrt{i^2 + j^2} \leq 1)$$

Equation 85 mathematically describes the fitness calculation for the false alarm rate model. This is a minimization problem; in order to use this fitness function as a maximization function, a (-) sign is added to the final value. This model remains untested.

4. ***Minimize probability of error by calculating intersection point for error distribution model***:

This model is designed to include the P_b, overall constellation energy (E_s) and the $\frac{E_b}{N_o}$ parameter into the same fitness function. Much like the Volume model, calculations are based on the error associated with selecting the wrong symbol and the assumption that the noise added to the incoming signal has a Gaussian distribution. To sum the error associated with selecting the wrong symbol, it is advantageous to find the distance of the point where the two distributions (one on each symbol location) intersect. If the two symbols have the same distribution, the same scale for the distribution, and the same standard deviation, the intersecting point is the mid-point between the locations of the two symbols. However, if the distributions are scaled differently, but still maintain the same standard deviation and type of distribution, there is a adjustment that can be made to find the new intersecting point. Furthermore, this adjustment is calculated in Equation 86 where the Gaussian distribution scaling is the Hamming code distance between symbols. Finally, given that the distribution of error and the standard deviation of that distribution is the same for each symbols is the same, the probability of false alarm can be calculated by summing up the tail of the Gaussian away from this intersection point. This final calculation can be found in Equation 87. In addition, this model can also be a multiobjective model. However, the objectives are in the form of the parameter $\frac{E_b}{N_o}$ requiring optimization (*i.e.*, for a two bit or 4ary constellation, $\frac{E_b}{N_o} = \{0, 1, \cdots, 8\}$, making for a nine objective problem. See Figure 66.a for an illustration of why this range of $\frac{E_b}{N_o}$s is selected for a 4ary constellation.) The relationship of $\frac{E_b}{N_o}$ to the standard deviation, σ, is given in Equation 88. Thus, for each objective, the Gaussian distributions changes shape as the $\frac{E_b}{N_o}$ changes - making the P_b different for each different $\frac{E_b}{N_o}$ even when the constellation remains the same.

$$K(i,j) = \frac{D_{i,j}}{2} - \sigma^2 \cdot \frac{\ln(\mathcal{H}_{i,j})}{D_{i,j}} \tag{86}$$

$$f_e = |\mathcal{E}_s - S_{bits}| + \sum_{i=1}^{\|\mathcal{S}\|} \sum_{j=i+1}^{\|\mathcal{S}\|} erfc\left(\frac{K(i,j)}{\sqrt{2} \cdot \sigma}\right) \tag{87}$$

$$Let,\ E_b = 1\ and\ \sigma = \sqrt{\frac{1}{N_o}} = \sqrt{\frac{\frac{E_b}{N_o}}{E_b}} \tag{88}$$

$$\mathcal{E}_s = \frac{1}{\|\mathcal{S}\|} \sum_{i \in \mathcal{S}} \hat{R}_i^2 \tag{89}$$

Finally, the power calculation is found in Equation 89 where $\hat{R}_i$ is the Amplitude, or radius, of Symbol, i, and it is used in Equation 87. When $\mathcal{E}_s = S_{bits}$, $\mathcal{E}_s$ is optimal. It is unfortunate that this model also proved to not be as good as the first model. A modification to this model followed and Equations 90, where $\mathcal{Q}(x)$, is defined and Equation 91 are the result. This version of the model in this area has not been fully tested.

$$\mathcal{Q}(x) = \frac{1}{2} \cdot erfc\left(\frac{x}{\sqrt{2}}\right) \tag{90}$$

$$f_e = |\mathcal{E}_s - S_{bits}| + \sum_{i=1}^{\|\mathcal{S}\|} \sum_{j=i+1}^{\|\mathcal{S}\|} \mathcal{H}_{i,j} \cdot \mathcal{Q}\left(\frac{\mathbf{D}_{i,j}}{2 \cdot \sigma}\right) \tag{91}$$

5. ***Simulation model***:

   The simulation model acts as a correlation receiver described in the Detection of Signals in Gaussian Noise section of [205]. The number of symbols in a constellation is $2^{\mathcal{S}_{bits}}$. The simulation generates $(10000 * log_2(symbols))$ bits randomly. These random bits are converted to 10000 symbols. The 10000 symbols are then converted to their respective coordinates in the constellation. These coordinates are then multiplied by the basis functions (see Equation 92) and the results are transmitted. Next, a specific amount of Gaussian noise is added to match the required $\frac{E_b}{N_o}$ for a particular evaluation. The noise and signal are then correlated with the basis functions. Based on the output of the correlators, symbols are estimated according to their distances from each symbol in the constellation. Although this particular model showed to be more accurate, it is much more computationally prohibitive. Moreover, it takes this fitness calculation 33.93 seconds, as opposed to the 15.6 ms for an 8-bit symbol analytical model calculation on Cluster 1 listed in Table 13 in Chapter IV. This model is used to validate constellations found by MOMGA-IIa and results are used for comparisons drawn in Section 8.5.

$$basis\ function(h, time) = \frac{cos\left(\frac{2*\pi*h*time}{samples}\right)}{\sqrt{\left(\frac{samples}{2}\right)}} \tag{92}$$

The next section discusses the design of experiments, including system hardware and algorithm parameters.

8.4 Design of Experiments for m-ary Digital Symbol Set Design Problem

The Brute Force fitness model is used in the results of this study. Therefore, MOMGA-IIa is run in single objective mode; although, analysis of each constellation has two objectives no matter the technique used to find the constellation. For each configuration, the algorithm is run ten times. Algorithm settings can be found in Section 8.4.2. Solutions are presented in two manners: symbol set constellation and resulting P_b for each $\frac{E_b}{N_o}$. Comparisons are made using the Monte Carlo simulation model for determining P_b.

8.4.1 System Configuration. All experiments in this investigation were completed on Cluster 1 in Table 13 in Chapter IV. However, testing and development was conducted on Clusters 2 and 3, as well. The use of 64-bit compilers and linkers are used to take advantage of Cluster 1's 64-bit architecture. In addition, all experiments were run in serial mode (modified fmGA).

8.4.2 Algorithm Parameter Selection. MOMGA-IIa has many parameter settings – including a primordial phase schedule which by itself has numerous settings. Parameters addressed in this investigation are configuration, experiments, string size, $\mathcal{S}_{bits}$, BB sizes, encoding, cut-and-splice probability, CTs, and generations for the primordial and juxtaposition phases. Tables 41 and 42 list the settings used. In most cases, settings for all variables are the same for each configuration. However, as the problem size increases, string size is adjusted to allow for the algorithm to scale and find solutions in a reasonable amount of time. Table 41 lists the string sizes and $SLICE$[8] values for each configuration.

BB sizes are also a factor in these experimentations. The sizes are kept the same (1-20) throughout each configuration. These sizes have shown in past experiments [41] to be good for difficult problems. The random seed is set for each experi-

[8] A $SLICE$ of bits is the number of bits representing one variable within the chromosome.

Table 42: Algorithm Settings

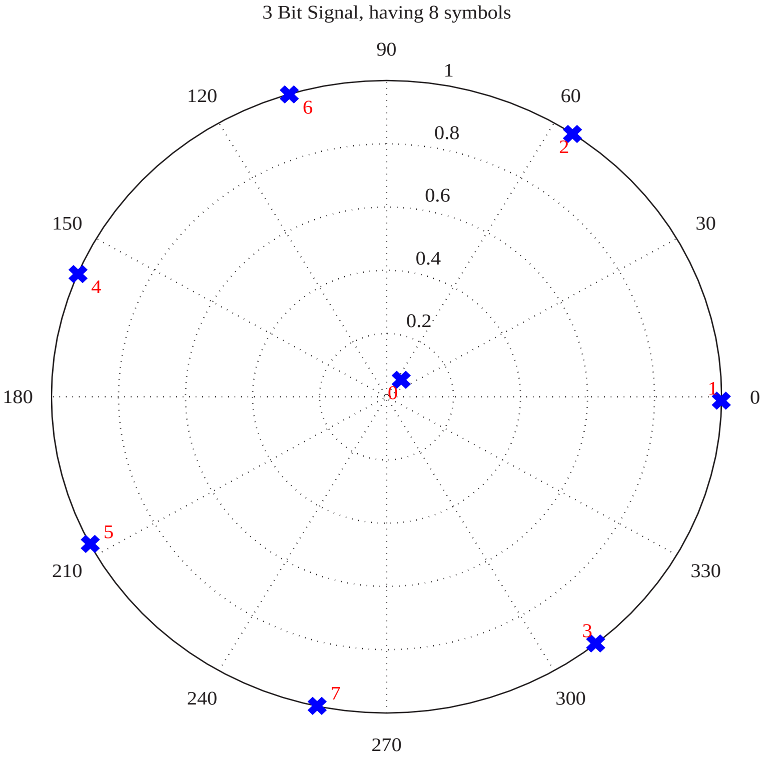
Variable		Setting
Random Seed	=	987654321
Experiments	=	10
Block Size (Min and Max)	=	1 20
Genic Alphabet	=	01
Encoding	=	BINARY
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Orthogonal Templates	=	32
Primordial_Generations	=	200 $\cdots$
Total_Generations	=	300 $\cdots$
n_a	=	500 $\cdots$

ment for solution repeatability. Population size is calculated according to Goldberg's population sizing equation [91]. Multiple competitive templates are used to increase algorithm effectiveness [47] – in these experiments seven competitive templates are used as well as 32 orthogonal competitive templates generated using a Latin Square. The orthogonal templates are built by filtering a randomly selected regular template through a bank of pre-generated orthogonal arrays.

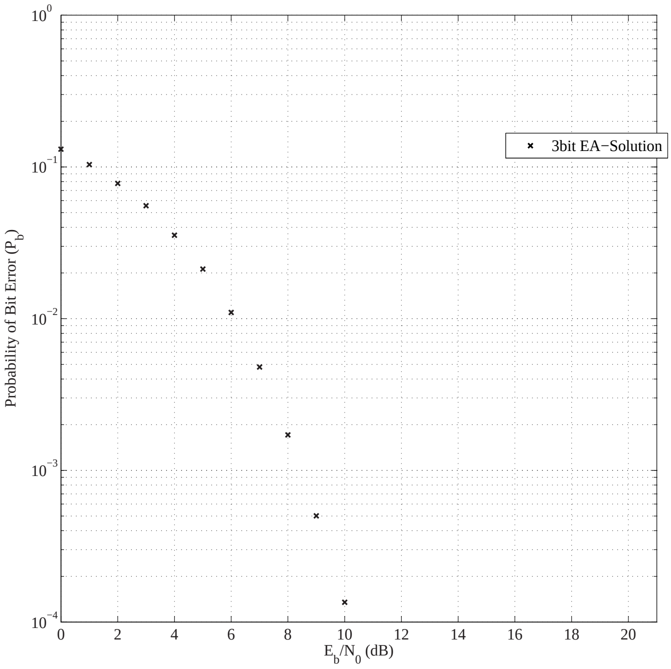
8.5 Results and Analysis

In this section, first the MOMGA-IIa-found constellations are graphically presented and discussed. Then, the results of a Monte Carlo simulation (see Section 8.3.2.5 on page 241) evaluation of MOMGA-IIa's constellations are discussed. Finally, the Monte Carlo simulation results are then compared, where able, to rectangular theoretical best.

8.5.1 MOMGA-IIa Discovered Constellations. 2-, 3-, 4-, and 5-bit solutions (constellations) found by MOMGA-IIa are illustrated in Figure 66, 68, 69, 70, 71, 72, and 73. Another interesting observation is that, other than the 4ary constellation, MOMGA-IIa constellations are not as symmetric as one might expect.

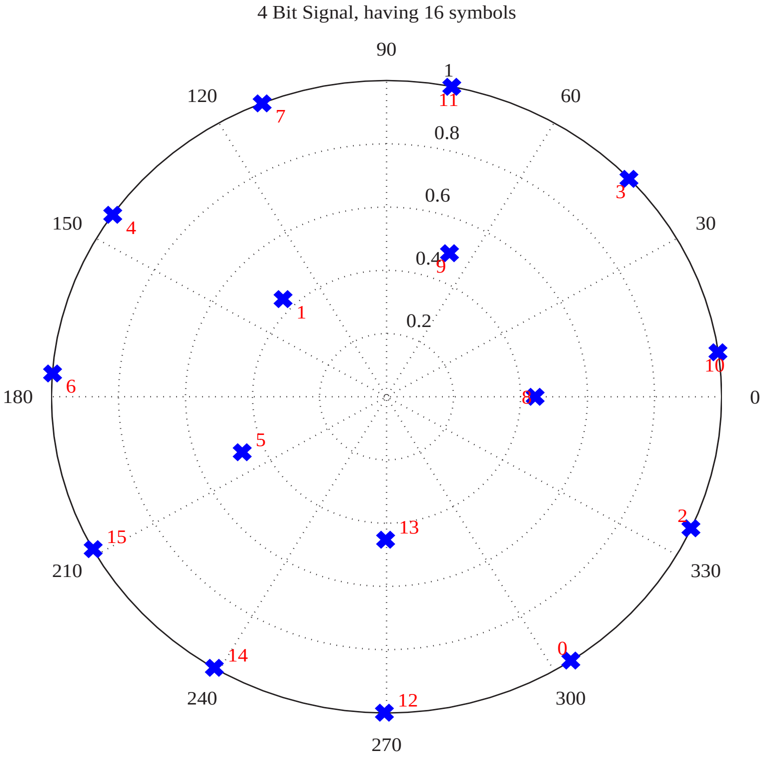


a) Best 8ary (3 bit) MOEA Constellation

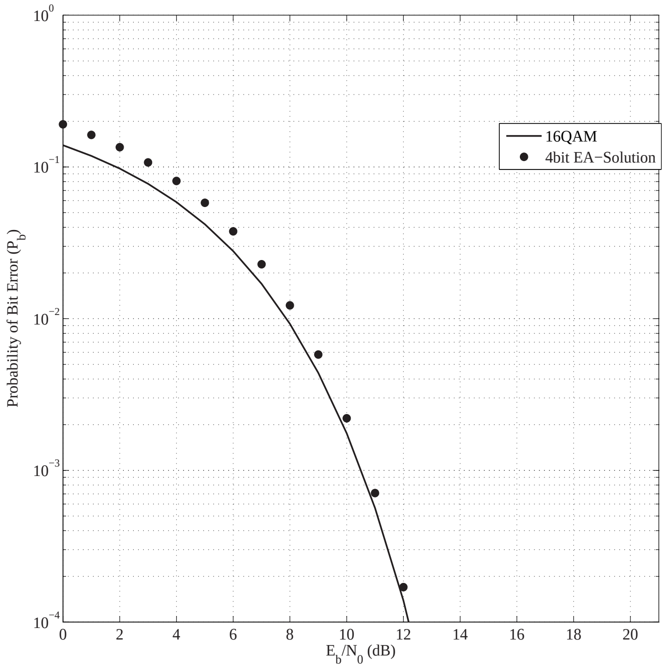


b) 8ary MOEA solution compares well to the 8ary rectangle constellation.

Figure 68: Presented is the best constellations found for 3-bit symbol set.

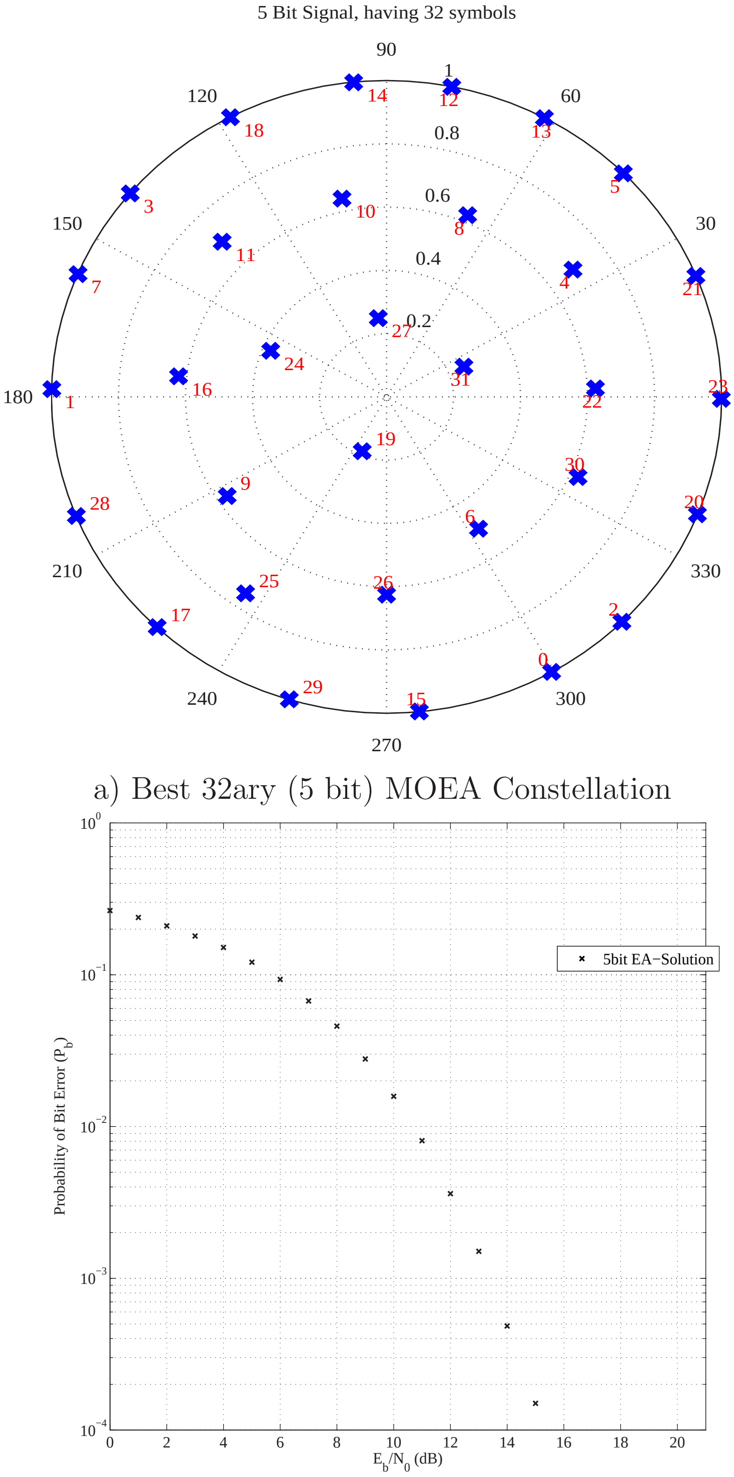


a) Best 8ary (4 bit) MOEA Constellation



b) 16ary MOEA solution compares well to the 16ary rectangle constellation.

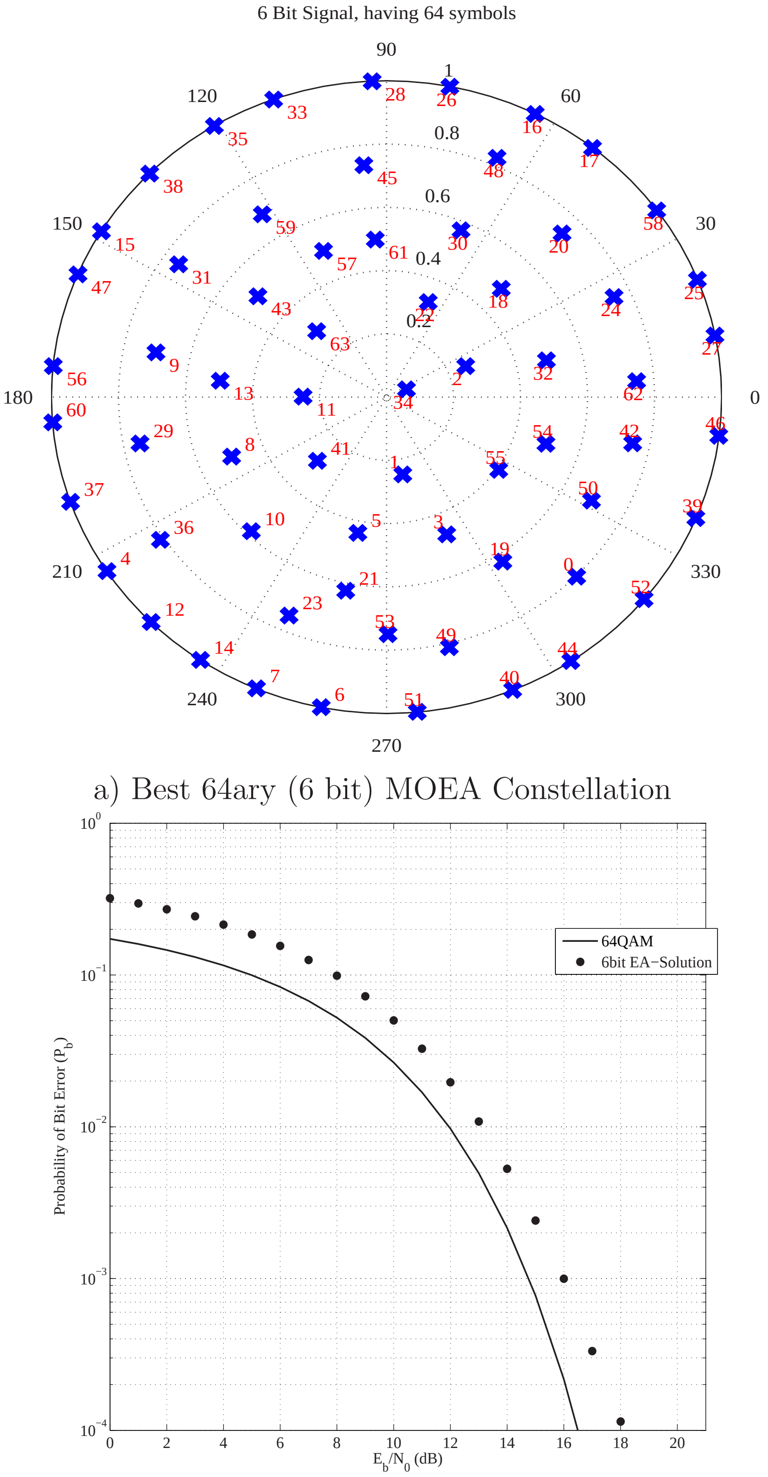
Figure 69: Presented is the best constellations found for 4-bit symbol set.



a) Best 32ary (5 bit) MOEA Constellation

b) 32ary MOEA solution and evaluated SNR plot.

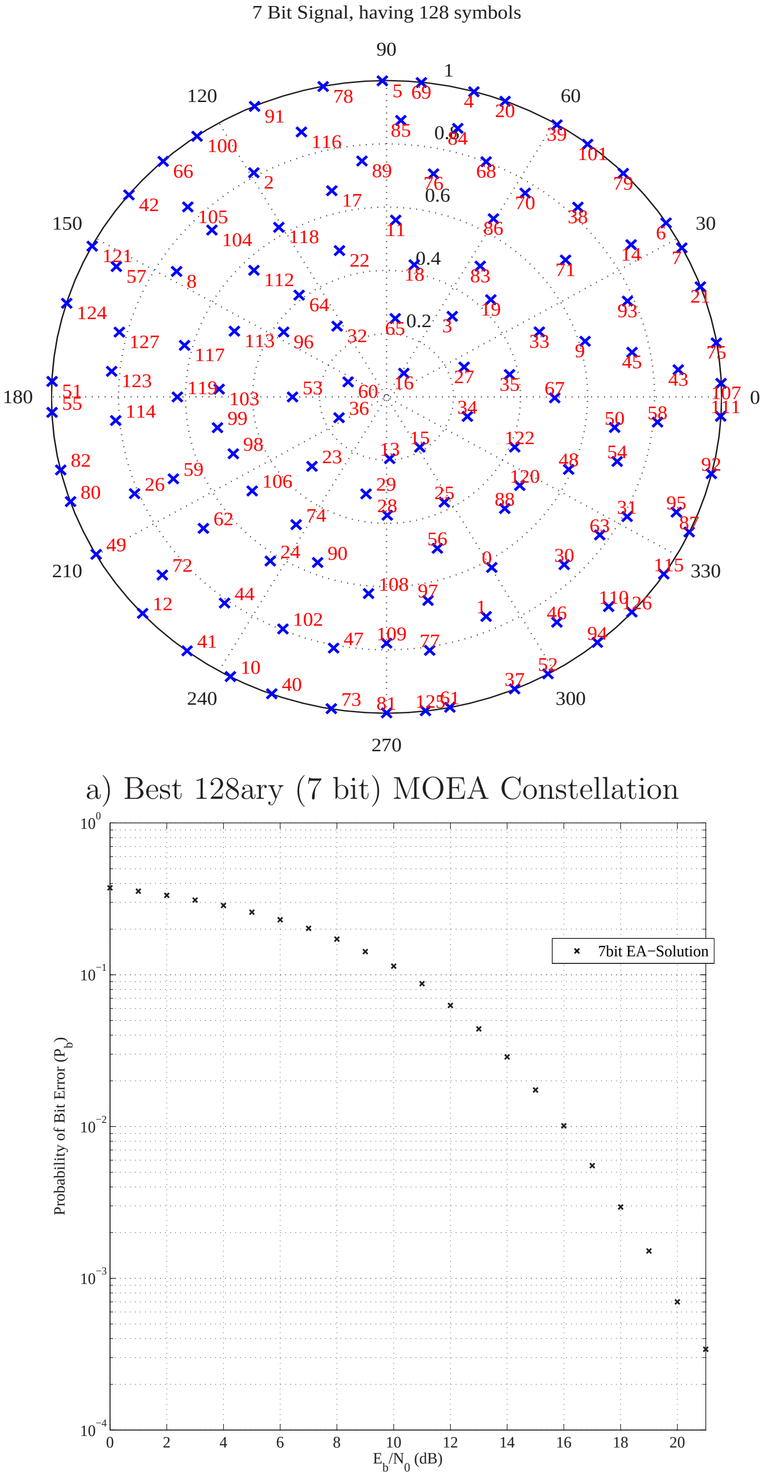
Figure 70: Presented is the best constellations found for 5-bit symbol set.



a) Best 64ary (6 bit) MOEA Constellation

b) 64ary MOEA solution compared to the 64ary rectangle constellation. Note the increase in P_b due to lack of model accuracy.

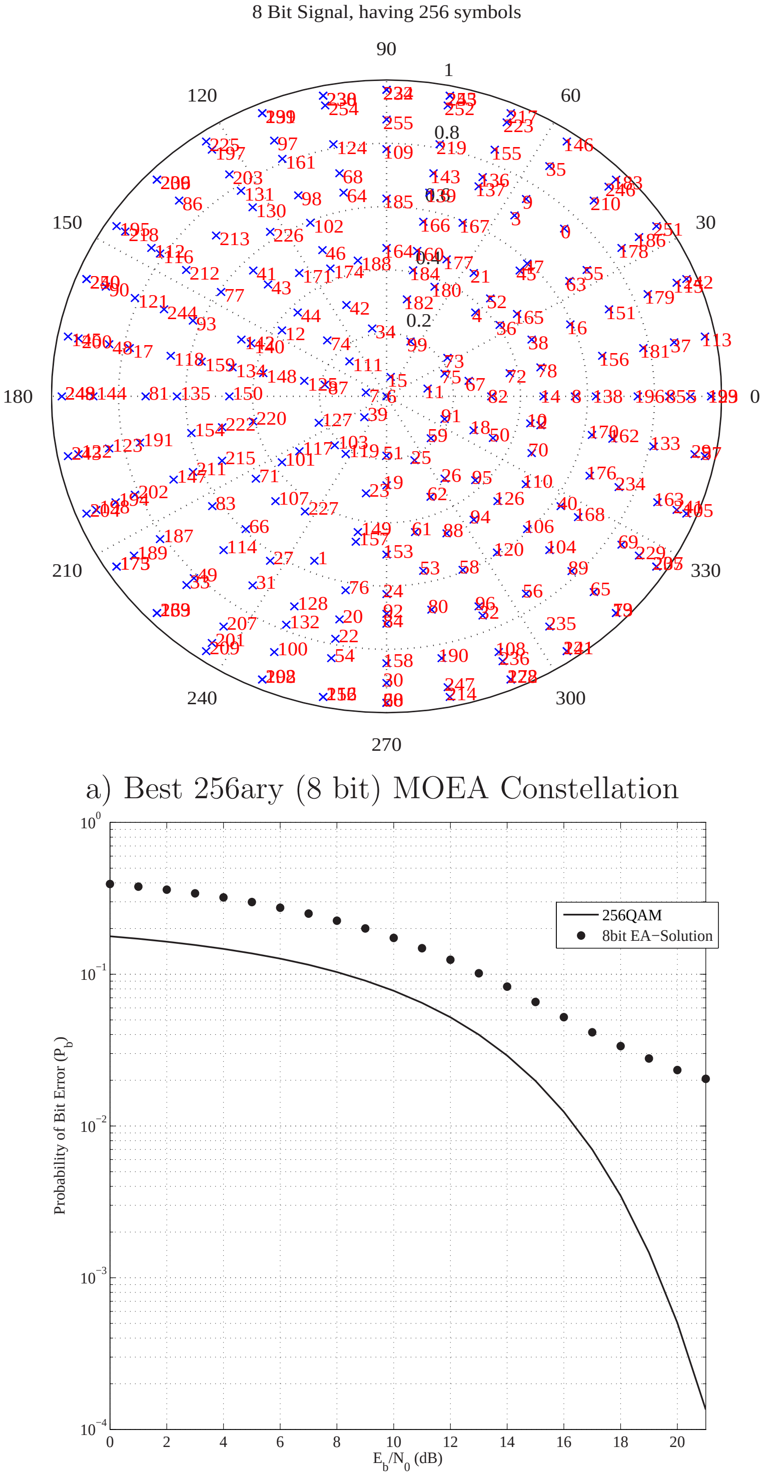
Figure 71: Presented is the best constellations found for 6-bit symbol set.



a) Best 128ary (7 bit) MOEA Constellation

b) 128ary MOEA solution's SNR plot.

Figure 72: Presented is the best constellations found for 7-bit symbol set.



a) Best 256ary (8 bit) MOEA Constellation

b) 256ary MOEA solution compared to the 256ary rectangle constellation. Note the increase in P_b due to lack of model accuracy.

Figure 73: Presented is the best constellations found for 8-bit symbol set. Although it is busy, the symbols are getting closer to one another, where the P_b increases due to inter-symbol interference.

8.5.2 Rectangular Constellations. The x-rectangular constellations have a different pattern than the EA found constellation patterns. A few examples of x-rectangular[9] constellations are illustrated in Figure 74. Note that all rectangular constellations are symmetric along both axes and there is no wheel spoke patterns like those found by the EA.

Evaluation of x-rectangular constellations can be accomplished with a Monte Carlo simulator; however, researchers using x-rectangular constellations have approximated the best $\frac{E_b}{N_o}$ versus P_b with Equation 93. In this equation, L represents the number of amplitude levels in a single dimension and can be calculated as $Log_2 L$ bits. This is the accepted formula to calculate P_b for any given allowable x-rectangular constellation [205]. Furthermore, this formula is used to evaluate MOMGA-IIa constellations.

$$P_B \simeq \frac{2(1 - L^{-1})}{log_2 L} * Q\left[\sqrt{\left(\frac{3log_2 L}{L^2 - 1}\right)\frac{2E_b}{N_0}}\right] \tag{93}$$

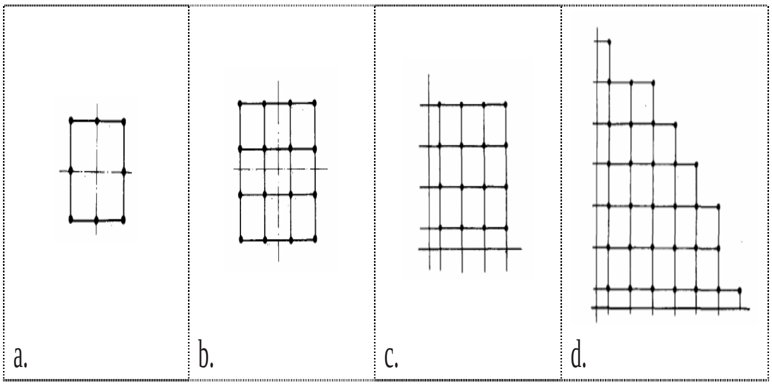


Figure 74: Illustrating x-rectangle constellation symbols.

8.5.3 Analysis. MOMGA-IIa constellations having two, three, four, and five bits all compete well with the x-rectangular constellation theoretical best. The comparisons are graphically presented in Figure 66. Unfortunately, comparisons with

[9] The x in x-rectangular represents the number of symbols in the constellation.

only rectangular constellations having 4, 16, 64, and 256 symbols can be made due to the limitation of the theoretical best equation. The only MOMGA-IIa constellation that did not come within 1 dB of the rectangular constellation theoretical best is the eight-bit or 256ary constellation.

In conclusion, the MOMGA-IIa has found comparable constellations using the brute force model described in 8.3.2.1. Although MOMGA-IIa solutions did not beat the x-rectangular constellation theoretical best, they did compete rather well. This is validated using standard Amplitude Modulation techniques in use today. Theoretically, there exists a constellation that has lower P_b than the x-rectangular constellations at certain $\frac{E_b}{N_o}$s; however, those constellations evade researchers in the field today and the MOMGA-IIa so far.

8.6 Summary

This chapter discussed the employment of MOMGA-IIa onto a digital symbol set design problem and the replacement of a highly computational simulation model with lower complexity mathematical model. The efficiency gain is well worth the employment of these mathematical models (see Section 8.2 on page 237 for time reduction when using an approximation model). MOMGA-IIa is used both as an EA (single objective) and MOEA (multiobjective). The algorithm finds good results for the model given to it; however, the models within this chapter do not represent well the Monte Carlo digital signal simulation they are meant to replace. It is for this reason that the results are not beating the rectangular QAM symbol set. It should be noted that all models are wrong, but some are useful [149]. In protein structure prediction, many different models have been used in hopes of more precisely capturing a sliver of the real work problem, so the solutions may make more sense. In addition, often times previously labeled good models are tested for validation with new equipment to ensure that an algorithm is solving the right problem. New discoveries or new equipment may lead to modifications in an old model or the development of an entirely new model that is more accurate in representing the problem. In the problem within this chapter, the model requires modification and possibly more theoretical development. It is for this reason that the MOMGA-IIa does not find better solutions. Next, the dissertation summary, conclusions and contributions are given.

IX. Contributions and Conclusions

This chapter concludes the body of this document with a discussion about contributions, the summary of how the goal and supporting objectives are met and future work. This research resulted in the development of a state-of-the-art explicit BB MOEA that solved or found good solutions for higher dimensional MOPs such as those found in relevant USAF real-world applications and benchmark test suite MOPs. The Protein Structure Prediction (PSP) Problem, Digital Symbol Set Design problem (DSSDP), and the futuristic OAV flight formation problems are the real-world applications studied. The new MOEA is thoroughly tested and compared to yesterday's state-of-the-art explicit BB MOEAs. In all experiments, it is shown that the new MOEA is statistically the same or better in effectiveness and efficiency than the previous algorithm.

On the way to the final development of this new technology, many research milestones produced additional contributions to the field. The document begins with a discussion of standardized definitions that are used throughout the MOEA field starting with the theoretical representation of an MOP, its solutions (decision variables) and associated objective vectors as values having infinite word length. A relationship is drawn between the theoretical representation and the computational-world representation of solutions and associated objective vectors. This discussion bridges the computational gap between the theoretical world and the computation world. It emphasizing the fact that the computational model of problems may be a altogether a different problem than the theoretical one it is intended to represents. Still, solutions relating to the computational model are related to the theoretical one in some manner that is problem specific.

The dialogue then moves away from the theoretical representation terminology embracing the computation model terminology. A short list of computational terms are presented to accent the fact that there are two domains considered when attempting to solve an MOP. These domains are called the solution space and objective space. The solutions space domain contains sets of decision variables while the objective space domain contains sets of vectors having values that are produced when a solution is evaluation. Furthermore, optimal solutions (decision variables) are said to be solutions that, once evaluated, have objective vectors that are nondominated. It is then conjectured that the set of optimal solutions are built from a optimal set of smaller units called BBs. *Contributions* begin with the analysis of the process by which the optimal set of BBs can be identified. Previous research ended with approximate definitions for good BBs (computational good partial solutions). *The limitation of these approximate good BB definitions is identified and new good BB definitions are forged. In addition, a new Optimal BB Set definition is developed as well as a upper bound to the expected number of unique BBs observed during a BBB search. Finally, the new good BB definitions are redefined as BB classifying systems and real-world BBB are characterized by the computational BB classifier they employed.* This concluded the BB analysis chapter and began the construction of a new MOEA based on these findings.

State-of-the-art explicit BBBs are hypothesized to perform better than implicit BBBs. This is also supported by the literature for a limited set of MOPs. Therefore, using the knowledge of what makes for a good BBB, *its BB classifying system*, a foundation EA is chosen for augmentation. The fmGA was picked because of its good BB classifying system. Although a multiobjective fmGA already did exist (MOMGA-II), it had been shown to be limited in scaling to large deceptions problems. Still, because it contained a good BB classifying systems it was hypothesizing that this type of EA should be as good, if not better, than any other explicit BBB. The previous version's code of construction was not consulted for augmentation except for the population sizing equation. This research contributions continue with the development of the new MOEA. Augmented into the new MOMEA are several new features. *An active archive, a competitive template management system and a BB solutions tracing mechanism is added into the new MOEA, now called the MOMGA-IIa.* The radical change in how an active archive and competitive template management system work together makes the MOMGA-IIa more efficient and effective. In fact, the new MOEA is shown to outperform the previous version over every MOP tested; however, this achievement was not as exciting as the new tracing mechanism allowing for the development of several new-to-the-field analysis tools. This marked the achievement of major contributions. *The BB evolution tracing allowed for a partial computational BB size assignment associated with PF_{known} vectors. This included a first ever, BB size assignment visualization of evolving BBs. In addition a first ever MOP partial epistatic level measurement and decision variable stability assessment contrived using a post mortem analysis of optimal solution findings.*. The innovative design of the MOEA allowed for these additional payoff contributions.

Once developed, testing of the MOEA using test suite MOPs was conducted. Although the metrics, test suite MOPs, and statistical methods used to identify differences between the two MOEAs metric results are not new, *the non-graphical analysis of a complete set of outcomes for each metric is new.* Most researchers actually show the data values associated with each test in a table or an elaborate comparison chart but never has someone concluded with the final outcomes in one table. This contribution is simple, effective, and easily understood if one knows what each metric measures. Future MOEA researchers are urged to use this metric summary table as a means to quickly illustrate effectiveness of tested MOEAs.

Other testing is accomplished using harder test problems called deception problems. In addition two USAF real-world application problems and one futuristic application problem is tested. Finally, *two new approximation techniques are tested to replace MOEA parallelization models.* The first is the replacement of a highly computational model using a neural network. The second is the replacement of a highly computational simulation using four mathematical models. Although replacement produced a negative result, these approximation model illustrated more than 16 times the speedup from the computation model. The final real-world problem tested is the *first ever attempt for an MOEA to optimize a m-ary digital symbol set design that can be generalized, once a solid model is found, to optimize symbol set designs according to channel noise characteristics*. This is a contribution in the digital communications field. In summary, this research has advanced technology in this field of study with a broad range of achievements. This dissertation serves as documentation of original work for extending the state-of-the-art MOEA research within the optimization research field. This research represents original ideas based in part from concepts of appropriately cited authors. Next, a summary of contributions are enumerated for quick identification. Following the summary of contributions, the goal and supporting objectives are reviewed with a brief statement conveying how each was met.

9.1 Contributions

Listed within this section is a brief summary of contributions achieved by this research. This summary is by no means a replacement for the detail summary found above; however, it does give the reader a quick reference from which to check while reflecting on the research stated within this document.

1. Explicit BB definitions are extended in Chapter II to include clarifying the meaning of good single and multiobjective BBs.
2. A more robust algorithm is developed where MOEA concepts are implicitly designed within the new algorithm.
3. A BB (solution) tracing mechanism is integrated within the new algorithm to enable a BBB researcher to evaluate required BB sizes for solving a particular problem.

BB size is not all that can be measured, but also the stability of variables within a problem can be measured. This includes identifying the usefulness and sensitivity within each decision variable.

4. The tracing mechanism allows for the development of a new metric serving the explicit BB MOEA community with a epistasis metric.
5. A new visualization technique is developed for the viewing of the genotype, phenotype and evolutionary process of the new algorithm finding solutions evaluating to Pareto front vectors while tracking the size of the BB required for finding each solution.
6. A new way to display metric results is identified. This method should be picked up by future MOEA researchers for quick and easy analysis between MOEA metric results.
7. Application of this MOEA on Air Force applications during this research and in future research is also a contribution.
8. software is written in a way that can be useful. New MOPs can be *Plugged* into the algorithm without integrating the new code directly into the algorithm.

   Matlab code is written to assist in the post mortem visualization of the BB trace.

This section is provided only to give a quick summarized list of each contributions provided by this research. Next, the goal and supporting objectives are restated accompanied by a summary of how each is met.

9.2 Meeting the goal and supporting objectives

This research effort did meet the intended objectives. This section describes how each supporting objective is achieved - thus meeting the final overall goal of this research. Table 43 is provided as a review of what is purposed in Chapter 1 as goal and supporting objectives.

Goal *Development of a more robust multiobjective evolutionary algorithm capable of solving larger range of NPC applications, which are relevant to United States Air Force's real-world applications. In addition, to advance the understanding design and application of BB based MOEAs.* The following objectives are met to attain the stated goal.

Objective 1: *Develop a new, more robust, algorithm to solve a larger range of MOPs. Test and validate MOEA metrics and statistical methods associated with comparing MOEAs.* Chapter III is dedicated to the design of a new, more robust algorithm. The new algorithm (MOMGA-IIa) offers better exploration and exploitation through the use of a competitive template management system (CTMS) than previous versions. The algorithm is shown to be more robust by being statistically more effective and efficient (never worse) on particular MOPs and applications than the previous version.

MOMGA-IIa statistically outperforms MOMGA-II when applied to all of the MOPs selected for study. The addition of problem domain knowledge can be easily integrated in an MOEA like this; however, it is thought that with the CTMS, this type of modification may be unnecessary in the future unless it is to bring more resolution to more feasible regions. The code is generalized enough to allow for easy addition of other MOPs; indeed, it can be said that the addition of new problems is as easy as filling out a template. The excellent performance and versatility of the MOMGA-IIa is an attractive tool for attempting to solve real-world MOPs. This illustrates that the MOMGA-IIa meets objective 1.

Objective 2: *Indicate how MOEA research advantageously applies to Air Force applications.*

Three Air Force applications are selected to apply this new robust algorithm toward. The first is the Protein Structure Prediction problem. Although this algorithm is used mainly as a tool to gather good records for the use in training a neural network for fitness function replacement, the algorithm itself solves this problem using both its multiobjective and single objective capabilities. The second application is the Organic Air Vehicle (OAV) problem where unmanned vehicle fight patterns are organized according to communication requirements between a group of heterogeneous OAVs. The final application of this MOEA is used to design digital symbol sets for lowering the probability of bit error rate during transmission through Gaussian white noise.

Although these problems are not completely solved (especially NPC problems) by any measure, it is illustrated within this document that MOMGA-IIa is statistically better than the earlier version of MOMGA series (MOMGA-II) and that MOMGA-IIa finds good solutions to these difficult real-world Air Force application problems. This being said, clearly MOMGA-IIa meets the objective 2.

Objective 3: *Extend MOEA understanding by extending the meaning of BBs in an explicit search algorithm.*

Within Chapter II on page 19, previous state-of-the-art BB definitions are explored. Newly forged definitions and perspectives redefine the state-of-the-art BB definitions through the use of more rigorous definitions and conjectures. A new different view of BBBs, what it means when someone references good BBs, and categories of types of good BBs are formed within. Moreover, conjectures and definitions are stated proposing a new reasoning for the underlying goals of today's state-of-the-art BBBs (a "how things work" for these BBBs). A division between good algorithm domain BBs and good problem domain BBs is delineated. Finally, the concept of an upper bound limit is derived that defines the limitation, in terms of available gain, for any explicit BBB. Finally, the concept of what an optimal BB set is defined, couched with the idea that the basic goal for every explicit BBB is seeking a set of BBs to put together to build the optimal solution - be they single or multi-objective.

The new definitions for describing good BBs and delineation of how BBs are defined in both the problem and algorithm domains extend the understanding of what BBs are for an explicit BBB, thus, meeting Objective 3.

Objective 4: *Indicate that explicit BB MOEA approaches are statistically similar in solving studied deception problems; however, different in search methods.*

Search method differences between the mBOA and the top-down BB MOEA approach, like MOMGA-IIa, are discussed in Chapter II where the underlying rules for both BBBs are well defined. Statistical similarities are empirically

Table 43: Goal continued and supporting Objectives

Goal	**Objectives**
Development of a more robust multiobjective evolutionary algorithm capable of solving larger range of NPC applications, which are relevant to United States Air Force's real world applications.	**1:** Develop a new algorithm that is more robust (effective) in solving a larger range of MOPs. Test and valid metrics and statistical methods are selected for comparing MOEAs.
	2: Indicate how MOEA research advantageously applies to Air Force applications
To advance the understanding, design and application of BB based MOEAs.	**3:** Extend MOEA understanding by extending the meaning of BBs in an explicit search algorithm
	4: Indicate that explicit BB MOEA approaches are statistically similar in solving deception problems; however, different in search methods.
	5: Use MOEA generic objectives as preserve, progress, and diversity for extending an explicit BB MOEA.
	6: Develop possible MOEA techniques that can be employed in an efficient parallel design and implementation.

validated in Chapter V, when solving the deception problems. Finally, other publications within the EC field are used in comparison with the Bayesian approach to further illustrate the similarities between these approaches. The IEA, a single objective explicit BBB, are described and compared to the BOA within Appendix F section F.3 on page 368 and all other BBB approaches can be found in Appendix E in section F.5.1 on page 372. These sections describe the major BBB approaches: single and multiobjective, plus past and present state-of-the-art. The IEA study statistically shows that the IEA and BOA are also similar in effectiveness. This is a second example making the point that these two approaches are similar in effectiveness. This reasoning is submitted to meet objective 4.

Objective 5: *Use MOEA generic objectives extending the preserve, progress, and diversity within an explicit BB MOEA.*

Researchers in the MOEA field recognize that every successful MOEA has four fundamental principles: preserve, progress, maintain diversity, and provide. Each of these MOEA principles has an impact on how well an MOEA might perform given a particular MOP. It is challenging to insert the best form of each of these principles within an MOEA without disrupting the harmony of the search process. Within the new MOEA, a few principles are met with an innovative design that makes for a compatible fit within the original design of the EA and then the previous versions of this MOEA. Mainly, the preserve and maintain diversity principles are enhanced within MOMGA-IIa by a unique and innovative design of an archived competitive template management system. The progress and provide principles are also implicit in design of the algorithm where final optimal solutions and related PF vectors are listed, along with statistical data, in files.

Clearly, the previous MOMGA-series algorithms were missing some fundamentals of MOEA technology. The MOMGA-IIa meets the shortfalls and is statistically shown to outperform the MOMGA-II on certain MOPs. This innovative design meets the requirement for objective 5.

Objective 6: *Develop possible MOEA techniques that can be employed in an efficient parallel implementation.*

The first fitness function approximation method studied is the replacement of the force field fitness function as a way to reduce the computational cost for evaluating a protein conformation. Fitness function replacement has been studied before; however, a function having this complexity and lack of oscillations has not. Chapter VI is dedicated to an evaluation on replacing the CHARMm force field model fitness function with a neural network. Although, it is decided that this not be implemented due to the low quality in solution reproduction, other methods are suggested to try. For the CHARMm it is thought that the lack of oscillation pattern within the function hinders the ANN in accuracy. An approximation function that is not accurate enough may actually misguide the MOEA to areas not having good solutions at all. Thus, it is thought that this particular ANN replacement is not ideal.

The second simulation model approximation method studied is the replacement of the Monte Carlo simulation used for testing digital symbol set designs. Four different approximation models are tested. Each are found to be more efficient than the simulation mode; however, none are found to be accurate enough to replace the Monte Carlo simulator. Furthermore, new symbol set designs did not best rectangular QAM designs.

These two studies along with the literature review of other approximation methods meets objective 6 requirements.

This section described how each objective is met in regards to attaining the overall goal of this research. Next is the conclusions drawn from this dissertation.

9.3 Summary

By meeting each supporting objective, this research has met the goal. This MOEA research is useful for both the Air Force and civilian applications. This research extends state-of-the-art explicit BB MOEA research and contributes to optimization research fields of study in several application areas: bioinformatics, unmanned vehicle flight patterns, and digital communication symbol set design. Test problems of many characteristics are used to illustrate that a new algorithm design is never worse and scales better than an earlier version of the MOMGA series explicit BB MOEA. This research is sponsored by Abel S. Nunez in the Air Force Research Laboratory, Sensors Directorate, (RF) Sensor Technology Division, Electronic Warfare Technology Branch, Dr. Robert L. Ewing in the Air Force Research Laboratory, Embedded Information Systems Engineering and Technologies Branch, and Dr. Ruth Pachter in the Air Force Research Laboratory, Materials Directorate.

9.4 Future Work

Future implementations for MOMGA can use an EDA and/or BOA for an additional BB selection method much like the operator selection mechanism in GENMOP that statistically selects operators according to their past performance. Furthermore, many new avenues of research may proceed from the development of this novel MOEA design: new parallel techniques, BB classifiers, co-evolutionary approaches applied to a new set of applications. Appendix G on page 378 is provided to give MOEA researchers some ideas for furthering this research.

Appendix A. MOEA Parameter Settings

This appendix contains all the parameter settings for each experiment within this document. Each section is label according to the MOP tested. The MOMGA-II and MOMGA-IIa are the algorithms under test.

A.1 MOP 1

Table 44: MOP 1 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	24
Block Size (Min and Max)	=	1 5
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	100 100 100 100 100
Total Generations	=	200 200 200 200 200
n_a	=	100 100 100 100 100

Cut gen	Str len	Threshold
5	24	22
6	14	11
7	10	5
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
15	3	2
22	2	1
24	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.2 MOP 2

Table 45: MOP 2 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	27
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	400 400 400 400 400 400 400 400 400 400
n$_a$	=	50 50 50 50 50 50 50 50 50 50

Cut gen	Str len	Threshold
5	27	22
6	23	11
7	20	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
22	3	2
23	2	1
27	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.3 MOP 3

Table 46: MOP 3 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	27
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	400 400 400 400 400 400 400 400 400 400
n_a	=	50 50 50 50 50 50 50 50 50 50

Cut gen	Str len	Threshold
5	20	22
6	14	11
7	11	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
15	3	2
19	2	1
20	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.4 MOP 4

Table 47: MOP 3 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	24
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	30 30 30 30 30 30 30 30 30 30
Total Generations	=	50 50 50 50 50 50 50 50 50 50
n_a	=	40 40 40 40 40 40 40 40 40 40

Cut gen	Str len	Threshold
5	24	20
6	20	11
7	10	5
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
15	3	2
19	2	1
24	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.5 MOP 6

Table 48: MOP 6 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	20
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	20 20 20 20 20 20 20 20 20 20
Total Generations	=	40 40 40 40 40 40 40 40 40 40
n_a	=	50 50 50 50 50 50 50 50 50 50

Cut gen	Str len	Threshold
5	20	12
6	14	11
7	11	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
15	3	2
19	2	1
20	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.6 Tanaka

Table 49: MOP tanaka MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	24
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	20 20 20 20 20 20 20 20 20 20
Total Generations	=	40 40 40 40 40 40 40 40 40 40
n_a	=	50 50 50 50 50 50 50 50 50 50

Cut gen	Str len	Threshold
5	24	12
6	14	11
7	11	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
15	3	2
19	2	1
24	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.7 DTLZ3

Table 50: MOP dtlz3 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	28
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	20 20 20 20 20 20 20 20 20 20
Total Generations	=	40 40 40 40 40 40 40 40 40 40
n_a	=	50 50 50 50 50 50 50 50 50 50

Cut gen	Str len	Threshold
5	28	12
6	23	11
7	20	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	2
13	5	2
14	4	2
22	3	2
23	2	1
28	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 20 t r 0	c 200 t r 0
	c 20 t r 0
	c 20 t r 0
	c 20 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0
m 3 0 0 0 0	m 3 0 0 0 0

A.8 mQAP 1

Table 51: MOP mQAP MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	50
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	500 500 500 500 500 500 500 500 500 500
n_a	=	100 100 100 100 100 100 100 100 100 100
Cut gen Str len Threshold		
5 50 30		
6 48 30		
7 37 25		
8 35 20		
9 33 20		
10 32 20		
12 29 9		
13 25 8		
14 14 7		
15 13 7		
17 12 5		
18 11 5		
19 10 5		
20 9 5		
21 8 4		
22 7 3		
25 6 2		
30 5 2		
33 4 2		
35 3 2		
48 2 2		
50 1 1		
Competitive Template File		
MOMGA-IIa		(**MOMGA-II**)
c 20 t r 0		c 200 t r 0
		c 20 t r 0
		c 20 t r 0
		c 20 t r 0
		c 20 t r 0
		c 20 t r 0
		o 20 t r 0
m 1 0 0 0 0		m 1 0 0 0 0
m 2 0 0 0 0		m 2 0 0 0 0

A.9 T1T2 30

Table 52: MOP T1T2 30 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	30
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(11)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	100 100 100 100 100 100 100 100 100 100
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	10 10 10 10 10 10 10 10 10 10

Cut gen	Str len	Threshold
5	30	20
6	29	10
7	28	10
8	27	10
9	25	10
10	23	10
11	13	7
12	12	5
13	11	5
15	10	5
17	9	5
19	8	4
20	7	3
21	6	2
22	5	5
23	4	4
25	3	3
28	2	2
30	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.10 T1T2 60

Table 53: MOP T1T2 60 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	60
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(11)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	250 250 250 250 250 250 250 250 250 250

Cut gen	Str len	Threshold
1	60	45
3	50	35
4	40	31
6	35	26
8	30	21
9	25	10
10	15	5
12	10	3
13	9	3
14	8	3
25	7	3
26	6	3
27	5	3
38	4	3
49	3	3
53	2	2
60	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.11 T1T2 90

Table 54: MOP T1T2 90 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	90
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(15)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	500 500 500 500 500 500 500 500 500 500

Cut gen	Str len	Threshold
1	90	25
2	80	25
3	70	25
4	60	21
5	40	9
9	12	5
10	10	5
12	9	2
13	8	2
14	7	2
25	6	2
26	5	2
27	4	2
38	3	2
59	2	2
90	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.12 T3T4 30

Table 55: MOP T3T4 30 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	30
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(11)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	100 100 100 100 100 100 100 100 100 100
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	100 100 100 100 100 100 100 100 100 100

Cut gen	Str len	Threshold
5	30	12
6	23	11
7	20	5
8	10	2
9	9	2
10	8	2
11	7	2
12	6	3
13	5	2
14	4	2
22	3	2
23	2	1
30	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.13 T3T4 60

Table 56: MOP T3T4 60 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File

Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	60
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(15)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	250 250 250 250 250 250 250 250 250 250

Cut gen	Str len	Threshold
1	60	45
3	50	35
4	40	31
6	35	26
8	30	21
9	25	10
10	15	5
12	10	3
13	9	3
14	8	3
25	7	3
26	6	3
27	5	3
38	4	3
49	3	3
53	2	2
60	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.14 T3T4 90

Table 57: MOP T3T4 90 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	90
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(20)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	500 500 500 500 500 500 500 500 500 500

Cut gen	Str len	Threshold
1	90	25
2	80	25
3	70	25
4	60	21
5	40	9
9	12	5
10	10	5
12	9	2
13	8	2
14	7	2
25	6	2
26	5	2
27	4	2
38	3	2
59	2	2
90	1	1

Competitive Template File	
MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

A.15 PSP

Table 58: MOP PSP MET MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File

Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	90
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(20)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	200 200 200 200 200 200 200 200 200 200
Total Generations	=	300 300 300 300 300 300 300 300 300 300
n_a	=	500 500 500 500 500 500 500 500 500 500

Cut gen	Str len	Threshold
2	240	210
4	230	200
6	220	180
7	215	148
8	210	138
9	200	128
10	190	118
11	180	108
13	150	102
19	139	100
20	83	17
32	58	10
33	20	9
34	19	8
35	18	7
37	17	6
38	16	5
39	15	5
55	14	6
67	13	6
70	12	5
80	11	5
95	10	5
100	9	2
150	8	2
165	7	2
210	6	2
215	5	2
220	4	2
225	3	2
230	2	1
240	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 200 t r 0	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

Table 59: mQAP size 10 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	100
Block Size (Min and Max)	=	1 10
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(10)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	300 300 300 300 300 300 300 300 300 300
Total Generations	=	500 500 500 500 500 500 500 500 500 500
n_a	=	500 500 500 500 500 500 500 500 500 500

Cut gen	Str len	Threshold
5	100	60
6	95	40
7	90	35
8	85	30
9	60	20
10	45	20
12	25	9
13	15	8
14	14	7
15	13	7
25	12	5
40	11	5
65	10	5
70	9	5
71	8	4
72	7	3
75	6	2
77	5	5
80	4	4
85	3	3
88	2	2
100	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 50 t r 0	c 200 t r 0
	c 50 t r 0
	c 50 t r 0
	c 5 t r 0
	c 5 t r 0
	c 5 t r 0
	c 2 t r 0
	c 200 t r 0
	o 100 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

Table 60: mQAP size 20 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	200
Block Size (Min and Max)	=	1 10(15)
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(22)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	300 300 300 300 300 300 300 300 300 300 (... 300 300 300 300 300)
Total Generations	=	500 500 500 500 500 500 500 500 500 500 (... 500 500 500 500 500)
n_a	=	500 500 500 500 500 500 500 500 500 500 (... 500 500 500 500 500)

Cut gen	Str len	Threshold
2	200	110
4	190	100
6	180	80
7	175	38
8	170	38
9	150	38
19	139	27
20	83	17
32	58	10
34	20	6
34	19	6
35	18	6
37	17	6
38	16	5
39	15	5
55	14	6
67	13	6
70	12	5
80	11	5
95	10	5
100	9	2
150	8	2
165	7	2
170	6	2
175	5	2
180	4	2
185	3	2
190	2	1
200	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 50 t r 0	c 200 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 5 t r 0
	c 5 t r 0
	c 5 t r 0
	c 2 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

Table 61: mQAP size 30 MOMGA-II and MOMGA-IIa settings for experimentation. Listed are the MOMGA-II setting and beside them are listed, in parenthesis, MOMGA-IIa setting.

Parameters File		
Random Seed	=	random setting
Monte Carlo function calls	=	0
Monte Carlo time out (seconds)	=	0
Exhaustive Search	=	N
Experiments	=	10
String Length	=	300
Block Size (Min and Max)	=	1 12(20)
Genic Alphabet	=	01
Encoding	=	0
Shuffle Number (> 1)	=	2
Cut Probability	=	0.02
Splice Probability	=	1.0
Overflow (> 1.0)	=	1.6
Competitive Template Guesses	=	1
Inverse Template	=	N(Y)
Orthogonal Templates	=	0(22)
Energy Farms	=	0
Conjugate Gradient	=	0.02
Probablity Baldwinian	=	0.5
Primordial Generations	=	300 300 300 300 300 300 300 300 300 300 300 300 (... 300 300 300 300 300 300 300 300 300)
Total Generations	=	500 500 500 500 500 500 500 500 500 500 500 500 (... 500 500 500 500 500 500 500 500 500)
n_a	=	500 500 500 500 500 500 500 500 500 500 500 500 (... 500 500 500 500 500 500 500 500 500)

Cut gen	Str len	Threshold
2	300	210
4	290	200
6	280	180
7	275	148
8	270	138
9	250	128
10	230	118
11	210	108
13	150	102
19	139	100
20	83	17
32	58	10
34	20	9
34	19	8
35	18	7
37	17	6
38	16	5
39	15	5
55	14	6
67	13	6
70	12	5
80	11	5
95	10	5
100	9	2
150	8	2
165	7	2
200	6	2
275	5	2
280	4	2
285	3	2
290	2	1
300	1	1

Competitive Template File

MOMGA-IIa	(**MOMGA-II**)
c 50 t r 0	c 200 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 50 t r 0
	c 5 t r 0
	c 5 t r 0
	c 5 t r 0
	c 2 t r 0
	c 200 t r 0
	o 200 t r 0
m 1 0 0 0 0	m 1 0 0 0 0
m 2 0 0 0 0	m 2 0 0 0 0

Appendix B. Raw Statistical Results

This appendix presents metric results for all experiments run. The algorithms under test are the MOMGA-IIa (IIa) and the MOMGA-II (II). Displayed in each graph are the mean (*) and standard deviation (bar) results of 30 experiments for each MOP solved. The 10 metrics are the following: Error Ratio (ER), Generational Distance (GD), Hyperarea Ratio (HA), Spacing (S), Overall Non-dominated Vector Generation (ONVG), Overall Non-dominated Vector Generation Ratio (ONVGR), Max PF error, epsilon indicator (ϵ), the utility R_2 indicator, and the utility R_3 indicator. Final statistical analysis between each of these algorithms is done using a Kruskal Wallis test[1]. Chapter IV presents the final results of the Kruskal Wallis test on the data for each MOP tested.

[1] This method is approved as being a good way to determine if two algorithms are different. The assumptions made are that each algorithm is given a different random seed for each of the 30 experiments, the algorithm itself is a random process, output or responses from the MOEA are continuous and the function or metric responses are also continues. Also, using the KWtest records differences in the medians of the results for each metric, but results that are claimed to be different under the KWtest certainly must also be concluded as different under the student-t and z test.

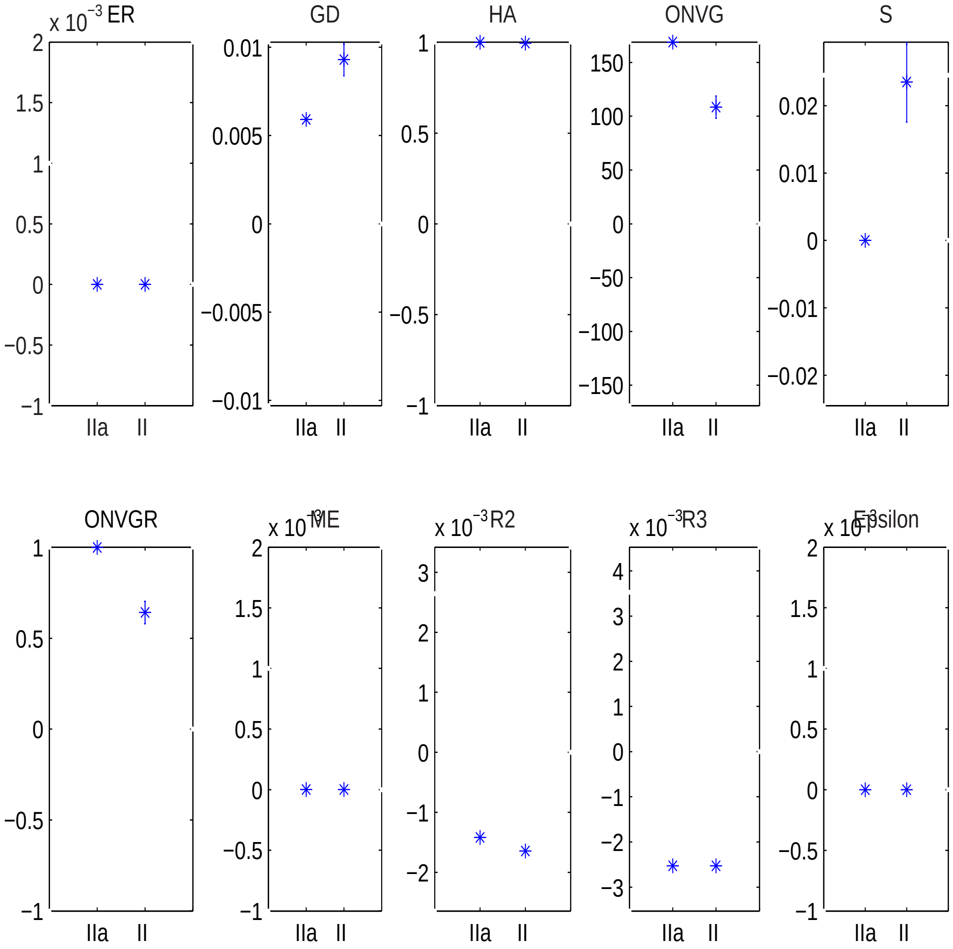


Figure 75: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL1. Results indicate visually that these MOEAs perform similarly over all metrics except ONVG and ONVGR where the MOMGA-IIa found more PF$_{true}$ vectors than MOMGA-II.

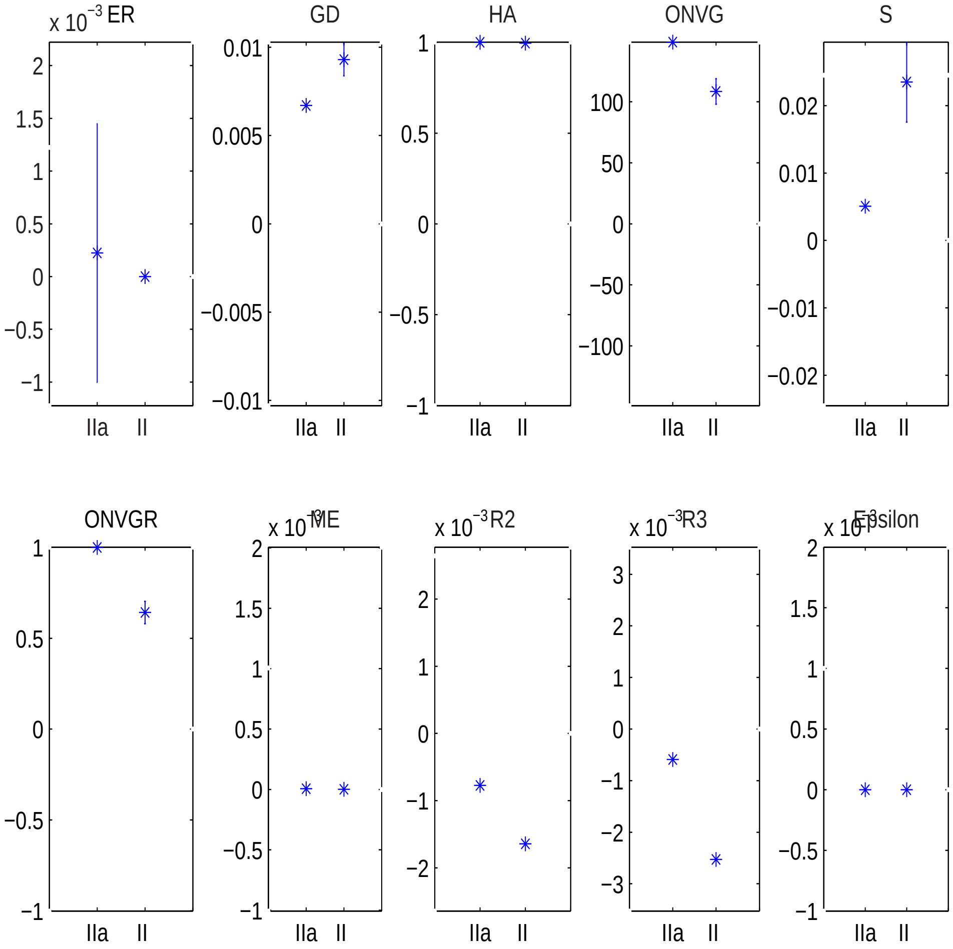


Figure 76: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL2. Results indicate visually that these MOEAs perform similarly over all metrics except ONVG and ONVGR where the MOMGA-IIa found more PF_{true} vectors than MOMGA-II.

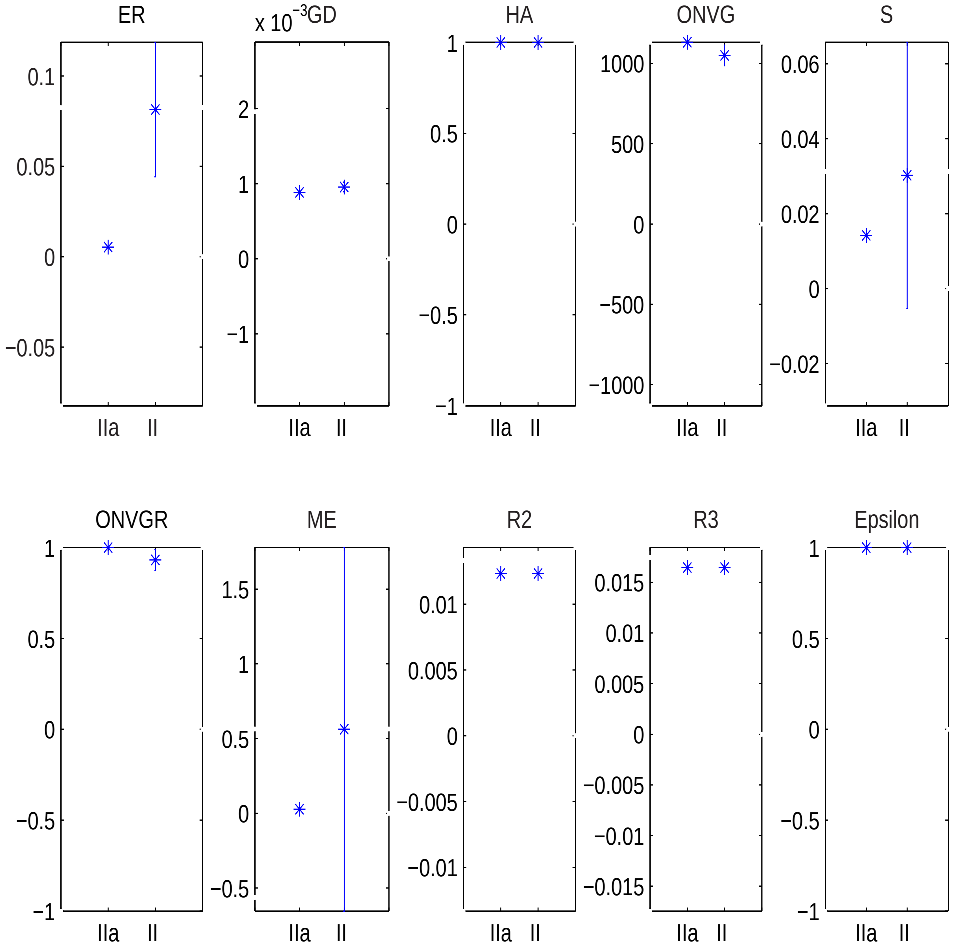


Figure 77: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL3. Results indicate visually that these MOEAs perform similarly over all metrics except ER where the MOMGA-IIa found all PF$_{true}$ vectors while the MOMGA-II did not.

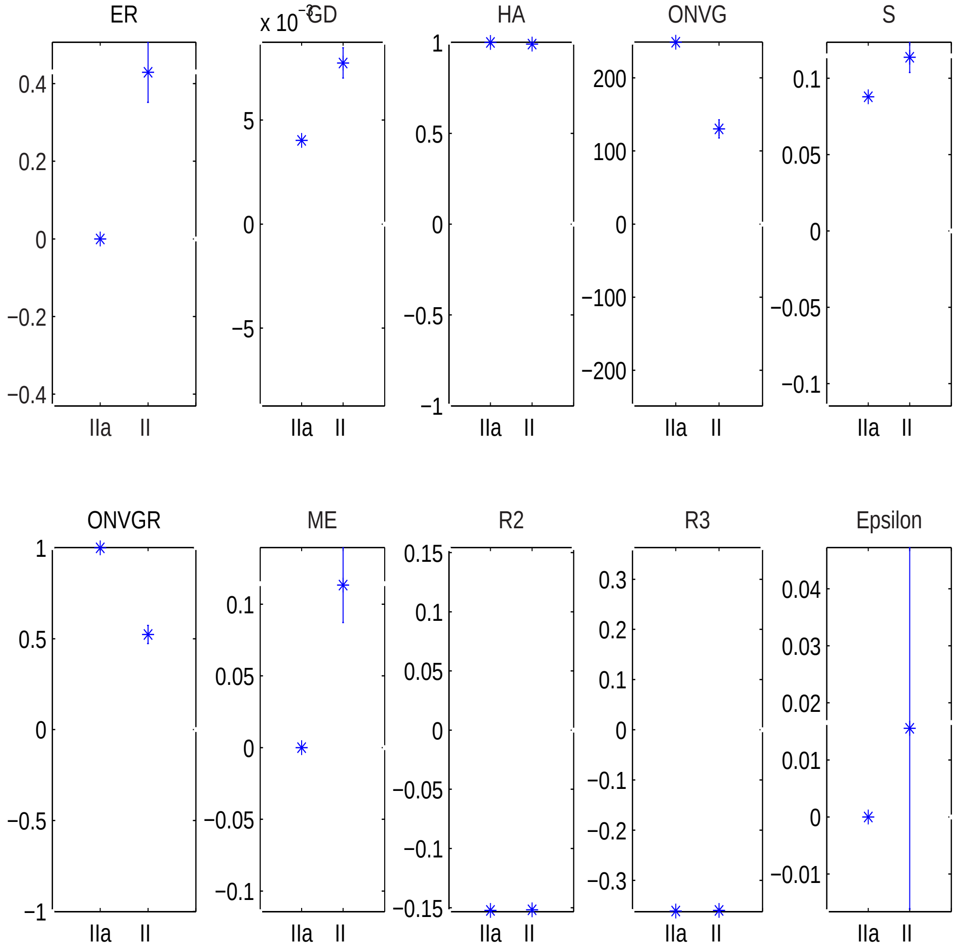


Figure 78: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL4. Results indicate visually that these MOEAs perform similarly over all metrics except ER, ONVG, ONVGR, and ME where the MOMGA-IIa found all PF_{true} vectors and is overall closer to the true Pareto front than the MOMGA-II.

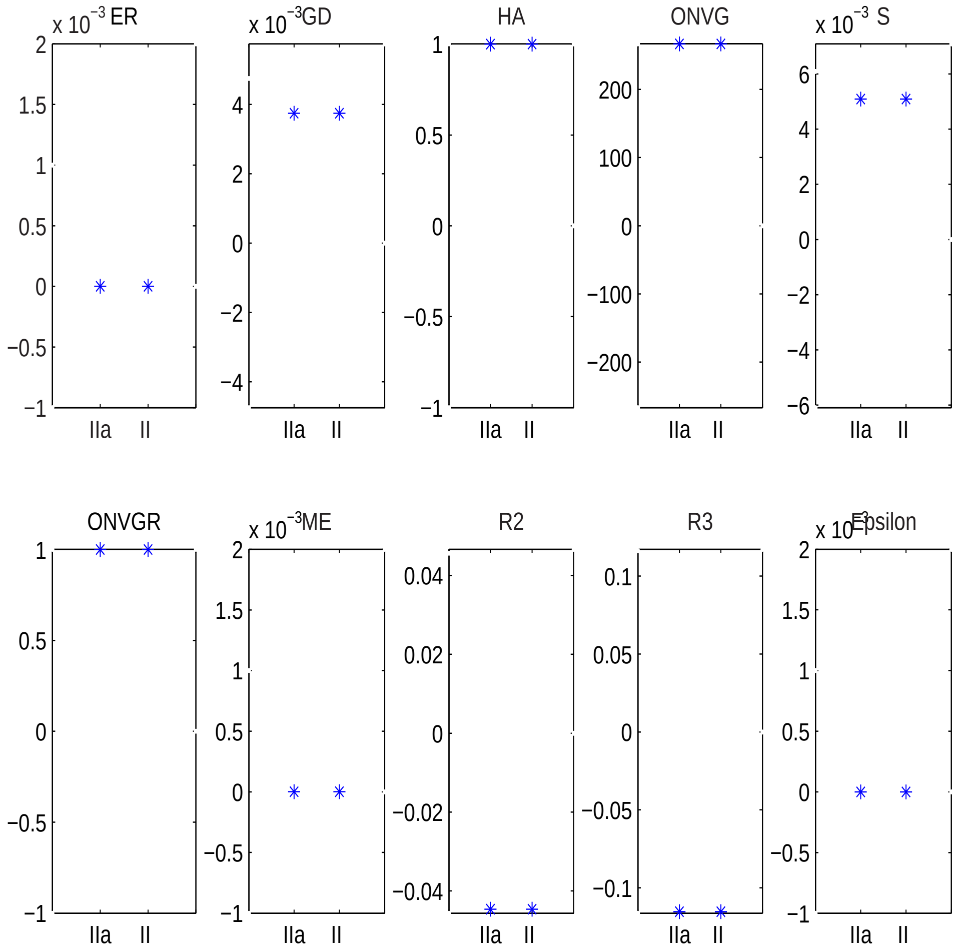


Figure 79: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is VL6. Results indicate visually that these MOEAs perform similarly over all metrics indicating that these two algorithms (after 30 experiments) on average have the same effectiveness on MOP 6.

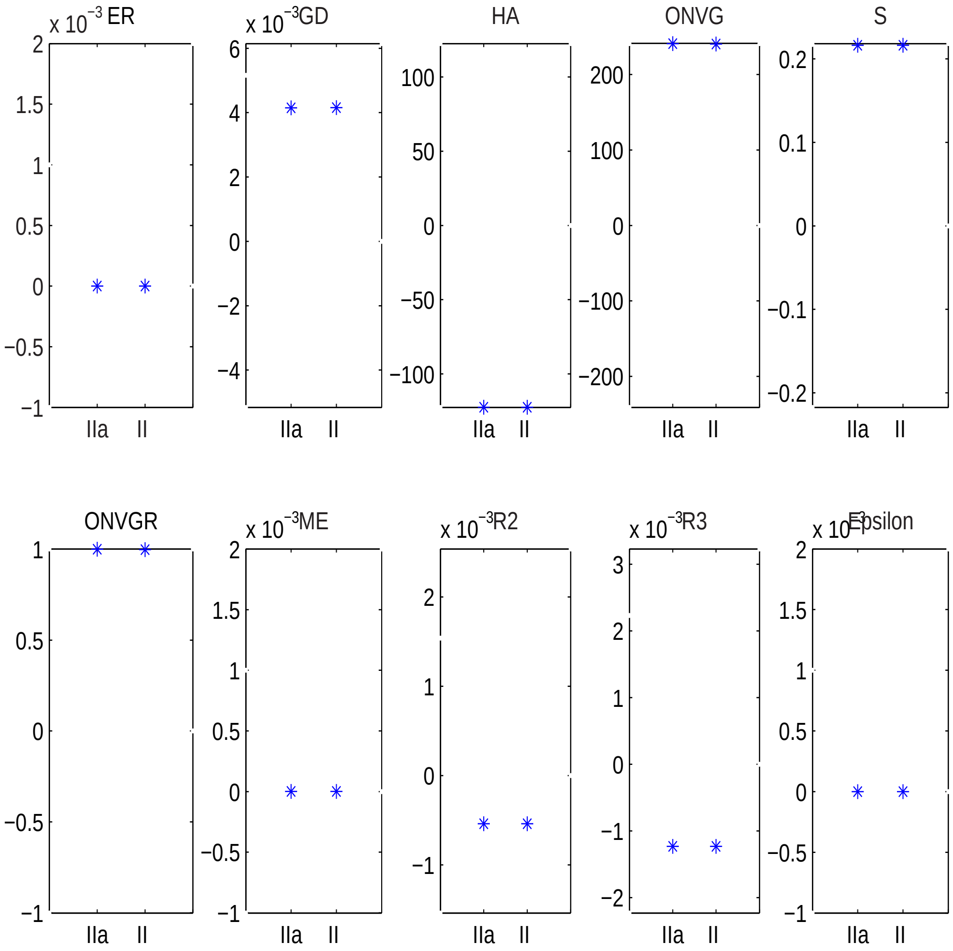


Figure 80: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is DTLZ3. Results indicate visually that these MOEAs perform similarly over all metrics indicating that these two algorithms (after 30 experiments) on average have the same effectiveness on DTLZ3. Not that the HA really indicates a hypervolume (HV) calculated by the performance package provided by Knowles and Ziztler (www.tik.ee.ethz.ch/pisa/). In addition R2, R3 and ϵ indicators are all measured using 3 dimensions.

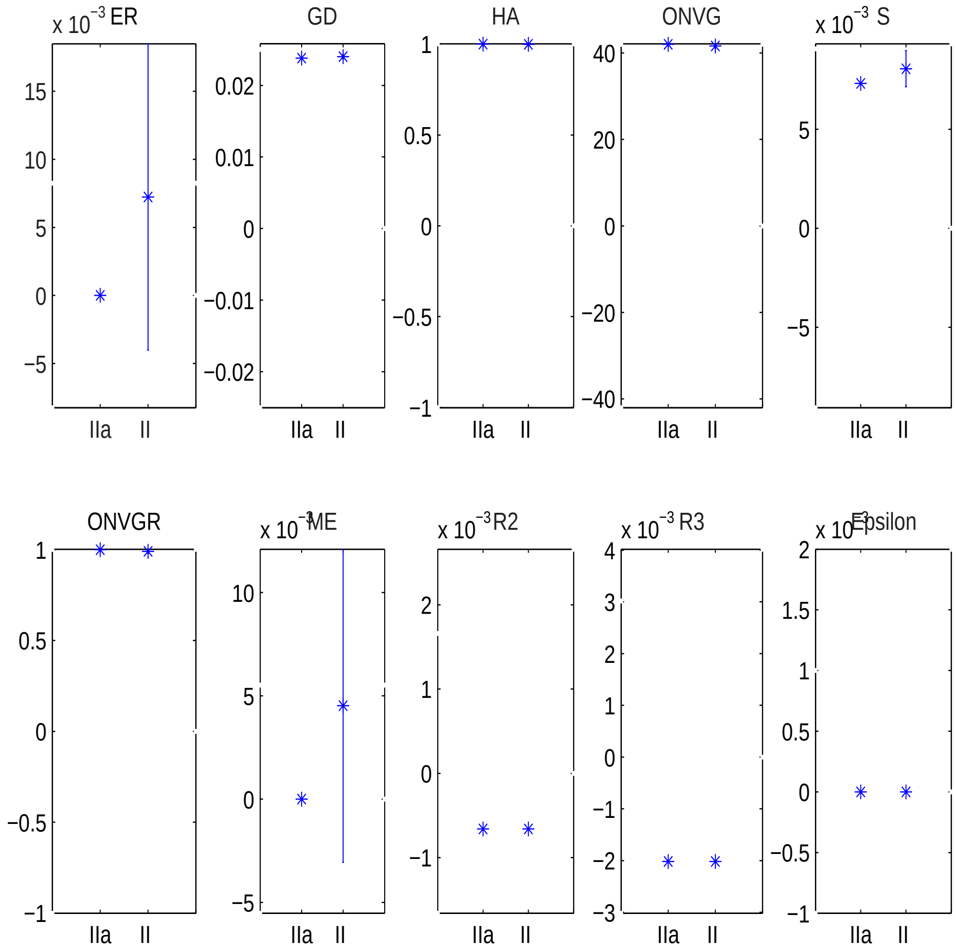


Figure 81: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is Tanaka. Results indicate visually that these MOEAs perform similarly over all metrics indicating that these two algorithms (after 30 experiments) on average have the same effectiveness on Tanaka.

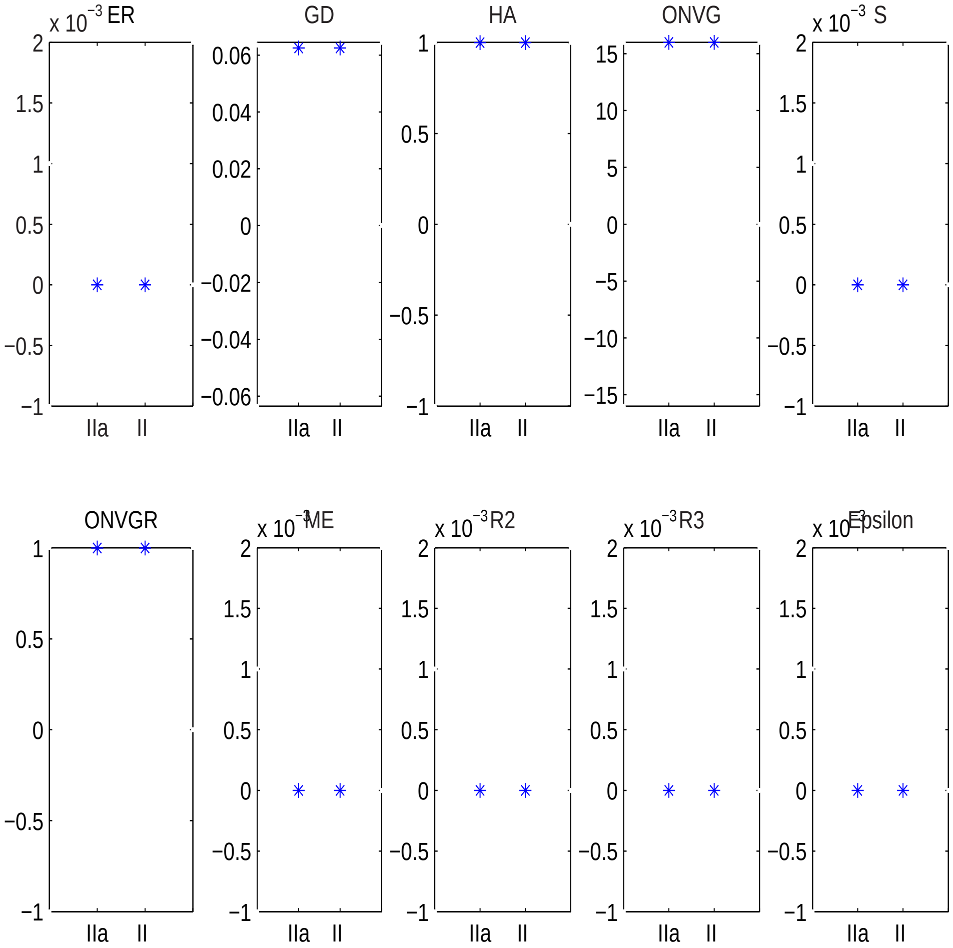


Figure 82: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T1T2:30. Results indicate visually that these MOEAs perform similarly over all metrics indicating that these two algorithms (after 30 experiments) on average have the same effectiveness on T1T2:30.

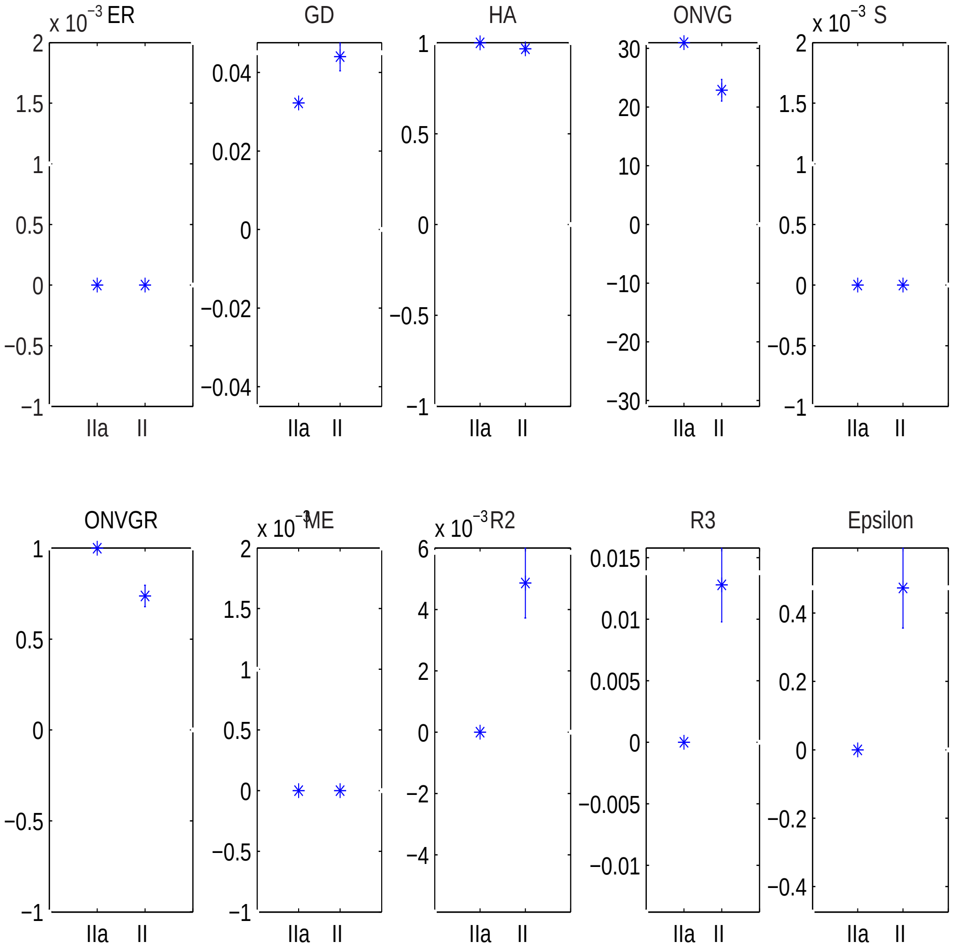


Figure 83: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T1T2:60. Results indicate visually that these MOEAs perform similarly over all metrics except HA, ONVG, ONVGR, and ϵ where the MOMGA-IIa found more PR$_{true}$ vectors than MOMGA-II.

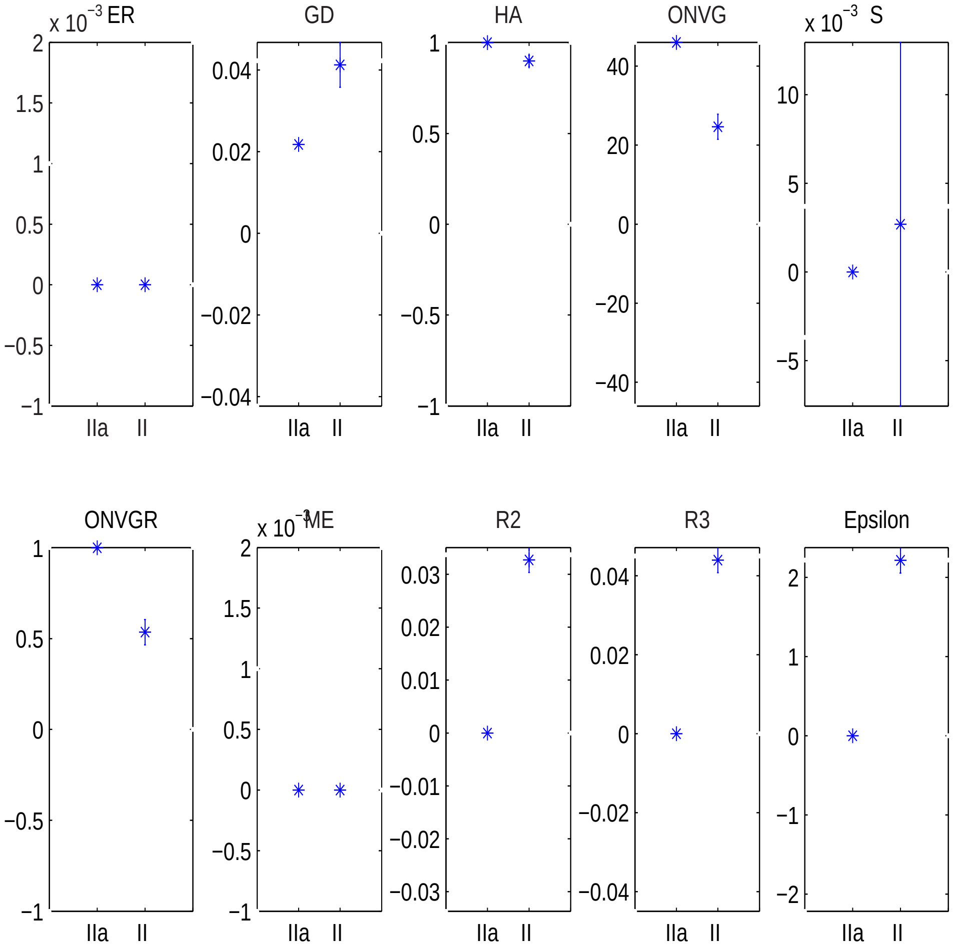


Figure 84: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T1T2:90. Results indicate visually that these MOEAs perform similarly over all metrics except GD, HA, ONVG, ONVGR, R2, R3, and a large ϵ. Differences occur in the metrics because the MOMGA-IIa found more PR$_{true}$ vectors than MOMGA-II and is much closer to the Pareto front than MOMGA-II. It is apparent that the MOMGA-II has a problem solving large stringed MOPs. This is an example where the MOMGA-II could not find on average even half the PF$_{true}$ vectors.

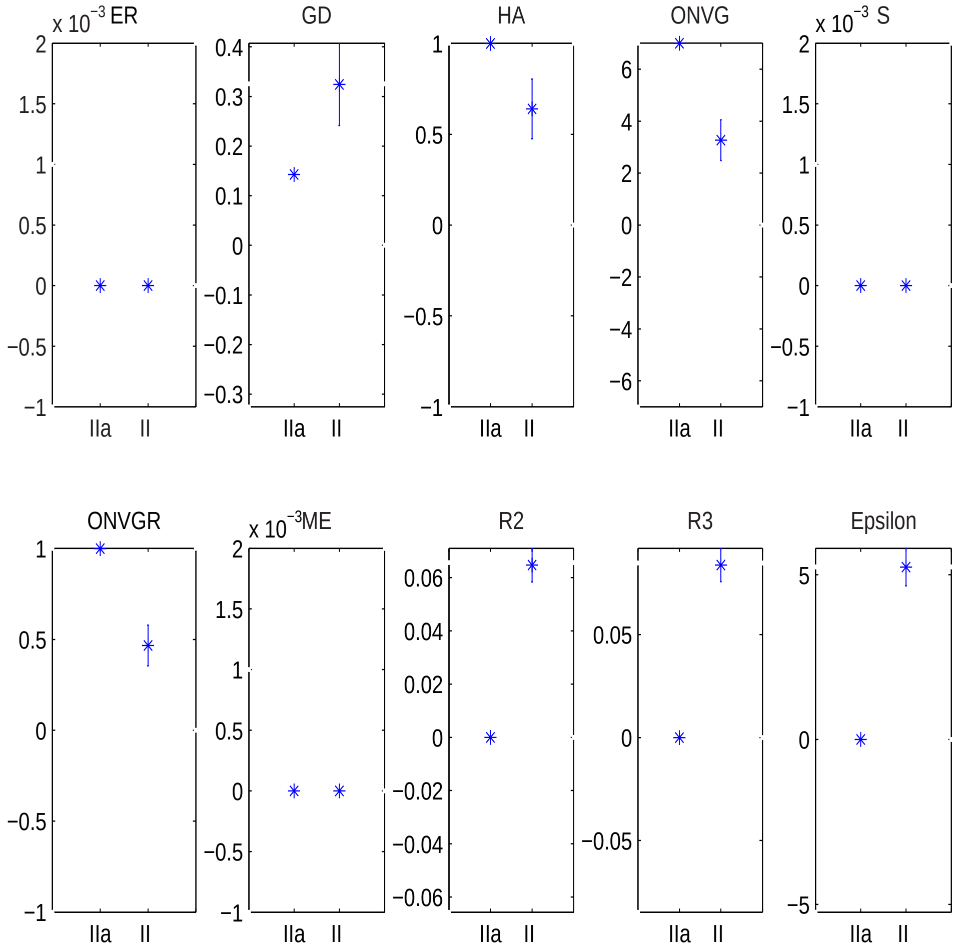


Figure 85: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T3T4:30. Results indicate visually that these MOEAs perform similarly over all metrics except GD, HA, ONVG, ONVGR, R2, R3 and ϵ. Differences occur in the metrics because the MOMGA-IIa found more PR$_{true}$ vectors than MOMGA-II and is much closer to the Pareto front than MOMGA-II. Being that T3T4 is a particularly nasty deception problem, it is apparent that the MOMGA-II has a problem solving small and large deception MOPs of this type. Once again, this is an example where the MOMGA-II could not find on average even half the PF$_{true}$ vectors.

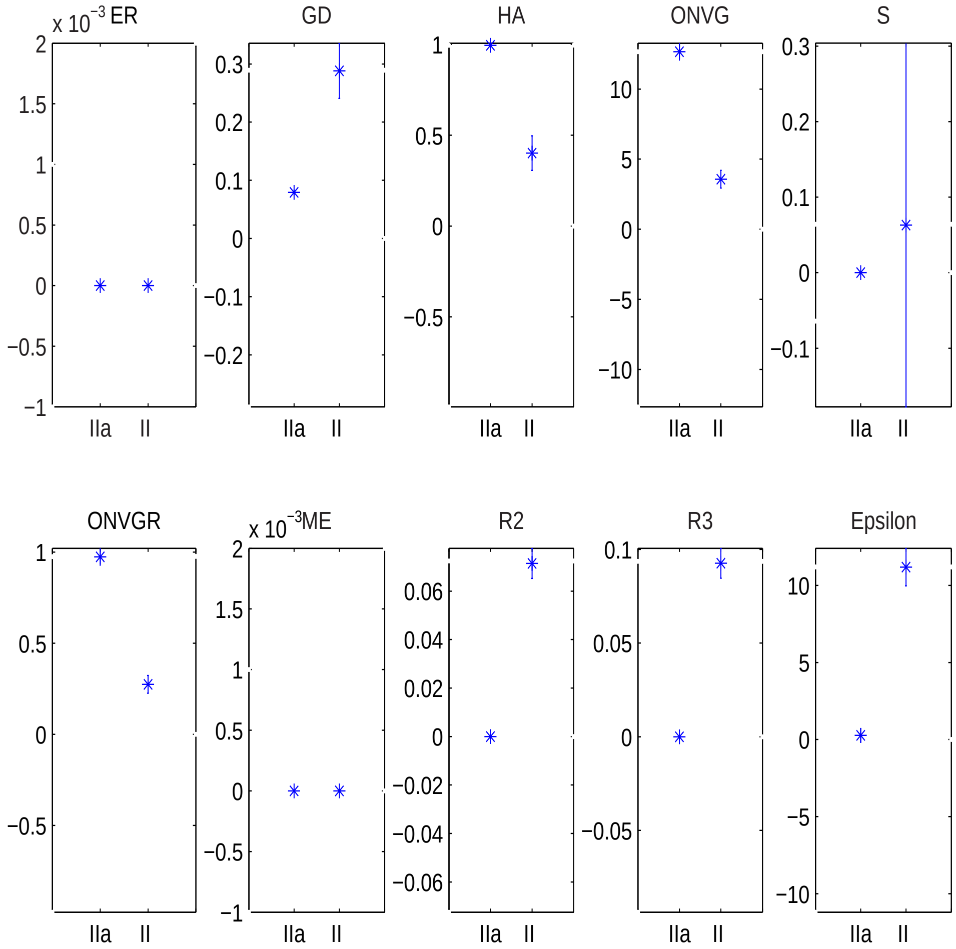


Figure 86: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T3T4:60. Results indicate visually that these MOEAs perform similarly over all metrics except GD, HA, ONVG, ONVGR, R2, R3 and ϵ. Differences occur in the metrics because the MOMGA-IIa found more PR$_{true}$ vectors than MOMGA-II and is much closer to the Pareto front than MOMGA-II. Being that T3T4 is a particularly nasty deception problem, it is apparent that the MOMGA-II has a problem solving small and large deception MOPs of this type. Once again, this is an example where the MOMGA-II could not find on average even half the PF$_{true}$ vectors.

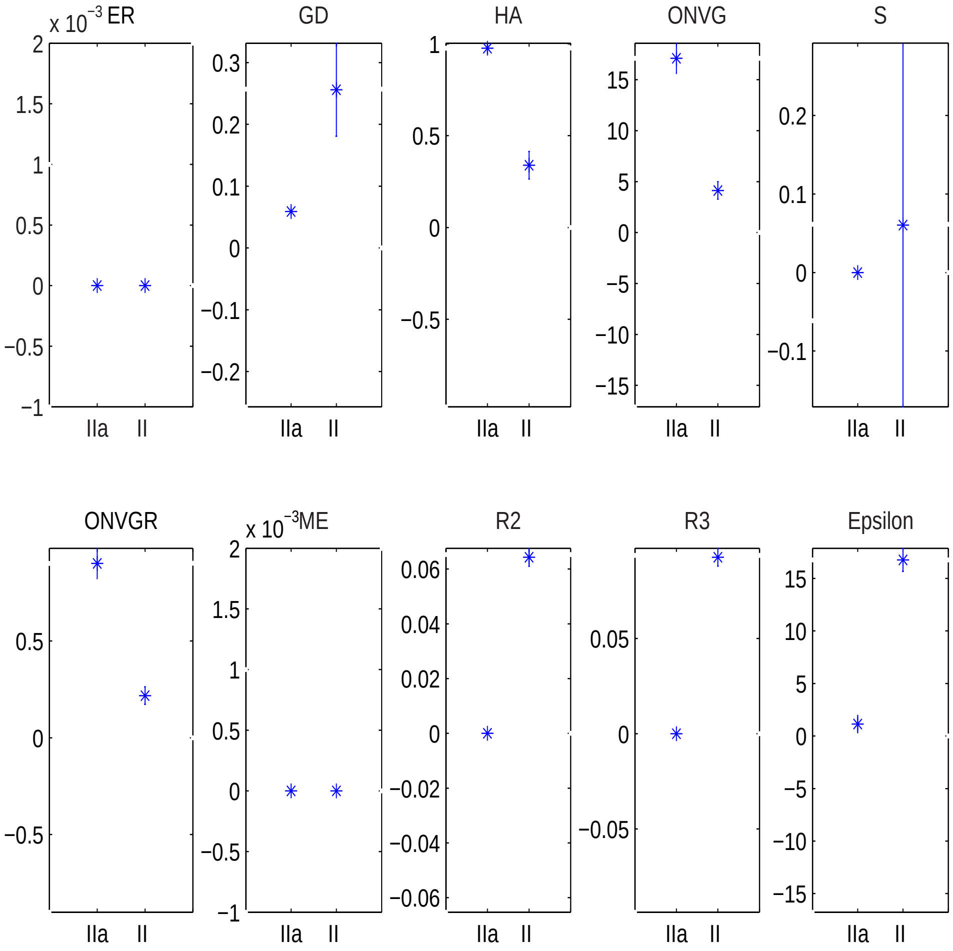


Figure 87: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is T3T4:90. Results indicate visually that these MOEAs perform similarly over all metrics except GD, HA, ONVG, ONVGR, R2, R3 and ϵ. Differences occur in the metrics because the MOMGA-IIa found more PR_{true} vectors than MOMGA-II and is much closer to the Pareto front than MOMGA-II. Being that T3T4 is a particularly nasty deception problem, it is apparent that the MOMGA-II has a problem solving small and large deception MOPs of this type. Once again, this is an example where the MOMGA-II could not find on average even half the PF_{true} vectors.

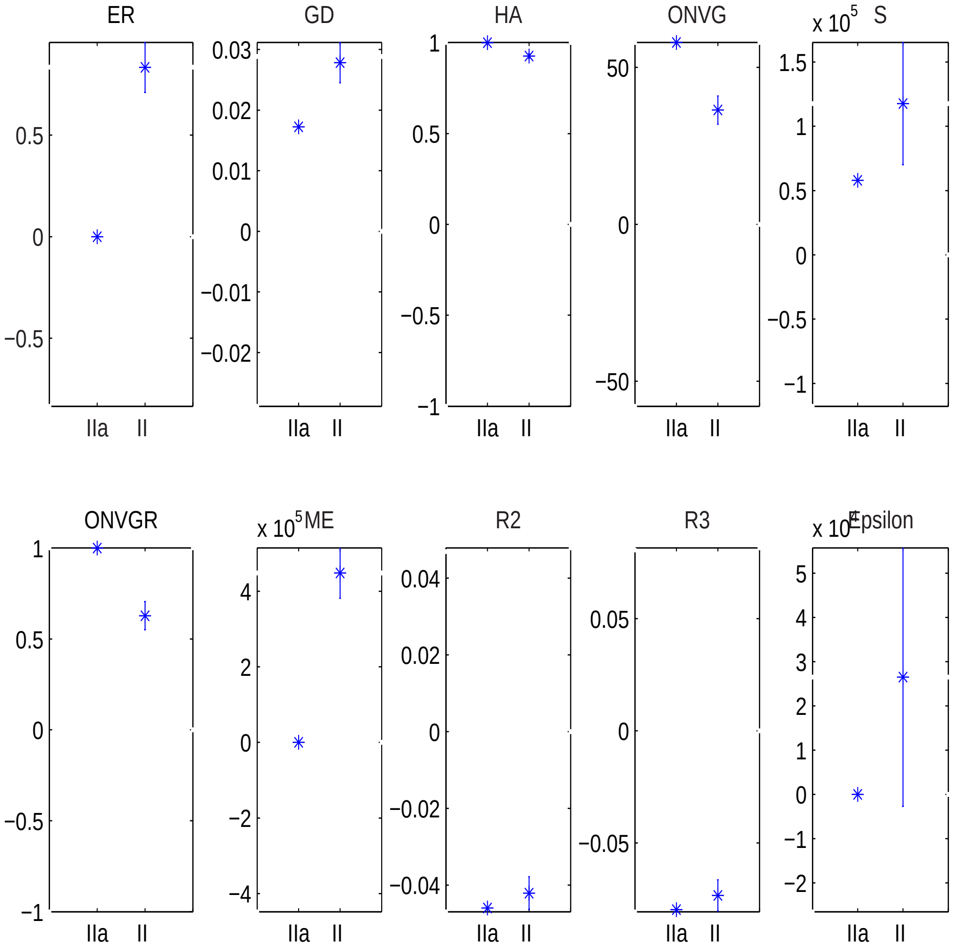


Figure 88: Statistical results of MOMGA-II vs. MOMGA-IIa over the 10 metrics used. The mean and standard deviation is generated using 30 experimental runs. The MOP under test is mQAP. Results indicate visually that these MOEAs perform similarly over all metrics except for metrics ER, HA, ONVG, S, ONVGR, ME, and ϵ. Differences occur in the metrics because the MOMGA-IIa found all PF$_{true}$ vectors and MOMGA-II did not. This is another illustration of the MOMGA-II not scaling well.

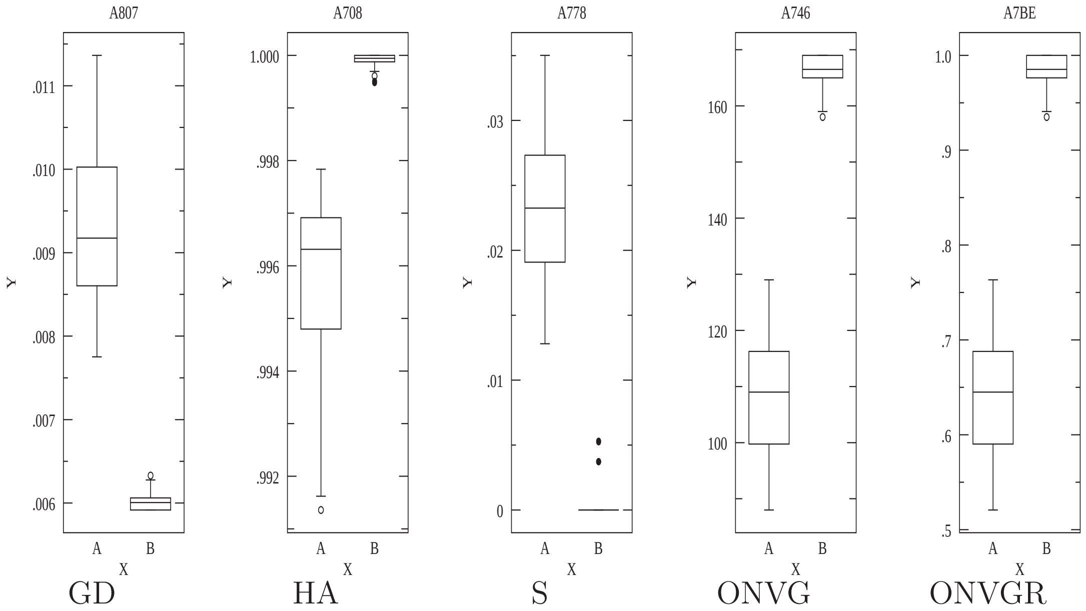


Figure 89: Illustrated are the statistical (student t-test) box plot results of the MOMGA-II vs. MOMGA-IIa over the five metrics showing a difference from the Kruskal Wallis testing. Results are generated using data collected over 30 experimental runs and student-t/box plot software found at `http://www.physics.csbsju.edu/`. The MOP under test is VL1. Results indicate visually that these MOEAs perform differently over all five metrics. Differences occur in the metrics because the MOMGA-IIa found all PF$_{true}$ vectors and MOMGA-II did not. This is another illustration of the MOMGA-II not scaling well.

Appendix C. Implicit vs. Explicit BBB results on Test Suites

This appendix is provided to present statistical results for implicit BB MOEAs (MOGA, NSGA, NPGA) and explicit BB MOEAs (MOMGA, MOMGA-II, MOMGA-IIa) when solving the test suite MOPs (VL) 1, 2, 3, 4, and 6.

C.1 Implicit vs. Explicit BBB results

Figures 90, 91, 92, 93, and 94 illustrate data collected by Dr David Van Veldhuizen [222], Dr Jesse Zydallis [244], and Richard Day to illustrate the differences between implicit and explicit MOEAs when solving test suite MOPs. In these figures, data is collected over 10 experiments suggesting a normal distribution. This is in contrast to the 30 experimental runs collected when comparing the MOMGA-II and MOMGA-IIa in the testing chapter of this dissertation.

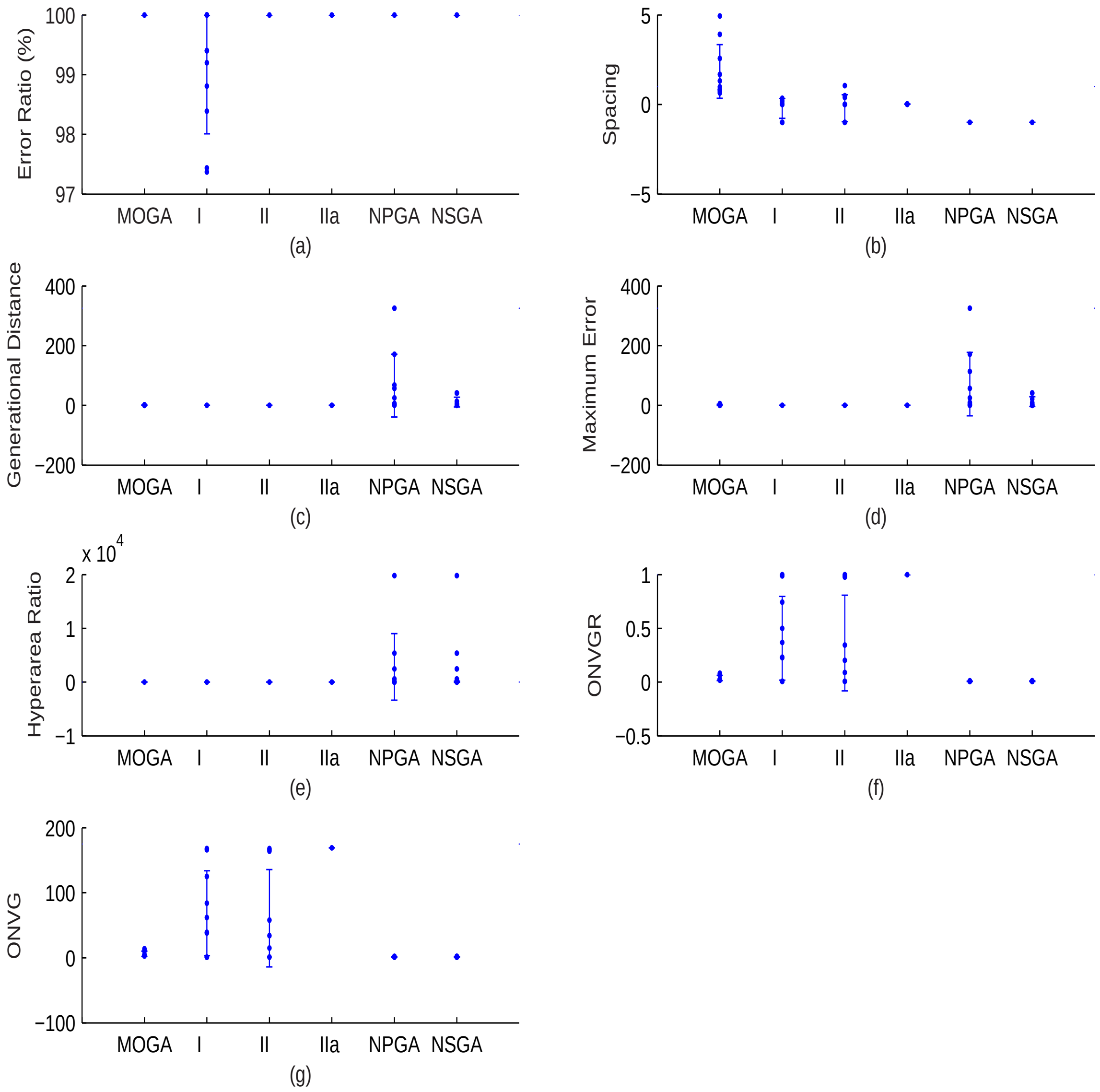


Figure 90: This figure illustrates results from MOGA, NSGA, NPGA, MOMGA (I), MOMGA-II (II) and MOMGA-IIa (IIa) when solving VL1. Experimental data is taken over 10 runs and error bars illustrate 1 standard deviation away from the mean. This graphical illustration is given to connect three generations of dissertations together to illustrate the progression of the MOMGA series MOEA.

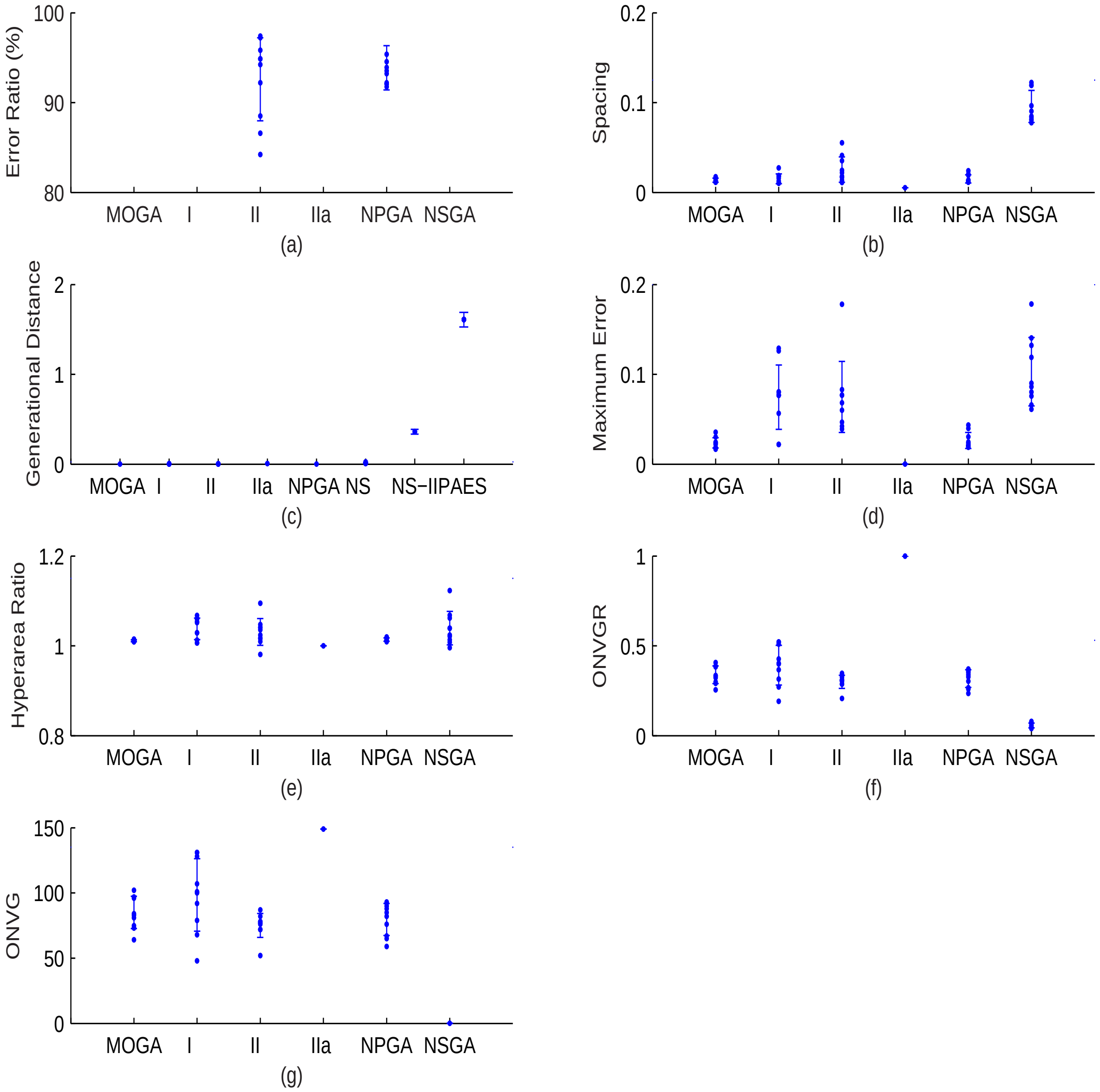


Figure 91: This figure illustrates results from MOGA, NSGA, NPGA, MOMGA (I), MOMGA-II (II) and MOMGA-IIa (IIa) when solving VL2. Experimental data is taken over 10 runs and error bars illustrate 1 standard deviation away from the mean. This graphical illustration is given to connect three generations of dissertations together to illustrate the progression of the MOMGA series MOEA.

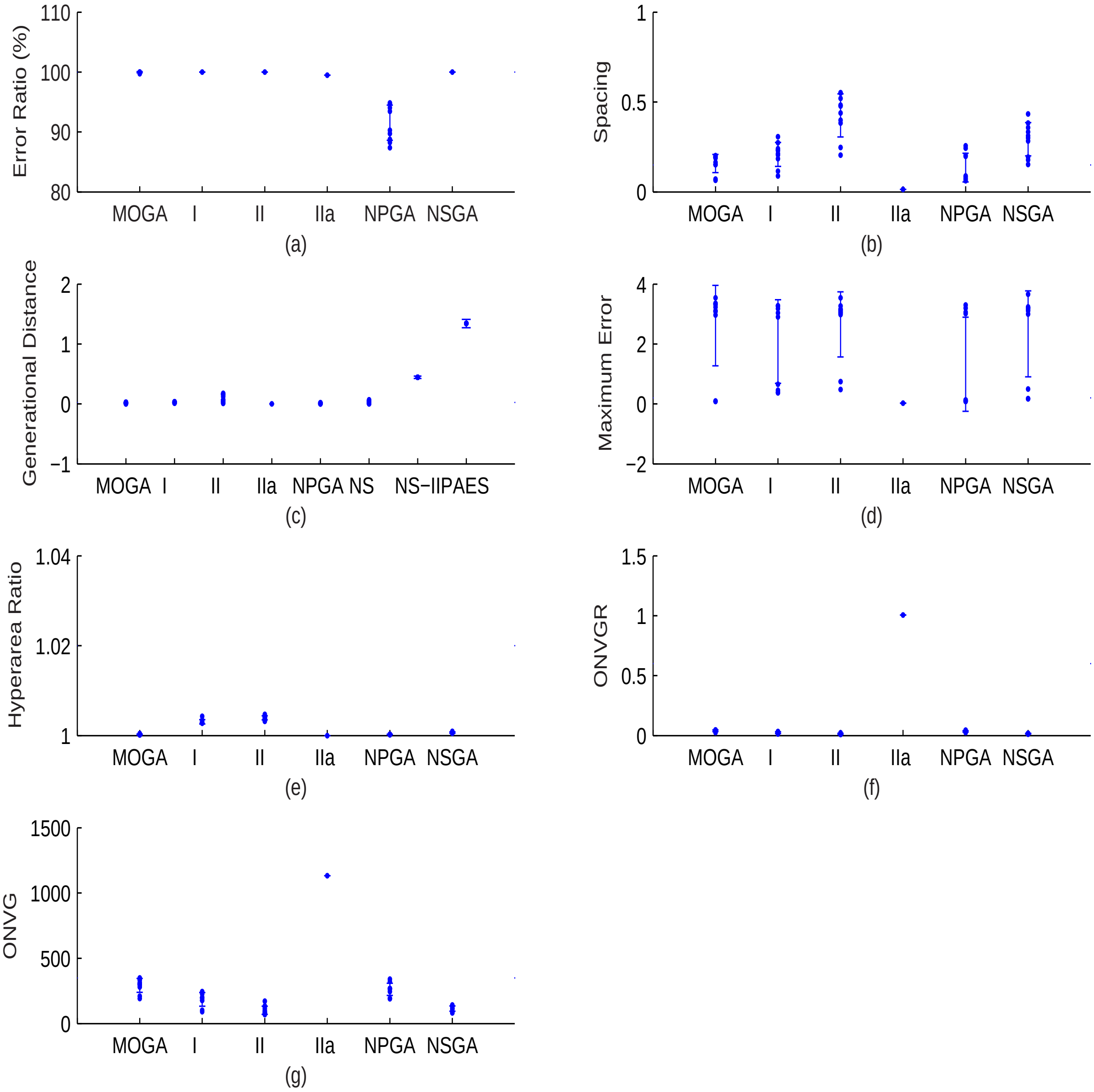


Figure 92: This figure illustrates results from MOGA, NSGA, NPGA, MOMGA (I), MOMGA-II (II) and MOMGA-IIa (IIa) when solving VL3. Experimental data is taken over 10 runs and error bars illustrate 1 standard deviation away from the mean. This graphical illustration is given to connect three generations of dissertations together to illustrate the progression of the MOMGA series MOEA.

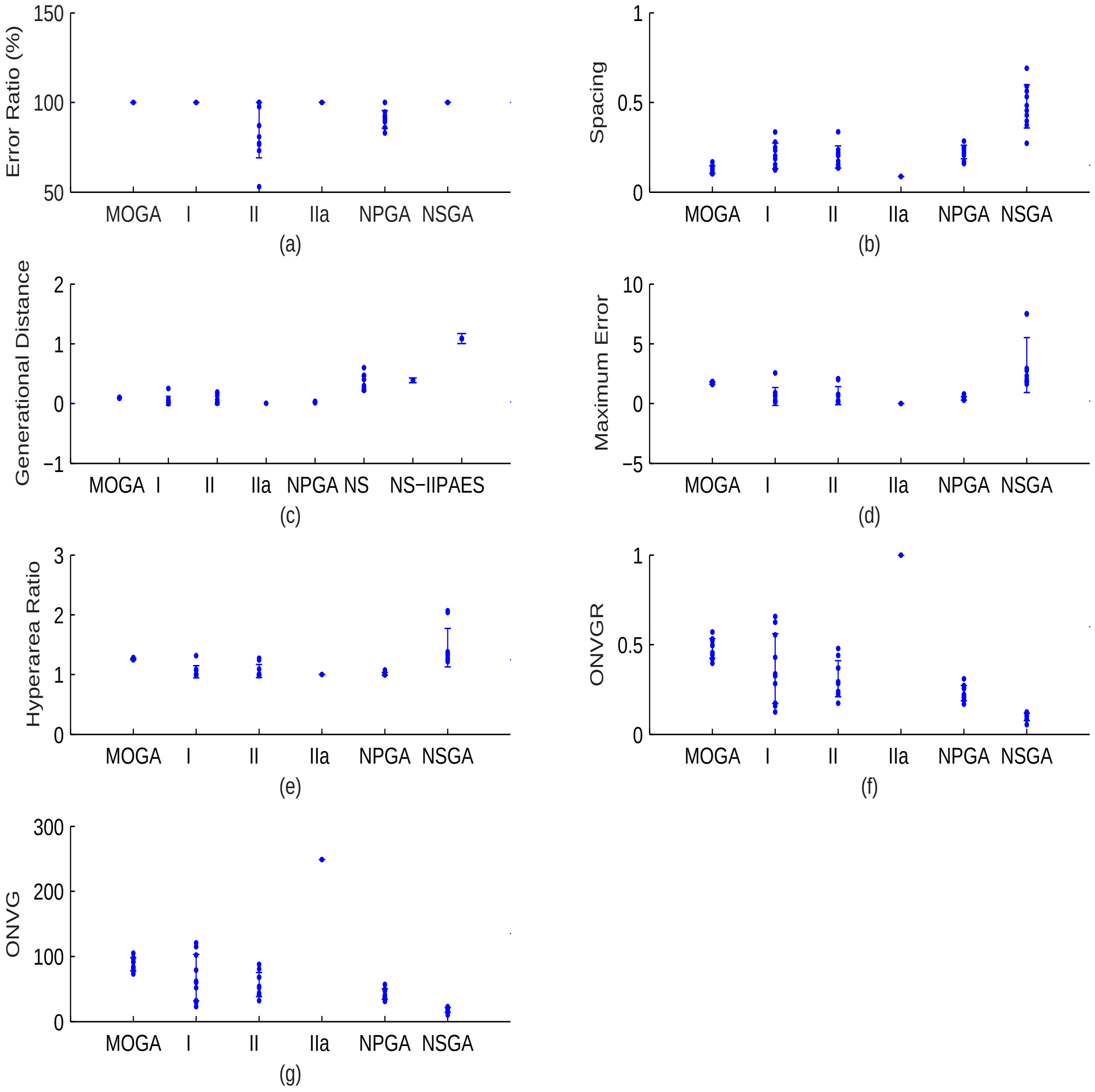


Figure 93: This figure illustrates results from MOGA, NSGA, NPGA, MOMGA (I), MOMGA-II (II) and MOMGA-IIa (IIa) when solving VL4. Experimental data is taken over 10 runs and error bars illustrate 1 standard deviation away from the mean. This graphical illustration is given to connect three generations of dissertations together to illustrate the progression of the MOMGA series MOEA.

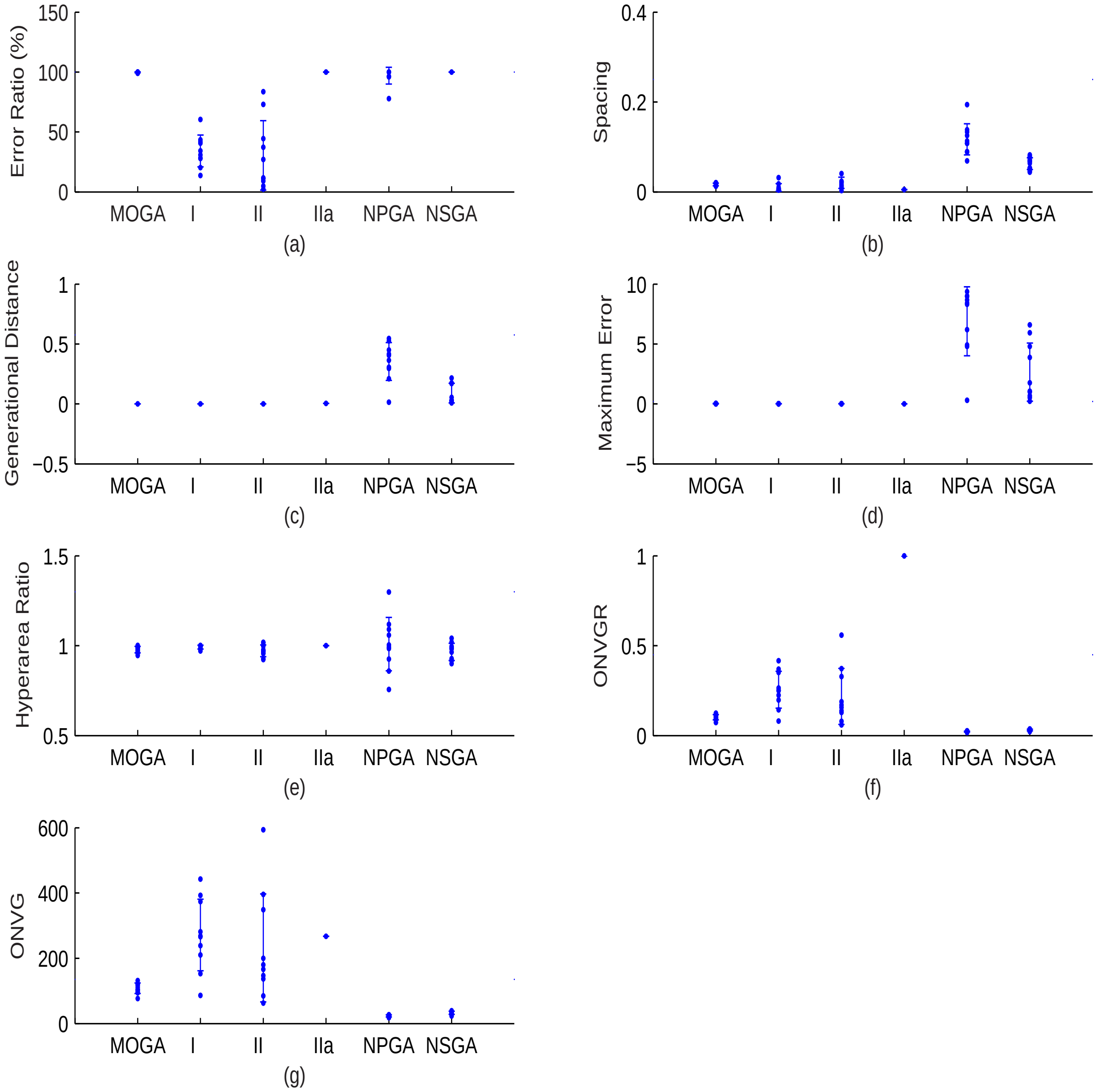


Figure 94: This figure illustrates results fromMOGA, NSGA, NPGA, MOMGA (I), MOMGA-II (II) and MOMGA-IIa (IIa) when solving VL6. Experimental data is taken over 10 runs and error bars illustrate 1 standard deviation away from the mean. This graphical illustration is given to connect three generations of dissertations together to illustrate the progression of the MOMGA series MOEA.

Appendix D. Explicit BBB Epistasis Results

This appendix is provided to present epistasis and schemata size associated with each BB found along the Pareto front. For each illustration, provided is the genotype, phenotype, epistasis and schemata size associated with Pareto front vectors found using the MOMGA-IIa. Results are illustrated for MOP 1, 2, 3, 4, 6, Tanaka, DTLZ3, T1T2:60, and T3T4:60.

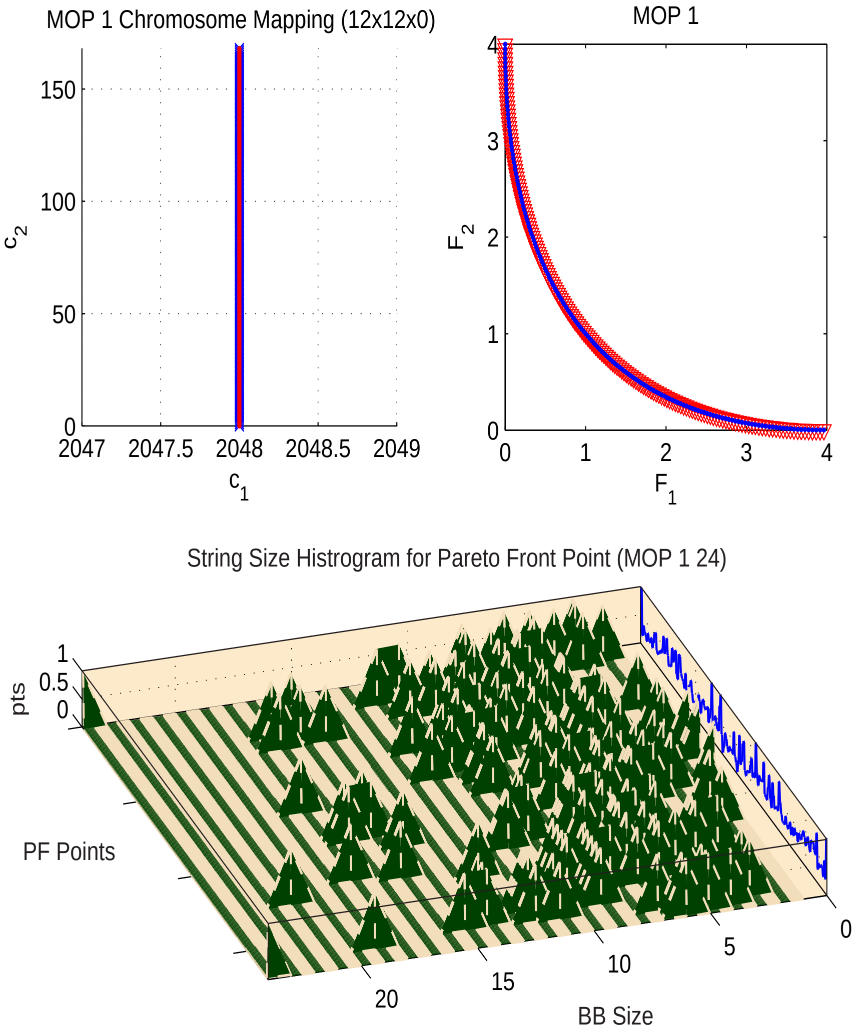


Figure 95: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the MOP 1 (VL1).

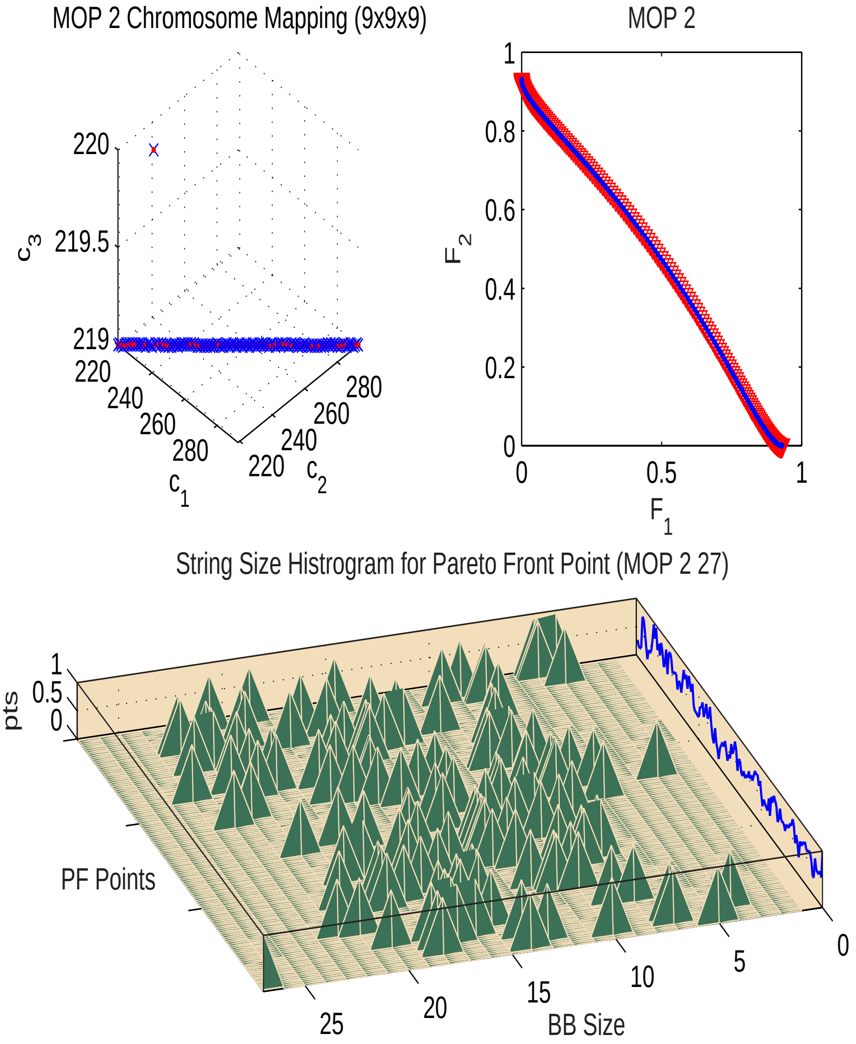


Figure 96: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the MOP 2 (VL2).

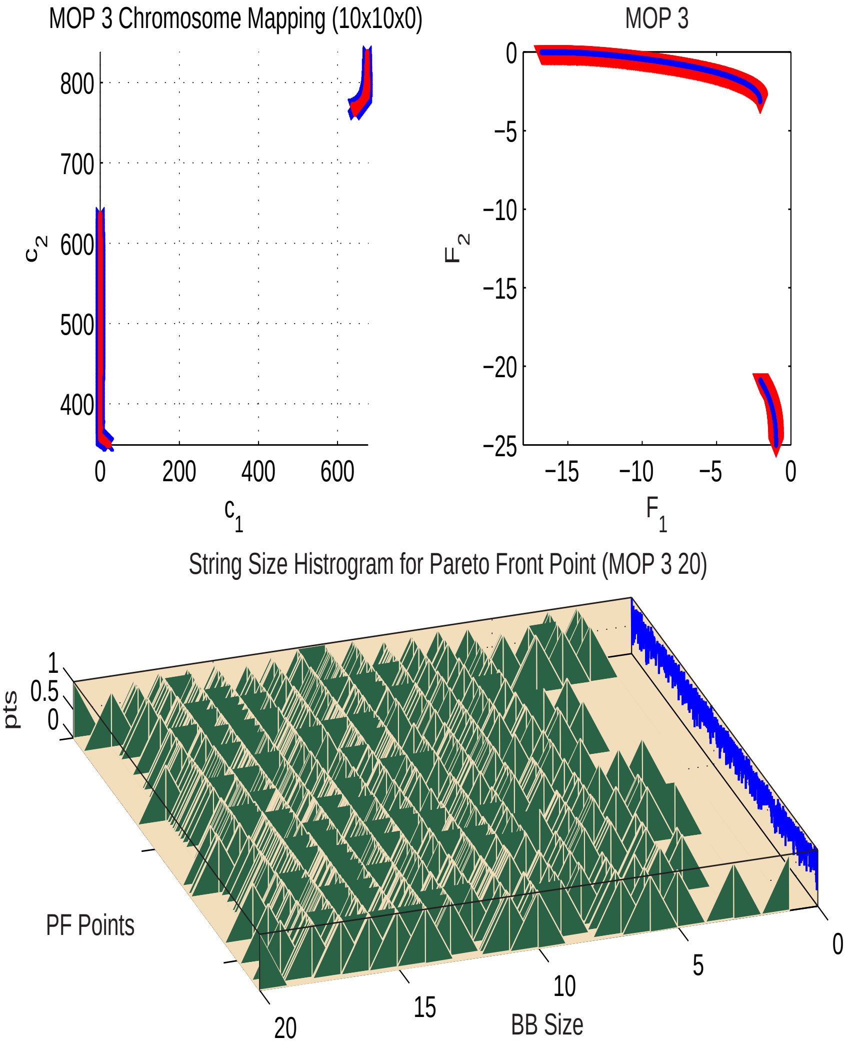


Figure 97: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the MOP 3 (VL3).

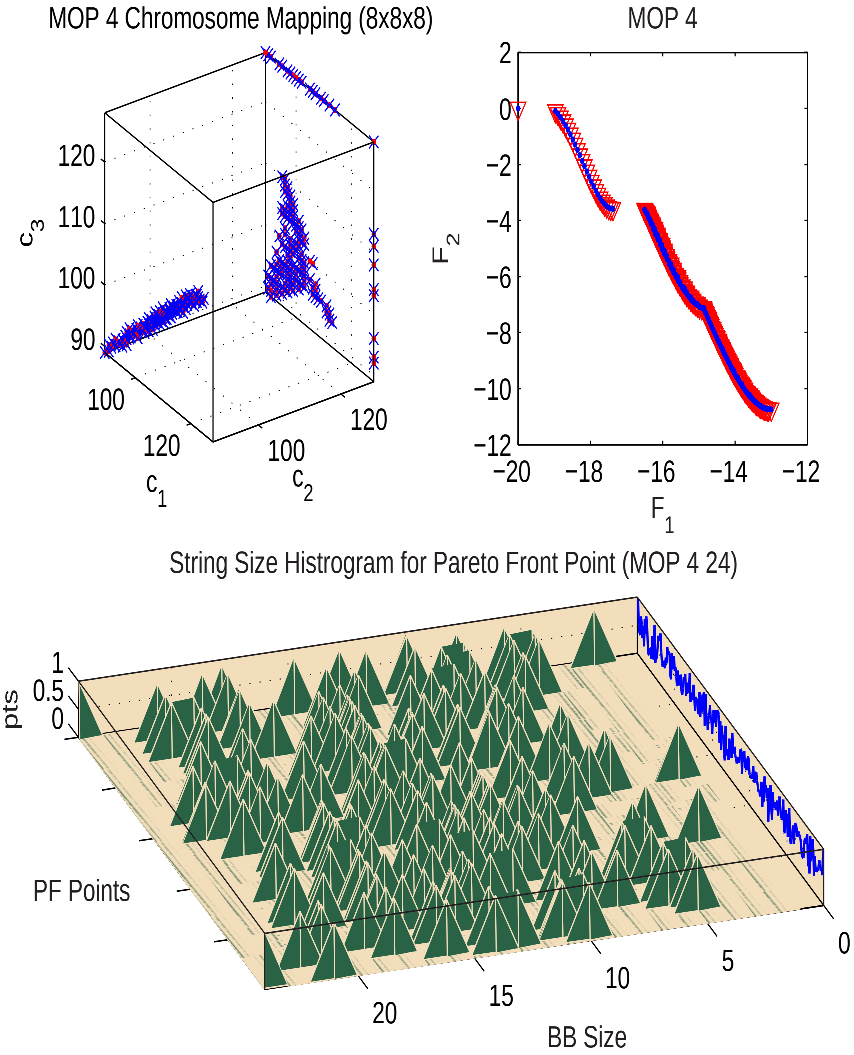


Figure 98: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the MOP 4 (VL4).

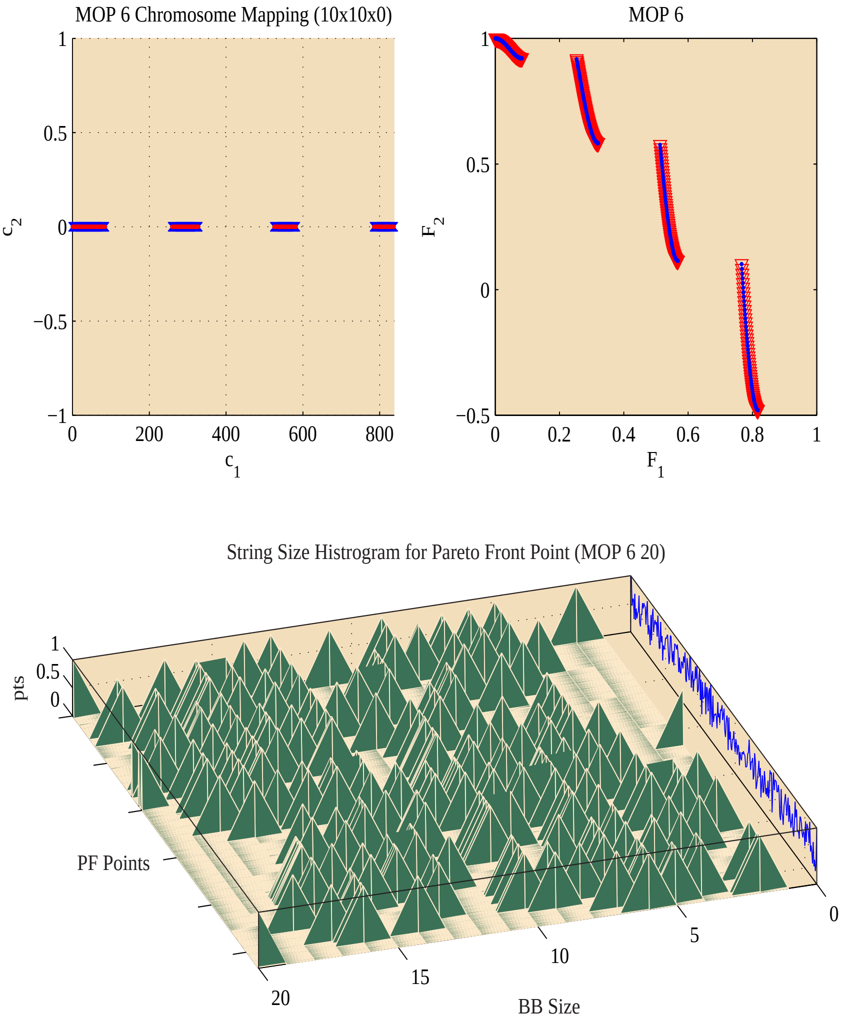


Figure 99: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the MOP 6 (VL6).

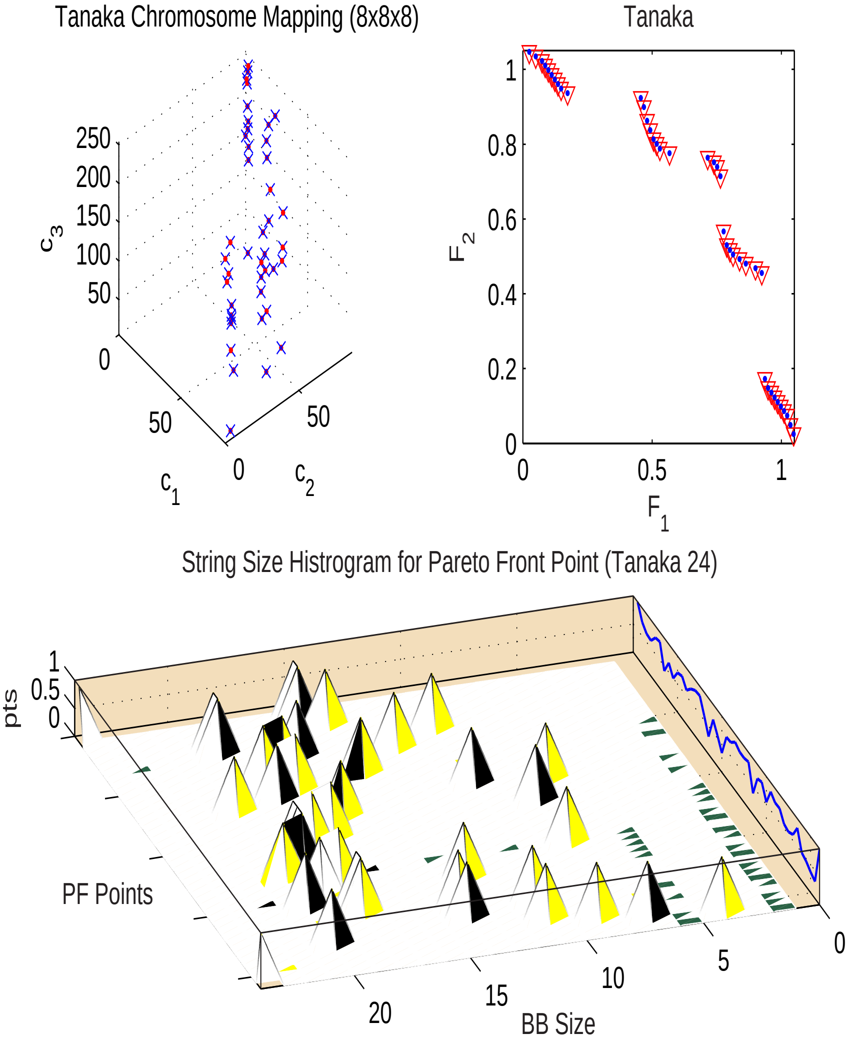


Figure 100: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the Tanaka (VL-C4).

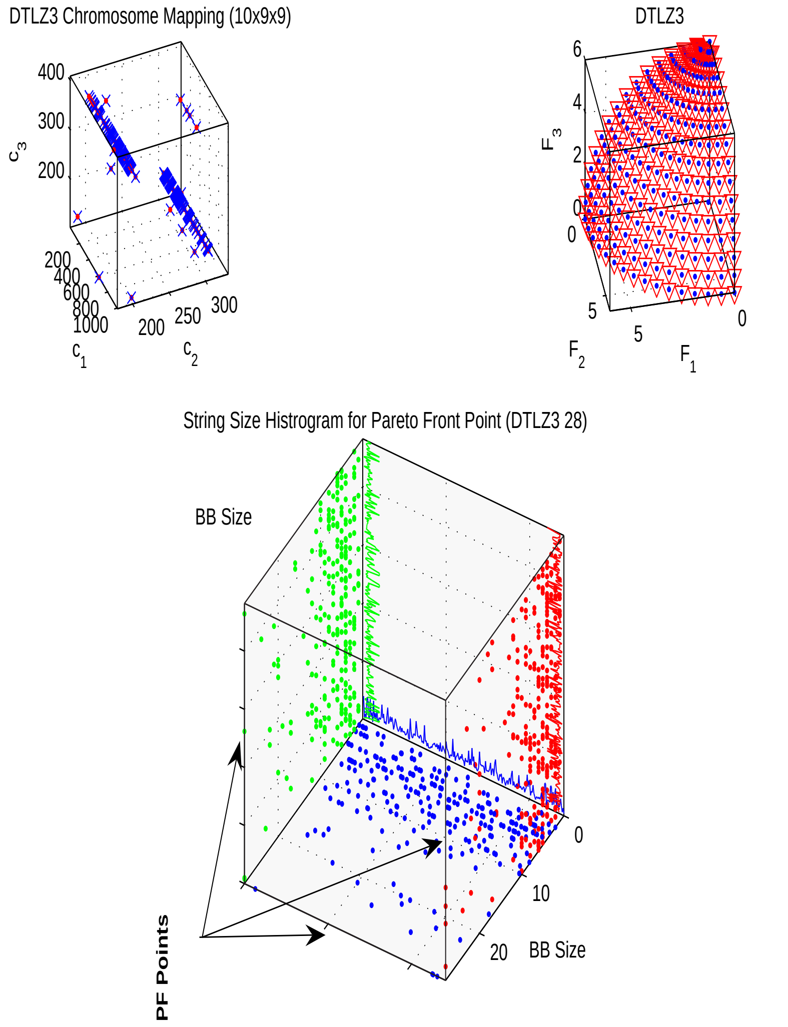


Figure 101: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the DTLZ3.

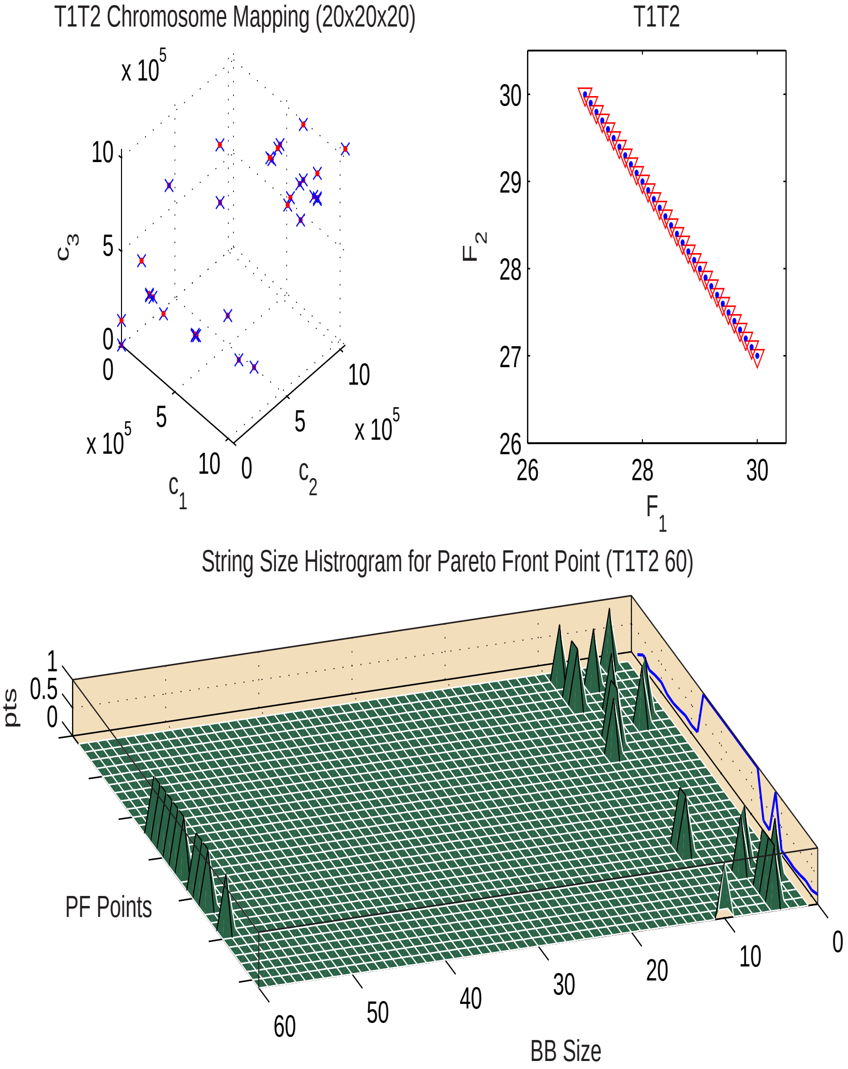


Figure 102: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the deception problem T1T2 of size 60 (T1T2:60).

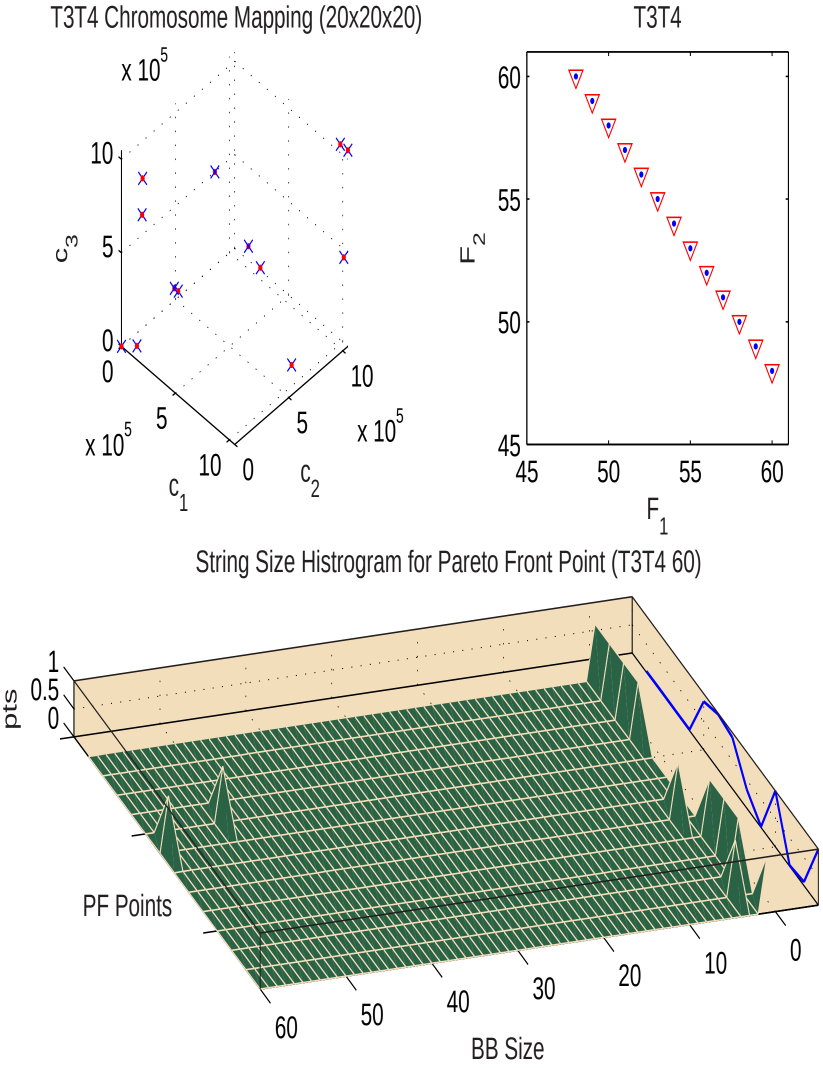


Figure 103: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the deception problem T3T4 of size 60 (T3T4:60).

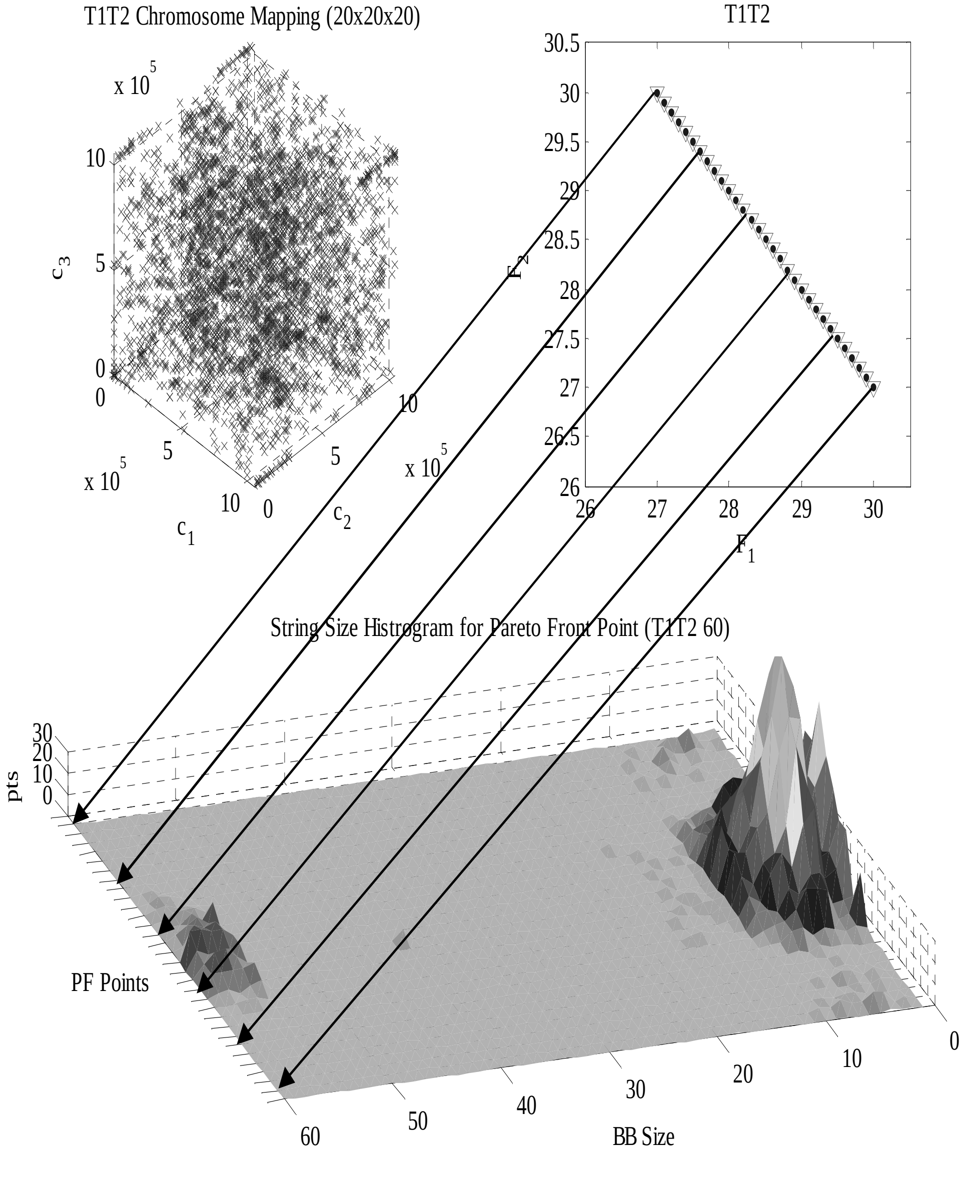


Figure 104: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the T1T2 60. Additionally, lines are draw to illustrate how the PF vectors are partially ordered on the lower BB size plot.

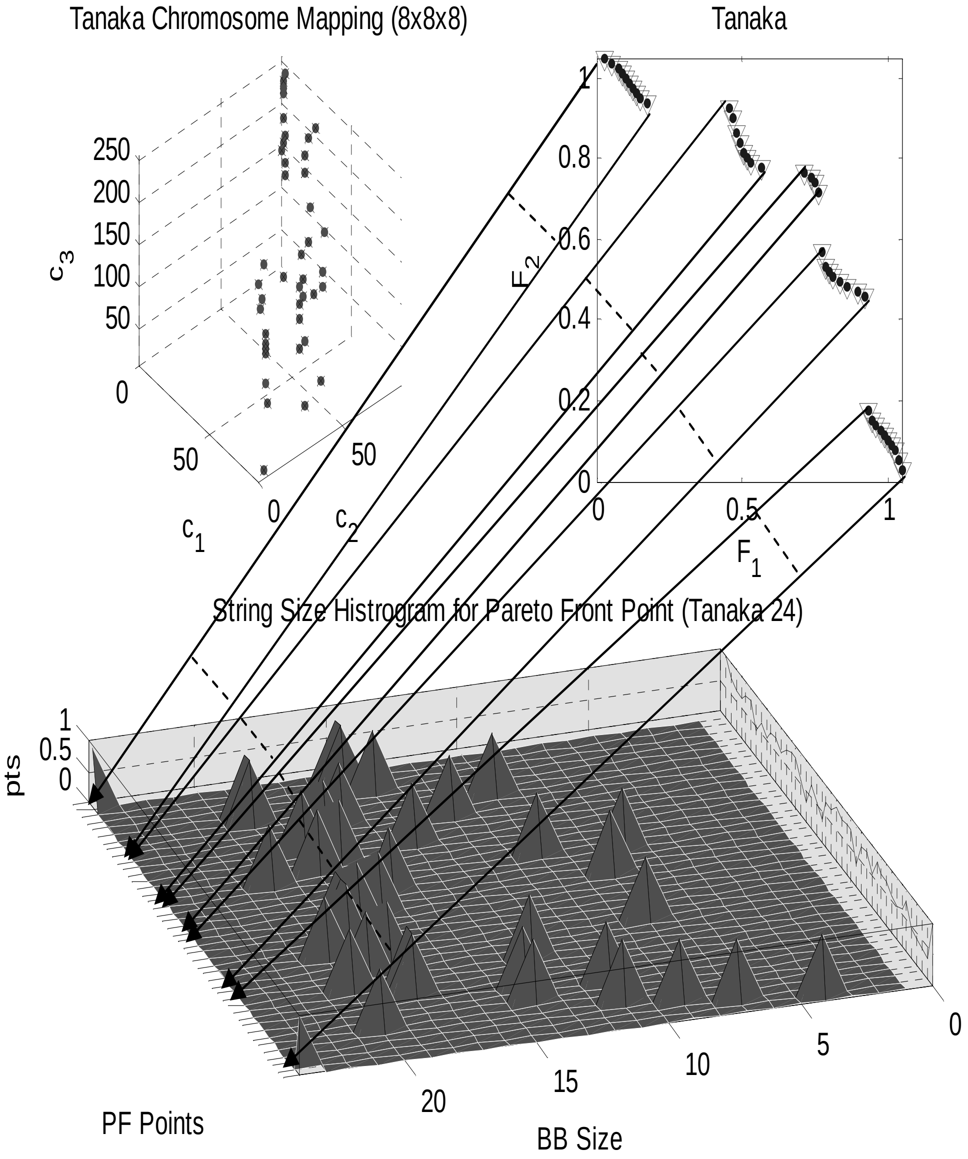


Figure 105: This figure illustrates the MOMGA-IIa tracing of BB sizes and epistasis when solving the Tanaka. Additionally, lines are draw to illustrate how the PF vectors are partially ordered on the lower BB size plot.

Appendix E. Optimization Techniques

There are many optimization techniques available to researchers, and no single technique is better at solving every problem [230]; however, some are said to be more robust, efficient or effective on particular problems. It is not the purpose of this appendix to include every single optimization technique, but to give a brief overview of a few optimization techniques and then go into a thorough history of implicit and explicit BB single and multi objective Evolutionary Algorithms (MOEAs). In addition, since the focus of this research is BBBs, comparisons between implicit and explicit search structures and operators are given.

The layout of this appendix is as follows: a short discussion of some optimal resource allocation techniques used by the Operational Research community[1]. Mathematical programming, linear programming, Monte Carlo Analysis and Markov Models topics are touched upon. These techniques are useful but have limitations when solving multiobjective problems. This is the justification for using evolutionary algorithms in their stead. Discussed next are single objective explicit BB evolutionary algorithms. This background is required for the extensive discussion of some multiobjective evolutionary algorithms (MOEAs).

E.1 Mathematical Programming

Mathematical programming has its roots from planning or scheduling of activities within a large organization [21]. Programmers found that they could represent the level of activities within the organizations as variables - which are to be solved for later. The constraints in the scheduling or planning are described mathematically with a set of equations or inequalities having the variables adjustable. Thus a solution meeting all the constraints is considered an acceptable plan.

[1] The operational research community is large and include groups concerned with optimizing systems. Included in this community is the Military Operations Research society.

In the beginning of using this method history showed that it was difficult to model a complex operation simply by specifying constraints. Plus, success of this method was dependent on the number of constraints to be inserted, too few constraints allowed for inferior solutions to satisfy them, and too many constraints would cause the method to rule out desirable solutions. Many times, the success of the programming depended on some key insight that provided a way around this complication.

As the mathematically programming field matured, an objective function was added to the mathematical constraint modeling. The objective function allowed for the programmer to label a particular solution with a cost or profit (merit). Solutions meeting the mathematical criteria were then ranked according to some objective function. Objective functions could either be a maximization or minimization function.

Today, there are many mathematical programming methods for solving multicriteria problems (see Table 82 on page 420); however, many of these generate Pareto optimal members, one at a time. In addition, most of the mathematical programming methods fail when the Pareto front is concave or facing a discontinuous front. [201] In addition, it is found that EAs efficiency increases with the problem size [142].

One particular example of an application of mathematical programming is linear programming. This is when all the costs, requirements and other interested quantities are formed strictly proportional to the levels of the activities, or sums of these terms. Often times this kind of mathematical programming is used within an evolutionary method. This is called a hybridize evolutionary method. Other examples of methods used to hybridize evolutionary methods are cutting-plane algorithms, branch-and-cut, branch-and-prize, and branch-and-cut-and-prize. It is important to understand that although these optimization techniques do not perform as well as an evolutionary heuristic, they can help in a manner where they are applied later in the evolutionary search.

E.2 Markov Chains

A Markov chain in the strict sense of the mathematics is a collection of random variables (where the index i runs through $\{0, 1, ...\}$) having the property that, given the present, the future is conditionally independent of the past [225]. There are many problem that can be modeled as a Markov Chains where the structure of the chain is dependent upon the parameter values chosen. These are called Parameter Dependent Markov Chain Optimization (PDMCO) problems and are normally solved by finding the optimal Markov Chain which essentially gives the optimal parameter values to solve the problem.

A Markov Chain can be thought of as a state diagram with probabilities assigned to transitions from one state to another. In addition, a chain my be adaptive or non-adaptive depending on the design. The fundamental property of Markov Chains are the following:

$$P\left\{Future/Present\ and\ Past\right\} = P\left\{Future/Present\right\}$$

Conditional probabilities $p_{i,j}$ assigned to transitions from one state to the next are called transitional probabilities. The matrix P_{matrix}=$[p_{i,j}]$ defines the probability of transitions from state $\breve{E}_i$ to state $\breve{E}_j$ - the following matrix is an example of P_{matrix}:

$$\begin{array}{cc|cccc|}
 & & \multicolumn{4}{c}{\text{(j)}} \\
 & & \breve{E}_1 & \breve{E}_2 & \cdots & \breve{E}_n \\
 & \breve{E}_1 & p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\
\text{(i)} & \breve{E}_2 & p_{2,1} & p_{2,1} & \cdots & p_{2,n} \\
 & \vdots & \vdots & \vdots & \cdots & \vdots \\
 & \breve{E}_n & p_{n,1} & p_{n,2} & \cdots & p_{n,n}
\end{array}$$

Transitions from state i to j that do not exist between states have a probability, $p_{i,j}$, of zero. An example of a Markov Chain used in solving the following equation $x = \mathbf{B}x + f$ can be found [25].

Markov chains, or Random Walks on Graphs are probably one of many important concepts in computer science. They appear in other fields of study as well, such as the following: statistical physics, biology, ecology, economy and the stock market, the study of the web, and have been useful in combinatorial applications such as approximation, *optimization* and counting algorithms. [1] One of the drawback to using Markov chains is in developing a probability model that represents the problem accurately; however, this optimization method is used for weather predication [171] and other applications.

E.3 Monte Carlo

A Monte Carlo analysis is sometimes called a Monte Carlo simulation. This simulation is named for Monte Carlo, Monaco. This is a place where the primary attractions are casinos. These casinos provide games of chance like roulette wheels, dice, and slot machines that exhibit the randomness found in a Monte Carlo simulation. Furthermore, the Monte Carlo simulation referred to in this investigation is one where values for independent variables are generated many times to simulate finding solutions randomly. This simulation is used in each and every experiment to show that the evolutionary technique under test is performing better than a random search. The general algorithm for a Monte Carlo simulation [101] is presented in Algorithm 10.

This algorithm is presented to provide background as to what it means to run a Monte Carlo simulation. The focus evolutionary algorithm developed within this research is better than a simple Monte Carlo simulation. Most, if not all, test functions evaluated in this study are compared against solutions found by the Monte Carlo simulation with the same number of function calls as the EA.

Algorithm 10 General algorithm for a Monte Carlo simulation.

```
procedure MARKOV-PROCESS(g (number of specified generations))
    Select a point[a] in phase space[b], x (initialization)
    for i = 1 to g do
        Create a new state from x, x' (Markov process[c])
        Compute the transition probability of state $x \to x'$ , $W(x, x')$
        Generate a (pseudo) random number, $\mathfrak{R}$, uniformly distributed in [0,1]
        if ($W \geq \mathfrak{R}$) then
            $x \leftarrow x'$
        end if
    end for
end procedure
```

[a] A point in the algorithm could be either a group of atoms' or a single atom's position

[b] Phase space could be selected subspace within the entire area (select few atoms) to optimize or it could be the entire space itself (all atoms)

[c] A first order Markovian process is when a random event or next event is dependant on only the most recent observation [204].

This ends the non-evolutionary optimization technique discussion. Next, the single objective Evolutionary Algorithm discussion beginning with Evolutionary Strategies.

E.4 Single Objective Evolutionary Algorithms (SOEA)

The EAs presented in this section are the foundation to most of the multiobjective evolutionary algorithms presented in Section E.5.

E.4.1 Evolutionary Strategies. Evolutionary Strategies (ES)s were developed in 1964 at the Technical University of Berlin (TUB). ES optimization is based on the hypothesis that during the biological evolution, generational replacement causes slight changes to better the populace. ES based algorithms imitate, in contrast to the genetic algorithms (GAs), the effects of genetic procedures on the phenotypic domain using *mutation* operators.

Furthermore, it is assumed that ES variable coding is accomplished with the idea that small changes in cause achieve small changes in the effect (i.e. epsilon changes over time results in epsilon improvements over time). The climax of ES theory is the discovery of the so-called *Evolution Window* where the evolutionary progress is said takes place only within a narrow band of mutation steps. These mutation steps are small in nature and self-adaptable within the algorithm to provide better results over time. ES notation is as follows:

$(1+1)$**-ES:** Two membered ES. A child is generated from its parent and then when comparing parents and child the two most fit survived to the next generation.

$(1+\check{\lambda})$**-ES:** Multimembered ES. Children are generated from parents. Selection for next generation of EVOPs is accomplished with parents and child.

$(1,\check{\lambda})$**-ES:** Multimembered ES. Children are generated from parents. Parents are discarded and selection for next generation of EVOPs is accomplished children.

$(\check{\mu}/\check{\rho},\check{\lambda})$**-ES:** Multimembered ES. All the parents have the same mating probabilities and as with the two-membered ES, the least fit member of the population including all the parents and the one offspring is eliminated in each generation.

where the $\check{\mu}$ is the total number of parents, $\check{\rho}$ marks the number of parents, which is recombined, and $\check{\lambda}$ stands for the number of offspring. Selection from among the offspring (comma notation) or among the offspring and parents together (plus notation) is represented. Nested ESs are written in the form $[\check{\mu}, \lambda(\mu, \check{\lambda})^{\check{\gamma}}]$-ES; where the outer brackets the $\check{\lambda}$ populations are judged according to their convergence speed, after each of these population has run $\check{\gamma}$-times through a $(\check{\mu}, \check{\lambda})$-selection scheme. $\check{\gamma}$ is called the isolation number. The nested ES forms the basis for the evolutionary self-adaptation of strategic parameters. Furthermore, a nested ES is qualified for multimodal and multiobjective optimization [11,12]. [187,197–200]

The pseudocode for an Evolutionary Strategy is given in Algorithm 11 on page 330. The main focus of this EA is the use of the mutation operator. Furthermore, any statistical model can be used to represent the allowable step size for each vari-

Algorithm 11 Evolutionary Strategy Algorithm

```
procedure ES($g, h, j$)        ▶ $g$ gens, $h$ - (,+), $j$ - 2 or multimembered ES
    Initialize population, $\mathbb{P}'_0$, = $\{\vec{d_1}, \vec{d_2}, ..., \vec{d_\mu}\}\epsilon\mathbb{P}^{\tilde{\mu}}$
    $\vec{d_i}$ : are filtered with a feasibility function using constraints
    Evaluate $\mathbb{P}'_0$ :where each member of the population is evaluated
    for i = 1 to g do
        recombine
        mutate
        evaluate
        select
    end for
end procedure
```

able. Next, Evolutionary Programming, which uses both crossover and mutation, is discussed.

E.4.2 Evolutionary Programming. Evolutionary Programming (EP) is similar to that of ES. Mutation is normally distributed and has some self-adaptation schedule into genotype mutations. Again, a state-of-the-art implementation of EPs are *meta*-EP.

Mutation: The asexual mutation operator mutates the population member with a standard deviation that is obtained for each component of the variable vector as the square root of a linear transformation of the fitness function. In overcoming tuning problems as the algorithm runs, they have added a vector of variances per individual. This vector is much similar to the parameter variables for the ES.

Recombination: EP does not use crossover or recombination, but relies heavily upon the mutation operator discussed.

Selection: The asexual essence of the mutation operator the offspring become the size of the population. Additionally, tournament selection is applied with ranking (in descending order).

E.4.3 Genetic Programming. It is most challenging to get a computer to accomplish a task without coaching it in how to achieve that task. Genetic Programming (GP) embodies this concept of a computer learning how to program itself, or auto programming. GP does this by genetically developing or allowing a population to evolve using Darwinian's *natural selection* along with biologically understood operations. These operations include, but are not held exclusively to the following: reproduction, crossover, mutation, and architecture-altering operations patterned after gene duplication and deletion.

E.4.3.1 Population Creation. The auto programming begins by random generation of many computer programs. These computer programs may be in the form of mathematical logic (represented by reverse polish) or in the form of program modules or sections. In the case of applying GP to any problem, including MOPs, modules can be thought of other search algorithms: local search algorithms, GAs, or EPs, to name a few. To begin the algorithm massively produces combinations of GAs that can work on a similar problems for each objective separately. Each search algorithm, GAs (simple GA, messy GA, fast messy GA, etc), can use different operators like conjugate gradient, local twist, or sweep, to swap out after a conclusive run for a particular configurations and objective.

Hierarchical GPs can be used to manage each configuration for each objective separately - keeping track of what configurations gives the best result for each objective.

Reproduction: This operator simply selects the next generation based on fitness values. Once all population member have been evaluated and other operations have been applied – configurations finding the best fitness are moved to the next population pool.

Crossover: This operator is sexual reproduction after the selection of two parental programs. When using crossover, it would be the selection of two configurations, and then swapping part of a configuration for the part of another configuration. Crossover points are randomly chosen in each parent. The subtree at the crossover point for the first parent is deleted and replaced with that at the second parent's crossover point subtree.

Mutation: The mutation operator selects a configuration to mutate based on fitness. It randomly selects a mutate point, deletes the subtree at that point, and grows another subtree at the mutation point to replace it.

Architecture-Altering Operations: This operator dynamically allows the removal and insertion of sub-routines within programs genetically. This operator allows for the actual program to be modified or shaped in a way so that is may adjust to solving the problem. Ultimately, this operator relieves the programmer from writing concrete specifications for a program. This operator mutates sub-routines that are already being *wholly* crossed and mutated in and out of population members.

E.4.4 Classical Genetic Algorithm. There are many variations on the Genetic Algorithm. The simple Genetic Algorithm and the steady state GA (ssGA) are two good examples of such algorithms. All are rooted back to the Darwin's theory of evolution, natural selection, and genetics [103]. Described in detail in this section is the simple Genetic Algorithm.

A Simple Genetic Algorithm (sGA) is at the foundation of genetic algorithm designs. It evolves a population of complete solutions using the basic operators of crossover and mutation. Algorithm 12 presents the pseudocode for the sGA.

This is a single objective generational GA. Evolutionary Operators (EVOPs) for this particular genetic algorithm are basic including crossover and mutation. The inverse operator is added by Holland [104] in an attempt to correct breaking linkages using single-point crossover. The operator mimics the property from nature where the function of a gene is *independent* of its location on the chromosome.

Algorithm 12 Simple Genetic Algorithm

```
procedure sGA($\mathcal{N}, g, f_k(\vec{x})$)
    $\mathcal{P}'_{new} \leftarrow rand(\mathcal{P})$                ▶ Population Randomly Initialized.
    $\mathcal{P}'_{old} \leftarrow \mathcal{P}'_{new}$
    $\mathcal{S} \leftarrow \bigcup_{i=1}^{h} \mathcal{P}'_{old}(rand)$        ▶ Selection $h$ members for recombination.
    $\mathcal{P}'_{new} = \mathcal{EVOP}(\mathcal{S})$      ▶ Conduct (EVOPs) on S to produce new members.
    if ($size(P_{new}) < p$) then
        step 4.
    end if
    $\mathcal{P}'_{old} \leftarrow \mathcal{P}'_{new}$
    Stop if termination criteria met; otherwise, step 3.
end procedure
```

An advantage of this algorithm is that it is easy to implement. Disadvantages of this algorithm may include that it tends to get stuck in local minima/maxima. Additionally this is an *implicit* BB building algorithm because it manipulates and seeks complete solutions – not partial solutions. Other types of regular Genetic Algorithms that modify the crossover operator or mutation operator also exist; however, this document places all that evaluate complete solutions as a single unit in this category. Next the messy Genetic Algorithm and fast messy Genetic algorithms are discussed.

E.4.5 EA Summary of GPs, ESs, EPs, and GAs: Most, if not all, EAs can be characterized by Equation 94 where $\mathcal{S}$ and $\mathcal{V}$ represent the selection operator and evolutionary variation operators respectively [16].

$$\mathcal{X}[t+1] = \mathcal{S}\left(\mathcal{V}\left(\mathcal{X}[t]\right)\right) \tag{94}$$

However, major differences are illustrated within algorithm structure and problem solution attainment. These difference are tied together with BB search method: implicit and explicit. This phenomenon is more relevant and apparent when solving more difficult multicriteria problems (MOPs). Implicit BBBs tend to record lower effectiveness performance than the explicit BBBs. Some extra data collected and presented illustrating the lower performance for MOEAs when solving test suite MOPs VL1, VL2, VL3, VL4, and VL5 can be found in Appendix C on page 306 in Figures 90, 91, 92, 93, and 94.

Algorithm 13 mGA algorithm

```
procedure mGA($\mathbf{o}$)                                  ▶ $\mathbf{o}$ is the size of BBs
    Randomly Create Competitive Template (CT)
    for i = 1 to epoch do
        Initialize BB Population $\mathbb{P}'_{bb^{\mathbf{o}}}$                  ▶ PEI Phase
        Evaluate BBs in $\mathbb{P}'_{bb^{\mathbf{o}}}$     ▶ each BB is overlaid onto the CT before eval
                                                        ▶ Begin Primordial Phase
        for j=1:primordial generations do
            Dope $\mathbb{P}'_{bb^{\mathbf{o}}}$ with good BBs
        end for
                                                          ▶ End Primordial Phase
                                                    ▶ Begin Juxtapositional Phase
        for j = 1 to Max Juxtapositional Generations do
            Cut-and-Slice
            Evaluate Each BB's fitness using CT
            Perform Tournament Thresholding Selection
        end for
                                                      ▶ End Juxtapositional Phase
        Update CT                                   ▶ Best known value becomes CT
    end for
end procedure
```

E.4.6 Explicit Building Block Search Genetic Algorithms. Over the last decade, some radical new ideas genetic algorithm design have solidified. Among these ideas is that idea that genetic algorithms, BBB, specifically seek good BBs for combining together to make good solutions. Discussed are five such algorithms, the messy GA, fast messy GA, Intelligent Evolutionary Algorithm, Extended Compact GA, and the Bayesian Optimization Algorithm BOA.

E.4.6.1 messy Genetic Algorithm (mGA). The mGA consists of an initialization (PEI), primordial and juxtaposition phases. The main difference between the mGA and a simple GA is in the fact that the mGA uses varying string lengths. The encoding of a chromosome is the same as what is described in Equation 6. The partially enumerative initialization builds a population of BBs consisting of all possible partial solutions of a specified length. Algorithm 13 presents the pseudocode for the mGA and Figure 10 on page 76 illustrates the program flow of one epoch as well as how the population members grows for one iteration of each phase.

$$|\mathcal{A}|^{\mathbf{o}} * \binom{\ell}{\mathbf{o}} \tag{95}$$

The differences between a mGA and a sGA are many! The initial populations are much different. The mGA produces a population of partial solutions (BBs of a specified length) whereas the population of the sGA is a group of chromosomes or complete solutions. The fitness function for the mGA is modified (uses a competitive template (CT)) to handle partial solutions (BBs); on the other hand, a sGA can only evaluate an entire chromosome's fitness for comparison. Associated with the type of population members held in the population is the population size itself. The mGA has a larger initial population size than that of a sGA. The mGA's population size can be calculated using Equation 95 where $\mathbf{o}$ is the block size, ℓ is the length of the string, and $|\mathcal{A}|$ is the cardinality of the alphabet [82]. The idea for such a large population size is to have every combination of a particular block size to be present for the insurance of having the answer to the problem held in the population. Furthermore, the mGA enriches the population pool (pop-pool) with good BBs in some cases allowing duplicate stings to reside and periodically reduces the total population size during selection. Finally, the mGA has variable length strings maintained in the population where the sGA population members are essentially always the same length - that of the original fixed string length. This is a generational algorithm that focuses on generations as steps within the algorithm itself. It is also a single objective *explicit* BB genetic algorithm.

Next, the fast messy Genetic Algorithm is discussed as a follow on to the mGA. Designers sought to improve the memory requirement and the BB search phase of the mGA. The result is the fast messy GA.

Algorithm 14 fmGA algorithm

```
procedure FMGA($\mathcal{N}'$,BBF schedule)        ▶ $\mathcal{N}'$ members evolved using BBF scedule
    Randomly Create Competitive Template (CT)
    for t = 1 to epochs do
                                                    ▶ PCI Phase
        Perform Probabilistically Complete initialization
        Evaluate each BB's fitness using CT
                                                    ▶ Building Block Filtering (BBF) Phase
        for i = 1 to Max BBF generations do
            if BBF schedule requires cutting at this generation then
                Perform BBF
            else
                Perform Tournament Thresholding Selection
            end if
        end for
                                                    ▶ Juxtapositional Phase
        for i = 1 Max Juxtapositional generations do
            Cut-and-Splice
            Evaluate each BB's fitness using CT
            Perform Tournament Thresholding Selection
        end for
        Update Competitive Template                 ▶ Using best known value
    end for
end procedure
```

E.4.6.2 fast messy Genetic Algorithm. The mGA's advantage over the sGA is its ability to *explicitly* create tightly linked BBs for the optimization of deception problem - basically, insuring it has all good BBs somewhere in the population of order **o** BBs. However, the mGA's insurance policy does not come at cost; indeed, it is extremely expensive to build every combination of a particular BB size to put into a population. This initialization dominates the entire algorithm [90]. The fmGA is designed to reduce this complexity by replacing the initialization phase and primordial phase with a probabilistic complete initialization and primordial phase consisting of selection and BB filtering. PCI and BBF are an alternate means to providing the juxtaposition phase with highly fit BBs [93]. The fmGA pseudocode is provided in Algorithm 14 and Figure 106 illustrates the program flow and chromosome sizes during program execution. When comparing the complexity of the fmGA to that of the mGA in Table 23 on page 170, it can be concluded that the fmGA has lower complexity.

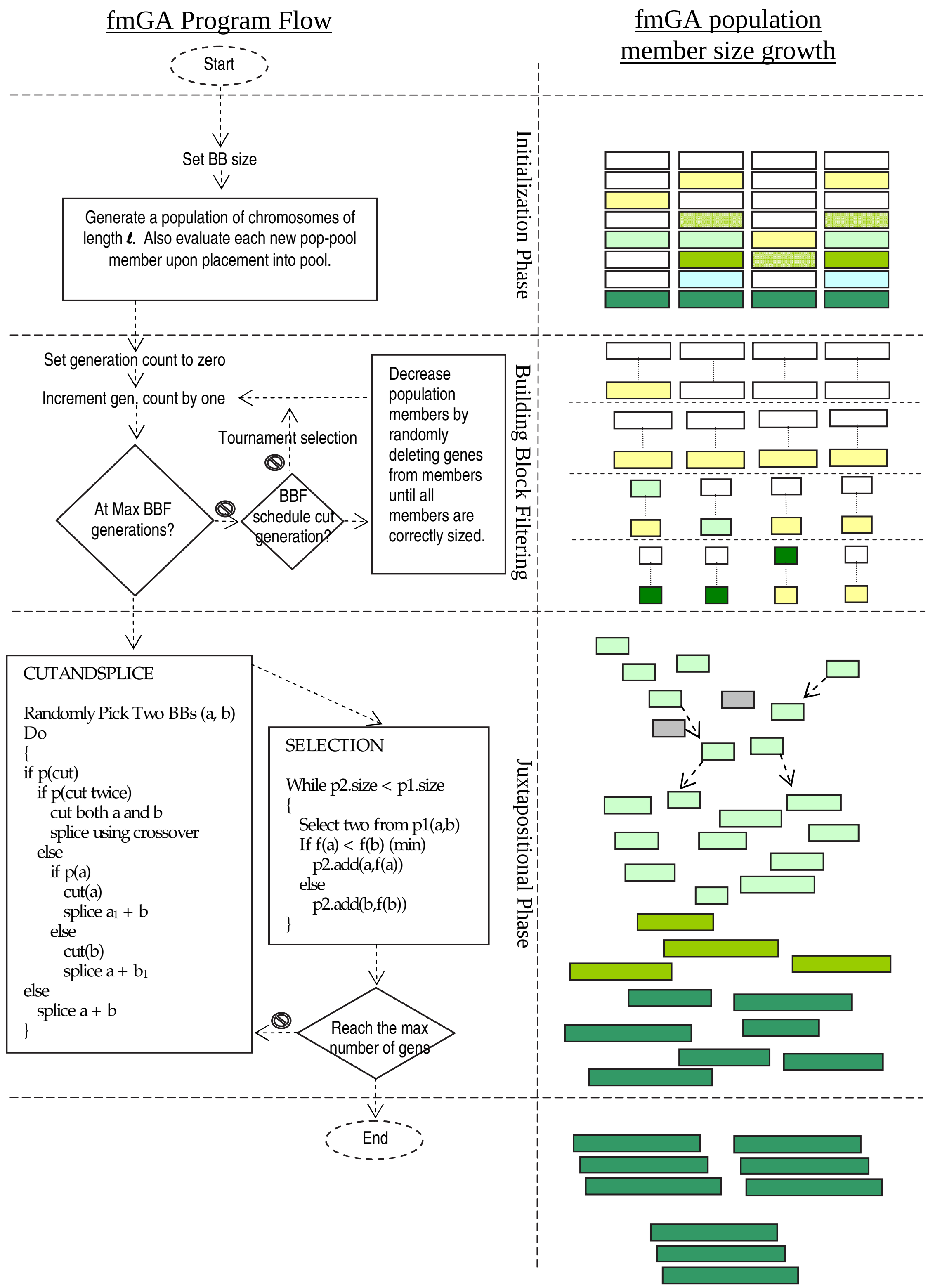
$$2^{\mathbf{o}} \frac{\binom{\ell}{\ell'}}{\binom{\ell-\mathbf{o}}{\ell'-\mathbf{o}}} 2|\mathcal{A}|(\alpha)\beta^2(\hat{m}-1) \tag{96}$$

The PCI phase creates an initial pop-pool size of n described by Equation 96, which is probabilistically equivalent to the pop-pool size at the end of the primordial phase of mGAs[2].

$$p(\ell', \mathbf{o}, \ell) = \frac{\binom{\ell-\mathbf{o}}{\ell'-\mathbf{o}}}{\binom{\ell}{\ell'}} \tag{97}$$

It is accepted as true that the population size is the multiplication of three equations: The gene-wise probability equation, the allele-wise combinatoric equation, and the BB evaluation noise equation [93]. Furthermore, it can be shown that the probability gene-wise equation is the probability of selecting a gene combination of size $\mathbf{o}$ in a string of length ℓ' having the total number of genes, ℓ, is given as Equation 97. If, $\hat{m}_g$, is assigned to the *inverse* of Equation 97, it is suggested that each subpopulation of size $\mathcal{N}_g$ has one needed string, on average, gene combination of size $\mathbf{o}$. Equation 98 defines $\mathcal{N}_g$. If this equation suggests that one of the required gene combinations for a particular BB size $\mathbf{o}$ is obtained, then it can also be expected that each and and every possible combination of $\mathbf{o}$ BB size is also sought, which makes for $2^{\mathbf{o}}$ allelic combinations or allele-wise combinatoric population size multiplier.

[2] The MOEA version of population sizing equations can be found in Section 3.2.2.


fmGA Program Flow
fmGA population member size growth
Start
Set BB size
Generate a population of chromosomes of length ℓ. Also evaluate each new pop-pool member upon placement into pool.
Initialization Phase
Set generation count to zero
Increment gen. count by one
Tournament selection
At Max BBF generations?
BBF schedule cut generation?
Decrease population members by randomly deleting genes from members until all members are correctly sized.
Building Block Filtering
Reach the max number of gens
End
Juxtapositional Phase

```
CUTANDSPLICE

Randomly Pick Two BBs (a, b)
Do
{
if p(cut)
   if p(cut twice)
      cut both a and b
      splice using crossover
   else
      if p(a)
         cut(a)
         splice $a_1$ + b
      else
         cut(b)
         splice a + $b_1$
else
   splice a + b
}
```

```
SELECTION

While p2.size < p1.size
{
   Select two from p1(a,b)
   If f(a) < f(b) (min)
      p2.add(a,f(a))
   else
      p2.add(b,f(b))
}
```

Figure 106: Illustrated in this figure is the program flow of the fast messy genetic algorithm and the population of BB and their growth while the algorithm progresses.

A second multiplier is then defined in Equation 99 called the BB evaluation noise equation. This equation makes for a population size calculation where the selection error between two competing BBs is no more than an α different and a difference between sampled BB means is β. Finally, the result is a simpler, more manageable, population sizing calculation Equation 100. [93]

$$\mathcal{N}_g = \frac{1}{\frac{\binom{\ell - \wr}{\ell' - \wr}}{\binom{\ell}{\ell'}}} \tag{98}$$

$$\mathcal{N}_a = 2|\mathcal{A}|(\alpha)\beta^2(\hat{m} - 1) \tag{99}$$

$$\mathcal{N} = \mathcal{N}_a \mathcal{N}_g \tag{100}$$

Once the population size is determined, the initial population is created and the algorithm begins. The length of strings, ℓ', is set to ℓ-$\mathbf{o}$. The primordial phase performs several tournament selection generations to build up copies of highly fit strings followed by BBF to reduce the string length toward the BB size $\mathbf{o}$. It should be noted that building block filtering is nothing more than randomly deleting genes from a particular string effectively reducing it [81]. To conclude, instead of having a huge initialization cost as with the mGA, the fmGA allows a more optimal initial population mechanism that is statistically equivalent to that of the mGA.

Algorithm 15 IGC algorithm

```
procedure IGC($\mathbb{P}'$)                                   ▶ $\mathbb{P}'$ is the population.
    Divide large chromosomes into smaller segments            ▶ Division Phase
    Identify potentially good smaller segments                ▶ Conquest Phase
    Combine all potentially good smaller segments into fully specified solutions
        ▶ Combination Phase
end procedure
```

E.4.6.3 Intelligent Evolutionary Algorithm (IEA). The Intelligent Evolutionary Algorithm is based on an older smaller algorithm called the Intelligent Gene Collector. The claim to fame for the IGC is in its orthogonal experimental design. IGC has three phases: division phase, conquest phase, and combination Phase. It is much like the fmGA where the IGC uses a divide and conquer approach by first chopping up fully specified solutions into smaller segments. One big difference between the fmGA and the IGC is that the IGC uses contiguous genes and the fmGA can use genes in any sequence. A second major difference is in the Division phase. Where the fmGA randomly deletes bits from chromosomes until all chromosomes are of the same size, the IGC makes the assumption that it is dealing with a population of good population members and tries to identify common genes within the population. Next, in the Conquest Phase, good gene segments are identified in the IGC. This phase has a huge memory/space advantage over the fmGA because the IGC can only look for contiguous bit segments, where the fmGA looks for all combinations of non-contiguous bit segments. Finally, in the combination phase, the IGC differs from the fmGA's juxtapositional phase in that the IGC tries to combine a set of Latin Square combinations of the divided chromosomes together until two children are created: one having the better gene segment from the derived corresponding parents, and the second child is selected in a similar manner except that the factor with the smallest MED adopts the other level. Child two is selected from the second best set of factor settings when comparing the combinations of a Latin Square set of factor settings (avoiding a full factorial test set).

On small variable problems the IEA using IGC does not perform vs. simple GA with elitist strategy and one point crossover (OEGA), uniform crossover (UEGA), two-point crossover (TEGA), BLX-α[3] (BLXGA), and BOA. However, on large test

[3] BLX-α is called a blend crossover operator and has an interesting property: the location of the child solution depends on the difference in parent solutions. If the difference between the parent solutions is small, the difference between the child and parent solutions is also small. Basically, the crossover or movement of variables creates two children that are moved $(1-\gamma)$ and γ distance from the original values: γ is uniformly distribution for a fixed value of α.

variable problems, the IEA using IGC is found to be better than all other algorithms but the BOA.

Algorithm 16 IEA algorithm

```
procedure IEA($\mathbb{N}'$, $p_s$, $p_c$, $p_m$)    ▶ prob of selection, $p_s$ and prob of crossover,
        $p_c$, probability of mutation $p_m$
    Randomly Generate Population of size $\mathbb{N}'$
    repeat
        Evaluation each member of the population
        Update elite sets
        Selection: ▶ Perform a conventional selection operation. (i.e. Elitism on
            $100(1 - p_s)\%$ of the population)
        Randomly select $p_c * \mathcal{N}'$ parents including $I_{best}$ for IGC operation
        Mutation with probability of $p_m$ : $p_m = 0$ for the best pop member
    until Termination Criteria Met
end procedure
```

The IGA is a generational EA that *explicitly* searches for good BBs. In fact this algorithm is no doubt a good competitor to both the fmGA and the BOA. The pseudocode for IEA is given in Algorithm 16.

E.4.7 Probabilistic model building GAs. In addition to the explicit BB GAs designs, there is a competing group of Probabilistic Model Building GA (PMBGA) or Iterated Density Estimation Algorithms (IDEAs) designs that also perform well on difficult problem. In fact, one of these algorithms, Bayesian Optimization Algorithm (BOA), is the flag ship for genetic algorithm research done at Illinois Genetic Algorithms Laboratories at the University of Illinois at Urbana-Champaign. This section discusses the following single objective probabilistic model building GAs: Extended Compact GA (ECGA), Estimation Distribution Algorithm (EDA) and the BOA. The following is an outline for a general PMBGA [203]:

1. Generate initial population of $\mathcal{N}$ members
2. Select $\mathcal{N}'$ good solution where $\mathcal{N}' <= \mathcal{N}$
3. Calculate the joint probability distribution of selected individuals

4. Generate offspring according to the calculated probability distribution and replace parents

5. Go to step 2 unless the termination criteria are meet

Crossover and mutation is not involved in the process of generating offspring being completely replaced by the sampling of probability distribution. Furthermore, in contrast with implicit BBs builder GAs, PMBGAs process *explicit* BBs by using a probability model.

Algorithm 17 Extended Compact Genetic Algorithm

```
procedure ECGA($\mathcal{N}', p_c$)
    Initialize Population, $\mathbb{P}'$, Randomly
    repeat
        Evaluate Fitness of population members
        Tournament Selection
        Build MPM model using MDL
        if Population has not Converged then
            for ($i = 1$ to $\mathcal{N}' \cdot p_c$) do
                Randomly choose subsets from the current individuals where the
                    gene groups are identified by the current MPM
            end for
            for ($i = 1$ to $\mathcal{N}' \cdot (1 - p_c)$) do
                $P_{new} = P_{crossed} \cup$ best ($\mathcal{N}' \cdot (1 - p_c)$) individuals in $\mathbb{P}'$
            end for
        end if
    until Population Converges
end procedure
```

E.4.7.1 Extended Compact GA (ECGA). The ECGA is proposed by [99] in 1999. Its success hinges on the supposition that the probability distribution matching the problem's is equivalent to learning the linkage. The metric used for measuring the goodness of a probability distribution match is based on a minimum description length (MDL) model [191]. The concept behind using MDL models is that, given all things equal, a simpler distribution is better than a complex one.

The probability distributions used in the ECGA with the MDL are marginal product models (MPMs). The MPM models are advantageous to those found in the Compact GA (cGA) [100] and Population Based Incremental Learning (PBIL) [8] because the MPM models can represent probability distributions for more than one gene at a time by facilitating a direct linkage map with each partition separating tightly linked genes. The pseudocode for this particular algorithm is presented in Algorithm 17.

The ECGA is shown to be effective in literature on some toy problems like the following: One Max, Trap, Folded Trap I and Folded Trap II. However, it is difficult to have an MDL evaluation for multidimensional distributions; therefore, it is doubtful that this particular technique can be brought to the multiobjective field of study without difficulties.

E.4.7.2 Estimation of Distribution Algorithm. Estimation of Distribution Algorithms (EDAs) is an area of Evolutionary Computation where the algorithms use neither crossover nor a mutation operator. New populations are generated by sampling the probability distribution. Distributions are estimated from a bank or database of individuals kept from the previous generation. Approaches for estimating the probability distribution for the bank of individuals are many.

Algorithm 18 Estimation of Distribution Algorithms

```
procedure EDA($\mathcal{N}'$,g)        ▶ M is the size of the population, g is generations
    $D_0 \leftarrow$ generate $\mathcal{N}'$ individuals at random
    for i = 1 to g do
        $D^{se}_{i-1} \leftarrow$ Select $j$ from $D_{i-1}$
        $p_i(x) = p(x|D^{se}_{i-1}) \leftarrow$ est the prob distribution of an individual being
            among the selected individuals
        $D_i \leftarrow$ Sample $\mathcal{N}'$ individuals (the new population) from $p_i(x)$
    end for
end procedure
```

There are three different EDA approaches, as expressed above: independent variable, bivariate dependencies, and multiple dependencies case. For each of the

Table 62: Summary of EDA research findings [141]

EDA	**Description of Results**	**Problems**
UMDA	Good for linear problems having independent variables. However, requires memetic heuristics for combinatorial optimization.	Feature selection and Classifier Systems etc.
PBIL	Excellent on problems having independent variables in a binary search space. Vector probabilities instead of population.	Optimal weights between Neural Network nodes, GP, and Intelligent Vehicle Domains
MIMIC	A generation search on the best permutation of variables matching the problem's probability distribution using Kullback-Leibler distance.	Applied to problems with variables having at most two-order dependencies
COMIT	Estimation of probability distributions using a tree structured Bayesian Network by way of the Maximum Weight Spanning Tree (MWST) Algorithm.	" " " "
BMDA	Construction of an acyclic dependency graph. Can be connected or disconnected	" "
FDA	Requirement: Additively Decomposed Function (ADF) and the factorization of the joint probability distribution remains same for all iterations. Combination of EA and (SA)	Additively decomposable problems
BOA	Bayesian Network PBM and Bayesian Dirichlet (BD) metric to estimate joint probability distribution. Incorporation of prior information about the problem.	Decomposable problems of bounded difficulty
ECGA	Factorization of joint probability distribution as a product of variable size marginal distributions.	Silicon Cluster Optimization problem
EBNA	A Bayesian Network is used for factorizing the joint probability distribution. It uses the BIC score, K2+Penalized score or PC algorithm for guiding the search of a good model structure	Feature Subset Selection, featuring weighting in k-NN, Job Shop Scheduling and TSP

approaches there are different models that have been employed to capture the correct distribution for the problem being solved. The following descriptions gives some examples each employed:

Independent Variables All variables considered as univariate

1. Univariate Marginal Distribution Algorithm (UMDA) by M$\ddot{u}$hlenbein in 1998 [152]
2. Population Based Incremental Learning (PBIL) by Baluja in 1994 [8]
3. Compact Genetic Algorithm (cGA) by Harik et al. in 1998 [100]

Bivariate Dependencies Pairwise interaction among variables

1. Mutual Information Maximizing Input Clustering (MIMIC) algorithm by De Bonet et al. in 1997 [15]
2. Combining Optimizers with Mutual Information Tress (COMIT) by Baluja and Davies in 1997 [7]
3. Bivariate Marginal Distribution Algorithm (BMDA) by Pelikan and M$\ddot{u}$hlenbein in 1999 [183]

Multiple Dependencies Multiple dependencies among variables

1. Factorized Distribution Algorithm (FDA) (Mhlenbein et al. 1998)
2. Extended Compact Genetic Algorithm (ECGA) (Harik, 1999)
3. Bayesian Optimization Algorithm (BOA) (Pelikan et al., 2000)
4. Estimation of Bayesian Network Algorithm (EBNA) (Larranaga et al., 2000)

These types of algorithms combine an algorithm search with a statistical sampling of findings from the search space. They attempt to find inter-bit relationships or dependencies within a chromosome bit-wise or variable-wise. These algorithms have met with much success (see Table 62) and must be heeded as good competitors in the single and multiobjecitve field of EA study. Next, a more detailed description of the BOA is considered.

E.4.7.3 Bayesian Optimization Algorithm BOA. Although this algorithm is listed in the EDA section, its importance is emphasized by the fact that it is mentioned three times in this appendix. This particular algorithm has shown to solve difficult problems and parallelize with nearly linear speedup [172]. For a statistical model it uses a depend probabilistic model by building a Bayesian network to represent the inter-bit relations found from a discovered good population of complete solutions. The pseudocode for BOA is presented in Algorithm 19.

Algorithm 19 BOA algorithm

```
procedure BOA($\mathcal{N}'$,g))                    ▶ $\mathcal{N}'$ members evolved g gens
    Randomly generate a Population, $\mathbb{P}'$, of candidate $\mathcal{N}'$ solutions
    for t = 1 to g do
        Use selection operators to build candidates
        Use variation operator to Build a Bayesian network from selected candidates
        Draw new candidate solutions by sampling the constructed Bayesian network
    end for
end procedure
```

In addition this algorithms is used as a competitor for the algorithm developed in this study - the MOMGA-IIa. The description of this algorithm completes this section on SOEAs. These are the basis from which most of the MOEAs are developed; therefore, it is vital to have a good background of the SOEAs before venturing into discussions about MOEAs.

E.5 Multiobjective Evolutionary Algorithms (MOEAs)

This section outlines, chronologically, the development of major MOEAs over the years. A general discussion of each algorithm is given as well as pseudocode. Each

Table 63: Summary of EVOPs, fitness, and representation for discussed SOEAs. The notation within the table is as follows: m is mutation, c+m is crossover plus mutation, div-comb is divide and conquer, and pdf is probability distribution function

SOEA	**EVOPS**	$\mathbb{R}$ or $\{0,1\}$	**Explicit or Implicit BBs**
ES	m	$\mathbb{R}$	Implicit
GP	c+m	$\mathbb{R}$	Implicit
sGA	c+m	$\{0,1\}$	Implicit
mGA	cut-splice	$\{0,1\}$	Explicit
fmGA	cut-splice	$\{0,1\}$	Explicit
IEA	div-comb	$\{0,1\}$	Explicit
ECGA	pdf	$\{0,1\}$	Explicit
EDAs	pdf	$\{0,1\}$	Explicit
BOA	pdf	$\{0,1\}$	Explicit

MOEA is classified as being either an explicit or implicit BBB algorithm. Generally, the roots of each algorithm come from an earlier, single objective, implementation discussed above.

E.5.1 Vector Evaluated Genetic Algorithm (VEGA). One of the first MOEAs developed was called the Vector Evaluated Genetic Algorithm (VEGA) - designed by Schaffer in 1985 [194, 195]. Schaffer's design is also referred to as criterion selection because the algorithm bases population selection independently for each objective function. Furthermore, it was designed as an *implicit* generational Genetic Algorithm and uses similar EVOPs to those used in a single objective GA. Algorithm 20 presents the pseudocode for this MOEA.

The algorithm begins by making a population of $\mathcal{N}'$ complete solutions. Then, k sub-populations are created by selecting out of the $\mathcal{N}'$ original individuals good individuals with respect to each of the k objectives. A shuffling of these sub-populations is then achieved to evenly prioritize each member. The k sub-populations become one big population again. Finally, EVOPs are applied to the single population and the process starts all over again. This algorithm was found to suffer from

Algorithm 20 VEGA algorithm

```
procedure VEGA($\mathcal{N}', g, f_k(\vec{x})$)    ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    for t = 1 to g do
        Initialize Population $\mathbb{P}'$ of size $\mathcal{N}'$
        Create $k$ sub-populations        ▶ Each sub population has $\frac{\mathcal{N}'}{k}$ members
        Fill each $j^{th}$ sub-population by selecting from $\mathcal{P}'$ the best individuals
            w.r.t.$j^{th}$ objective
        Shuffle entire population to mix individuals
        Apply EVOPs on mixed individuals
    end for
end procedure
```

speciation[4]. The problem occurs because this MOEA's selection mechanism picks individuals who excel in one objective space, without regard to any other objective spaces. This is an undesirable effect because it opposes the goal of finding a balanced or compromised solutions set. Protection of the middling chromosomes is vital and can be fostered by adding a selection mechanism that encourages crossbreeding between different spiciest instead of the random mate selection of a tradition GA. [34] Many researchers have found speciation as a real problem [38, 170] in VEGA and used a k-branch tournaments to allow for the preservation of diversity [114].

Although the VEGA suffers from speciation, it is easy to implement and has been found to be useful in other domains. The next MOEA discussed is the Multi-objective Genetic Algorithm (MOGA) - first suggested by Goldberg in 1989.

E.5.2 Multiobjective Genetic Algorithm (MOGA). Fonseca and Fleming in 1993 redesigned Goldberg's Multiŏbjective Genetic Algorithm (MŎGA). The new MOGA ranks a certain individual corresponding to the number of chromosomes in the current population by which it is dominated. In other words, an individual $\vec{x}_i$ at generation t, is dominated by $p_i^{(t)}$ individuals in the current generation, thus an

[4] Speciation is a technical term for the evolution of a particular species within a populace. In genetic algorithm terms, it can be referred an algorithm getting caught in a local minimum by continuously generating similar population members - i.e. population members of a similar species.

individual is assigned a rank by the following rule: $rank(\vec{x}_i, t) = 1 + p_i^{(t)}$ [75]. The Algorithm 21 presents the pseudocode for the MOGA.

Algorithm 21 MOGA algorithm

```
procedure MOGA($\mathcal{N}'$, $g$, $f_k(\vec{x})$)   ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    Initialize Population $\mathbb{P}'$
    Evaluate Objective Values
    Assign Rank based on Pareto Dominance
    Compute Niche Count
    Assign Linearly Scaled Fitness
    Shared Fitness
    for i=1 to g do
        Selection via Stochastic Universal Sampling
        Single Point Crossover
        Mutation
        Evaluate Objective Values
        Assign Rank Based on Pareto Dominance
        Compute Niche Count
        Assign Linearly Scaled Fitness
        Assign Shared Fitness
    end for
end procedure
```

The fitness assignment for this algorithm is modified to support the rank based fitness assignment. First, the population is sorted according to rank. Next, fitness is assigned from best to worst according to some function. Finally, the average of fitness of individuals with the same rank so that all can be sampled at the same rate. Using this procedure, keeps the global fitness constant and maintains appropriate selective pressure. This algorithm is not in large use in the community and has been shown to not perform as well as the multiobjective simulated annealing [214].

E.5.3 Niched-Pareto Genetic Algorithm 1 and 2 (NPGA). In 1993 Horn and Nafpliotis proposed a tournament selection MOEA based on Pareto dominance and called it the Niched-Pareto Genetic Algorithm (NPGA) [105, 106]. The original NPGA takes two individuals and compares them against a sampling of the population. If one of the individual evaluates to be dominated while the other evaluates

Algorithm 22 NPGA algorithm

```
procedure NPGA($\mathcal{N}, g, f_k(\vec{x})$)    ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    Initialize Population $P$
    Evaluate Objective Value
    for i=1 to g do
        Specialize Binary Tournament Selection
        Begin
            if Only Candidate 1 dominated then
                Select Candidate 2
            else if Only Candidate 2 dominated then
                Select Candidate 1
            else if Both are Dominated or Non-Dominated then
                Perform specialized fitness sharing
                Return Candidate with lower niche count
            end if
        End
        Single Point Crossover
        Mutation
        Evaluate Objective Values
    end for
end procedure
```

to be non-dominated, the individual evaluating to be non-dominated is declared the winner. Ties are decided using a fitness sharing technique [95]. Algorithm 22 present the NPGA pseudocode.

Erickson et al. launched a new NPGA in 2001. This new design is called the NPGA-2 and uses Pareto ranking but keeps tournament selection. Niche counting is achieved by using the so-called *continuously updated fitness sharing* technique where individuals in the partially filled next generation as opposed to using the current generation [173]. The algorithm for NSPG-2 is presented in Algorithm 23. Both of these algorithms are generational *implicit* BBBs.

E.5.4 Non-dominated Sorting Genetic Algorithm I and II (NSGA). The Non-dominated Sorting Genetic Algorithm (NSGA) is another modification to the proposed Goldberg approach [207]. The pseudocode for this MOEA is given in Algorithm 24.

Algorithm 23 NPGA-2 algorithm

```
procedure NPGA-2($\mathcal{N}', g, f_k(\vec{x})$) ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    Initialize Population $\mathbb{P}'$
    Evaluate Objective Values
    for i=1 to g do
        Specialize Binary Tournament Selection using rank as domination degree
        Begin
            if Only Candidate 1 dominated then
                Select Candidate 2
            else if Only Candidate 2 dominated then
                Select Candidate 1
            else if Both are dominated or non-dominated then
                Perform specialized fitness sharing
                Return Candidate with lower niche count
            end if
        End
        Single Point Crossover
        Mutation
        Evaluate Objective Values
    end for
end procedure
```

Algorithm 24 NSGA-I algorithm

```
procedure NSGA-I($\mathcal{N}', g, f_j(\vec{x}_k)$) ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    Initialize Population $\mathbb{P}'$
    Evaluate Objective Values
    Assign Rank Based on Pareto dominance in Each Wave
    Compute Niche Count
    Assign Shared Fitness
    for i=1:g do
        Selection via Stochastic Universal Sampling
        Single Point Crossover
        Mutation
        Evaluate Objective Values
        Assign Rank Based on Pareto dominance in Each Wave
        Compute Niche Count
        Assign Shared Fitness
    end for
end procedure
```

The MOEA strips off layers of individuals evaluating to be non-dominated, one layer at a time, within the population. Each layer is assigned a rank and all individuals within that layer assumes an adjusted fitness value associated with this assigned rank as its fitness value. Before selection is performed, the entire population is ranked according to dominance. Thus, all solutions evaluating to non-dominated vectors are ranked together and given dummy fitness value which is proportional to the entire population size. Diversity is maintained within the population by sharing dummy fitness values among population individuals classified together. Each level of non-dominated vectors are taken separately until all members of the population are classified. Before selection, individuals with the highest rank always get more copies than the rest of the population. This allows for a better search of the PF_{known} regions. As a result, one might think that this MOEA converges rather quickly; however, this is prevented with the fitness sharing mechanism. This MOEA is shown to be successful in research; however, a modification to reduce computational complexity was completed by [56, 57]. Elitism and a crowded comparison operator that achieves parameterless diversity are added to the new MOEA called NSGA-II.

The non-dominated sorting algorithm-II (NSGA-II) is a generic non-explicit BB MOEA applied to multiobjective problems (MOPs) – based on the original design of NSGA. It builds a population of compete individuals, ranks and sorts each individual according to non-domination level, applies Evolutionary Operations (EVOPs) to create new pool of offspring, and then combines the parents and offspring before partitioning the new combined pool into fronts. The NSGA-II then conducts niching by adding a crowding distance to each member. It uses this crowding distance in its selection operator to keep a diverse front by making sure each member stays a crowding distance apart. This keeps the population diverse and helps the algorithm to explore the fitness landscape. Presented in Algorithm 25 is its pseudocode. This MOEA is used in most MOEA comparisons. It has also been used as a foundation for other successful algorithm designs like the multiobjective BOA [122].

Algorithm 25 NSGA-II algorithm

```
procedure NSGA-II($\mathcal{N}'$, $g$, $f_k(\vec{x}_k)$)    ▶ $\mathcal{N}'$ members evolved $g$ gens to solve $f_k(\vec{x})$
    Initialize Population $\mathbb{P}'$
    Begin
        Generate random population - size $\mathcal{N}'$
        Evaluate Objective Values
        Assign Rank (level) Based on Pareto dominance - sort
        Generate Child Population
            Binary Tournament Selection
            Recombination and Mutation
    End
    for i = 1 to g do
        for each Parent and Child in Population do
            Assign Rank (level) based on Pareto - sort
            Generate sets of non-dominated vectors along PF$_{known}$
            Loop (inside) by adding solutions to next generation starting from the
                first front until $\mathcal{N}'$ individuals found determine crowding distance
                between points on each front
        end for
        Select points (elitist) on the lower front (with lower rank) and are outside
            a crowding distance
        Create next generation
            Binary Tournament Selection
            Recombination and Mutation
    end for
end procedure
```

E.5.5 Strength Pareto Evolutionary Algorithm I and II (SPEA). The Strength Pareto Evolutionary Algorithm (SPEA) uses the dominance criterion for the fitness assignment and selection of solutions. One of the major problems with this algorithm is when a noisy fitness function is introduced, solutions evaluating to dominated vectors may evaluate to non-dominated vectors misleading the selection operator. This MOEA was introduced by Zitzler and Thiele in 1999 and has the flavor of a mixture of several MOEAs. The Algorithm pseudocode is specified in Algorithm 26. The algorithm begins by initializing a population. At each generation, individuals evaluating to non-dominated vectors are copied to a so-called external

Algorithm 26 SPEA algorithm

```
procedure SPEA($\mathcal{N}', g, f_k(\vec{x})$)
    Initialize Population $\mathbb{P}'$
    Create empty external set $\mathbb{E}'$
    for i:=1 to g do
        $\mathbb{E}' = \mathbb{E}' \cup \mathcal{ND}(\mathbb{P}')$   ▶ Copy members evaluating to be non-dominated of
            P to E
        $\mathbb{E}' = \mathcal{ND}(E)$   ▶ Keep only member evaluating to non-dominated vectors
            in $E$
        Prune $\mathbb{E}'$ (using clustering) if max capacity of $\mathbb{E}'$ is exceeded
        $\forall_{i \in \mathbb{P}'}$ Evaluate($\mathbb{P}'_i$)   ▶ Evaluate fitness for all member of $\mathbb{E}'$ and $\mathbb{P}'$
        $\forall_{i \in \mathbb{E}'}$ Evaluate($\mathbb{E}'_i$)
        $\mathcal{MP} \leftarrow \mathcal{T}(\mathbb{P}' \cup \mathbb{E}')$   ▶ Use binary tournament selection with
                                          ▶ replacement to select individuals from $\mathbb{P}' + \mathbb{E}'$
                                          ▶ (multiset union) until the mating pool is full
        Apply crossover and mutation on $\mathcal{MP}$
    end for
end procedure
```

set. Strength values are given to each individual within the external set. Diversity is kept by the *average linkage method* [167], which is a cluster pruning technique.

Algorithm 27 SPEA-II algorithm

```
procedure SPEA-II($\mathcal{N}', g, f_k(\vec{x})$)
    Initialize Population $\mathbb{P}'$
    Create empty external set $\mathbb{E}'$
    for i=1 to g do
        Compute fitness of each individual in $\mathbb{P}'$ and $\mathbb{E}'$
        Copy all individual evaluating to non-dominated vectors $\mathbb{P}'$ and $\mathbb{E}'$ to $\mathbb{E}'$
        Use the truncation operator to remove elements from $E$ when the capacity
            of the file has been extended
        If the capacity of $\mathbb{E}'$ has not been exceeded then use dominated individuals
            in $\mathbb{P}'$ to fill $\mathbb{E}'$
        Perform binary tournament selection with replacement to fill the mating
            pool
        Apply crossover and mutation to the mating pool
    end for
end procedure
```

The follow-up MOEA to the SPEA is the SPEA-II. Pseudocode can be found in Algorithm 27. The redesign called for a fitness adjustment, crowding mechanism and boundary preservation mechanism [237, 238]. Listed, the following briefly describes each:

1. The fitness assignment is improved by taking into account how many individuals are dominate to and dominated by each individual. [238]
2. A nearest neighbor density estimation technique allows for a more precise guidance of the search process. [238]
3. Archival truncation methods guarantee the preservation of boundary solutions. [238]

The SPEA-II and NSGA-II seem to be the two of the most prominent MOEAs used when comparing a newly designed MOEA. Prevalent in these generational MOEA are the fact that they are *implicit* BBBs and they reply heavily on some kind of niching, crowding or fitness sharing scheme.

Algorithm 28 PAES algorithm

```
procedure PAES($f_k(\vec{x})$)
    repeat
        Initialize Single Population parent, $\mathcal{C}$, and add to archive, $\mathbb{A}$
        Mutate $\mathcal{C}$ to produce child $\mathcal{C}'$ and evaluate fitness
        if $\mathcal{C} \succ \mathcal{C}'$ then
            discard $\mathcal{C}'$
        else if $\mathcal{C} \succ \mathcal{C}'$ then
            replace $\mathcal{C}$ with $\mathcal{C}'$, and add $\mathcal{C}$ to $\mathbb{A}$
        else if $\exists_{\mathcal{C}" \in \mathbb{A}}(\mathcal{C}" \succ \mathcal{C}')$ then
            discard $\mathcal{C}'$
        else
            apply test $(\mathcal{C}, \mathcal{C}', \mathbb{A})$ to determine which becomes the new current solution and whether to add $\mathcal{C}'$ to $\mathbb{A}$
        end if
    until termination criteria is met
end procedure
```

E.5.6 Pareto Archived ES I and M-PAES. The Pareto Archived Evolution Strategy PAES) was designed and implemented by Knowles and Corne in 2000 [135].

This MOEA uses Pareto based selection combined with an ES (described in Section E.4.1). Using a $(1+1)$-ES in combination with a historical archive that holds old solutions evaluating to non-dominated vectors. Comparisons are made to the historical archive when new individuals are created. In addition, PAES contains a crowding procedure that divides the objective space up in a recursive way to achieve diversity. The algorithm is presented in Algorith 28.

Other implementation,$(1+\lambda)$-ES and $(\mu+\lambda)$-ES,of this MOEA were proposed by authors of the PAES; however, these were deemed to not improve the overall performance of this MOEA. A *memetic*[5] version of PAES, called M-PAES was developed as a follow up to this algorithm in 2000 [130]. Finally, this MOEA is a convergent *implicit* BB MOEA.

Algorithm 29 PESA algorithm

```
procedure PESA($\mathcal{N}'$, $f_k(\vec{x})$)
    Initialize Population $\mathbb{P}'_i$ of size $\mathcal{N}'$ Randomly
    Evaluate each member of $\mathbb{P}'_i$
    Initialize the external population $\mathbb{P}'_e$ to the empty set
    repeat
        Incorporate individuals evaluating to non-dominated vectors from $\mathbb{P}'_i$ into $\mathbb{P}'_e$
        Delete the current contents of $\mathbb{P}'_i$
        repeat
            With probability $p_c$, select two parents from $\mathbb{P}'_e$    ▶ $p_c$ is the probability of crossover
            Produce a single child via crossover
            Mutate the child created in the previous step
            With probability $(1 - p_c)$, select one parent
            Mutate the selected parent to produce a child
        until $\mathbb{P}'_i$ is filled
    until termination criteria is met
    Return($\mathbb{P}'_e$)    ▶ Return the members of $\mathbb{P}'_e$ as the result
end procedure
```

[5] A memetic algorithm donotes the use of local search heuristic with a population-based strategy. The word memetic has its roots in the word meme - which is introduced in 1990 by Richard Dawkins in his book "The Selfish Gene." [34]

E.5.7 Pareto Envelop-based Selection Algorithm I and II (PESA). The Pareto Envelop-based Selection Algorithm (PESA) is suggested by Corne et al. in 2000 [136]. The pseudocode for the method is given in Algorithm 29. The method consists of a small internal population and a larger external population. A hyper-grid division of phenotype space is used to maintain selection diversity (application of a crowding measure) as the MOEA runs. Furthermore, this crowding measure is used to allow solutions into the external population via an archive of solutions evaluating to non-dominated vectors. A revised version of this MOEA is called PESA II. The difference between the PESA-I and II is that selection is region-based and selection becomes a hyperbox, not just an individual (i.e. first select a hyperbox, then select an individual within that hyperbox). The motivation behind this approach is to reduce computational cost associated with Pareto ranking [137]. Finally, these MOEAs are convergent *implicit* BB MOEAs.

Algorithm 30 MOSGA [3]

```
procedure MOSGA($\mathcal{N}', g, f_k(\vec{x})$)
    Initialize Population $\mathbb{P}'$
    repeat
        for (i = 1 to g) do
            Randomly Select p parents from $\mathbb{P}'$
            Apply EVOPs to create a child
            Calculate the rank of the child
            Rank the entire population with the new child
            Locate the most similar individual
            if New childs ranking is better than the similar individual then
                Replace the similar individual with new child
                Update the ranking of the entire population
            end if
        end for
    until Stopping criterion is met
end procedure
```

E.5.8 Multiobjective Struggle GA (MOSGA). The Multiobjective Struggle Genetic Algorithm (MOSGA) [2] combines the struggle crowding genetic algorithm [211] with a Pareto based ranking. The algorithm has the same pattern as the struggle algorithm where two parents are chosen at random from the population, and the normal crossover and mutation is performed to create a child. The child then competes with the most similar individuals in the entire population. The child replaces similar individuals if the child has a better ranking - counteracting genetic drift. The ranking method employed is the same as what is presented by Fonseca and Fleming in 1993 [75].

Although this MOEA has the flavor of being the MOGA, this algorithm is devised to counteract genetic drift which is known to spoil population diversity [2]. An advancement to this algorithm is a technique to assess the robustness of optimal solutions generated by the MOEA. Generational information is extracted from the MOEA to construct a response surface and a good estimate of the robustness of the Pareto front. Again, this algorithm is a generational MOEA and also an *implicit* BB MOEA.

Algorithm 31 OMOEA [232]

```
procedure OMOEA I($\mathcal{N}'$, $f_k(\vec{x})$)    ▶ $\mathcal{N}'$ members evolved until a specified precision is found for $f_k(\vec{x})$
    Input decision space $\chi$ as initial niche.
    Evolve niches into $\mathbb{P}'_{\mathcal{N}}(1)$
    Split the niche into a group of $\Psi_{\mathcal{N}}$ sub-niches.
    Initialize $\mathbb{P}'$ and $\Psi$
    $\mathbb{P}' \leftarrow \mathbb{P}'_{\mathcal{N}}(1)$; $\Psi \leftarrow \Psi_{\mathcal{N}}$
    gen=1
    repeat
        for (Each $\chi^{(s)}_{\mathcal{N}} \in \chi$) do
            Evolve $\chi^{(s)}_{\mathcal{N}}$ and yield $P^{(s)}_{\mathcal{N}}(1)$
            Split the niche into a group of $\Psi_{\mathcal{N}}$ niches.
        end for
        $\Psi \leftarrow \bigcup \Psi_{\mathcal{N}}$;$\mathbb{P}'$:$\mathbb{P}' \leftarrow \bigcup \mathbb{P}'^{(s)}_{\mathcal{N}}(1)$
        $gen = gen + 1$
    until (current $\mathbb{P}'$ does not reach the required precision, and the solution number of $\mathbb{P}'$ is not more than a critical value)
    Output $\mathbb{P}'$ as the satisfying close-to-Pareto-optimal set of MOP
end procedure
```

E.5.9 Orthogonal Multiobjective Evolutionary Algorithm I and II (OMOEA). The Orthogonal Multiobjective Evolutionary Algorithm (OMEA) process begins with a strict definition of the MOP constraints involved for a particular problem to solve. These constraints are considered when Pareto dominance is defined. The algorithm starts by defining a single niche in the decision space χ. This niche is recursively split into a group of sub-niches over and over again until a stopping criteria is satisfied. This partitioning forces a uniform search. The pseudocode for OMOEA is given in Algorithm 31 where $\mathbb{P}'$ denotes the global population and Ψ denotes the set of all sub-niches. [234]

Generally, this MOEA performs well; however, a couple of short comings were found [233]:

1. Strong interaction (high epistasis) between variables degrades the performance of OMOEA in both precision and distribution of the PF$_{known}$ vectors.
2. As the number of objective increase, the number of solutions increase exponentially.

Algorithm 32 OMOEA-II [30]

```
procedure OMOEA-II($\mathcal{N}$, $f_k(\vec{x})$)
    Randomly create population $\mathbb{P}_0$ with size $\mathcal{N}$.
    Counter $t \leftarrow 0$
    repeat
        Apply Crossover Operator on $\mathbb{P}_t$ resulting in $\mathbb{P}'_t$ offspring    ▶ $|\mathbb{P}_t| = |\mathbb{P}'_t|$
        $\mathbb{P}"_t = \mathbb{P}_t \cup \mathbb{P}'_t$
        Perform Selection on $\mathbb{P}"_t$ resulting in $\mathbb{P}_{t+1}$
        $t = t + 1$
    until Stopping Criteria Satisfied
    Output $\mathbb{P}_t$
end procedure
```

These short comings listed above are not unheard of for MOEAs. As a matter of principle, MOEA designers must recognize both of these problems when developing a new MOEA. The answer to these problems is purposed in [233] and called the

OMOEA-II. The modification to the OMOEA is to reduce the size of the orthogonal array in order to exploit optimality within a relatively small space. The pseudocode for OMOEA-II is presented in Algorithm 32. Finally, this is a convergence MOEA that *implicitly* seeks BBs.

Algorithm 33 GENMOP algorithm

```
procedure GENMOP($\mathcal{N}, g, f_k(\vec{x})$)
    Initialize Parent Population $\mathbb{P}_p$ of size $\mathcal{N}$
    Evaluate, Rank, Normalize and Save Parent Population
    for i=1 to g do
        Initialize Children and Mating Pool
        Fill Mating Pool with Parents by Rank
        for j = 1 to size(children pool) do
            Statistically select EVOP (weighted section based on previous
                good/bad children record)
            Apply selected EVOP on Children and Mating Pool once
            Store EVOP used with new child
        end for
        Mutate new Children
        Evaluate new Children
        Combine Parents with new Children into a new Parent Pool
        Rank, Normalize and Save new Parent Pool
    end for
end procedure
```

E.5.10 General Multiobjective Evolutionary Algorithm (GENMOP). The General Multiobjective Evolutionary Algorithm (GENMOP) is a general MOEA designed at the Air Force Institute of Technology AFIT). GENMOP employs numerous operators to select from when conducting EVOPs. As the search progresses, it more often chooses EVOPs that repeatedly produce better solutions. The algorithm works on the supposition that operators that continuously produce better solutions will, in the future, continue to produce good solutions. The pseudocode for this is given in Algorithm 33. In addition to the pseudocode a program flow/population growth diagram is presented in Figure 107 to illustrate the flow population members throughout execution of the MOEA. GENMOP is a generational *implicit* MOEA that can be

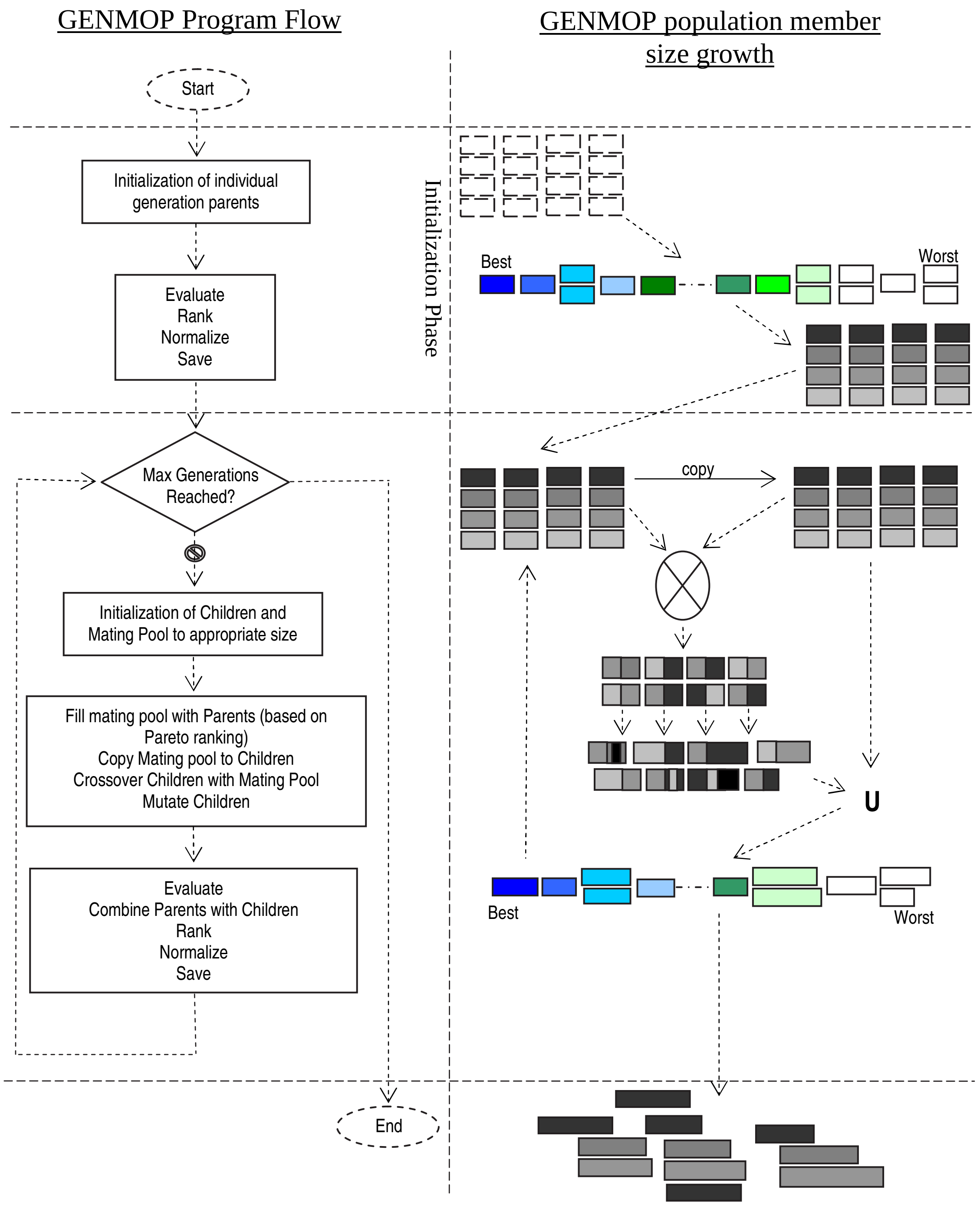

GENMOP Program Flow
GENMOP population member size growth
Start
Initialization of individual generation parents
Evaluate
Rank
Normalize
Save
Initialization Phase
Best
Worst
Max Generations Reached?
copy
Initialization of Children and Mating Pool to appropriate size
Fill mating pool with Parents (based on Pareto ranking)
Copy Mating pool to Children
Crossover Children with Mating Pool
Mutate Children
U
Evaluate
Combine Parents with Children
Rank
Normalize
Save
Best
Worst
End

Figure 107: Illustrated is the program flow of the GENMOP. Population of variable length solutions and the evolution process while the algorithm progresses is illustrated. GENMOP pseudocode can be found in Algorithm 33.

used on generic problems because it can adapt its operator use to those that provide better solutions.

Algorithm 34 MOMGA algorithm

```
procedure MOMGA($\mathcal{N}, g, f_k(\vec{x})$)
    for i = 1 to epoch do
                                                        ▶ PEI Phase
        Perform Partially Enumerative Initialization
        Evaluate each each population member's fitness w.r.t.$k$ templates
                                                        ▶ Primordial Phase
        for i = 1 to Max Primordial Generations do
            Perform Tournament Thresholding Selection
            if Appropriate number of generations accomplished then
                Reduce Population Size
            end if
        end for
                                                        ▶ Juxtapositional Phase
        for i = 1 to Max Juxtapositional Generations do
            Cut-and-Slice
            Evaluate Each Population member's fitness w.r.t.$k$ templates
            Perform Tournament Thresholding Selection and Fitness Sharing
            $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
        end for
        Update $k$ templates            ▶ Using best known value in each objective
    end for
end procedure
```

E.5.11 Multiobjective messy GA (MOMGA). The Multiobjective messy GA (MOMGA) is a multiobjective implementation of the messy GA. It works in three phases: initialization to build a pool of BBs, primordial phase for finding important BBs, and the juxtapositional phase for combining these BBs to form optimal or near-optimal solutions [94, 216]. In the primordial phase, partial strings are initialized. However, it differs from the original messy GA in that the MOMGA uses multiple competitive templates, each correspond to an individual objective function. MOMGA begins with random templates and then finds best templates for each objective from the best solutions obtained at the end of each era, thereby finding the competitive template for each objective for an era, a matter which is important for

Table 64: Summary of EVOPs, fitness, sharing, and representation for discussed implicit BB MOEAs.

MOEA	**EVOPS**	**Fitness**	**Sharing**	$\mathbb{R}$ or $\{0,1\}$	**Explicit or Implicit BB**
VEGA	c+m	-	Phenotypic Fitness σ_{share}	$\{0,1\}$	Implicit
MOGA	c+m	Linear interpolation using Fonseca and Fleming's Pareto ranking [75]	Phenotypic Fitness σ_{share}	$\mathbb{R}$ $\{0,1\}$	Implicit
NPGA 1	c	Tournament	Phenotypic Fitness σ_{share}	$\{0,1\}$ $\mathbb{R}$	Implicit
NPGA 2	c	Rank Dominance	Phenotypic Continuously Update fit. Technique	$\mathbb{R}$	Implicit
NSGA I	c	Dummy fitness using Goldberg's Pareto ranking [89]	Phenotypic (σ_{share} - Fitness)	$\{0,1\}$ $\mathbb{R}$	Implicit
NSGA II	c+m	-	Phenotypic Fitness σ_{share}	$\{0,1\}$ $\mathbb{R}$	Implicit
SPEA I	c+m	Dummy fitness using Goldberg's Pareto ranking [89]	Phenotypic Fitness σ_{share}	$\{0,1\}$	Implicit
SPEA-II	c+m	Strength value base on dominance	Density function niching	Prog Tree	Implicit
PAES-I	m	(1+1)single grid	Phenotypic Fitness σ_{share}	$\{0,1\}$ $\mathbb{R}$	Implicit
M-PAES	m	(1+1)single grid	Phenotypic Fitness σ_{share}	$\{0,1\}$ $\mathbb{R}$	Implicit
PESA	c+m	-	Phenotypic (Region - sharing)	$\{0,1\}$	Implicit
PESA-II	c+m	-	Phenotypic (Hyperbox - sharing)	$\{0,1\}$	Implicit
MOSGA	c+m	Linear interpolation using Fonseca and Fleming's Pareto ranking [75]	Phenotypic Fitness σ_{share}	$\{0,1\}$	Implicit
OMEA		-	Niching	$\mathbb{R}$	Implicit
OMEA-II		-	Niching	$\mathbb{R}$	Implicit
GENMOP	c+m	-	Phenotypic Fitness σ_{share}	$\mathbb{R}$	Implicit

proper evaluation of partial solutions in an era. Tournament selection and cut-and-slice operators are also inherited from the original messy GAs. Finally, the dominance measure used for selection and diversity preserving mechanism of NPGA is used, the resulting algorithm has both the desired properties of a multiobjective optimizer. This algorithm has also been suggested as a parallel MOEA. In fact, if the cardinality of this set is not the same as the desired population size, other individuals from the offspring population can be included. This strategy shows that convergence of this algorithm to the Pareto-optimal set - which is fantastic. A parallel version of the MOMGA lack the second task of maintaining diversity of Pareto-optimal solutions. Thus, explicit diversity preserving mechanism should be added to the MOMGA to make it more usable in practice.

This MOEA is a generational explicit BB building algorithm. The pseudocode for the MOMGA is presented in Algorithm 34.

E.6 Summary

This appendix discusses a broad variety of MOEAs. For completeness the rooted SOEAs are given as well to illustrate the design development process and thoughts between having a single objective algorithm and making it a multiobjective algorithm. In addition, these algorithm are classified with respect to the different being either an implicit or explicit BBB. Plus, if the algorithm is an implicit BBB, is it an interior or exterior BBB. The focus of this research is on implicit exterior BBs builders because it is the thought that this particular type of BBB is just as good, if not better, than any other type of builder.

Algorithm 35 MOMGA-II algorithm

```
procedure MOMGA-II($f_k(\vec{x})$)
    for i = 1 to epoch do
                                                                ▶ PCI Phase
        Perform Probabilistically Complete initialization
        Evaluate each pop member's fitness w.r.t.$k$ templates
                                         ▶ Building Block Filtering (BBF) Phase
        for i = 1 to Max BBF generations do
            if BBF schedule requires cutting at this generation then
                Perform BBF
            else
                Perform Tournament Thresholding Selection
            end if
        end for
                                                       ▶ Juxtapositional Phase
        for i = 1 Max Juxtapositional generations do
            Cut-and-Splice
            Evaluate each population member's fitness w.r.t.$k$ templates
            Perform Tournament Thresholding Selection and fitness Sharing
            $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
        end for
        Update $k$ templates             ▶ Using best known value in each objective
    end for
end procedure
```

Appendix F. Comparison of explicit building block MOEAs

This appendix is dedicated to describing and comparing explicit BBBs. Statistically, the mBOA and MOMGA-IIa are shown to effectively be the similar on deception problems; however, these two explicit BBBs classify BBs differently. A comparison of the mBOA to the MOMGA-IIa can be found within Section F.5.1 on page 372. Single objective explicit BBBs and implicit BBBs are discussed in Appendix E on page 324. A brief discussion of the MOMGA, MOMGA-II(a), IMOEA, and probabilistic model building algorithm are given before the main comparison between the multiobjective probabilistic model genetic algorithm and other multiobjective explicit BBBs is given. This appendix's main objective is to present a complete description of today's state-of-the-art explicit BBBs and delineate differences between each.

F.1 Multiobjective messy GA (MOMGA)

The Multiobjective messy GA (MOMGA) is a multiobjective implementation of the messy GA. It works in three phases: initialization to build a pool of BBs, primordial phase for finding important BBs, and the juxtapositional phase for combining these BBs to form optimal or near-optimal solutions [94, 216]. In the primordial phase, partial strings are initialized. However, it differs from the original messy GA in that MOMGA uses multiple competitive templates, each corresponding to an individual objective function. MOMGA begins with random templates and then finds the best templates for each objective from the best solutions obtained at the end of each era/epoch, thereby finding the competitive template for each objective for an era/epoch, a matter which is important for proper evaluation of partial solutions in an era/epoch. Tournament selection and cut-and-slice operators are also inherited from the original messy GAs. Finally, the dominance measure used for selection and diversity-preserving mechanism of NPGA is used; the resulting algorithm has both the desired properties of a multiobjective optimizer. This algorithm has also been suggested as a parallel MOEA. In fact, if the cardinality of this set is not the same as the desired population size, other individuals from the offspring population can be included. This strategy shows that this algorithm must convergence to the Pareto-optimal set given enough time. A parallel version of MOMGA lacks the second task of maintaining diversity of Pareto-optimal solutions (either within the genotype or phenotype domain). Thus, an explicit diversity-preserving mechanism should be added to the MOMGA to make it more usable in practice.

This MOEA is a generational explicit BBB. The pseudocode for MOMGA is presented in Algorithm 34 on page 362.

Algorithm 36 MOMGA-II algorithm

```
procedure MOMGA-II($f_k(\vec{x})$)                              ▷ Solve $f_k(\vec{x})$
    for h = 1 to epoch do
                                                                 ▷ PCI Phase
        Perform Probabilistically Complete initialization
        Evaluate each pop member's fitness w.r.t.k templates
                                                  ▷ BB Filtering (BBF) Phase
        for i = 1 to Max BBF generations do
            if BBF schedule requires cutting at this generation then
                Perform BBF
            else
                Perform Tournament Thresholding Selection
            end if
        end for
                                                    ▷ Juxtapositional Phase
        for i = 1 Max Juxtapositional generations do
            Cut-and-Splice
            Evaluate each population member's fitness w.r.t.k templates
            Perform Tournament Thresholding Selection and fitness Sharing
            $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
        end for
        Update $k$ templates          ▷ Using best known value in each objective
    end for
end procedure
```

F.2 Multiobjective fast messy GA MOMGA-II and MOMGA-IIa

A brief introduction to these *explicit* BBBs is given in Sections 2.10.5 on page 60 and 2.10.6 on page 61. The next appendix is spent exclusively on a brief history of MOMGA, detailed history of design of MOMGA-II and a detailed design description of MOMGA-IIa. Thus, this section is vacant of a discussion about the details of MOMGA-II and MOMGA-IIa. However, the pseudocode for the algorithms is given in Algorithms 36 on page 367 and 37 on page 369. MOMGA, MOMGA-II and MOMGA-IIa are alike in many ways. Each is an explicit BBB that has three phases within. The difference is that the phases of the MOMGA differ slightly from the phases of MOMGA-II and MOMGA-IIa, plus, MOMGA-IIa has an extra phase. Table 65 presents the corresponding phases between these three MOEAs.

Table 65: Listed is each phase of MOMGA and the corresponding phases in MOMGA-II and MOMGA-IIa.

MOMGA Phase		MOMGA-II/MOMGA-IIa Phase
	⟶	/Preparation
PEI	⟶	PCI
Primordial	⟶	BB Filtering
Juxtapositional	⟶	Juxtapositional

Phases for these MOEAs are renamed due to a redesign brought about by the high population size requirement of MOMGA. MOMGA is originally based upon the mGA, and it inherited the population sizing problem associated with that single objective algorithm.

F.3 Intelligent Multiobjective EA (IMOEA)

Intelligent Multiobjective Evolutionary Algorithm (IMOEA) incorporates elitism with a capacity of $N_{\mathcal{E}_{max}}$ to maintain diversity and improve the performance of the IMOEA. The IMOEA is similar to the IEA in that a Latin Square of combinations is created to produce children from one IGC operation; however, within the IMOEA,

Algorithm 37 MOMGA-IIa algorithm

```
procedure MOMGA-IIa($f_k(\vec{x})$)                                   ▶ Solve $f_k(\vec{x})$
    for h = 1 to epoch do
                                                                      ▶ PCI Phase
        Perform Probabilistically Complete initialization
        Evaluate each pop member's fitness w.r.t.($k * \hat{r} + \hat{\imath} + \hat{o}$) templates
                                                          ▶ BB Filtering (BBF) Phase
        for i = 1 to Max BBF generations do
            if BBF schedule requires cutting at this generation then
                Perform BBF
            else
                Perform Tournament Thresholding Selection
            end if
        end for
                                                             ▶ Juxtapositional Phase
        for i = 1 Max Juxtapositional generations do
            Cut-and-Splice
            Evaluate each pop member's fitness w.r.t.($k * \hat{r} + \hat{\imath} + \hat{o}$) templates
            Perform Tournament Thresholding Selection and fitness Sharing
            $P_{Known}(t) = P_{current}(t) \cup P_{known}(t-1)$
        end for
        Update $k * \hat{r}$ templates ▶ Using the Competitive Template Management
            System
        Filter $\hat{\imath}$ and $\hat{o}$ templates based on $k * \hat{r}$ updated templates
    end for
end procedure
```

multiple objective values representing each member makes for different selection and mating operators. Moreover, the IMOEA's selection and mating operators result in the increase from two to eight offspring - including two parents, two children, and four by-products.

The IMOEA is tested against SPEA, SPEA-II, NSGA, NSGA-II, NPGA, VEGA using the MOPs ZDT [135]. The IMOEA is illustrated visually to outperform all the competitor MOEAs on these test problems in [102]. Furthermore, this algorithm is shown to work well without linkage learning. However, parameter encoding to lower the degree of epistasis can help the results of the IMOEA. Finally, the IMOEA is a generational *explicit* BB MOEA, with a unique design where it does

Algorithm 38 IMOEA

```
procedure IMOEA($\mathbb{P}'$, $f_k(\vec{x})$)                    ▶ $f_k(\vec{x})$
    Randomly generate population $\mathbb{P}'$ members
    Create empty elite set $\mathcal{E}$
    Create empty temporary elite set $\mathcal{E}'$
    repeat
        Evaluate each pop member w.r.t.each objective function
        Re-assign fitness values for each member using GPSIFF
        Add the solutions evaluating to non-dominated vectors in $\mathcal{E}'$ to $\mathcal{E}$
        Empty $\mathcal{E}'$
        Remove dominated members in $\mathcal{E}$
        if ($size(\mathcal{E}) > N_{\mathcal{E}_{max}}$) then
            Randomly remove members in $\mathcal{E}$ until $size(\mathcal{E}) == N_{\mathcal{E}_{max}}$
        end if
        Select $\mathbb{P}' - \mathbb{P}'_{ps}$ individuals from the population using binary tournament
            selection and randomly select $\mathbb{P}'_{ps}$ individuals from $\mathcal{E}$ to form a new
            population, where $\mathbb{P}'_{ps} = \mathbb{P}'_p * p_s$.
        if ($\mathbb{P}' > N_{\mathcal{E}}$) then
            $\mathbb{P}' = N_{\mathcal{E}}$
        end if
        for (Each IGC operation on the $\mathbb{P}' * p_c$ selected parents) do
            Add individuals evaluating to non-dominated vectors derived from by-
                products OA combinations (by-products) and two children to $\mathcal{E}'$
        end for
        Apply mutation operation with $p_m$ to the population.
    until Termination Criteria Met
end procedure
```

not specifically hunt for linkages within the chromosome in order to find optimal solutions. IMOEA's pseudocode can be found in Algorithm 38.

F.4 Multiobjective Probabilistic model building GAs

Multiobjective Probabilistic model building Genetic Algorithms (MOPMBGAs) are few in the field of evolutionary computation. Much like the single objective equivalent, MOPMBGAs are motivated by an idea of building a probabilistic model of the population to preserve important (good) BBs in the subsequent generations. One such approach is the Multiobjective Bayesian Optimization Algorithm (mBOA) where the BOA is placed inside the NSGA-II for probabilistic model building before the population building purposes. Selection is still performed by the NSGA-II. In this section, a discussion of the mBOA is given.

Algorithm 39 mBOA algorithm

```
procedure mBOA($\mathcal{N}, g, f_k(\vec{x})$)
    Initialize Population $\mathbb{P}'$ where $|\mathbb{P}'| = \mathcal{N}$
    Begin
        Generate random population - size $\mathcal{N}'$
        Evaluate Objective Values
        Assign Rank (level) Based on Pareto dominance - sort
        Generate Child Population
           Binary Tournament Selection
           Recombination and Mutation
    End
    for t = 1 to g do
        for each Parent and Child in Population do
            Assign Rank (level) based on Pareto - sort
            Generate layers of sets vectors that are non-dominated
            Loop (inside) by adding solutions to next generation starting from the
                first front until $\mathcal{N}'$ individuals found determine crowding distance
                between points on each front
        end for
        Select points (elitist) on the lower front (with lower rank) and are outside
            a crowding distance
        Create next generation
            Binary Tournament Selection
            Recombination and Mutation
    end for
end procedure
```

F.5 Multiobjective Bayesian Optimization Algorithm (mBOA)

The mBOA was also used to solve deception MOPs in previous research [121]. mBOA is identical to the single objective Bayesian Optimization Algorithm (BOA) [121] minus the selection procedure. The mBOA's selection procedure is replaced by the non-dominated vector sorting and selection mechanism of NSGA-II. The BOA generates a child population of size $\mathcal{N}'$ from a parent population. The child and parent population is then merged and the combined population is Pareto ranked. Based on the Pareto ranking and crowding distance function, a new population is created from which BOA builds a new probabilistic model to generate children again. The mBOA pseudocode is presented in Algorithm 39 on page 371. Furthermore, the mBOA is a generational MOEA having an *explicit* BBB flavor.

F.5.1 MOPMBGA and Explicit MOBBB Statistical Similarities. Statistical similarities between multiobjective probabilistic model building algorithms and explicit BBBs must be accomplished empirically through testing and evaluation of solutions. Two such experiments exist where direct comparisons of final results from these algorithms exist. The first is within this document, where in Chapter V on page 169, a discussion of the mBOA and the MOMGA-IIa is used to solve deception problems with high epistasis. In this example, MOMGA-IIa and mBOA both find all PF$_{true}$ vectors successfully; however, MOMGA-IIa finds a larger number of duplicate vectors representing the same final Pareto front vectors but having a different genotype makeup. Table 66 lists the number of duplicates found. The duplicate-finding ratios are given below each MOP. For each experiment, MOMGA-IIa finds many more genotypically unique duplicate optimal solutions representing the same Pareto front vector set than the mBOA. In some respects, it can be said that MOMGA-IIa performed better on these test problems; however, the overall results show that both perform equally well in finding the Pareto front vectors - not in the number of optimal solutions. Thus, the effectiveness of these MOEAs are statistically similar for solving the deception problems T1T2, T3T4, and T5T2.

The second example is from [102] where a comparison of several EAs is presented. BOA is within the group of EAs tested. BOA, in Ho et ed. [102] finding is found to be extremely computationally expensive. Solutions found by BOA are only superior to those of SGA, OEGA, AGA, and SAGA. Furthermore, BOA takes, on average, about $1,115.95sec$ for a single run on the functions presented in Table 67, while the other EAs take less than $1sec$ using the CPU AMD 800 MHz. It

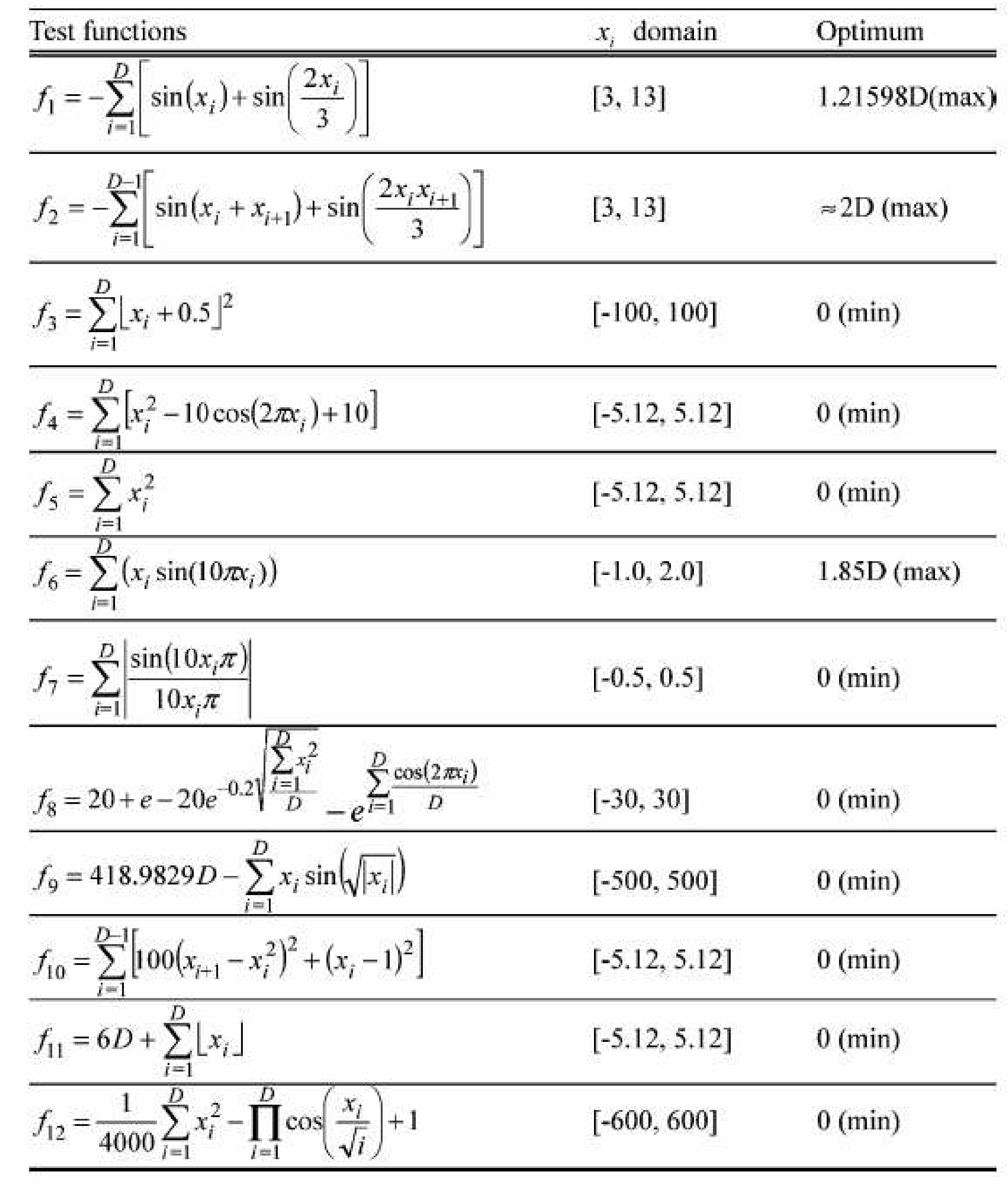
Table 66: Number of genotypically unique optimal solutions evaluating to duplicate Pareto front vectors (PF$_{true}$) for the Deception problems using the MOMGA-IIa and mBOA.

MOEA	T1T2 30	T1T2 90	
MOMGA-IIa	**32768**	**57661**	
mBOA	224	591	
Dupe Ratios (IIa to mBOA)	**146.3 to 1**	**97.6 to 1**	
	T3T4 30	T3T4 60	T3T4 90
MOMGA-IIa	**64**	**565**	**1280**
mBOA	30	102	327
Dupe Ratios (IIa to mBOA)	**2.1 to 1**	**5.5 to 1**	**3.9 to 1**

reveals that BOA takes much longer to solve large parameter optimization problems (LPOPs). This is a serious limitation when considering that EAs are generally used to solve NPC problems in a short period of time. Table 68 lists the results of Ho's testing.

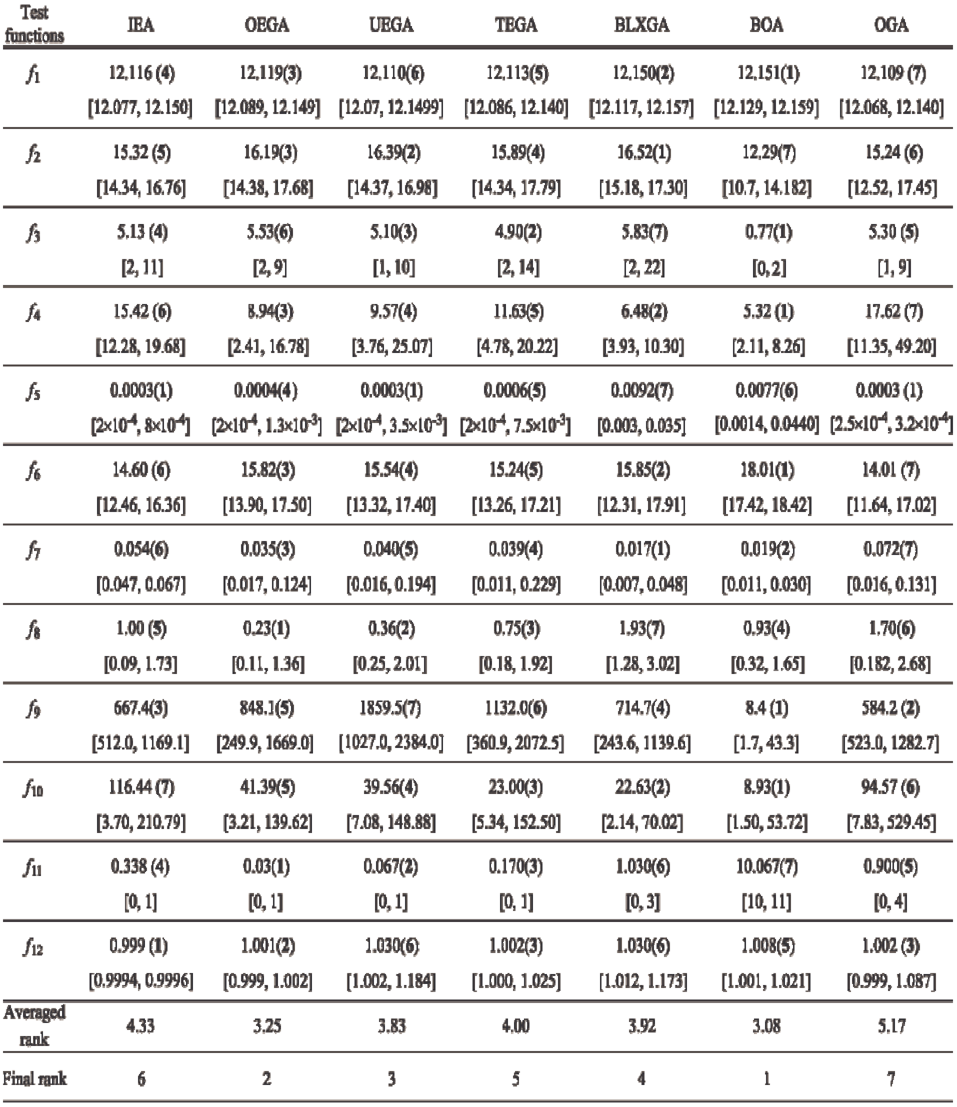
By these experimental examples, the BOA is considered a good explicit BBB if ample time exists to solve a problem. In fact, it is shown to be ranked 1st when solving all the benchmark functions selected by Ho. However, it is also illustrated that other explicit BBBs perform similarly on many problems, as well - at lower computational cost. This leads a researcher to identify that the top-down BB approach of the BOA is effectively statistically similar, over time, to the bottom-up approaches of the IEA and MOMGA-IIa. It may even be the case that the bottom-up approaches (IEA and MOMGA-IIa) might be, on average, more efficient than the top-down approach of the BOA; however, this is offered as a conjecture - not a proof. It should be emphasized that although these BBBs do perform statistically similarly, there is a definite difference in the manner in which they find good BBs. These differences are presented in Chapter II on page 19, where each BBB is identified as having a

Table 67: Benchmark functions used by [102] in his investigation on EA performance on LPOPs.



Test functions	x_i domain	Optimum
$f_1 = -\sum_{i=1}^{D}\left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right]$	[3, 13]	1.21598D(max)
$f_2 = -\sum_{i=1}^{D-1}\left[\sin(x_i + x_{i+1}) + \sin\left(\frac{2x_i x_{i+1}}{3}\right)\right]$	[3, 13]	≈2D (max)
$f_3 = \sum_{i=1}^{D}\lfloor x_i + 0.5\rfloor^2$	[-100, 100]	0 (min)
$f_4 = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12, 5.12]	0 (min)
$f_5 = \sum_{i=1}^{D} x_i^2$	[-5.12, 5.12]	0 (min)
$f_6 = \sum_{i=1}^{D}(x_i \sin(10\pi x_i))$	[-1.0, 2.0]	1.85D (max)
$f_7 = \sum_{i=1}^{D}\left\lvert\frac{\sin(10x_i\pi)}{10x_i\pi}\right\rvert$	[-0.5, 0.5]	0 (min)
$f_8 = 20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}} - e^{\sum_{i=1}^{D}\frac{\cos(2\pi x_i)}{D}}$	[-30, 30]	0 (min)
$f_9 = 418.9829D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]	0 (min)
$f_{10} = \sum_{i=1}^{D-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	[-5.12, 5.12]	0 (min)
$f_{11} = 6D + \sum_{i=1}^{D}\lfloor x_i\rfloor$	[-5.12, 5.12]	0 (min)
$f_{12} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]	0 (min)

Table 68: Lists a summary of EA performance on LPOPs [102].



Test functions	IEA	OEGA	UEGA	TEGA	BLXGA	BOA	OGA
f_1	12.116 (4)	12.119(3)	12.110(6)	12.113(5)	12.150(2)	12.151(1)	12.109 (7)
	[12.077, 12.150]	[12.089, 12.149]	[12.07, 12.1499]	[12.086, 12.140]	[12.117, 12.157]	[12.129, 12.159]	[12.068, 12.140]
f_2	15.32 (5)	16.19(3)	16.39(2)	15.89(4)	16.52(1)	12.29(7)	15.24 (6)
	[14.34, 16.76]	[14.38, 17.68]	[14.37, 16.98]	[14.34, 17.79]	[15.18, 17.30]	[10.7, 14.182]	[12.52, 17.45]
f_3	5.13 (4)	5.53(6)	5.10(3)	4.90(2)	5.83(7)	0.77(1)	5.30 (5)
	[2, 11]	[2, 9]	[1, 10]	[2, 14]	[2, 22]	[0, 2]	[1, 9]
f_4	15.42 (6)	8.94(3)	9.57(4)	11.63(5)	6.48(2)	5.32 (1)	17.62 (7)
	[12.28, 19.68]	[2.41, 16.78]	[3.76, 25.07]	[4.78, 20.22]	[3.93, 10.30]	[2.11, 8.26]	[11.35, 49.20]
f_5	0.0003(1)	0.0004(4)	0.0003(1)	0.0006(5)	0.0092(7)	0.0077(6)	0.0003 (1)
	[2×10^{-4}, 8×10^{-4}]	[2×10^{-4}, 1.3×10^{-3}]	[2×10^{-4}, 3.5×10^{-3}]	[2×10^{-4}, 7.5×10^{-3}]	[0.003, 0.035]	[0.0014, 0.0440]	[2.5×10^{-4}, 3.2×10^{-4}]
f_6	14.60 (6)	15.82(3)	15.54(4)	15.24(5)	15.85(2)	18.01(1)	14.01 (7)
	[12.46, 16.36]	[13.90, 17.50]	[13.32, 17.40]	[13.26, 17.21]	[12.31, 17.91]	[17.42, 18.42]	[11.64, 17.02]
f_7	0.054(6)	0.035(3)	0.040(5)	0.039(4)	0.017(1)	0.019(2)	0.072(7)
	[0.047, 0.067]	[0.017, 0.124]	[0.016, 0.194]	[0.011, 0.229]	[0.007, 0.048]	[0.011, 0.030]	[0.016, 0.131]
f_8	1.00 (5)	0.23(1)	0.36(2)	0.75(3)	1.93(7)	0.93(4)	1.70(6)
	[0.09, 1.73]	[0.11, 1.36]	[0.25, 2.01]	[0.18, 1.92]	[1.28, 3.02]	[0.32, 1.65]	[0.182, 2.68]
f_9	667.4(3)	848.1(5)	1859.5(7)	1132.0(6)	714.7(4)	8.4 (1)	584.2 (2)
	[512.0, 1169.1]	[249.9, 1669.0]	[1027.0, 2384.0]	[360.9, 2072.5]	[243.6, 1139.6]	[1.7, 43.3]	[523.0, 1282.7]
f_{10}	116.44 (7)	41.39(5)	39.56(4)	23.00(3)	22.63(2)	8.93(1)	94.57 (6)
	[3.70, 210.79]	[3.21, 139.62]	[7.08, 148.88]	[5.34, 152.50]	[2.14, 70.02]	[1.50, 53.72]	[7.83, 529.45]
f_{11}	0.338 (4)	0.03(1)	0.067(2)	0.170(3)	1.030(6)	10.067(7)	0.900(5)
	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 3]	[10, 11]	[0, 4]
f_{12}	0.999 (1)	1.001(2)	1.030(6)	1.002(3)	1.030(6)	1.008(5)	1.002 (3)
	[0.9994, 0.9996]	[0.999, 1.002]	[1.002, 1.184]	[1.000, 1.025]	[1.012, 1.173]	[1.001, 1.021]	[0.999, 1.087]
Averaged rank	4.33	3.25	3.83	4.00	3.92	3.08	5.17
Final rank	6	2	3	5	4	1	7

different type of BB classifier rule (see Table 4 on page 64 for the list of types of classifiers identified for each BBB).

F.6 MOEA Summary

A good summary of specifications for the SOEAs examined is in Appendix E on page 324 and listed in Table 63 on page 347. Also, a summary of specification for implicit and explicit MOEAs discussed in this appendix is given in Tables 69 and 64. The trend for each is that implicit BBBs came first, followed closely behind by explicit BBBs. The coding complexity of algorithms using explicit search techniques seems to be more difficult than that of the implicit methods. Furthermore, more structural emphasis on managing partial solutions and determining goodness of these partial solutions can become a maintenance nightmare; however, EAs like the mGA, fmGA, IEA and other MOEAs based on the structure of these have found ways to manage this complexity problem. In addition, explicit BB builders have numerous methods for seeking these good BBs. Researchers that design BB builders must be aware of the two major ways to search for good BBs: *interior* and *exterior.* An interior BB search first finds a good complete solution and then segments that solution seeking for the good BBs within. An exterior BB search builds BBs and subjects these BBs to tests for goodness, normally by placing them within another complete solution. The thought process behind having an interior BB search begins with an implicit BB search, which is rather like having a mixture of both implicit and explicit interior BB search algorithms.

The difference between implicit and explicit BBBs is in how these builders seek BBs within a chromosome. The implicit or *implicitus* BBBs use EVOPs in hopes of shuffling bits together in a manner to create BBs within the entire chromosome. This is an implied BB recognized only within the chromosome but not directly identified as a BB by the algorithm. BBs in an implicit BBB are capable of being recognized but they are unexpressed, whereas, an explicit BBB fully and clearly expresses the BBs

Table 69: Summary of EVOPs, fitness, sharing, and representation for discussed *explicit* BB MOEAs.

MOEA	**EVOPS**	**Fitness**	**Sharing**	$\mathbb{R}$ or $\{0,1\}$
MOMGA	cut-splice	Tournament	Phenotypic (σ_{share} - Fitness)	$\{0,1\}$
MOMGA-II	cut-splice	Tournament	Phenotypic (σ_{share} - Fitness)	$\{0,1\}$
MOMGA-IIa	cut-splice	Tournament	Phenotypic (σ_{share} - Fitness)	$\{0,1\}$
IMOEA	div-comb	MED	Phenotypic (weighted σ_{share} - Fitness)	$\{0,1\}$
mBOA	none	Bayesian Dirichlet (BD)	Phenotypic (σ_{share} - Fitness)	$\{0,1\}$

leaving nothing implied. Hence, each BB that is found can be presented separately, outside of the full chromosome, where it was overlaid to make a good solution that evaluated to a non-dominated vector.

F.7 Summary

This appendix discusses a broad variety of MOEAs. For completeness, the rooted SOEAs are given, as well, to illustrate the design development process and thoughts between having a single objective algorithm and making it a multiobjective algorithm. In addition, these algorithms are classified with respect to the difference being either an implicit or explicit BBB. Plus, if the algorithm is an implicit BBB, it is an interior or exterior BBB. The focus of this dissertation is on implicit exterior BBBs because it is thought that this particular type of BBB is just as good as any other type of builder in finding good BBs.

Appendix G. Future Work

This appendix is provided to give insight to future design modifications and applications that can be used in conjunction with MOEA research.

G.1 MOMGA-IIa selection operator change

MOMGA-IIa stores all values associated with each competitive template; however, it only uses the best of these values when determining selection. A study of selecting either based on an averaging these values for each objective when selecting the best BB within the tournament selection might show to improve the overall performance of the MOMGA-IIa.

G.2 BB deception avoidance

MOMGA-IIa has three types of competitive templates. Each embodies what you want from an evolutionary algorithm: exploration and exploitation (in both genotype and phenotype domains). However, there may be problems structured to take advantage of certain weaknesses of this design (if there are any). Let's call this problem, if it exists, a BB deception problem. As described by Dr Peterson, a BB deception problem is a problem where two solutions, having different genotypes, map to the exact same phenotypic vector. To make this problem different than the deception problems already solved within this research, let us also say that two solutions mapping to the same phenotypic vector do not have linkages in common (even shifted linkages). The MOMGA-IIa would keep track of the second incoming solutions mapping to the same phenotypic vector if the duplicate flag is set on or weak dominance is used within. Although it is conjectured that this problem can still easily be solved by an unmodified MOMGA-IIa, to ensure these types of solutions are kept around for later BB searching on that solution a new mechanism to keep a new type of competitive template would need to be added. The new competitive template might be called the shadow template because it evaluates to the same phenotypic vector as an already stored template and thus defeating the fabled BB deception problem.

G.3 Sorting of Non-dominated vectors

The MOMGA-IIa Pareto front structure advances the state-of-the-art in explicit BBBs by sorting according to objective values as solutions are found; however, the search must stop while the sort commences. A suggested alternative implementation using a parallel sorting node is given here to allow the search algorithm to continue without a hiccup when processing new solutions for target vector slot assignment and non-dominance. One such implementation might use a resident Common Object Request Broker Architecture (CORBA) Object Request Broker (ORB)[1] handle all sorting and incoming solution/vectors.

Another BB search technique using the DFT is described here, but not tested. This method is good at finding inter-bit linkages.

G.4 Linkage Learning within any Genetic Algorithm

Learning the linkages is important when solving problems using genetic algorithms. In fact, linkage learning is considered to be a large concern of most MOEA researchers. Even for single objective evolutionary algorithms linkage learning is challenging; however, when you add multiple dimensions (objectives) to the prob-

[1] The Object Request Broker (ORB) provides a mechanism for transparently communicating client requests to target object implementations. It simplifies distributed programming by coupling the client from the details of the method invocations. Client requests then appear to be local procedure calls. The ORB is responsible for finding the object implementation where the client has invoked an operation, transparently activating it if necessary, delivering the request to the object, and returning any response to the caller. See `http://www.cs.wustl.edu/~schmidt/corba-overview.html` for more details.

lem, linkages for each dimension can make identification of patterns difficult. When difficult to learn, problems are said to have high epistasis[2].

The following method is the first ever proposed using a Discrete Fast Fourier Transform (DFFT) as a device to identify both inter-bit and intra-bit patterns.

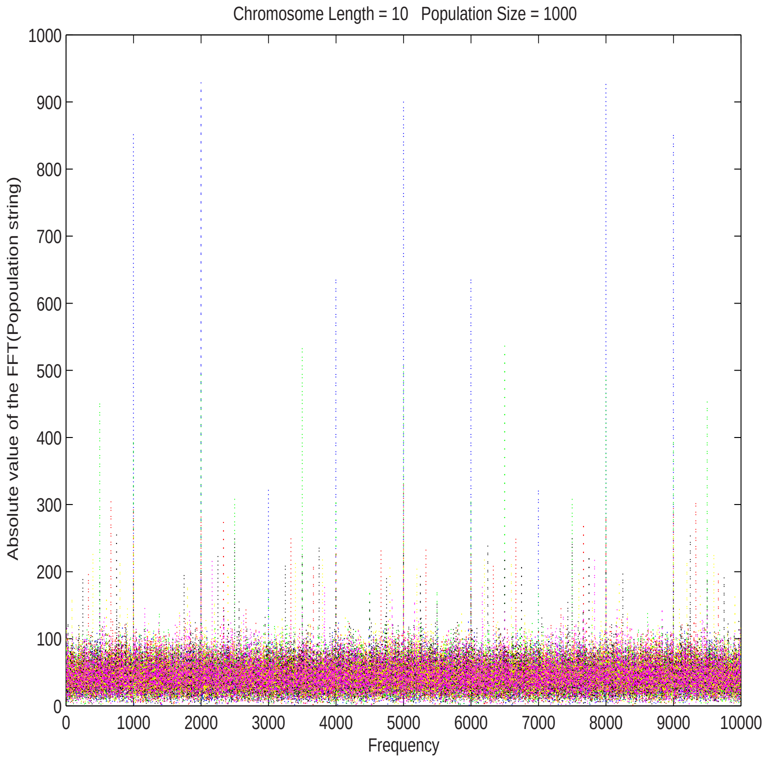


Figure 108: The plot illustrates the FFT results of a randomly generated population of 100 chromosomes of length 10.

Finding Inter-bit Linkages:

The linkage identification method is similar for both the inter-bit and intra-bit linkage detection; however, a modification of the population to find linkages is necessary. The inter-bit linkage identification method (FFT-IFFT method) is as follows:

[2] An interaction between nonallelic genes, especially an interaction in which one gene suppresses the expression of another - `www.dictionary.com`.

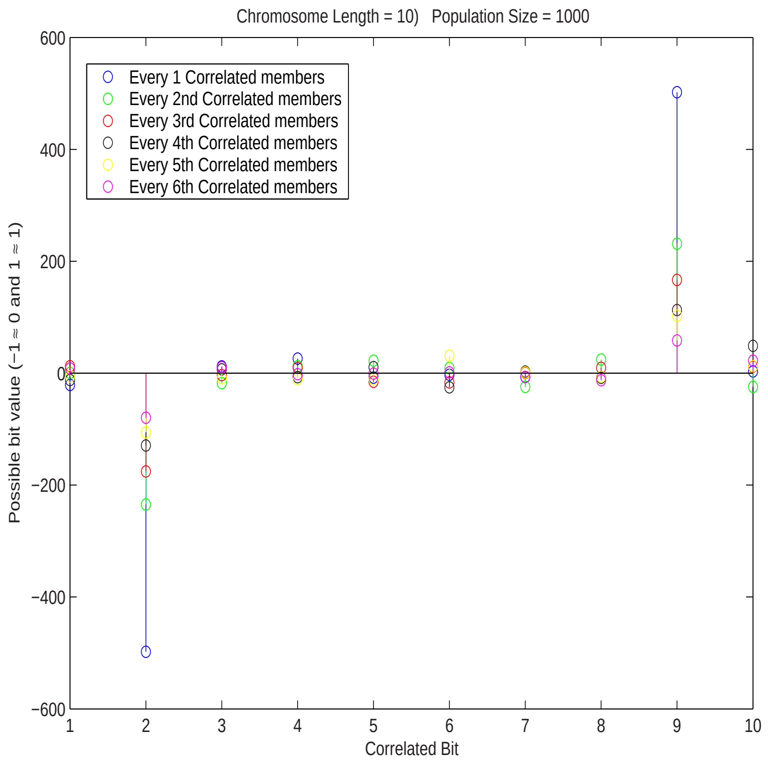


Figure 109: The plot illustrates the results of the inverse FFT after selecting values having meaningful interactions.

$$\mathbb{P} = \left\{ \begin{array}{ccc} b_{0,0}, & : & \cdots, b_{0,n} \\ b_{\mathcal{N},0}, & : & \cdots, b_{\mathcal{N},n} \end{array} \right\} \tag{101}$$

$$\bar{\mathbb{P}} = \left\{ \; b_{0,0}, \quad : \quad \cdots, b_{0,n}, \cdots, \cdots, b_{\mathcal{N},0}, \cdots, b_{\mathcal{N},n} \; \right\} \tag{102}$$

Correlations are identified by the values of the bit positions in FFT-IFFT method breaking a predetermined or learned correlations threshold value, φ. An example of an expected output from an FFT of $\bar{\mathbb{P}}$ is given in Figure 108. Moreover, an example of expected output from the IFFT of the new vector $\vec{nv}$ is given in

FFT-IFFT method

1. The entire population, $\mathbb{P}$, is lined up in one long vector, $\bar{\mathbb{P}}$. (Eq. 101)
2. An FFT is taken of the vector, $\bar{\mathbb{P}}$. (Eq. 102)
3. Every $j * n$ value from the FFT is made into a new vector, $\vec{nv}$ where $j = 1, \cdots, \mathcal{N}$
4. An Inverse FFT (IFFT) is then taken of the new vector, $\vec{nv}$
5. A threshold is determined to identify if a correlation exists.

Figure 109. Finally, degradation of frequency of correlated bits and results of the IFFT are given in Figure 110.

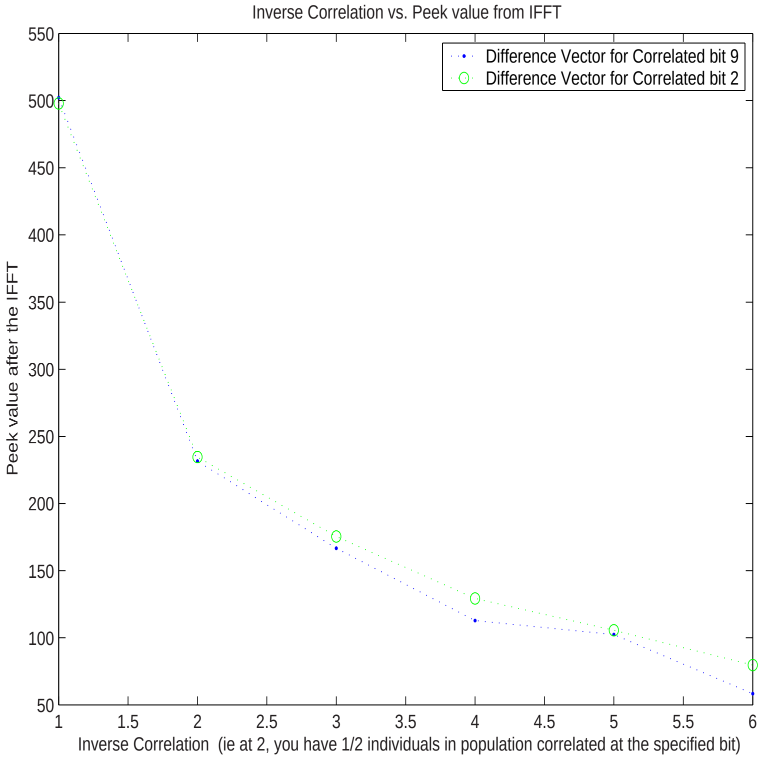
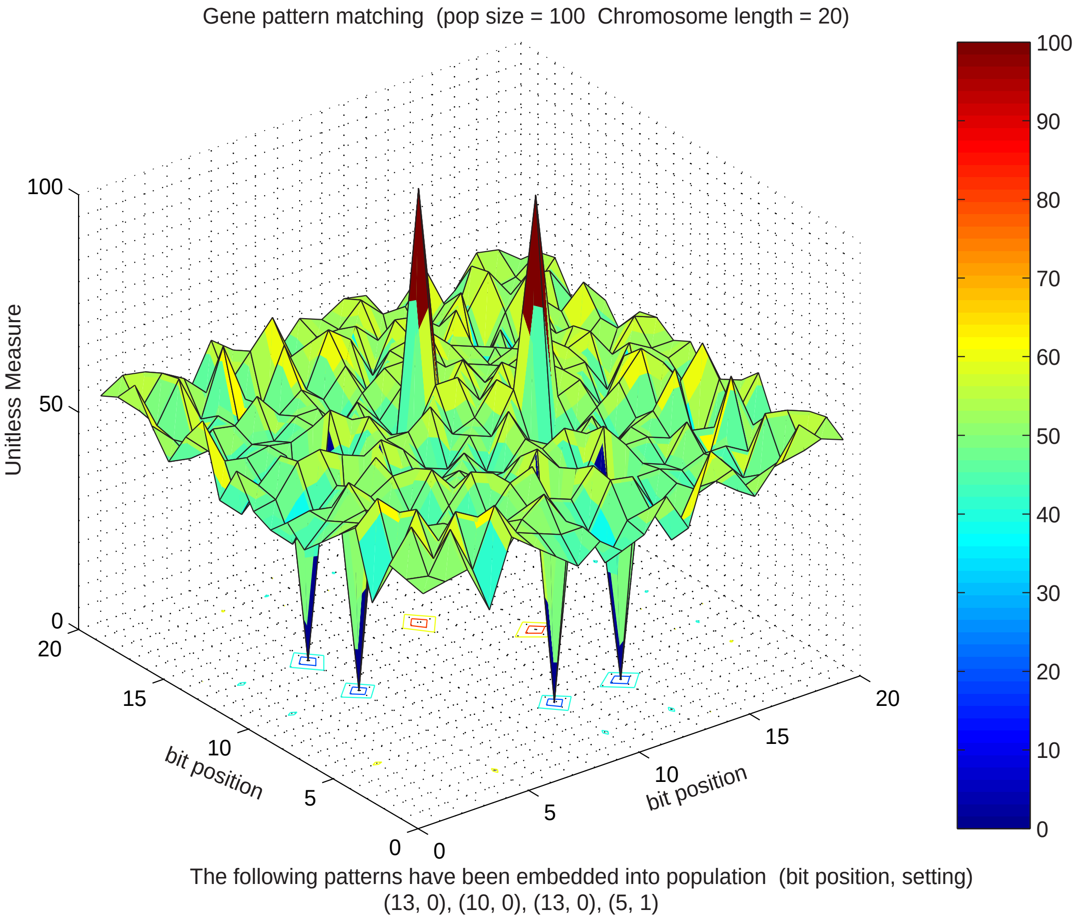


Figure 110: This plot illustrates the degradation of IFFT values resulting 100% and less values associated with particular values in a population of chromosomes.

Finding Intra-bit Linkages:

A modification of the population to find linkages is accomplished in the form of Xoring each bit of each member with itself. The linkages are then identified by


Gene pattern matching (pop size = 100 Chromosome length = 20)

The following patterns have been embedded into population (bit position, setting)
(13, 0), (10, 0), (13, 0), (5, 1)

Figure 111: This plot illustrates the capabilities of the Xor-FFT-IFFT method

transposing the matrix and the IFFT of the FFTed vectorized transposed matrix. The inter-bit linkage identification method (Xor-FFT-IFFT method) is as follows:

Figure 111 illustrates an example of the resultant matrix found using the Xor-FFT-IFFT method. Note that the row/columns of the matrix identify the bits interacting with each other and the matrix does not identify what the bits are set too. This can be found using a mean value method on the original population, $\mathbb{P}$, for one bit – the rest can be derived after one correlated bit setting is identified.

Drawbacks to this approach are mainly in the fact that extra computation may be unnecessary because there may not be any linkages to find. Also determining the

Xor-FFT-IFFT method

1. Each member within the population is Xored with each of its own bits
   This generates a new population of $\mathcal{N}^2$ x m bits, $\mathbb{P}'$.
2. The population, $\mathbb{P}'$, is then transposed, $\mathbb{P}$".
3. The transposed population, $\mathbb{P}$", is then lined up in one long vector, $\bar{\mathbb{P}}$". (Eq. 101)
4. An FFT is taken of the vector, $\bar{\mathbb{P}}$". (Eq. 102)
5. Every $j * m$ value from the FFT is made into a new vector, $\vec{xv}$
   where $j = 1, \cdots, \mathcal{N}^2$
6. An Inverse FFT (IFFT) is then taken of the new vector, $\vec{xv}$
7. The result of the IFFT is then placed into an adjacency matrix.
8. A threshold is determined to identify if a correlation exists[3].

threshold at which decide that bits are correlated may be a point of contention or problem specific.

G.5 Future Fitness Function Replacement

Future consideration for replacing the fitness function of the PSP problem should look into more suitable neural networks like the bipolar neural network that specifically can be trained to emulate spike filled landscapes such as what the PSP problem has. In addition, additional models or revalidation of the current PSP model may need to be addressed. Possible algorithm domain alternatives may be the BOA or the mBOA; however, beware of the computational time associated with running such an algorithm on this type of difficult problem.

G.6 Future Work m-ary signal symbol set design

Since this is a new angle on an old problem, many different techniques to increase the effectiveness of MOMGA-IIa in solving this problem. Right now, the model needs to be modified to yield maximum fitness value at the same constellation that the Monte Carlo simulation validates to be the lowest P_b. So far, a better model escapes us. Our current models show little correlation to the current simulation model. Once a model is found for finding good constellations for use on signals subject to additive White Gaussian Noise, a new model can be derived to find constellations for noisy signals having other types of noise. In fact, the ultimate goal would be to have a channel noise probe identify the noise over a channel and back propagates this information to an EA that searches for an optimal symbol set for that channel. Once the symbol set was defined for the unknown noise in the channel, communicators would have the best symbol set for that digital system.

G.7 Publications

The research completed in support of this document has led to several publications [41, 44–53, 126, 127, 145, 146] in four different problem domain areas. These publications are of interest to the Air Force as they illustrate an efficient and effective means of solving real-world Air Force MOPs that when used in conjunction with other approaches. Other real-world applications having different characteristics (different dimensionality: $\{2, 3, \cdots, k\}$ and/or fitness landscapes {rugged, smooth, multimodal, $\cdots$} or pareto front characterizations { disconnected, concave, discrete,$\cdots$}).

G.8 Application Overview

This section describes different applications where MOEA research can be applied. Discussion includes numerous pedagogical and real-world applications. Examples of past and present applications being solved using MOEAs are the following: Modified Multiobjective Knapsack Problem, Advanced Logistics Problem, Protein Structure Prediction Problem, Organic Air Vehicle (OAV) problem, Deception problem, MOEAs on Field Programmable Gate Array (FPGAs), Sensor Management, Digital Amplitude-Phase Keying with M-ary Alphabets, Signal Identification Problem, Signal Jamming Problem, and Network Security to name a few.

G.8.1 Modified Multiobjective Knapsack Problem. The Modified Multiobjective Knapsack Problem (MMKP)is modeled from the single objective knapsack problem. The single objective knapsack problem is a classical problem where an Airman is faced with filling his chemical bag with items – maximizing the total value of items carried in the bag. To make the problem more difficult, each item has a specified weight and value associated with it; furthermore, the chemical bag also has a specific capacity which it cannot exceed [37]. The MOP formulation of this problem has a variable number of bags and each item must be placed in all or none of the knapsacks – thereby keeping with military uniformity so each bag has exactly the same contents. Zydallis used the MOMGA-II to solve this problem in 2003 [244].

G.8.2 Advanced Logistics Problem. The Advanced Logistics Problem (ALP) is a real-world problem involving logistics research in resource allocation. According to [124], ALP is developing advanced information technology to support planning, execution, monitoring and replanning throughout the logistics pipeline, enabling the warfighter to project and sustain overwhelming combat power sooner and with less reliance on large Department of Defense (DoD) inventories. This research is advantageous to Battlefield Awareness Data Dissemination (BADD)[4] and has been solved using MOEA technology [244].

G.8.3 Protein Structure Prediction Problem. The Protein Structure Prediction (PSP) problem is a Grand Challenge problem among biochemists, computer scientists and engineers alike. Solving this problem involves correctly predicting the geometrical conformation of a fully folded protein. Past research focused on CHARMm energy minimization and the use of a genetic algorithm, fast messy genetic algorithm (fmGA), to obtain good solutions to this optimization problem. Research continued in this field by applying a Multiobjective approach to solving this problem using a modified fast messy GA - *MOEA*. By dividing the CHARMm energy model into separate objectives, it is shown in [41] that an MOEA can find

[4] BADD is providing warfighters at echelons from the Task Force Commander down to Battalion level or lower, and especially mobile warfighters, with advanced battlefield awareness applications that are driven by near-real time data that is delivered by advanced data dissemination methods.

structural configurations of a protein that yield lower energy values and ultimately more correct conformations.

G.8.4 Organic Air Vehicle (OAV) requirements. The Organic Air Vehicle requirement problem maps nicely to the Multiobjective Quadratic Assignment Problem (mQAP). The mQAP is an NP-complete problem with a multitude of real-world applications. The specific application addressed in this paper is the minimization of communication flows in a heterogenous mix of unmanned aerial vehicles. Developed is a multiobjective approach to solving the general mQAP for this UAV application. The combinatoric nature of this problem calls for a stochastic search algorithm; moreover, the multiobjective fast messy Genetic Algorithm (MOMGA-IIa) [43, 243] is used for experimentation. Results indicate that many of the pareto optimal points are found; however, again the MOMGA-II is limited by the archive and competitive template generation technique. The MOMGA-IIa shows better results overall.

G.8.5 Deception Problem. Deception problems are among the hardest problems to solve using ordinary genetic algorithms. Recent studies show that Bayesian optimization can help in solving problems such as these. This investigation compares the results acquired from the new multiobjective fast messy genetic algorithm (MOMGA-IIa), multiobjective fast/messy genetic algorithm (MOMGA-II), multiobjective Bayesian optimization algorithm (mBOA), and the non-dominated sorting genetic algorithm-II (NSGA-II) when applied to three different deception problems. The three deceptive problems are: interleaved minimal deceptive problem, interleaved 5-bit trap function, and interleaved 6-bit bipolar function. Problem sizes are increased to show how the MOMGA-IIa can scale up to solve even larger problems more easily than the MOMGA-II.

G.8.6 MOEAs on FPGAs Problem. Placing an MOEA on a Field Programmable Gate Array (FPGA) for the purposes of solving a problem has interest in many fields. Sensor Management, Signal Identification, Signal Jamming, and UAV coordination control are just a few that would require this type of integration. However, the integration task for putting an MOEA on a FPGA is not easy. Areas of concern are memory availability, computational power and mapping of problem to a discretized computer problem via digital logic design for a computer chip. The technology trend of chips is moving fast and support for larger programs is here today; however, much technology is still required to make this effort a reality.

G.8.7 Sensor Management. Sensor management is a large field. UAVs having autonomous control and coordination techniques to accomplish a mission need sensor management modules to govern fuel consumption, battery conservation, compromises between mission success and loss of equipment, coordinated intelligence gathering, effective formation patterns, enemy assault and target recognition. The entirety of the problem can be consolidated into a discretized computer solvable problem for an MOEA to solve; however, this is still a wide open optimization problem and field of study for MOEA application.

G.8.8 Digital Amplitude-Phase Keying with M-ary Alphabets. Signal sets employing amplitude and phase keying (APK) with large alphabets conserve bandwidth and do not require as high a signal-to-noise ratio (SNR) as phase-shift keying (PSK). Optimum designs are sought for alphabets containing 4 to 128 symbols. Designs are based on symbol-error-probability bound for both average and peak SNR. To lower symbol-error-probability a design placing symbols at farther distances from one another is best.

By means of an error probability bound, APK signal sets have been compared in an empirical search for the optimum design as a function of alphabet size. For all alphabet sizes greater than 4, new designs have been presented [213] that outperform previously proposed sets on the basis of both peak SNR and average SNR. For Number of Symbols $>$ 8, APK offers an advantage in average and peak SNR relative to PSK that increases with alphabet size. Given an accurate model to optimize, MOEAs can be used to design these symbol sets to be placed in optimal positions to acquire comparatively good peak and average SNR values. This problem is solved using the MOMGA-IIa; however, the goodness of solutions to the problem are limited by the model design – not due to the limitation of the algorithm. Further model development is required in this area; however, results presented are comparable to the rectangular symbols set positions used in everyday digital communication systems.

G.8.9 Signal Identification Problem. There is military interest in identifying M-ary signals for exploitation. Signals are normally broadcasted in Phase or Amplitude Shift keying modes. In addition, these signals can be shifted in N increments to provide more or less transmission bandwidth. Exploiting enemy transmission, on the fly, can be advantageous to the air force for intelligence gathering. To exploit a intercepted transmission the signal must be identified as either a Phase or Amplitude shift keying signal; then, the signal bandwidth or bit level must be identified. Much study has been accomplished on identifying each component of a transmitted signal. A researcher using each of these pattern recognition techniques to identify a transmitted signal while maximizing correlation of the transmitted signal with the expected results of each type of test for signal identification, this problem then becomes a pattern classification problem which can be solved by an MOEA. Again, this problem is wide open to MOEA research.

G.8.10 Signal Jamming Problem. Signal jamming interests military operations to prevent communication between enemy units (be they ground units or aircraft operators/equipment). Stand alone devices built primarily for the purpose of matching and then jamming an enemy signal in the field is not unimaginable. Field Programmable Gate Arrays can be integrated with other components to *small* programs in an attempt to match and then jam these signals. MOEA FPGA integration can become involved in this process; however, the MOEA must be designed with the limited memory and signal adaptation in mind. This problem area too is open for MOEA research.

G.8.11 Network Security. Attacks against computer networks today are sophisticated. Adversaries are exploiting weaknesses by using new attacks or modifying existing ones. Past research used two types of multiobjective approaches, lexicographic and pareto-based, in an evolutionary programming algorithm to develop a new method for detecting these attacks. MOEA development can extend the Computer Defense Immune System: an artificial immune system for virus and computer intrusion detection. The approach *vaccinates* the system by evolving antibodies as finite state transducers to detect attacks; this technique may allow the system to detect attacks with features similar to known attacks. Testing indicates that the algorithm performs satisfactorily in generating finite state transducers capable of detecting attacks. Further testing of this algorithm can include a MOEA having a preprocessing mechanism for the developing of antibodies. A study of this kind remains an area needing to be studied.

Appendix H. Multiobjective Discussion [216]

This appendix is used mainly for a brief discussion of Pareto terminology. The more detailed discussion can be found in [216] where this is drawn.

H.1 Multiobjective Optimization

The method for finding the global maximum or minimum of any function is referred to as Global Optimization. In general, this is presented in Definition 27 as stated in Bäck [4]:

Definition 27 (Global Minimum): *Given a function $f : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq \emptyset$, for $\vec{x} \in \Omega$ the value $f^* \triangleq f(\vec{x}^*) > -\infty$ is called a global minimum if and only if*

$$\forall \vec{x} \in \Omega : \quad f(\vec{x}^*) \leq f(\vec{x}) \; . \tag{103}$$

Then, $\vec{x}^*$ is the global minimum solution(s), f is the objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution(s) is called the global optimization problem. □

This formulation must be modified to reflect the nature of multiobjective problems where there may not be one unique solution but a set of solutions found through the analysis of associated Pareto Optimality Theory. Many times multiobjective problems force the decision maker to make a choice which is essentially a tradeoff of one solution over another in objective space.

Multiobjective problems are those where the goal is to optimize k objective functions simultaneously. This may involve the maximization of all k functions, the minimization of all k functions or a combination of maximization and minimization of these k functions. A MOP and a MOP global minimum (or maximum) is formally defined by Van Veldhuizen as [216]:

Definition 28 (MOP Global Minimum): *Given a function* $F : \Omega \subseteq \mathbb{R}^\ell \rightarrow \mathbb{R}^k$, $\Omega \neq \emptyset$, $k \geq 2$, *for* $\vec{x} \in \Omega$ *the set* $\mathcal{PF}^* \triangleq F(\vec{x}_i^*) > (-\infty, \ldots, -\infty)$ *is called the global minimum if and only if*

$$\forall \vec{x} \in \Omega : \;\; F(\vec{x}_i^*) \preceq F(\vec{x}) \; . \tag{104}$$

Then, $\vec{x}_i^*$ *is the global minimum solution set (*i.e., $\mathcal{P}^*$*),* F *is the multiple objective function, and the set* Ω *is the feasible region. The problem of determining the global minimum solution set is called the MOP global optimization problem.* □

This MOP consists of k objectives reflected in the k objective functions, m constraints on the objective functions and n decision variables. The k objective functions may be linear or nonlinear in nature. The evaluation function, $F : \Omega \longrightarrow \Lambda$, is a mapping from the decision variables ($\vec{x} = x_1, \ldots, x_n$) to output vectors ($\vec{y} = a_1, \ldots, a_k$) [216].

It is necessary to define additional terminology to remain consistent with the terminology used in the EA field. The term *objective* is used to refer to the goal of the MOP to be achieved and *objective space* is used to refer to the coordinate space within which vectors resulting from the MOP evaluation are plotted [216].

H.1.1 Pareto Terminology. The concept of Pareto Optimality is integral to the theory and analysis of MOPs. A way to determine if one solution is *better* than another is a necessity as well as in all problems. Pareto concepts allow for the determination of a set of optimal solutions in MOPs. Although single-objective optimization problems may have a unique optimal solution, MOPs usually have a possibly uncountable set of solutions, which when evaluated produce vectors whose components represent trade-offs in decision space. Some key Pareto concepts, for minimization MOPs, are defined mathematically by Van Veldhuizen as [216]:

Pareto optimal solutions are those solutions within the search space whose corresponding objective vector components cannot be all simultaneously improved. These solutions are also termed *non-inferior*, *admissible*, or *efficient* solutions, with the entire set represented by $\mathcal{P}^*$. Their corresponding vectors are termed *non-dominated*; selecting a vector(s) from this vector set (the Pareto front set $\mathcal{PF}^*$) implicitly indicates acceptable Pareto optimal solutions, decision variables or genotypes. These solutions may have no clearly apparent relationship besides their membership in the Pareto optimal set. It is simply the set of all solutions whose associated vectors are non-dominated; it is stressed that these solutions are classified as such based on their *phenotypical* expression. Their expression (the non-dominated vectors), when plotted in criterion phenotype or objective space, is known as the *Pareto front* [216, 246].

A MOEA's complex structure can lead to confusion in discussing the algorithmic process that takes place. To prevent further inconsistencies in discussions of MOEAs, Van Veldhuizen [216] developed Pareto terminology to clarify MOEA discussions. He stated at any given generation of a MOEA a "current" set of Pareto optimal solutions (with respect to the *current* MOEA generational population) exists and is termed $P_{current}(t)$, where t represents the generation number. There are also a number of MOEAs that use a secondary population, also referred to as an archive or an external archive, to store non-dominated solutions found through the generations [216, 218]. Since this secondary population contains Pareto optimal solutions generated at a certain point in time, each time another point is considered for addition to the secondary population, the point must be looked at for non-dominance with respect to the points currently in the secondary population. This secondary population is denoted $P_{known}(t)$. The t reflects the potential changes to the secondary population as the MOEA executes. Additionally, $P_{known}(0)$ is defined as the empty set ($\emptyset$) and P_{known} alone as the *final* set of Pareto optimal solutions returned by the MOEA at termination [216, 246].

Different secondary population storage strategies exist; the simplest is when $P_{current}(t)$ is added at each generation (i.e., $P_{current}(t) \bigcup P_{known}(t-1)$). At any given time, $P_{known}(t)$ is thus the set of Pareto optimal solutions *yet found by the MOEA* *through generation t*. Of course, the *true* Pareto optimal solution set (termed P_{true}) is not explicitly known for problems of any difficulty. P_{true} is defined by the functions composing an MOP; it is fixed and does not change. Because of the manner in which Pareto optimality is defined $P_{current}(t)$ is always a non-empty solution set [216].

$P_{current}\,(t)$, P_{known} , and P_{true} are sets of MOEA genotypes where each set's phenotypes form a Pareto front. The associated Pareto front terms for each of these solution sets is $PF_{current}\,(t)$, PF_{known} , and PF_{true} . Thus, when using an MOEA to solve MOPs, the implicit assumption is that one of the following holds: $P_{known} = P_{true}$, $P_{known} \subset P_{true}$, or PF$_{known} \in PF^{\epsilon}_{true}$ over some norm (Euclidean, RMS, etc.).

Solutions on the Pareto Front represent optimal solutions in the sense that improving the value in one dimension of the objective function vector leads to a degradation in at least one other dimension of the objective function vector. This forces the decision maker to make a tradeoff decision when presented with a number of optimal solutions for the MOP at hand, i.e. the Pareto Front. There exists a difference in terminology between an acceptable compromise solution and a Pareto Optimal Solution [77]. The decision maker typically chooses only one of the associated Pareto Optimal solutions, $\vec{u} \in \mathcal{PF}^*$, as being the acceptable compromise solution, even though all of the Pareto Optimal solutions are optimal. The decision maker bases this solution choice off of which solutions take into account the human's preference. The human preference factor forces engineers and scientists to attempt to find all of the points on the Pareto front since all points are not weighted equally in the decision maker's mind.

Definition 29 (Pareto Front width distribution): *The width of the Pareto front created by the Pareto epsilon Dominance factor is described by the Pareto front width distribution. By placing 3D Gaussian distributions (Parzon Windows) on each vector on a Pareto epsilon front, a distribution can be illustrated having a multidimensional Gaussian Distribution Characteristics. See Definition 6 on page 12 for a definition of Pareto epsilon dominance.* □

Appendix I. Additional metrics

This appendix is added to describe some of the metrics/indicators that are not used to describe the merit of MOEA found approximation sets. These additional sections are not an attempt to list each and every metric available to an MOEA research, but only to list the other metrics considered.

I.1 R1 Indicator

The R1 metric calculates the probability that an approximation set A is better than a set B over a set of utility functions, $\mathbf{U}$, and R1R is identical to R1 when it is used with a reference set [88].

$$R1(A, B, \mathbf{U}, p) = \left\{ \int_{u \in \mathbf{U}} C(A, B, u) p(u) du, \ \textit{subject to Equation } 105 \right\}$$

$$\ddot{}\,_{u}(\tilde{A}, B\!\!\!C)= \begin{cases} 1 & : \ if\ u(A) > u(B) \\ 1/2 & : \ if\ u(A) = u(B) \\ 0 & : \ if\ u(A) < u(B) \end{cases} \tag{105}$$

Let A and B be two approximation sets, U is a set of utility functions, $u : \mathbb{R}^k \mapsto \mathbb{R}$. This function maps each point in objective space into a measure of utility, $p(u)$ is the probability density of the utility $u \in \mathbf{U}$, and $u(A) = max_{z \in A}\{u(z)\}$ and also for $u(B)$. Joshua Knowles suggests that the R1 indicator requires a set of utility functions which must be defined. Recently, Fonseca et ed's presented a method for defining the utility function for R1 at EMO [129]. Furthermore, Zydallis stated that an indicators such as R_R use low computational resources and can differentiate between different levels of complete out performance if given a reference set. In addition he stated that these indicators are somewhat complex to understand and require the use and determination of utility functions, reducing the attractiveness of the metric. [244]

I.2 Two Set Coverage (CS)

The two set coverage metric is an MOEA comparison metric which can be called a relative coverage comparison of two approximation sets. Consider A, $B \subset A$ as two sets of phenotype decision vectors. `CS` is defined as the mapping of the order pair (A, B) to the interval [0,1] according to equation 106.

$$\texttt{CS}(A, B) \triangleq \frac{|\{a'' \in B; \exists a' \in A : a' \succeq a''\}|}{|B|} \tag{106}$$

If all solutions in A evaluate to vectors that are (weak) non-dominated by vectors resulting from evaluating solutions in B, then by definition `CS` = 1. `CS` = 0 implies the opposite. In general, `CS`(A, B) and CS(B,A) both have to be considered due to set intersections not being empty. This metric can be used for A = P$_{known}$ or PF$_{known}$. The advantage of this metric is that it is easy to calculate and provides a relative comparison based upon dominance numbers between generations of MOEAs. Observe that it is not a distance measure of how close these set are as that is a different metric. [31]

I.3 Distributed Spacing (ι)

A measure expressing how well an MOEA maintained a distributed set of PF$_{known}$ vectors over a non-dominated region is (ι). This metric is defined as:

$$\iota \triangleq \left(\sum_{i=1}^{\hat{q}+1} (\frac{\jmath_i - \bar{\jmath}_i}{\sigma_i})^2 \right)^{\frac{1}{2}} \tag{107}$$

where $\hat{q}$ is the number of desired non-dominated vectors and the $(\hat{q}+1)^{th}$ subregion is the dominated region, $\jmath_i$ is the number of individuals evaluating to vectors in the i^{th} subregion (niche) of the non-dominated region, $\bar{\jmath}_i$ is the expected number of individuals evaluating to vectors in the i^{th} subregion of the non-dominated region, and σ_i^2 is the variance of individuals evaluating to vectors serving the i^{th} subregion of the non-dominated region. They show that if the distribution of points is ideal with $\bar{\jmath}_i$ number of points in the i^{th} subregion, the performance measure $\iota = 0$. Thus, a low performance measure characterizes an algorithm with a good distribution capacity. This metric may be modified to measure the distribution of vectors within the Pareto front. In that case both metrics (S and ι) then measure only uniformity of vector distribution and thus complement the generational distance and maximum Pareto front error metrics.

I.4 Generational Non-dominated Vector Generation (GNVG)

This metric tracks how many non-dominated vectors are produced at each MOEA generation and is defined as:

$$GNVG \triangleq |\mathrm{PF}_{current}(t)| \tag{108}$$

I.5 Non-dominated Vector Addition (NVA)

As globally non-dominated vectors are sought, one hopes to add new non-dominated vectors (that may or may not dominate existing vectors) to PF_{known} each generation. This metric is then defined as:

Appendix J. Parallel Computing

This appendix is provided to give an explanation and references for other parallel computing models used within the MOEA field. These models are not unique and well known throughout the field.

J.1 Parallel Models

Another advantage to using the new design of strings of characters and a vector of integers plus allocating the storage of fitness values in a separate array of doubles is the ease of parallel implementation. Furthermore, it has been shown that MOEA efficiency improves when using any one of the parallel models [222]. The following three models are possible parallel implementations of the MOMGA-IIa: Island model, Master-slave model, and diffusion model.

J.1.1 Island model. The island model is based on the setting where islands in the ocean are close, but out of direct evolutionary contact. However, because the islands are close, once in while exchange of species occurs. The pMOEA model resembles this by separately evolving EA populations on separate nodes with the random or generational exchange of percentage of good population members. Island models can be implemented on any cluster type architecture including distributed or shared memory clusters. Communication occurs in logical or physical geometric structures like rings, meshes, toruses, triangles, and hypercubes.

Within the island model each processor (or island) in this paradigm, executes a MOEA simultaneously with other processors. A single sub-population evolves on each processor. On occasion, processors randomly exchange or send the best solutions (or BBs) to other islands based on the neighborhood structure. Each processor evolves a population using either the same or a different MOEA. After completion of generations on each island, populations are combined and the best population member (or BB) is presented as the solution. The five different types of island models are the following: homogeneous, heterogeneous, a combination of different optimization approaches, a partitioning of the genotype region, and finally, a different distribution or layout scheme of the new competitive template target vector scheme discussed in this chapter. The first four listed are covered in depth in Zydallis' dissertation [244].

J.1.2 Farming (Master-Slave) model. The farming model code modification is motivated by having the need for two parallel models working as one. The farming model is a simple dynamic load balancing implementation of the juxtaposition phase's evaluation function. Upon start-up, the fmGA initializes a pool of processors that is used in parallel to evaluate the fitness function of all new partial population members during phase of the algorithm. The only stipulation is that the total number of compute nodes must divide evenly by the number of algorithm nodes. For example, in the Figure 112 each Algorithm node is represented by a square, and circles represent compute nodes. The configuration on the left is showing a configuration of one Algorithm node (which runs the fmGA) and three compute nodes. When this Algorithm node reaches the cut and splice phase within the juxtaposition phase, it builds the new population, and then dynamically (according to the currently population size) divides up the evaluations between the three compute nodes, all left over evaluations are performed by the Algorithm Node. The configuration on the right side of Figure 112 shows how the communication occurs between Algorithm nodes as well as from Algorithm node to Compute nodes. This communication is essential when the Algorithm nodes are working together from a common population. In addition, there is another stipulation that no two Algorithm nodes can have a common compute node.

The farming model, built in code using Message Passing Interface (MPI), follows the visual representation discussed above. The most important part is initializing the groups correctly. Once this is complete, all following MPI calls are the same

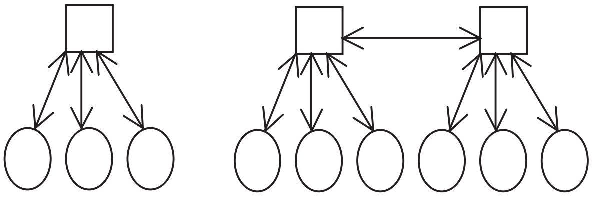
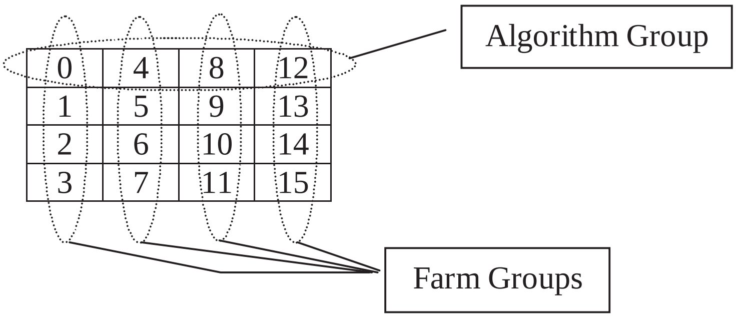


Figure 112: Farming model visualization of communication between algorithm nodes and farm or compute nodes. Algorithm nodes are represented by the square boxes and Compute nodes are the ovals.

with one modification to the "group" identifier. Normally, all sends and receives are channeled to the entire group, MPIWORLD. Now, all calls are either to the a configured Algorithm Group or Farm Group. Furthermore, each farm group can only communicate with one algorithm node. Figure 113 illustrates how the nodes are grouped. Communication only occur within a group. It is this restriction that forces the group relationships defined for this model. Further, it is easy to see that Algorithm nodes communicate only with Algorithm nodes and Farm nodes within that Algorithm node's farm group. Furthermore, farm nodes can only communicate with nodes that are within its own farm group – this always includes one Algorithm node.



0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Figure 113: Visualization of nodes grouped into Algorithm and Farm arrays.

Consequently, the farming model code is contained inside the MOfmGA and consequently does come with the MOMGA-IIa package; however, it is untested.

Algorithm 40 Initialize Farms

```
procedure INIT_FARMING_MODEL($an$,$fn$)    ▶ $an$ is number of algorithm nodes and $fn$ is the number of farming nodes for each $an$.
    NumNodes = $(an + (an * fn))$
    NumFarms = $(an * fn)$
    NumFarmsPerAlg = fn
    for ($i = 0$;$i < NumFarmsPerAlg$;$k + +$) do
        alg_proc[i] = i*(fn+1)
        if (i∗(fn+1) == this_node's_world_number) then
            my_group = i
            IamBoss = TRUE
        end if
        for ($j = 0$;$j < (fn + 1)$);$j + +$) do
            farm_proc[i].ranks[j] = i*(fn+1) + j
            if ((i*(fn+1)+j) == this_node's_world_number) then
                $j > 0$
                my_group = i
                IamFarm = TRUE
            end if
        end for
    end for
end procedure
```

J.1.3 Diffusion model. The diffusion model, sometimes referred to as the local population model, constrains the selection of parents to a local neighborhood – normally the processor on which the algorithm runs. Each individual is handled separately; selection and mating partners are always selected within the local neighborhood by local selection. A diffusion of information through the population takes place. Another model would be an overlapped diffusion model having a shared memory architecture where the memory is marked in such a way that it is overlapped in certain areas. Individuals located in the overlapped area are actually used for selection purposes in two or more algorithm contained regions. Figure 114 on page 402 presents a visual example of regionalistic chromosomes and the overlapped areas discussed above. In this model, the overlapped chromosomes require a lock bit for use when being replaced with offspring. Zydallis also has a more in depth description of the diffusion model [244].

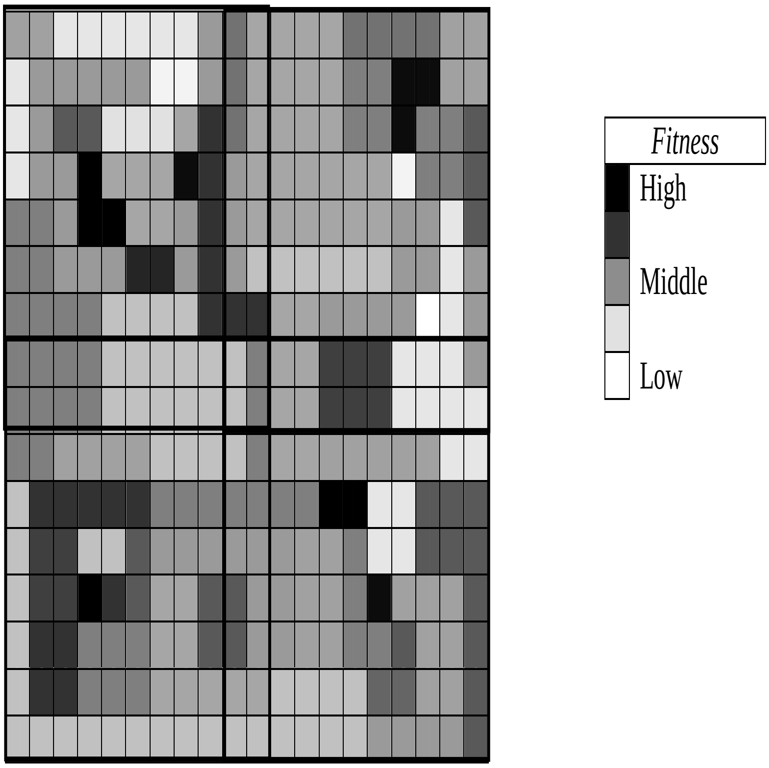


Figure 114: This figure illustrates the fitness landscape of the diffusion model where patches of good chromosomes may be. Notice that the selection regions marked by bold. In this figure, selection regions do overlap and can effect the overall evolution of the separated regions.

J.2 Summary

This appendix sole purpose is to describe MOEA parallel design already tested and used within the MOEA field. In fact most of these models are not much different than the parallel models purposed for single objective evolutionary algorithms.

Appendix K. Alternative methods for producing orthogonal arrays

This appendix is provided to present brief descriptions of methods for acquiring well distributed orthogonal vectors. Most of these methods are used for design of experiments (DOE) when deciding how many experiments to run and how to change the levels (settings) for variables for a good representation (subset) of the full factorial design that would be required if one where to test every possible experiment combination.

K.1 Full factorial

The full factorial method is one where each and every combination of the incoming vector is developed. This is the method that is too computationally cumbersome and unrealistic to use on problems having greater than 20 variables each having levels greater than 2.

K.2 Latin Square

The term Latin Square was first coined by Euler in 1782. Normally, Latin Squares are used for factors having more than two levels and interactions between factors are nonexistent. The following equation denotes L as the Latin Square operator that generates an $\hat{o}$ by ℓ matrix of values that are a series or orthogonal rows (arrays).

$$L_{\hat{o}}(|\mathcal{A}|^{\ell}) = [a_{i,j}]_{\hat{o}\ by\ \ell}$$

where the j^{th} factor in the i^{th} combination has the level $a_{i,j} \in \mathcal{A}$

For example: assume there is a problem having four factors, $\ell = 4$, and each factor has three levels: $\mathcal{A} = \{0, 1, 2\}$. Notice that the alphabet is now changed to include an extra level (*i.e.*, trinary instead of binary). The ortogonal bank of array that would be created if the user specified nine, $\hat{o} = 9$, chromosomes to be created would be the following:

$$L_9(3)^4 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix}$$

Table 70: Bank of orthogonal templates generated using Algorithm 8 with $\hat{o} = 5$, $\ell = 20$, and $Q = 2$.

$$\begin{bmatrix} 00000000000000000000 \\ 00000000000000011111 \\ 10000001111111100000 \\ 00000001111111111111 \\ 10011110000111100001 \end{bmatrix}$$

K.3 Taguchi arrays

A Japanese engineer named *Genichi Taguchi* proposed a handful of approaches to experimental designs referred to as the "Taguchi Methods." These methods utilize two-, three-, and mixed-level fractional factorial designs. One set of designs called *large screening* seemed to be particularly favorite of Taguchi. This method is called "off-line quality control" because it can be used to ensure good performance in the design stage of products or processes. Figure 115 presents a visualization of how the experiments are factored. The inner array is indicated by $\vec{v_i}$ and the outer $\vec{z_j}$ arrays for a equally partitioned factored experiment. [13]

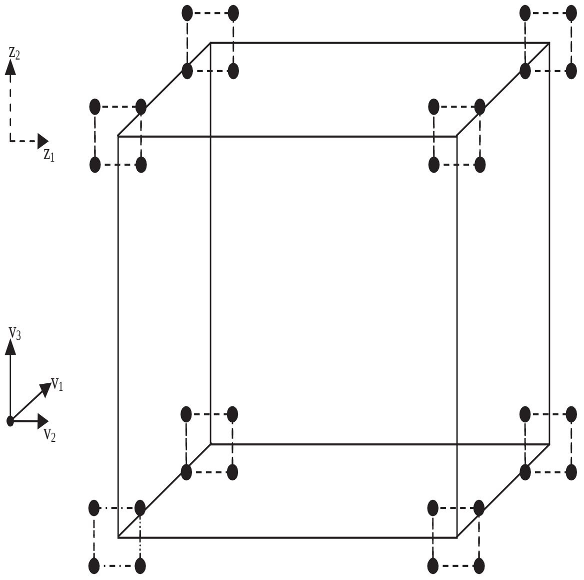


Figure 115: This figure illustrates a 8x4 experimental settings (or orthogonal vectors) using the Taguchi methods. Notice the inner and outer array. Designs like these are developed to repeatedly test noisy factors (outer array) and while testing each design factors (inner array).

Table 71: Plus and Minus Signs indicating templates used for Plackett-Burman two-level Design of Experiments.

K = *Exp*-1 vars, *Exp* runs	Plus and Minus Signs for two-level Plackett-Burman Designs
11/12	++-+++- - -+-
19/20	++- -++++-+-+- - - -++-
23/24	+++++-+-++- -++- -+-+- - - -
35/36	-+-+++- - -+++++-+++- -+- - - -+-+-++- -+-

K.4 Plackett-Burman arrays

Another common two-level fractional factorial designs via fold over are attributed to Plackett and Burman (1946) [185]. Table K.3 lists templates used by Packett and Burman for creating designs for *Exp* = 12, 20, 24, 28, and 36. To create these designs the procedure is as follows:

1. Take the template ± row from the Table for the appropriately sized run, *Exp*, and write it in a table row (or column).

2. The next row (or column) is generated from the previous one by moving the elements of the row (or column) to the right (or down) one position and placing the last element in the first position.

3. Repeat step 2 until the table is complete.

4. Finally, a row of minus signs is then added to complete the design.

Appendix L. 3-bit Examples Using BB definitions

The following 3-bit problem to illustrate the application of the definitions and the need for a fitness ranking. To begin, Figure 116 illustrates the fitness functions used in this example.

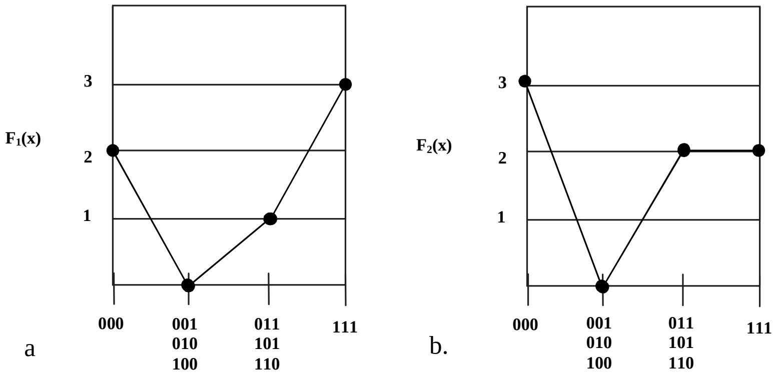


Figure 116: Presented is the 3-bit bi-objective fitness functions used in the following two examples illustrating the problem with using unmodified fitness values.

$$\begin{aligned}
\mathbb{P} &= \{000, 001, 010, 011, 100, 101, 110, 111\} && (109)\\
\mathbb{BB}^1 &= \{0xx, x0x, xx0, 1xx, x1x, xx1\} = \left\{\tilde{b}_{1,1}, \cdots, \tilde{b}_{1,6}\right\} && (110)\\
\mathbb{BB}^2 &= \{00x, 0x0, x00, 01x, 0x1, x01, 10x, 1x0, x10, 11x, 1x1, x11\} &&\\
&= \left\{\tilde{b}_{2,1}, \cdots, \tilde{b}_{2,12}\right\} && (111)
\end{aligned}$$

This is not to say that a BBB having the good BB definition 11 on page 23 combined with this fitness ranking adjustment contains the necessary components to be called the optimum BBB. The ranking fitness adjustment only ensures that if the true optimal solution is found (using a BB), that particular BB is identified as a good BB and not lost by definition. Without the fitness ranking, optimal BBs can be missed. The upcoming example in the next section illustrates the point.

L.1 Un-adjusted fitness example

Two fitness function examples have been contrived here, $F_1(\vec{x})$ and $F_2(\vec{x})$ (see Figure 116), to assist in explaining how the weighting of fitness can misguide these good BB definitions into labeling a BB as good when it is *not* really a part of building an optimal solution. This ultimately means that the BBB is misguided and cannot put its set of *identified* good BBs together to build the optimal solution or solutions.

Table 72: Mean objective 1 value for $\mathbb{BB}^1$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{1,1,1}} =$	(2+0+2+0+0+1+0+1)/8 =	0.75
$\mu_{\tilde{b}_{1,1,2}} =$	(2+0+2+0+0+1+0+1)/8 =	0.75
$\mu_{\tilde{b}_{1,1,3}} =$	(2+2+0+0+0+0+1+1)/8 =	0.75
$\mu_{\tilde{b}_{1,1,4}} =$	(0+1+1+3+0+1+1+3)/8 =	1.25
$\mu_{\tilde{b}_{1,1,5}} =$	(0+1+0+1+1+3+1+3)/8 =	1.25
$\mu_{\tilde{b}_{1,1,6}} =$	(0+0+1+1+1+1+3+3)/8 =	1.25

$\mathcal{W}_{obj,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	<, =, >	g/e/b
$\mu_{\mathcal{W}_{1,1,1}} =$	(2*0.75 + 3*1.25)/5 =	1.05	> 0.75	$\tilde{b}_{1,1}$ is bad
$\mu_{\mathcal{W}_{1,1,2}} =$	(2*0.75 + 3*1.25)/5 =	1.05	> 0.75	$\tilde{b}_{1,2}$ is bad
$\mu_{\mathcal{W}_{1,1,3}} =$	(2*0.75 + 3*1.25)/5 =	1.05	> 0.75	$\tilde{b}_{1,3}$ is bad
$\mu_{\mathcal{W}_{1,1,4}} =$	(3*0.75 + 2*1.25)/5 =	0.95	< 1.25	$\tilde{b}_{1,4}$ is good
$\mu_{\mathcal{W}_{1,1,5}} =$	(3*0.75 + 2*1.25)/5 =	0.95	< 1.25	$\tilde{b}_{1,5}$ is good
$\mu_{\mathcal{W}_{1,1,6}} =$	(3*0.75 + 2*1.25)/5 =	0.95	< 1.25	$\tilde{b}_{1,6}$ is good

- BB inspection with respect to function $F_1(\vec{x})$

The first example illustrates how to calculate and test a selected BB for goodness in $F_1(\vec{x})$ (Objective 1). Every BB of size 1 and 2 in the three-bit solution space is evaluated for goodness. In order to test for BB goodness, first a BB size, **o**, is chosen. Then a BB of size **o** is picked for inspection. Each possible BB of size $\mathbf{o} \in \{1, 2\}$ is given in Equations 110 and 111. Next, the BB under inspection is overlaid onto each and every population member. The resultant of the overlaying function is evaluated for fitness. Every population member in the three-bit solution space is given in Equation 109. The mean value, $\mu_{\tilde{b}}$, of the BB under inspection after being overlaid onto each and every population member is stored. This value is then compared to the mean fitness value, $\mu_{\mathcal{W}}$, of every other BB being overlaid onto the entire population. If the $\mu_{\tilde{b}}$ is greater than $\mu_{\mathcal{W}}$ then the BB is said to be a true good BB and a g appears in the BB's row in the table. If the means are equal, the BB is equivalent and a e appears in the BB's row within the table; otherwise the BB is bad and a b appears in

Table 73: Mean objective 1 value for $\mathbb{BB}^2$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{1,2,1}} =$	(2+0+2+2+0+0+2+0)/8 =	1.0
$\mu_{\tilde{b}_{1,2,2}} =$	(2+2+0+2+0+2+0+0)/8 =	1.0
$\mu_{\tilde{b}_{1,2,3}} =$	(2+2+2+0+2+0+0+0)/8 =	1.0
$\mu_{\tilde{b}_{1,2,4}} =$	(0+1+0+0+1+1+1+1)/8 =	0.625
$\mu_{\tilde{b}_{1,2,5}} =$	(0+0+1+1+1+0+1+1)/8 =	0.625
$\mu_{\tilde{b}_{1,2,6}} =$	(0+0+0+1+0+1+1+1)/8 =	0.5
$\mu_{\tilde{b}_{1,2,7}} =$	(0+1+0+0+1+1+0+1)/8 =	0.5
$\mu_{\tilde{b}_{1,2,8}} =$	(0+1+1+0+1+0+1+1)/8 =	0.625
$\mu_{\tilde{b}_{1,2,9}} =$	(0+0+0+1+0+1+1+1)/8 =	0.5
$\mu_{\tilde{b}_{1,2,10}} =$	(1+3+1+1+3+3+1+3)/8 =	2.0
$\mu_{\tilde{b}_{1,2,11}} =$	(1+1+3+1+3+1+3+3)/8 =	2.0
$\mu_{\tilde{b}_{1,2,12}} =$	(1+1+1+3+1+3+3+3)/8 =	2.0

$\mathcal{W}_{i,j,k}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	$<, =, >$	g/e/b
$\mu_{\mathcal{W}_{1,2,1}} =$	(2*1.0 + 3*0.625 + 3*0.5 + 3*2.0)/11 =	1.0341	> 1.0	$\tilde{b}_{2,1}$ is b
$\mu_{\mathcal{W}_{1,2,2}} =$	(2*1.0 + 3*0.625 + 3*0.5 + 3*2.0)/11 =	1.0341	> 1.0	$\tilde{b}_{2,2}$ is b
$\mu_{\mathcal{W}_{1,2,3}} =$	(2*1.0 + 3*0.625 + 3*0.5 + 3*2.0)/11 =	1.0341	> 1.0	$\tilde{b}_{2,3}$ is b
$\mu_{\mathcal{W}_{1,2,4}} =$	(3*1.0 + 2*0.625 + 3*0.5 + 3*2.0)/11 =	1.0682	> 0.625	$\tilde{b}_{2,4}$ is b
$\mu_{\mathcal{W}_{1,2,5}} =$	(3*1.0 + 2*0.625 + 3*0.5 + 3*2.0)/11 =	1.0682	> 0.625	$\tilde{b}_{2,5}$ is b
$\mu_{\mathcal{W}_{1,2,6}} =$	(3*1.0 + 3*0.625 + 2*0.5 + 3*2.0)/11 =	1.0795	> 0.5	$\tilde{b}_{2,6}$ is b
$\mu_{\mathcal{W}_{1,2,7}} =$	(3*1.0 + 3*0.625 + 2*0.5 + 3*2.0)/11 =	1.0795	> 0.5	$\tilde{b}_{2,7}$ is b
$\mu_{\mathcal{W}_{1,2,8}} =$	(3*1.0 + 2*0.625 + 3*0.5 + 3*2.0)/11 =	1.0682	> 0.625	$\tilde{b}_{2,8}$ is b
$\mu_{\mathcal{W}_{1,2,9}} =$	(3*1.0 + 3*0.625 + 2*0.5 + 3*2.0)/11 =	1.0795	> 0.5	$\tilde{b}_{2,9}$ is b
$\mu_{\mathcal{W}_{1,2,10}} =$	(3*1.0 + 3*0.625 + 3*0.5 + 2*2.0)/11 =	0.9432	< 2.0	$\tilde{b}_{2,10}$ is g
$\mu_{\mathcal{W}_{1,2,11}} =$	(3*1.0 + 3*0.625 + 3*0.5 + 2*2.0)/11 =	0.9432	< 2.0	$\tilde{b}_{2,11}$ is g
$\mu_{\mathcal{W}_{1,2,12}} =$	(3*1.0 + 3*0.625 + 3*0.5 + 2*2.0)/11 =	0.9432	< 2.0	$\tilde{b}_{2,12}$ is g

the BB's row within the table. Tables 72 and 73 represent the results of a full factorial calculation of BB sizes one and two.

This example illustrates the calculation and the fact that the function, $F_1(\vec{x})$, works nicely for the definitions as they are prescribed to work. However, the next function, $F_2(\vec{x})$, is an example where the definitions do not work as prescribed because the weight of the total fitness function misguides the definitions into identifying a BB as good when in fact it is not (*i.e.*, The identified good BB is not found within the optimal solution.)

- BB inspection with respect to function $F_2(\vec{x})$

This second example illustrates how to calculate and test a selected BB for goodness in $F_2(\vec{x})$ (Objective 2). In the same manner as the first example, every BB of size one and two in the 3-bit solution space is evaluated for goodness. Listed in Tables 74 and 75 is the mean value, $\mu_{\tilde{b}}$, found for each BB of sized o. Tables 72 and 73 list the mean value associated with every other BB, $\mu_{\mathcal{W}}$. Classification of a good, equivalent, and bad BB is done in the same manner. If the mean value of the BB under inspection is greater than that of the other same sized BBs, the BB is considered good, g. If the means are equal, the BB is equivalent, e. Finally, if the mean value of the BB under inspection is lower than the mean values of the other BBs, it is considered to be a bad, b, BB.

The major difference is presented in Tables 74 and 75 where the good and bad classified BBs do not reflect BBs that make up optimal solutions w.r.t. the fitness function $F_2(\vec{x})$. Moreover, these BBs may evaluate, on average, to a better fitness value than all other BBs of the same size; but, w.r.t.the optimal solution for the fitness function, the BBs are wrongly identified as good BBs. These wrongly identified BBs are indicated with a $^\sharp$ in Tables 74 and 75.

An interesting consequence from using these definitions in this manner is that the weighting is self correcting or appears to be self correcting. The results of every single size one BB tested are listed in Table 74. Within this table, each

Table 74: Mean objective 2 value for $\mathbb{BB}^1$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{2,1,1}} =$	(3+0+0+3+2+0+0+2)/8 =	1.25
$\mu_{\tilde{b}_{2,1,2}} =$	(3+0+3+0+0+2+0+2)/8 =	1.25
$\mu_{\tilde{b}_{2,1,3}} =$	(3+3+0+0+0+0+2+2)/8 =	1.25
$\mu_{\tilde{b}_{2,1,4}} =$	(0+2+2+2+0+2+2+2)/8 =	1.5
$\mu_{\tilde{b}_{2,1,5}} =$	(0+2+0+2+2+2+2+2)/8 =	1.5
$\mu_{\tilde{b}_{2,1,6}} =$	(0+0+2+2+2+2+2+2)/8 =	1.5

$\mathcal{W}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	<,=,>	g/e/b
$\mu_{\mathcal{W}_{2,1,1}} =$	(2*1.25 + 3*1.5)/5 =	1.4	> 1.25	$\tilde{b}_{1,1}$ is bad$^\sharp$
$\mu_{\mathcal{W}_{2,1,2}} =$	(2*1.25 + 3*1.5)/5 =	1.4	> 1.25	$\tilde{b}_{1,2}$ is bad$^\sharp$
$\mu_{\mathcal{W}_{2,1,3}} =$	(2*1.25 + 3*1.5)/5 =	1.4	> 1.25	$\tilde{b}_{1,3}$ is bad$^\sharp$
$\mu_{\mathcal{W}_{2,1,4}} =$	(3*1.25 + 2*1.5)/5 =	1.35	< 1.5	$\tilde{b}_{1,4}$ is good$^\sharp$
$\mu_{\mathcal{W}_{2,1,5}} =$	(3*1.25 + 2*1.5)/5 =	1.35	< 1.5	$\tilde{b}_{1,5}$ is good$^\sharp$
$\mu_{\mathcal{W}_{2,1,6}} =$	(3*1.25 + 2*1.5)/5 =	1.35	< 1.5	$\tilde{b}_{1,6}$ is good$^\sharp$

BB is wrongly classified as good or bad; however, as the BBs under inspection become larger (closer to the size of a complete solution), the effects of the distribution of fitness over the entire landscape becomes less influential to the evaluation on a particular BB. This *self correction* is illustrated in Table 75 where the good BBs are indeed identified as good. The reason for this is because the BB becomes close to being the fully instantiated solution and evaluation of a BB is almost like evaluating the whole solutions (minus a few bits which are provided by the complete solution used for the overlaying function). Finally, if the BB size become as large as the size of a complete solution, overlaying the BB into each population member becomes pointless because the BB is the same size as each population member. Thus, the average fitness of the BB (as a complete solution) can be had by just evaluating the BB itself. This is another reason for defining BBs as having at least one allele position unspecified.

Table 75: Mean objective 2 value for $\mathbb{BB}^2$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{2,2,1}} =$	(3+0+3+3+0+0+3+0)/8 =	1.5
$\mu_{\tilde{b}_{2,2,2}} =$	(3+3+0+3+0+3+0+0)/8 =	1.5
$\mu_{\tilde{b}_{2,2,3}} =$	(3+3+3+0+3+0+0+0)/8 =	1.5
$\mu_{\tilde{b}_{2,2,4}} =$	(0+2+0+0+2+2+2+2)/8 =	1.25
$\mu_{\tilde{b}_{2,2,5}} =$	(0+0+2+2+2+0+2+2)/8 =	1.25
$\mu_{\tilde{b}_{2,2,6}} =$	(0+0+0+2+0+2+2+2)/8 =	1.0
$\mu_{\tilde{b}_{2,2,7}} =$	(0+2+0+0+2+2+0+2)/8 =	1.0
$\mu_{\tilde{b}_{2,2,8}} =$	(0+2+2+0+2+0+2+2)/8 =	1.25
$\mu_{\tilde{b}_{2,2,9}} =$	(0+0+0+2+0+2+2+2)/8 =	1.0
$\mu_{\tilde{b}_{2,2,10}} =$	(2+2+2+2+2+2+2+2)/8 =	1.5
$\mu_{\tilde{b}_{2,2,11}} =$	(2+2+2+2+2+2+2+2)/8 =	1.5
$\mu_{\tilde{b}_{2,2,12}} =$	(2+2+2+2+2+2+2+2)/8 =	1.5

$\mathcal{W}_{i,j,k}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	$<, =, >$	g/e/b
$\mu_{\mathcal{W}_{2,2,1}} =$	(2*1.5 + 3*1.25 + 3*1.0 + 3*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,1}$ is g
$\mu_{\mathcal{W}_{2,2,2}} =$	(2*1.5 + 3*1.25 + 3*1.0 + 3*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,2}$ is g
$\mu_{\mathcal{W}_{2,2,3}} =$	(2*1.5 + 3*1.25 + 3*1.0 + 3*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,3}$ is g
$\mu_{\mathcal{W}_{2,2,4}} =$	(3*1.5 + 2*1.25 + 3*1.0 + 3*1.5)/11 =	1.3182	> 1.25	$\tilde{b}_{2,4}$ is b
$\mu_{\mathcal{W}_{2,2,5}} =$	(3*1.5 + 2*1.25 + 3*1.0 + 3*1.5)/11 =	1.3182	> 1.25	$\tilde{b}_{2,5}$ is b
$\mu_{\mathcal{W}_{2,2,6}} =$	(3*1.5 + 3*1.25 + 2*1.0 + 3*1.5)/11 =	1.3409	> 1.0	$\tilde{b}_{2,6}$ is b
$\mu_{\mathcal{W}_{2,2,7}} =$	(3*1.5 + 3*1.25 + 2*1.0 + 3*1.5)/11 =	1.3409	> 1.0	$\tilde{b}_{2,7}$ is b
$\mu_{\mathcal{W}_{2,2,8}} =$	(3*1.5 + 2*1.25 + 3*1.0 + 3*1.5)/11 =	1.3182	> 1.25	$\tilde{b}_{2,8}$ is b
$\mu_{\mathcal{W}_{2,2,9}} =$	(3*1.5 + 3*1.25 + 2*1.0 + 3*1.5)/11 =	1.3409	> 1.0	$\tilde{b}_{2,9}$ is b
$\mu_{\mathcal{W}_{2,2,10}} =$	(3*1.5 + 3*1.25 + 3*1.0 + 2*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,10}$ is g$^\sharp$
$\mu_{\mathcal{W}_{2,2,11}} =$	(3*1.5 + 3*1.25 + 3*1.0 + 2*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,11}$ is g$^\sharp$
$\mu_{\mathcal{W}_{2,2,12}} =$	(3*1.5 + 3*1.25 + 3*1.0 + 2*1.5)/11 =	1.2955	< 1.5	$\tilde{b}_{2,12}$ is g$^\sharp$

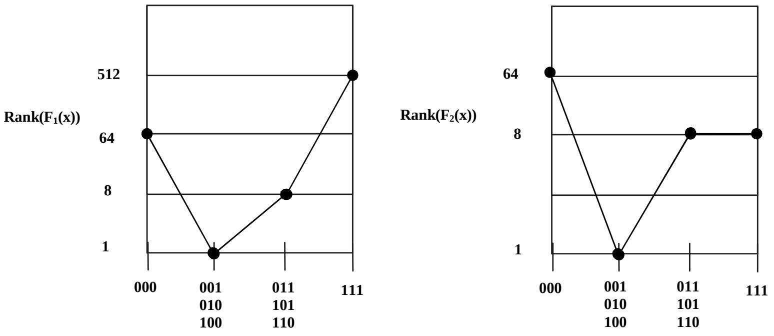


Figure 117: Presented is the 3-bit bi-objective fitness functions used in the two examples illustrating the problem having modified fitness values to a ranked fitness values.

This self-correcting adds support for use of a ranked fitness value. The next set of examples are used to illustrate the effects of using a modified fitness previously described in Equation 29 on page 50.

L.2 Adjusted fitness example

Adjusting any finite set of fitness values to a set of ranked based fitness values is not difficult. Suppose you have $\mathcal{N}'$ fitness values, having any range. Each fitness value is associated to a particular entity or population member. Then, each fitness value is ordered from best (max) to worst (min). Finally, each fitness is then mapped recursively using the following rule in Equation 29. This ranking guarantees that the better level always out-weighs any next best finite set containing at most $\mathcal{N}' - 1$ members.

The following two examples use the same 3-bit functions; however, the fitness values associated with each function are modified to reflect the ranked based fitness described by Equation 29 on page 50. Figure 117 illustrates the new fitness landscape. In the following examples, the definitions are reapplied to each BB in the same manner as it was in the previous two examples.

- BB inspection with respect to function $RF\left(F_1(\vec{x})\right)$

Table 76: Mean objective 1 value for $\mathbb{BB}^1$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{1,1,1}} =$	(64+1+1+8+64+1+1+8)/8 =	18.5
$\mu_{\tilde{b}_{1,1,2}} =$	(64+1+64+1+1+8+1+8)/8 =	18.5
$\mu_{\tilde{b}_{1,1,3}} =$	(64+64+1+1+1+1+8+8)/8 =	18.5
$\mu_{\tilde{b}_{1,1,4}} =$	(1+8+8+512+1+8+8+512)/8 =	132.25
$\mu_{\tilde{b}_{1,1,5}} =$	(1+8+1+8+8+512+8+512)/8 =	132.25
$\mu_{\tilde{b}_{1,1,6}} =$	(1+1+8+8+8+8+512+512)/8 =	132.25

$\mathcal{W}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	<,=,>	g/e/b
$\mu_{\mathcal{W}_{1,1,1}} =$	(2*18.5 + 3*132.25)/5 =	86.75	> 18.5	$\tilde{b}_{1,1}$ is bad
$\mu_{\mathcal{W}_{1,1,2}} =$	(2*18.5 + 3*132.25)/5 =	86.75	> 18.5	$\tilde{b}_{1,2}$ is bad
$\mu_{\mathcal{W}_{1,1,3}} =$	(2*18.5 + 3*132.25)/5 =	86.75	> 18.5	$\tilde{b}_{1,3}$ is bad
$\mu_{\mathcal{W}_{1,1,4}} =$	(3*18.5 + 2*132.25)/5 =	64	< 132.25	$\tilde{b}_{1,4}$ is good
$\mu_{\mathcal{W}_{1,1,5}} =$	(3*18.5 + 2*132.25)/5 =	64	< 132.25	$\tilde{b}_{1,5}$ is good
$\mu_{\mathcal{W}_{1,1,6}} =$	(3*18.5 + 2*132.25)/5 =	64	< 132.25	$\tilde{b}_{1,6}$ is good

The results of this example are found to be the same as the example where the fitness is left unmodified. If the results became different, the rank based fitness would be changing a good example to bad.

- BB inspection with respect to function $RF\left(F_2(\vec{x})\right)$

As listed in Tables 76-79, the wrongly good/bad BBs indicated by $^\sharp$ are corrected in this example. The ranked fitness function allows for the original good BB definitions to work properly. The definitions, using the ranked based fitness, identify the correct good BBs regardless of the deception present in the function.

Illustrating the Multiobjective good BB definition use, the following two examples are provided. Both examples use the ranked fitness adjustment. Furthermore, the first has the BBs of size 1 and the second as BBs of size 2.

- Order-1 MBB inspection with respect to function $RF\left(F_1(\vec{x})\right)$ and $RF\left(F_2(\vec{x})\right)$

Table 77: Mean objective 1 value for $\mathbb{BB}^2$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{1,2,1}} =$	(64+1+64+64+1+1+64+1)/8 =	32.5
$\mu_{\tilde{b}_{1,2,2}} =$	(64+64+1+64+1+64+1+1)/8 =	32.5
$\mu_{\tilde{b}_{1,2,3}} =$	(64+64+64+1+64+1+1+1)/8 =	32.5
$\mu_{\tilde{b}_{1,2,4}} =$	(1+8+1+1+8+8+8+8)/8 =	5.375
$\mu_{\tilde{b}_{1,2,5}} =$	(1+1+8+8+8+1+8+8)/8 =	5.375
$\mu_{\tilde{b}_{1,2,6}} =$	(1+1+1+8+1+8+8+8)/8 =	3.625
$\mu_{\tilde{b}_{1,2,7}} =$	(1+8+1+1+8+8+1+8)/8 =	3.625
$\mu_{\tilde{b}_{1,2,8}} =$	(1+8+8+1+8+1+8+8)/8 =	5.375
$\mu_{\tilde{b}_{1,2,9}} =$	(1+1+1+8+1+8+8+8)/8 =	3.625
$\mu_{\tilde{b}_{1,2,10}} =$	(1+512 +1+1+512 +512 +1+512)/8 =	256.5
$\mu_{\tilde{b}_{1,2,11}} =$	(1+1+512 +1+512 +1+512 +512)/8 =	256.5
$\mu_{\tilde{b}_{1,2,12}} =$	(1+1+1+512 +1+512 +512 +512)/8 =	256.5

$\mathcal{W}_{i,j,k}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	$<, =, >$	g/e/b
$\mu_{\mathcal{W}_{1,2,1}} =$	(2*32.5+ 3*5.375+ 3*3.625+ 3*256.5)/11 =	78.3182	> 32.5	$\tilde{b}_{2,1}$ is b
$\mu_{\mathcal{W}_{1,2,2}} =$	(2*32.5+ 3*5.375+ 3*3.625+ 3*256.5)/11 =	78.3182	> 32.5	$\tilde{b}_{2,2}$ is b
$\mu_{\mathcal{W}_{1,2,3}} =$	(2*32.5+ 3*5.375+ 3*3.625+ 3*256.5)/11 =	78.3182	> 32.5	$\tilde{b}_{2,3}$ is b
$\mu_{\mathcal{W}_{1,2,4}} =$	(3*32.5+ 2*5.375+ 3*3.625+ 3*256.5)/11 =	80.7841	> 5.375	$\tilde{b}_{2,4}$ is b
$\mu_{\mathcal{W}_{1,2,5}} =$	(3*32.5+ 2*5.375+ 3*3.625+ 3*256.5)/11 =	80.7841	> 5.375	$\tilde{b}_{2,5}$ is b
$\mu_{\mathcal{W}_{1,2,6}} =$	(3*32.5+ 3*5.375+ 2*3.625+ 3*256.5)/11 =	80.9432	> 3.625	$\tilde{b}_{2,6}$ is b
$\mu_{\mathcal{W}_{1,2,7}} =$	(3*32.5+ 3*5.375+ 2*3.625+ 3*256.5)/11 =	80.9432	> 3.625	$\tilde{b}_{2,7}$ is b
$\mu_{\mathcal{W}_{1,2,8}} =$	(3*32.5+ 2*5.375+ 3*3.625+ 3*256.5)/11 =	80.7841	> 5.375	$\tilde{b}_{2,8}$ is b
$\mu_{\mathcal{W}_{1,2,9}} =$	(3*32.5+ 3*5.375+ 2*3.625+ 3*256.5)/11 =	80.9432	> 3.625	$\tilde{b}_{2,9}$ is b
$\mu_{\mathcal{W}_{1,2,10}} =$	(3*32.5+ 3*5.375+ 3*3.625+ 2*256.5)/11 =	57.9545	< 256.5	$\tilde{b}_{2,10}$ is g
$\mu_{\mathcal{W}_{1,2,11}} =$	(3*32.5+ 3*5.375+ 3*3.625+ 2*256.5)/11 =	57.9545	< 256.5	$\tilde{b}_{2,11}$ is g
$\mu_{\mathcal{W}_{1,2,12}} =$	(3*32.5+ 3*5.375+ 3*3.625+ 2*256.5)/11 =	57.9545	< 256.5	$\tilde{b}_{2,12}$ is g

Table 78: Mean objective 2 value for $\mathbb{BB}^1$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{2,1,1}} =$	(64+1+1+64+8+1+1+8)/8 =	18.5
$\mu_{\tilde{b}_{2,1,2}} =$	(64+1+64+1+1+8+1+8)/8 =	18.5
$\mu_{\tilde{b}_{2,1,3}} =$	(64+64+1+1+1+1+8+8)/8 =	18.25
$\mu_{\tilde{b}_{2,1,4}} =$	(1+8+8+8+1+8+8+8)/8 =	6.25
$\mu_{\tilde{b}_{2,1,5}} =$	(1+8+1+8+8+8+8+8)/8 =	6.25
$\mu_{\tilde{b}_{2,1,6}} =$	(1+1+8+8+8+8+8+8)/8 =	6.26

$\mathcal{W}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	$<,=,>$	g/e/b
$\mu_{\mathcal{W}_{2,1,1}} =$	(2*18.5+ 3*6.25)/5 =	11.15	> 18.5	$\tilde{b}_{1,1}$ is good
$\mu_{\mathcal{W}_{2,1,2}} =$	(2*18.5+ 3*6.25)/5 =	11.15	> 18.5	$\tilde{b}_{1,2}$ is good
$\mu_{\mathcal{W}_{2,1,3}} =$	(2*18.5+ 3*6.25)/5 =	11.15	< 18.5	$\tilde{b}_{1,3}$ is good
$\mu_{\mathcal{W}_{2,1,4}} =$	(3*18.5+ 2*6.25)/5 =	13.6	> 6.25	$\tilde{b}_{1,4}$ is bad
$\mu_{\mathcal{W}_{2,1,5}} =$	(3*18.5+ 2*6.25)/5 =	13.6	> 6.25	$\tilde{b}_{1,5}$ is bad
$\mu_{\mathcal{W}_{2,1,6}} =$	(3*18.5+ 2*6.25)/5 =	13.6	> 6.25	$\tilde{b}_{1,6}$ is bad

Using the data in Tables 76 and 78, classification of each BB is accomplished. The results of each BB is plugged into the inequality cited in Equation 17, which result in the final multiobjective classification of each BB.

Notice that all BBs of size 1 are equivalent in the multiobjective formulation of these two functions. Table 80 lists the results of BBs being evaluated in the single objective case and then carries the findings over into the multiobjective case. This is important, although expected. Small BBs of each type are required for solving both objective function; therefore, identification of each is helpful for the future juxtapositioning these BBs together to build optimal solutions. In this case, an equivalent BB ultimately should be called good. The equivalent label is just to denote the fact that it is not really a totally dominate BB (good for all functions).

- Order-2 MBB inspection with respect to function $RF\left(F_1(\vec{x})\right)$ and $RF\left(F_2(\vec{x})\right)$

Table 81 presents the finding of the single and multiobjective size 2 BB inspection results for the ranked fitness functions. These results are also expected

Table 79: Mean objective 2 value for $\mathbb{BB}^2$ after being overlaid onto $\mathbb{P}$.

$\tilde{b}_{objective,BB\ size,BB\ no.}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\tilde{b}_{i,j,k}}$
$\mu_{\tilde{b}_{2,2,1}} =$	(64+1+64+64+1+1+64+1)/8 =	32.5
$\mu_{\tilde{b}_{2,2,2}} =$	(64+64+1+64+1+64+1+1)/8 =	32.5
$\mu_{\tilde{b}_{2,2,3}} =$	(64+64+64+1+64+1+1+1)/8 =	32.5
$\mu_{\tilde{b}_{2,2,4}} =$	(1+8+1+1+8+8+8+8)/8 =	5.375
$\mu_{\tilde{b}_{2,2,5}} =$	(1+1+8+8+8+1+8+8)/8 =	5.375
$\mu_{\tilde{b}_{2,2,6}} =$	(1+1+1+8+1+8+8+8)/8 =	4.5
$\mu_{\tilde{b}_{2,2,7}} =$	(1+8+1+1+8+8+1+8)/8 =	4.5
$\mu_{\tilde{b}_{2,2,8}} =$	(1+8+8+1+8+1+8+8)/8 =	5.375
$\mu_{\tilde{b}_{2,2,9}} =$	(1+1+1+8+1+8+8+8)/8 =	4.5
$\mu_{\tilde{b}_{2,2,10}} =$	(8+8+8+8+8+8+8+8)/8 =	8.0
$\mu_{\tilde{b}_{2,2,11}} =$	(8+8+8+8+8+8+8+8)/8 =	8.0
$\mu_{\tilde{b}_{2,2,12}} =$	(8+8+8+8+8+8+8+8)/8 =	8.0

$\mathcal{W}_{i,j,k}$	$\mathbb{P} = (p_1, p_2, p_3, p_4, p_5, p_6)$	$\mu_{\mathcal{W}}$	<, =, >	g/e/b
$\mu_{\mathcal{W}_{2,2,1}} =$	(2*32.5+ 3*5.375+ 3*4.5+ 3*8.0)/11 =	10.7841	< 32.5	$\tilde{b}_{2,1}$ is g
$\mu_{\mathcal{W}_{2,2,2}} =$	(2*32.5+ 3*5.375+ 3*4.5+ 3*8.0)/11 =	10.7841	< 32.5	$\tilde{b}_{2,2}$ is g
$\mu_{\mathcal{W}_{2,2,3}} =$	(2*32.5+ 3*5.375+ 3*4.5+ 3*8.0)/11 =	10.7841	< 32.5	$\tilde{b}_{2,3}$ is g
$\mu_{\mathcal{W}_{2,2,4}} =$	(3*32.5+ 2*5.375+ 3*4.5+ 3*8.0)/11 =	13.25	> 5.375	$\tilde{b}_{2,4}$ is b
$\mu_{\mathcal{W}_{2,2,5}} =$	(3*32.5+ 2*5.375+ 3*4.5+ 3*8.0)/11 =	13.25	> 5.375	$\tilde{b}_{2,5}$ is b
$\mu_{\mathcal{W}_{2,2,6}} =$	(3*32.5+ 3*5.375+ 2*4.5+ 3*8.0)/11 =	13.3295	> 4.5	$\tilde{b}_{2,6}$ is b
$\mu_{\mathcal{W}_{2,2,7}} =$	(3*32.5+ 3*5.375+ 2*4.5+ 3*8.0)/11 =	13.3295	> 4.5	$\tilde{b}_{2,7}$ is b
$\mu_{\mathcal{W}_{2,2,8}} =$	(3*32.5+ 2*5.375+ 3*4.5+ 3*8.0)/11 =	13.25	> 5.375	$\tilde{b}_{2,8}$ is b
$\mu_{\mathcal{W}_{2,2,9}} =$	(3*32.5+ 3*5.375+ 2*4.5+ 3*8.0)/11 =	13.3295	> 4.5	$\tilde{b}_{2,9}$ is b
$\mu_{\mathcal{W}_{2,2,10}} =$	(3*32.5+ 3*5.375+ 3*4.5+ 2*8.0)/11 =	13.0114	> 8.0	$\tilde{b}_{2,10}$ is b
$\mu_{\mathcal{W}_{2,2,11}} =$	(3*32.5+ 3*5.375+ 3*4.5+ 2*8.0)/11 =	13.0114	> 8.0	$\tilde{b}_{2,11}$ is b
$\mu_{\mathcal{W}_{2,2,12}} =$	(3*32.5+ 3*5.375+ 3*4.5+ 2*8.0)/11 =	13.0114	> 8.0	$\tilde{b}_{2,12}$ is b

Table 80: MBB results for $\mathbb{BB}^1$

$\tilde{b}_{objective,BB\ size,BB\ no.}$	i=1	i=2	Classification
$\tilde{b}_{i,1,1}$ is	bad	good	equivalent
$\tilde{b}_{i,1,2}$ is	bad	good	equivalent
$\tilde{b}_{i,1,3}$ is	bad	good	equivalent
$\tilde{b}_{i,1,4}$ is	good	bad	equivalent
$\tilde{b}_{i,1,5}$ is	good	bad	equivalent
$\tilde{b}_{i,1,6}$ is	good	bad	equivalent

Table 81: MBB results for $\mathbb{BB}^2$

$\tilde{b}_{objective,BB\ size,BB\ no.}$	i=1	i=2	MBB Classification
$\tilde{b}_{i,1,1}$ is	bad	good	equivalent
$\tilde{b}_{i,1,2}$ is	bad	good	equivalent
$\tilde{b}_{i,1,3}$ is	bad	good	equivalent
$\tilde{b}_{i,1,4}$ is	bad	bad	bad
$\tilde{b}_{i,1,5}$ is	bad	bad	bad
$\tilde{b}_{i,1,6}$ is	bad	bad	bad
$\tilde{b}_{i,2,7}$ is	bad	bad	bad
$\tilde{b}_{i,2,8}$ is	bad	bad	bad
$\tilde{b}_{i,2,9}$ is	bad	bad	bad
$\tilde{b}_{i,2,10}$ is	good	bad	equivalent
$\tilde{b}_{i,2,11}$ is	good	bad	equivalent
$\tilde{b}_{i,2,12}$ is	good	bad	equivalent

because the chosen functions together are classified as a deception problem. Thus, the BBs remain in the equivalent category because the solutions for the hyperplane are on the opposite end of the BB spectrum. Furthermore, a number of bad BBs are identified correctly because their patterns do not make up any of the good solutions.

Appendix M. Miscellaneous Figures and Tables

This appendix is added only to support the document and allow for non-essential graphs and tables to be added for the reader's convenience. Presented is a set of definitions describing Nondeterministic Polynomial Time Problems. Also listed are the major categories of operational research and subcategories within[1]. The word *Programming* used within Table 82 is used as *planning*; the necessary relationship to computer programming was incidental to the choice of name [40].

Definition 30 (Deterministic Turing Machine Polynomial Time Problems): A problem is assigned to the P (polynomial time) class if the number of steps is bounded by a polynomial. (i.e. the problem is in P_{time} if it has a deterministic Turing machine polynomial time solution.) □

Definition 31 (Nondeterministic Turing machine): *The word* Nondeterministic *in the definition for a Nondeterministic Polynomial Time Problems (Definition 32) is referring to the number of* parallel *Turing machine which can take many computational paths simultaneously required to evaluate every possible instance of a solutions in a particular problem space. The restriction for this is that the parallel Turing machines cannot communicate. [226]* □

Definition 32 (Nondeterministic Turing Machine Polynomial Time Problems): A problem is in NP_{time} if it has a nondeterministic Turing machine polynomial time solution; this means that the solution can be checked within polynomial time. *[123]* □

[1] A problem is NP-complete, it means that a particular solution can be checked in polynomial time, but to solve the whole problem (which often requires checking many possible solutions) requires an exponential time algorithm. Because an exponential function increases at a much more rapid rate than a polynomial, these problems are said to be intractable. For a (reasonable) problem size of 20, a polynomial algorithm might require $t \sim 20^c$ time steps, compared with $t \sim c^{20}$ for an exponential time algorithm (where c is a constant). [123]

Definition 33 (Nondeterministic Polynomial Time Hard Problem): *An NP_{Time} problem is* NP-hard *if a data structure can be translated into one for solving any other NP-problem. [226]* □

Definition 34 (Nondeterministic Polynomial Time Complete Problems): *A problem which is NP and NP-hard (NP-hard) is called an NP-complete problem.* □

Table 82: Methods of Computer Science/Engineering and Operations Research

Mathematical Programming Techniques	***Stochastic Process Techniques***
Nonlinear programming	Statistical decision theory
Geometric programming	Markov processes
Quadratic programming	Queueing theory
Linear programming	Renewal theory
Integer programming	Simulation methods
Stochastic programming	Reliability theory
Separable programming	
Multiobjective programming	
Fractional programming	

Algorithms	***Statistical Methods***
Calculus methods	Regression analysis
Calculus of variations	Cluster analysis, pattern recognition
Dynamic programming	Design of experiments
Network methods:	Discriminate analysis
CPM and PERT	(factor analysis)
Game Theory	
Neural networks	
Evolutionary Computation	
Genetic Algorithms	
Genetic Programming	
Evolutionary Programming	
Evolutionary Strategies	
Simulated Annealing [168]	
TABU search	

Table 83: NPC applications and implementation example

NP-Compete Problem	Example
Traveling Salesman (laundry van problem)	Minimize distance
Graph Colouring	Minimize colors to one another
Maximum Independent Set (Max Clique)	Max set size; Min geometry
Vehicle Routing	Min Time, energy, and/or geometry
Scheduling	Min Time, missed deadlines, waiting time, resources used
Layout	Min space, overlap, costs
NP-Complete Problem Combinations	Vehicle routing and scheduling
0/1 Knapsack - Bin Packing	Max profit, Min weight
Protein Structure Prediction	Min energy and min entropy
Multiobjective Quadratic Assignment Problem	Min energy for static patterns
Boolean satisfiability problem (SAT)	Evaluate True
Hamiltonian cycle problem	Minimize distance
Subgraph isomorphism problem	Min Time to Match Pattern

Table 84: Fitness Landscape Characteristics

NP-Compete Problem	Landscape (Chapter VIII)
Deception Problems	Smooth (Chapter V)
Protein Structure Prediction	Sparse and Spiked (deltas) (Chapter VI)
MO Quadratic Assignment Problem	Rough with shallow wells (Chapter VII)
m-ary Digital symbol set design problem	Fitness Function Dependent
Traveling Salesman (laundry van problem)	Rough but not deep
Graph Colouring - $O(2^n)$	
Maximum Independent Set (Max Clique $O(3^{\frac{n}{3}})$)	
Vehicle Routing	(TSP)
Scheduling	Depends (various problems)
0/1 Knapsack - Bin Packing	Smooth
Boolean satisfiability problem (SAT)	spikes (deltas)
Hamiltonian cycle problem	(TSP)

M.1 Results CHARMm energy function approximation using an ANN

This section is mainly here to support Chapter VI on page 191. Supporting procedures and results are presented here in a effort to unencumber the reader when reading the body of the chapter.

M.2 Parameter Determination

The Matlab 6.5 neural network toolbox is used to build both the RBFNN and the MLPNN used in this study. The following are the command lines used to build, train, and test the neural networks.

1. Newrb[2] command was used for the RBFNN

   - net = newrb(a1', b1', 10)
   - Y(r) = sim(net, norm_data(1:24,data_set_size))
   - X(r) = norm_data(25,data_set_size)

2. Newff command was used for the MLPNN

   - net = newff(minmax(a1'), [3 1],
     'tansig' 'purelin','traingdx')
   - net.trainParam.epochs = 500
   - net.trainParam.goal = 1e-4
   - [net, tr] = train(net, a1', b1')
   - test = sim(net,norm_data(1:24,data_set_size))

Training Parameters: The training parameter is chosen to be selected first. Matlab 6.5's neural network toolbox has many different settings for this parameter. Training parameters tested are listed in Table 85.

Transform parameter: The transfer parameter is chosen to be selected second. Matlab 6.5's neural network toolbox has many different settings for this parameter. Among those tested for selection are listed in Table 86.

[2] The Newrb command creates a RBFNN and auto calculates parameter settings including the number of neurons required for optimal effectiveness – including the optimal hidden layers with the specified number of neurons.

Table 85: This table lists all tested training parameters. The best training parameter is indicated by bold face type.

#	Training Parameter: Description
a	trainrp: Resilient backpropagation
b	trainbr: Bayesian regularization
c	trainlm: Levenberg-Marquardt
d	traingd: Gradient Descent (GD)
e	traingdm: GD with momentum
f	**traingdx**: Adaptive learning rate
g	traincgf: Fletcher-Reeves conjugate gradient
h	traincgp: Polak-Ribiere conjugate gradient
i	traincgb: Powell-Beale conjugate gradient
j	trainscg: Scaled conjugate gradient
k	trainscg: BFGS quasi-Newton method
l	trainoss: One step secant method

Table 86: This table lists all tested transfer parameters. The best training parameter is indicated by bold face type.

#	Transfer Parameter: Description
a	compet: Competitive xfer fnc
b	hardlim: Hard-limit (HL) xfer fnc
c	hardlims: Symmetric HL xfer fnc
d	logsig: Log-sigmoid xfer fnc
e	poslin: Pos linear xfer fnc
f	purelin: HL xfer fnc
g	radbas: Radial basis xfer fnc
h	satlin: Saturating linear xfer fnc
i	satlins: Symmetric saturating linear xfer fnc
j	softmax: Soft max xfer fnc
k	**tansig**: Hyperbolic tangent sigmoid xfer fnc
l	tribas: Triangular basis xfer fnc

The transfer parameter is used to simulate the network when the *sim* command is called. More simply put, the behavior of this parameter, mathematically, is placed in the position of the neuron in the network.

M.2.1 Overtraining ANN Example. In the pedagogical example of training data to fit a noisy sine wave, like what is illustrated in Figure 120, it is essential to notice when the network begins to become overtrained. In Figure 120.d, the divergence of good training is apparent after the 3^{rd}-order polynomial. A closer analysis of the training and variance of the RMS error reveals that, as the training increases past the 5^{th}-order, even the RMS variances of the test and training data no longer overlap – indicating the system is overtrained. This constitutes a definite divergence for the system to be able to classify future incoming data into the original sine wave signal. One could argue that if the points landed directly on the x values, the system would not only accurately classify the data, but it would get an exact match for the y value. However, if the incoming data fell in between x values, a system of 5^{th}-order or higher would not classify properly.

Table 87: Summary of Training Generations Results

Generations	Neurons Layers (1/2/3)	MSE[a]	Variance
10	(25/25/1)	4.73	0.0093
100	(25/25/1)	2.07	0.0017
200	(25/25/1)	1.76	0.0013
300	(25/25/1)	0.63	0.00019
400	(25/25/1)	****	****
500	(25/25/1)	1.84	0.0032
600	(25/25/1)	****	****

Figure 118 illustrates the deep wells of energy fitness found by inserting randomly generated dihedral angles for evaluation in the CHARMm energy function. The low fitness areas in the energy landscape are sparse, and it cannot be discerned in this image if the wells have a sloped area leading into the wells. Figure 119 illustrates the fitness landscape as it has deep wells when associated with certain dihedral angle positions. Note that the deep wells are in areas where the protein conformation is good. This is indicated by a lower energy fitness value for that combination of dihedral angles. Figure 119 illustrates a simplified version of the model that this

[a] Mean squared error - This mean squared error is evaluated using the leave one out technique for the entire data set. Moreover, each point gets a turn at being the test set while the rest of the data becomes the training set.

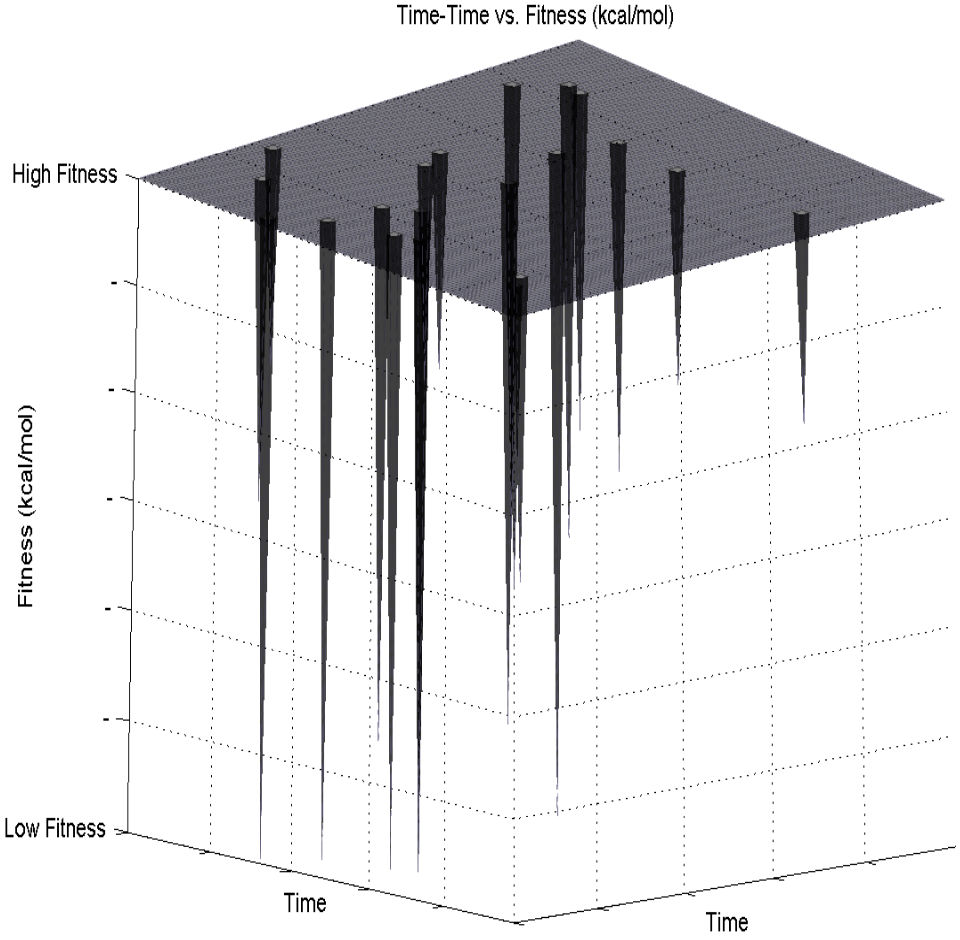


Figure 118: This figure illustrates the fitness of the CHARMm energy function when applying random values for dihedral angles. This is not a true surface plot of the fitness landscape. It reveals the fact that the lowest energy values are sparsely located in the landscape. [163]

neural network must learn to be effective. It is vital not to overtrain the network else results are expected not to represent the low energy areas.

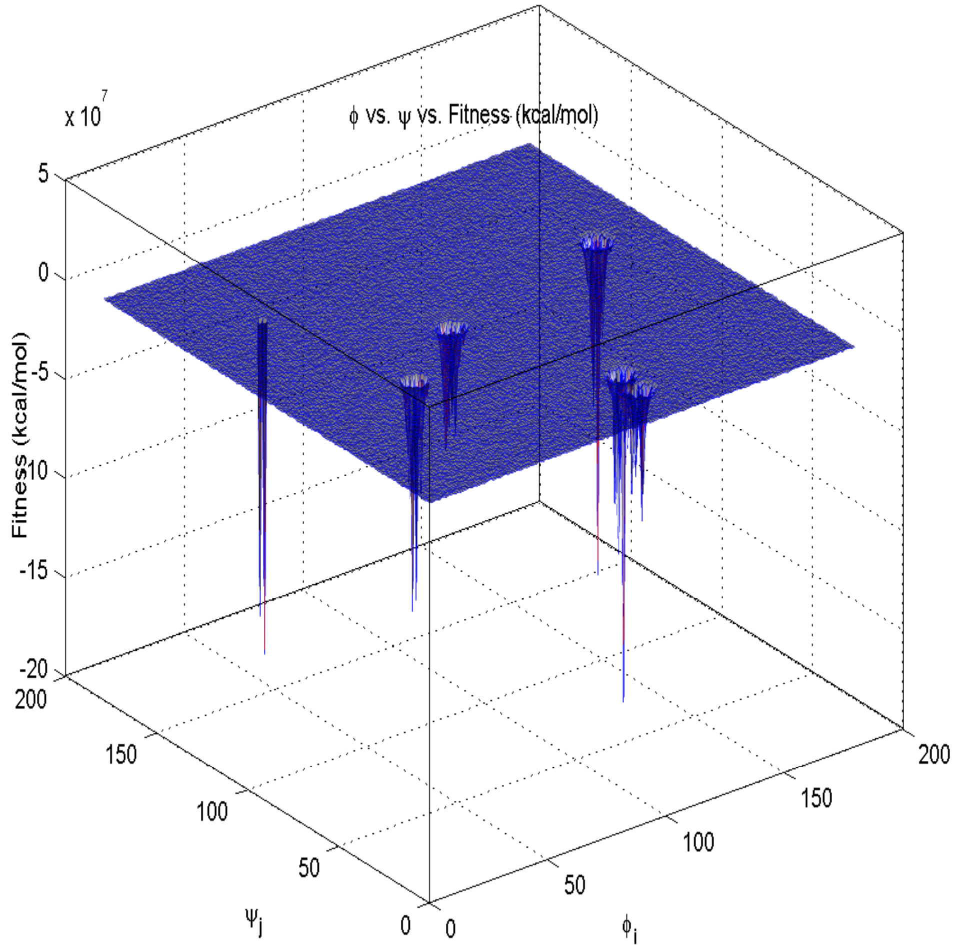
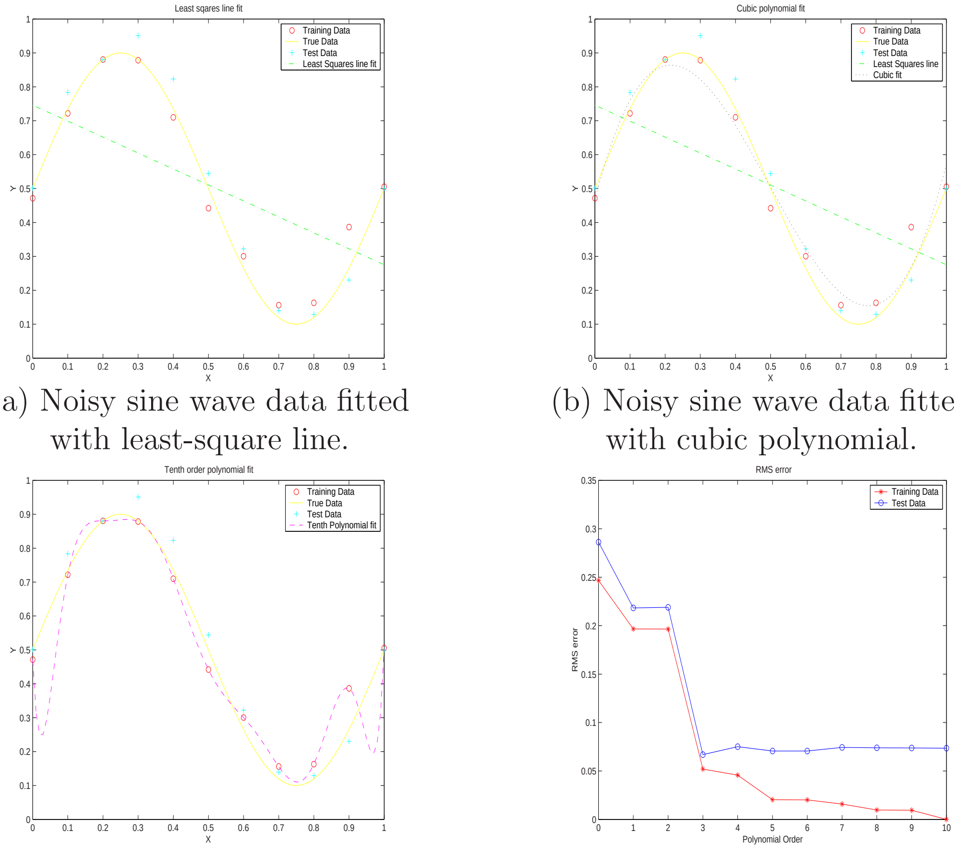


Figure 119: This figure illustrates a simulation of the CHARMm energy fitness landscape when changing only two dihedral angles. Notice the sparseness of good fitness values and the deep wells with rugged edges and rugged tight slopes leading into each one.



(a) Noisy sine wave data fitted with least-square line.

(b) Noisy sine wave data fitted with cubic polynomial.

(c) Noisy sine wave data fitted with 10^{th} order polynomial.

(d) RMS error associated with each fitted polynomial. Notice that the point of over training occurs after the 3^{rd} order polynomial.

Figure 120: Figures *a*-*c* illustrate the fitting of a line to noisy sine wave data. Figure *a* fits a least-square line to the data. Figure *b* fits a cubic line to the data. Figure *c* fits a tenth order line to the data. Figure *d* illustrates that the higher the polynomial, the lower the RMS difference between the noisy data and the fitted data becomes. However, at the same time, the difference between the real data and the fitted data drops at first, but then it actually begins to grow as the fitted line becomes close to fitting the noisy data. This is similar to overtraining a neural network. There is a point where the continued training actually works against representing the true data.

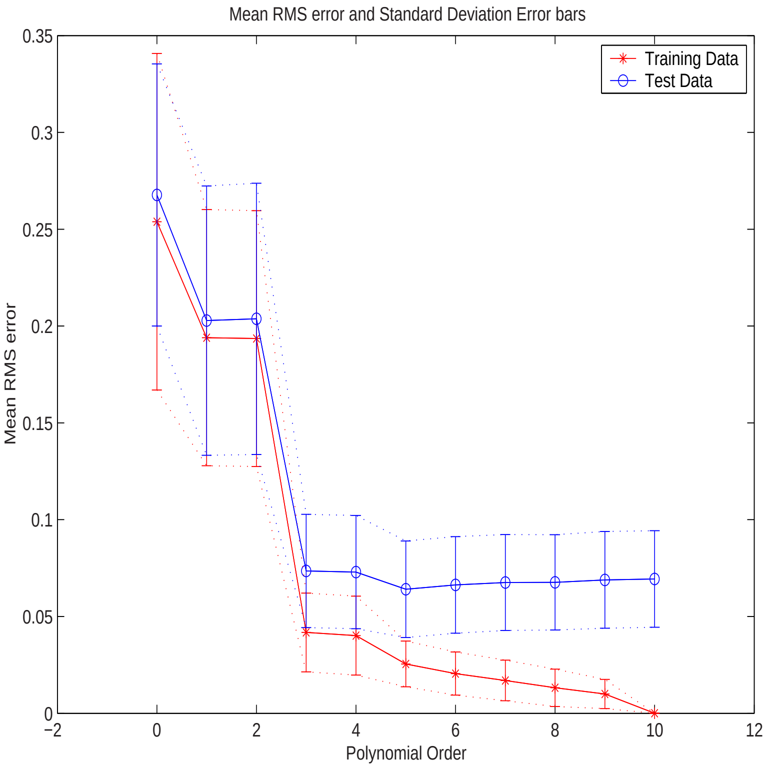


Figure 121: This figure illustrates the variance observed in the RMS error when fitting polynomials to data of a noisy sine wave function. Notice the divergence and separation of overlapping variance bars at the 5^{th} order polynomial. This figure is a closer analysis of RMS error originally shown in Figure 120.d.

Bibliography

1. Aharonov D. "Markov Chains in Theoretical Computer Science." Hebrew University, Jerusalem, Israel `http://www.cs.huji.ac.il/~doria/markovchains.huji2002.html`, May 2002.

2. Andersson J. and Krus P. "Metamodel Prepresentations for Robustness Assessment in Multiobjective Optimisation," *International Conference on Engineering Design Iced 01 GLASGOW* (August 21-23 2001).

3. Andersson J., Krus P., and Wallace D. "Multi-objective Optimisation of Hydraulic Actuation Systems," *ASME Design Automation Conference, Baltimore* (Sept 11-13 2000).

4. Bäck T. *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

5. Bäck T., Fogel D. B., and Michalewicz Z., editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

6. Baltzer L., Nilsson H., and Nilsson J. "De Novo Design of Proteins–What are the Rules?," *American Chemical Society*, *101*:3153=3163 (2001).

7. Baluja S. and Davies S. *Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of Search Space*. Technical Report CMU-CS-97-107, Pittsburgh Pennsylvania: Carnegie Mellon University, 1997.

8. Baluja S. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning,*. Technical Report CMU-CS-94-163, Pittsburgh, PA: Carnegie Mellon University, 1994.

9. Barr R., Golden B., Kelly J., Resende M., and Stewart W. "Designing and Reporting on Computational Experiments with Heuristic Methods," *Journal of Heuristics*, *1*(1):pp. 9–32 (Fall 1995).

10. Barrow J. *Theories of Everything*. Oxford Univ. Press, 1991.

11. Binh T. T. and Korn U. "An Evolution Strategy for the Multiobjective Optimization," *Second International Conference on Genetic Algorithms (Mendal 96)*, pages 23–28 (1996). Brno, Czech Republic.

12. Binh T. T. and Korn U. "Multiobjective Evolution Strategy for Constrained Optimization Problems," *In Proceedings of the 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics*, pages 357–362 (1997). Berlin, Germany.

13. Bisgaard S. and Steinberg D. "The Design and Analysis of 2k-p Prototype Experiments," *Technometrics*, *39*(1):52–62 (1997).

14. Bishop C. M. *Neural Networks for Pattern Recognition*, chapter 9, 377–380. Great Clarendon St, Oxford New York: Oxford, 2002.

15. Bonet J. S. D., Isbell C. L., and Viola P. "MIMIC: Finding Optima by estimating probability densities," *Advances in Neural Information Processing Systems*, *9* (1997).

16. Bonissone P. P. "Hybrid Soft Computing: Where are we going?." Presentation:, Aug 24 2000.

17. Brinkman D. J. and Gates G. H. "Implimentation of a CHARMm Energy Function." /usr/genetic/Toolkit/CHARMm, Setpember 1994.

18. Brinkman D. J., Merkle L. D., Lamont G. B., and Pachter R. "Parallel Genetic Algorithms and Their Application to the Protein Folding Problem," *Intel Supercomputer Users Group Annual Meeting* (October 1993).

19. Brooks B., Bruccoleri R., Olafson B., States D., Swaminathan S., and Karplus M. "CHARMm A Program for Macromolecular Energy, Minimization, and Dynamics Calculations," *Journal of Computational Chemistry*, *4*(2):187–217 (1983).

20. Brooks B., Bruccoleri R., Olafson B., States D., Swaminathan S., and Karplus M. *CHARMm A Program for Macromolecular Energy, Minimization, and Dynamics Calculations*, *4*, 187–217. John Eiley and Sons, INC, 1983.

21. Brothers L. A. "Operations Analysis in the United States Air Force," *Journal of the Operations Research Society of America*, *2*(1) (1954).

22. Bryngelson J. D., Billings E. M., Mouritsen O. G., Hertz J., Jensen M. H., Sneppen K., and Flyvbjerg H. *From Interatomic Interactions to Protein Structure*, *480*, chapter Physics of Biological Systems : From Molecules to Species, 80–116. Springer-Verlag New York, 1997.

23. Burkard R., Karisch S., and Rendl F. "A Quadratic Assignment Problem Library," *Journal of Global Optimization*, 391–403 (1997).

24. Cantu-Paz E. *Efficient and Accurate Parallel Genetic Algorithms*. Boston: Kluwer Academic Publishers, 2000.

25. Carter E. F. "Random Walks, Markov Chains and the Monte Carlo Method." Taygeta Scientific Inc, `http://www.taygeta.com/rwalks/node7.html`, Feb 2005.

26. Çela E. *The Quadratic Assignment Problem - Theory and Algorithms*. Boston, MA: Kluwer Academic Publishers, 1998.

27. Chang I. "The Urban Battleground," *www.nytimes.com* (March 28 2003).

28. Cheng V. W.-K. and Stark W. E. “Performance of Trellis-Coded Direct-Sequence Spread Spectrum with Noncoherent Reception in a Fading Environment,” *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, *47*(2):579–590 (May 1998).

29. Chow C. O. and Dimyati K. “Congestion Control of Poisson Distributed ABR Applications in ATM Networks using Neural Network,” *IEEE TENCON*, pg 759–762 (August 2002).

30. Coello C. A. C., Aguirre A. H., and Zitzler E., editors. *Evolutionary Multi-Criterion Optimization, Third International Conference,EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, *3410*. Lecture Notes in Computer Science, Springer, 2005.

31. Coello C. A. C. and Lamont G. B. *Applications of Multi-Objective Evolutioinary Algorithms* (1 Edition), *1*. 5 Toh Tuck Link, Singapore 596224: World Scientific, 2004.

32. Coello Coello C. A. “A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques,” *Knowledge and Information Systems. An International Journal*, *1*(3):269–308 (August 1999).

33. Coello Coello C. A. “A Short Tutorial on Evolutionary Multiobjective Optimization.” *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 21–40, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

34. Coello Coello C. A., Van Veldhuizen D. A., and Lamont G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, May 2002.

35. Committee on Physical M. and Sciences E. *Grand Challenges 1993: High Performance Computing and Communications*. Office of Science and Technology Policy, 1982.

36. Constable P. “Angry Afghans Protest Attack on Wedding Party,” *Washington Post Foreign Service* (3 July 2002). 2002 The Washington Post Company.

37. Coppersmith D. *Open Problems in Communication and Computation*, chapter 4.6 Knapsack Used in Factoring, pp. 117–119. New York: Springer-Verlag, 1987. Ed. T. M. Cover and B. Gopinath.

38. Crossley W. A., Cook A. M., Fanjoy D. W., and Venkayya V. B. “Using the Two-Branch Tournament Genetic Algorithm for Multiobjective Design.” *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. April 1998.

39. Czyzak P. and Jaszkiewicz A. "Pareto Simulated Annealing: A Metaheuristic Tech- nique For Multiple-Objective Combinatorial Optimization," *Journal of Multi-Criteria Decision Analysis*, (7):34–47 (1998).

40. Danzig G. B. *Linear Programming and Extensions*. Reading, Princeton NJ: Princeton University Press, 1963. 1914, Series: Rand Corporation research study.

41. Day R. O. *A Multiobjective Approach Applied to the Protein Structure Prediction Problem*. Masters Thesis, Air Force Institute of Technology, March 2002. Sponsor: AFRL/Materials Directorate.

42. Day R. O., Kleeman M. P., and Lamont G. B. "Solving the Multi-objective Quadratic Assignment Problem Using a fast messy Genetic Algorithm." *Congress on Evolutionary Computation (CEC'2003)1*. 2277–2283. Piscataway, New Jersey: IEEE Service Center, December 2003.

43. Day R. O., Kleeman M. P., and Lamont G. B. "Multi-Objective fast messy Genetic Algorithm Solving Deception Problems," *Proceedings of the 2002 Congress on Evolutionary Computation CEC2004* (2004).

44. Day R. O., Kleeman M. P., and Lamont G. B. "Multiobjective Quadratic Assignment Problem solved by an explicit Building Blocked Search Algorithm – MOMGA-IIa," *5th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP2005)* (30 March - 1 April 2005). Lausanne, Switzerland.

45. Day R. O. and Lamont G. B. "Force Field Approximations Using Artificial Neural Networks," *The Congress on Evolutionary Computation (CEC)* (June 20-23 2004). Portland, Oregon.

46. Day R. O. and Lamont G. B. "Multi-Objective fast messy Genetic Algorithm Solving Deception Problems," *Congress on Evolutionary Computation; Portland, Oregon*, *4*:1502–1509 (June 19 - 23 2004).

47. Day R. O. and Lamont G. B. "Extended Multi-objective fast messy Genetic Algorithm Solving Deception Problems," *Proceeding for the Third International Conference on Evolutionary Multi-Criterion Optimization*, (32):pp 296–310 (March 2005). Guanajuato, Mexico.

48. Day R. O. and Lamont G. B. *Parallel Computing For Bioinformatics & Computational Biology* (1 Edition), chapter Parallel Evolutionary Computations In Discerning Protein Structures, In Press. Wiley, 2005.

49. Day R. O., Lamont G. B., and Pachter R. "Protein Structure Prediction by Applying an Evolutionary Algorithm," *The Second IEEE International Workshop On High Performance Computational (HiCOMB)* (April 22-25 2003). Nice, France.

50. Day R. O., Zydallis J. B., and Lamont G. B. "Competitive Template Analysis of the Fast Messy Genetic Algorithm When Applied to the Protein Structure Prediction Problem," *ICCN*, 4 (April 2002).

51. Day R. O., Zydallis J. B., and Lamont G. B. "Solving the Protein Structure Prediction Problem Through a Multiobjective Genetic Algorithm," *ICCN*, 4 (April 2002).

52. Day R. O., Zydallis J. B., Lamont G. B., and Pachter R. "Genetic Algorithm Approach to Protein Structure Prediction with Secondary Structures," *EUROGEN*, 6 (September 2000).

53. Day R. O., Zydallis J. B., Lamont G. B., and Pachter R. "Analysis of Fine Granularity in Parallelization and Building Block Sizes of the Parallel Fast Messy GA used on the Protein Structure Prediction Problem," *World Congress on Computational Intelligence*, 6 (December 2001). Special Biological Area.

54. Deb K. *Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems*. Technical Report CI-49/98, University of Dortmund, Germany: Dortmund: Department of Computer Science/LS11, 1998.

55. Deb K. "Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design." *Evolutionary Algorithms in Engineering and Computer Science* edited by Kaisa Miettinen, et al., chapter 8, 135–161, Chichester, UK: John Wiley & Sons, Ltd, 1999.

56. Deb K. "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, *186*(2/3):311–338 (2000).

57. Deb K. "Multiobjective Evolutionary Algorithm: Past, Present, Future," *Proceedings of the fourth Advanced Computing On Design and Mnufacturing (ACDM-2000)*, 225–236 (2000). PEDC, University of Plymouth, UK, Springer London.

58. Deb K. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001.

59. Deb K. and Goldberg D. E. "Analyzing Deception in Trap Functions." *Foundations of Genetic Algorithms 2* edited by L. Darrell Whitley, 93–108, San Mateo, CA: Morgan Kaufmann, 1993.

60. Deb K., Horn J., and Goldberg D. E. *Multimodal Deceptive Functions*. Technical Report IlliGAL Report No 92003, University of Illinois, Urbana, 1992.

61. Deb K. and Kumar A. "Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems," *Complex Systems*, *9*:431–454 (1995).

62. Deb K., Thiele L., Laumanns M., and Zitzler E. *Scalable Test Problems for Evolutionary Multi-Objective Optimization*. Technical Report 112, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.

63. Deb K., Thiele L., Laumanns M., and Zitzler E. "Scalable Multi-Objective Optimization Test Problems." *Congress on Evolutionary Computation (CEC'2002)1*. 825–830. Piscataway, New Jersey: IEEE Service Center, May 2002.

64. Deerman K. R. *Protein Structure Prediction Using Parallel Linkage Investigating Genetic Algorithms*. MS THESIS, Air Force Institute of Technology, Wright Patterson AFB, OH, March 1999. Sponsor: AFRL/Material Directorate.

65. Deerman K. R., Lamont G. B., and Pachter R. "Linkage Investigation Genetic Algorithms and Their Application ot the Protein Structure," *ACM Symposium on Applied computing (SAC01)* (March 11-14 2001). Las Vegas, Nevada.

66. Demuth H. and Beale M. *Neural Network Toolbox* (version 3 Edition). The MATH WORKS Inc., 1998.

67. Derrida B. "Random Energy Model: Limit of a Family of Disordered Models." *Phys. Rev. Lett.* 45. 1980.

68. Dittmer T. W. *Digitally Modulated RF System*. Primer, Quincy, IL USA: Harris Corporation, 1999.

69. Duda R. O., Hart P. E., and Stork D. G. *Pattern Classification* (2 Edition), chapter 6, 295. 605 Third Ave, New York, NY 10158: Wiley-Interscience, 2001.

70. Engineous Software I. "Introduction to iSIGHT Application Development,". iSIGHT Patent Pending, DHS Patent #US6086617, 2000 CentreGreen Way, Suite 100, Cary, NC 27513., March 2005. iSIGHT is a trademark of Engineous Software, Inc.

71. English T. M. "Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch." *Evolutionary Programming V: Proc. of the Fifth Annual Conf. on Evolutionary Programming*, edited by Lawrence J. Fogel, et al. 163–169. Cambridge, MA: MIT Press, 1996.

72. et al Soukhanov . *WEBSTER'S II New Riverside University Dictionary*. Boston, MA: The Riverside Publishing Company, 1984.

73. Foley J. B., "State Deptartment Report on Accidental Bombing of Chinese Embassy." Oral Presentation, July 1999. U.S. Department of State.

74. Fonseca C. M. and Fleming P. J. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization." *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by Stephanie Forrest. 416–423. San Mateo, California: Morgan Kauffman Publishers, 1993.

75. Fonseca C. M. and Fleming P. J. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization." *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by Stephanie Forrest. 416–423. San Mateo, CA: Morgan Kaufman, 1993.

76. Fonseca C. M. and Fleming P. J. "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, *3*(1):1–16 (Spring 1995).

77. Fonseca C. M. and Fleming P. J. "Multiobjective Optimization." *Evolutionary Computation 2*, edited by Thomas Bäck, et al., 25–37, Institute of Physics Publishing, 2000.

78. Forrest S. and Mitchell M. "Relative Building-Block Fitness and the Building-Block Hypothesis." *Foundations of Genetic Algorithms – 2 (FOGA-92)*, edited by Darrell Whitley. 109–126. 1992.

79. Gambardella L. M., Taillard E. D., and Dorigo M. "Ant Colonies for the Quadratic Assignment Problems," *Journal of the Operational Research Society*, *50*:167–176 (1999).

80. Gates G. H. *Predicting Protein Structure using Parallel Genetic Algorithms*. MS THESIS, Air Force Institute of Technology, Wright Patterson AFB, OH, December 1994. Sponsor: AFOSR WL/Material Directorate.

81. Gates G. H. *Predicting Protein Structure using Parallel Genetic Algorithms*. MS THESIS, Air Force Institute of Technology, Wright Patterson AFB, OH, December 1994. Sponsor: AFOSR WL/Material Directorate.

82. Gates G. H., Merkle L. D., Lamont G. B., and Pachter R. "Simple Genetic Algorithm parameter Selection for Protein Structure Prediction," *International Conference on Evolutionary Algorithms* (December 1995). Perth, Austria.

83. Gates G. H., Pachter R., Merkle L. D., and Lamont G. B. "Parallel Simple GAs vs Parallel Fast Messy GAs for Protein Floding Problem," *Intel Supercomputer Users Groupt Annual Meeting* (June 1995).

84. Geoffrion A. M. "Proper Efficiency and the Theory of Vector Maximization," *Journal of Mathematical Analysis and Applications*, *22*(3):618–630 (June 1968). Printed in Belguim.

85. Gilbreth F. B. and Carey E. G. *Cheaper by the Dozen* (Paperback Edition). Perennial, June 2002. 224 pages.

86. Go N. and Scheraga H. A. "Calculation of the Conformation of Pentapeptide cyclo-(Glycylglycylglycylprolylprolyl). I. AComplete Energy Map," *Macromolecules*, *3*:188–194 (December 1969). Department of Chemistry, Cornell University, Ithaca, New York.

87. Go N. and Scheraga H. A. "Ring Closure and Local Conformational Deformation of Chain Molecules," *Macromolecules*, *3*:178–187 (December 1969). Department of Chemistry, Cornell University, Ithaca, New York.

88. Goldberg D. *Simple Genetic Algorithms and the Minimal, Deceptive Problem*, chapter Genetic Algorithms and Simulated Annealing, pp: 74–88. Morgan Kaufmann, Pubs, 1987.

89. Goldberg D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.

90. Goldberg D. E. "Messy Genetic Algorithms Revisited: Studies in Mixed Size and Scale.," *Complex Systems*, 415–444 (1990).

91. Goldberg D. E., Deb K., and Clark J. H. "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Systems*, *6*(19):pp. 333–362 (1992).

92. Goldberg D. E., Deb K., and Horn J. "Massive Multimodality, Deception, and Genetic Algorithms." *Parallel Problem Solving from Nature, 2*, edited by R. Männer and B. Manderick. Amsterdam: Elsevier Science Publishers, B. V., 1992.

93. Goldberg D. E., Deb K., Kargupta H., and Harik G. "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms." *Proceedings of the Fifth International Conference on Genetic Algorithms*. July 1993.

94. Goldberg D. E., Korb B., and Deb K. "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, *3*(5):493–530 (1990).

95. Goldberg D. E. and Richardson J. "Genetic Algorithms with Sharing for Multimodal Function Optimization." *Genetic Algorithms and their Applications (ICGA'87)*, edited by John J. Grefenstette. 41–49. Lawrence Erlbaum Associates, Publishers, 1987.

96. Gosselin P. "Questions surround a precision bombing," *The Boston Globe* (Feb 1991). Globe Staff, `www.boston.com`.

97. Hahn P., Hall N., and Grant T. "A Branch-and Bound Algorithm for the Quadratic Assignment Problem based on the Hungarian Method," *European Journal of Operational Research* (August 1998).

98. Hansen M. P. and Jaszkiewicz A. *Evaluating The Quality Of Approximations To The Non-Dominated Set*. Technical Report IMM-REP-1998-7, 1998.

99. Harik G., "Linkage Learning via Probabilistic Modeling in the ECGA," 1999.

100. Harik G., Lobo F., and Goldberg D. "The Compact Genetic Algorithm." *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*. 523–528. IEEE Press, 1998.

101. Heermann D. W. *Computer Simulation Methods in Theoretical Physics*. Springer-Verlag, 1990.

102. Ho S.-Y., Shu L.-S., and Chen J.-H. "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," *Evolutionary Computation, IEEE Transactions on*, *8*(6):522–541 (Dec 2004). ISSN: 1089-778X.

103. Holland J. H. "Adaptation in Natural and Artificial Systems," *University of Michigan Press* (1975).

104. Holland J. J. *Adaptation in Natural and Artificial Systems*. Ann Arbor MI: The University of Michigan: The University of Michigan Press, 1975.

105. Horn J. and Nafpliotis N. "Multiobjective Optimization Using the Niched Pareto Genetic Algorithm.". 1993.

106. Horn J., Nafpliotis N., and Goldberg D. E. "A Niched Pareto Algorithm for Multiobjective Optimization." *Proc. of the First IEEE Conf. on Evolutionary Computation1*. 82–87. Piscataway, NJ: IEEE Service Center, 1994.

107. Horng J.-T., Chen C.-C., Liu B.-J., and Kao C.-Y. "Resolution of Quadratic Assignment Problems Using an Evolutionary Algorithm," *2*:902–909 (2000).

108. Hu J., Goodman E., K.Seo , Fan Z., and Rosenberg R. "The Hierarchical Fair Competition (HFC) Framework for Sustainable Evolutionary Algorithms," *Evolutionary Computation*, *13*(1) (2005). In Press.

109. Hu J., Seo K., Fan Z., Rosenberg R., and Goodman E. "HEMO: A Sustainable Multi-Objective Evolutionary Optimization Framework," *Proc. 2003 Genetic and Evolutionary Computing Conference*, pp. 1029–1040 (July 2003). Chicago, Springer, Lecture Notes in Computer Science.

110. Hu J. J. and Goodman E. D. "Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms," *Proceedings, Congress on Evolutionary Computation, CEC 2002, IEEE World Congress on Computational Intelligence* (May 2002). Honolulu, Hawaii.

111. Jackson R., P. T. Boggs S. G. N., and Powell S. "Guidelines for Reporting Results of Computational Experiments. Report of the Ad Hoc Committee," *Mathematical Programming*, (49):pp. 413–426 (1990/1).

112. Jain R. *The Art of Computer Systems Performance Analysis*. Wiley, 1991.

113. Joiner L. L. and Komo J. J. "Improved Performance of Reduced M-ary Orthogonal Signaling Using Reed-Solomon Codes," *MILCOM*, *02/6*(0-7803-5538-5/99):4 (1999).

114. Jones B. R., Crossley W. A., and Lyrintzis A. S. "Aerodynamic and Aeroacoustic Optimization of Airfoils via a Parallel Genetic Algorithm." *Proceedings of*

the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-98-4811. AIAA, 1998.

115. Kaiser C. E., Lamont G. B., Merkle L. D., Gates G. H., and Pachter R. "Exploiting Domain Knowledge in Genetic Algorithms for Polypeptide Structure Predition (Abstract)," *212th American Chemical Society National Meeting* (August 1996). Orlando, Florida.

116. Kaiser C. E., Lamont G. B., Merkle L. D., Gates G. H., and Pachter R. "Real Valued Hybrid Genetic Algorithms for Polypeptide Structure Prediction," *28th American Chemical Society Central Regional Meeting* (June 1996). Dayton, Ohio.

117. Kaiser C. E., Lamont G. B., Merkle L. D., Gates G. H., and Pachter R. "Exogenous Parameter Selection in Real-Valued Genetic Algorithms," *International Conference on Evolutionary Computation (ICE97)* (April 1997). Indianapolis, Indiana.

118. Kaiser C. E., Lamont G. B., Merkle L. D., Gates G. H., and Pachter R. "Polypeptide Structure Prediction: Real-Valued versus Binary Hybrid Genetic Algorithms," *ACM Symposium on Applied Computing* (February 28-March 2 1997). Atlanta, Georgia.

119. Kaiser C. E., Lamont G. B., Merkle L. D., Gates G. H., and Pachter R. "Real-valued Genetic Algorithm Case Studies in Protein Structure Prediction," *SIAM Conference on Parallel Processing for Scientific Computing* (March 1997). Minneapolis, Minnesota.

120. Karplus M. "CHARMm energy functions." `yuri.harvard.edu`, 2001.

121. Khan N. *Bayesian Optimization Algorithms for Multiobjective and Hierarchically Difficult Problems*. Master Thesis, University of Illinois at Urbana-Champaign, 117 Transportation Building, July 2003.

122. Khan N., Goldberg D. E., and Pelikan M. *Multi-Objective Bayesian Optimization Algorithm*. Technical Report 2002009, Urbana, Illinois: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, March 2002.

123. Khimasia M. M. L. "NP Complete Problems." `http://www.tcm.phy.cam.ac.uk/$\sim$mmlk2/report13/node31.html`, 1996.

124. Kingston J. "The DARPA High Performance Knowledge Bases Programme," *Knowledge Acquisition, Modeling and Management, 10th European Workshop, EKAW'97, Sant Feliu de Guixols, Catalonia, Spain, October 15-18, 1997, Proceedings*, *1319*:pp. 383–388 (1997). J Kingston of the University of Edinburgh: ISBN 3-540-63592-0.

125. Kleeman M. P. *Optimization of Heterogeneous UAV Communications Using the Multiobjective Quadratic Assignment Problem*. MS THESIS, Air Force Institute of Technology, March 2004. Sponsor AFRL.

126. Kleeman M. P., Day R. O., and Lamont G. B. "Analysis of a Parallel MOEA solving the Multiobjective Quadratic Assignment Problem," *Genetic and Evolutionary Computation Conference (GECCO)* (June 26-30 2004). 2 Page Poster Paper.

127. Kleeman M. P., Day R. O., and Lamont G. B. "Multi-Objective Evolutionary Search Performance with Explicit Building-Block Sizes for NPC Problems," *The Congress on Evolutionary Computation (CEC)* (June 20-23 2004). Portland, Oregon.

128. Knowles J. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD Thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

129. Knowles J. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. Presented Tutorial II, Faraday Building, Sackville Street, PO Box 88, Manchester M60 1QD: University of Manchester, UK, March 2005. Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), Guanajuato, Mxico.

130. Knowles J. and Corne D. "M-PAES: A Memetic Algorithm for Multiobjective Optimization." *2000 Congress on Evolutionary Computation1*. 325–332. Piscataway, New Jersey: IEEE Service Center, July 2000.

131. Knowles J. and Corne D. *Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem*. Technical Report TR/IRIDIA/2002-25, IRIDIA, 2002. (Accepted for presentation/publication at the *2003 Evolutionary Multi-criterion Optimization Conference (EMO-2003)), Faro, Portugal.*

132. Knowles J. and Corne D. "On Metrics for Comparing Nondominated Sets.," *Congress on Evolutionary Computation (CEC'2002)*, *1*:711–716 (May 2002). Piscataway New Jersey: IEEE Service Center.

133. Knowles J. and Corne D. "Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multiobjective Quadratic Assignment Problem." *Soft Computing Systems: Design, Management and Applications*, edited by A. Abraham, et al. 271–279. Amsterdam: IOS Press, 2002. ISBN 1-58603-297-6.

134. Knowles J. and Corne D. "Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem." *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 2003, Proceedings*, number 2632 in LNCS, edited by Carlos Fonseca, et al. 295–310. Springer, 2003.

135. Knowles J. D. and Corne D. W. "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, *8*(2):149–172 (2000).

136. Knowles J. D. and Corne D. W. "The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization," *In Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, 839–848 (2000).

137. Knowles J. D., Corne D. W., and Oates M. "PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization," *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 283–290 (2001).

138. Krasnogor N., Pelta D., Mocciola P., Lopex P. E. M., and de la Canal E. "Enhanced Evolutionary Search of Folding Using Parsed Proteins," *Operational Research Symposium* (1997). Bs. As. Argentina.

139. Kuhn H. W. and Tucker A. W. "Nonlinear Programming." *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. 481–492. Berkeley, California, 1951.

140. Kuhn T. *The Structure of Scientific Revolutions*. Univ. of Chicago Press, 1962.

141. Kumar P. T. and Hitoshi I. "Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms," *9th MPS symposium on Evolutionary Computation, IPSJ, Japan* (2002).

142. Kumar T. "Comparison of Optimization Techniques in Large Scale Transportation Problems." Minnesota State University, Mankato, MN, mentor: Susan M. Schilling, 2003.

143. Kung H., Luccio F., and Preparata F. P. "On Finding the Maxima Of A Set Of Vectors," *Journal of the Association for Computing Machinery*, *22*(4):469–476 (1975).

144. Lamont G. *MOEAs: What, When, and Why*. Presented Tutorial I, Dayton Ohio: Air Force Institute of Technology, March 2005. Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), Guanajuato, Mxico.

145. Lamont G. B., Day R. O., and Keller T. "Design of Military Applications with Evolutionary Algorithms," *Genetic and Evolutionary Computation Conference (GECCO) Workshop on Biological Applications of Genetic and Evolutionary* (June 26-30 2004).

146. Lamont G. B., Kleeman M. P., and Day R. O. *Applications of Multi-Objective Evolutionary Algorithms Book*, chapter Multiobjective Evolutionary Algorithms for Computer Science, pg 451 478. World Scientific, 2004. Chapter 19.

147. Lamont G. B. and Merkel L. D. "Introduction to Bioinformatics for Computer Scientists." Chapter in W. Corne's book, August 2002.

148. Laumanns M., Zitzler E., and Thiele L. "On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization." *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 181–196, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

149. Launer R. L. and Wilkinson G. N. *Robustness in Statistics*, chapter Robustness in the Strategy of Scientific Model Building. New York: Academic Press, 1979.

150. Loiola E. M., de Abreu N. M. M., Boaventura-Netto P. O., Hahn P., and Querido T. *An Analytical Survey for the Quadratic Assignment Problem*. Technical Report, Council for the Scientific and Technological Development, of the Brazilian gov, 2004.

151. Maniezzo V. and Colorni A. "The Ant System Applied to the Quadratic Assignment Problem," *IEEE Transactions on Knowledge and Data Engineering*, *11*:769–778 (1999).

152. M*ü*hlenbein H. "The Equation for Response to Selection and Its Use for Prediction," *Evolutionary Computation*, *5*(3):303–346 (1998).

153. M*ü*hlenbein H. and Paass G. "From Recombination of Genes to the Estimation of Distributions I. Binary Parameters." *Parallel Problem Solving from Nature – PPSN V*, edited by A.E. Eiben, et al. 178–187. Springer, 1998.

154. Merkle L. D. *Generalization and Parallelization of Messy Genetic Algorithms and Communication in Parallel Genetic Algorithms*. Thesis, Air Force Institute of Technology, December 1992. Sponsor: WL/Material Directorate.

155. Merkle L. D. *Analysis of Linkage-Friendly Genetic Algorithms*. PhD Dissertation, Air Force Institute of Technology, Department of Electrical and Computer Engineering, School of Engineering, WPAFB, Oh, August 1997. technical advisor - Professor Gary B. Lamont.

156. Merkle L. D., Gates G. H., and Lamont G. B. "Scalability of an MPI-Based Fast Messy Genetic Algorithm," *ACM Symposium on Applied Computing (SAC98)* (February 1998). San Antonio, Texas.

157. Merkle L. D., Lamont G. B., Gates G. H., and Pachter R. "Application of the Parallel fast Messy Genetic Algorithm to the Protein Folding Problem," *Proceedings of Intel Supercomputing Users Group Annual Meeting* (June 1994). pp. 189-195.

158. Merkle L. D., Lamont G. B., Gates G. H., and Pachter R. "Parallel Messy Algorithms for the Protein Folding Problem," *Proceedings of Intel Supercomputing Users Group Annual Meeting* (June 1994). pp. 189-195.

159. Merkle L. D., Lamont G. B., Gates G. H., and Pachter R. "Hybrid Genetic Algorithms for Minimization of a Polypeptide Specific Energy Model," *IEEE Conference on Evolutionary Computation* (May 1996). Japan.

160. Merkle L. D., Lamont G. B., Gates G. H., and Pachter R. "Hybrid Genetic Algorithms for Polypepetide Energy Minization," *ACM Symposium on Applied Computing* (February 1996). Philadelphia, Pennsylvania.

161. Merz P. and Freisleben B. "A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem," *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, *3*:1999–2070 (1999).

162. Merz P. and Freisleben B. "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," *IEEE Transactions on Evolutionary Computation*, *4*:337–352 (2000).

163. Michaud S. R. *Protein Structure Prediction using Refined Parallel Fast Messy Genetic Algorithms*. MS THESIS, Air Force Institute of Technology, March 2001. Sponsor: AFRL/Material Directorate.

164. Michaud S. R., Zydallis J. B., Lamont G. B., and Pachter R. "Load Balancing the Parallel Fast Messy Genetic Algorithm for Increased Computational Efficiency in Attempting to Solve the Protein Structure Prediction Problem with a Heterogeneous Cluster of PCs," *Tenth SIAM Conference on Parallel Processing for Scientific Computing (PP'01)* (March 12-14 2001). Portsmouth, Virginia.

165. Michaud S. R., Zydallis J. B., Lamont G. B., and Pachter R. "Scaling a Genetic Algorithm to Medium-Sized Peptides by Detecting Secondary Structures with an Analysis of Building Blocks," *First International Conference on Computational Nanoscience (ICCN'01)* (March 19-21 2001). Hilton Head Island, South Carolina.

166. Miller I. and Freund J. E. *Probablity and Statisitcs for Engineers* (2nd Edition), chapter 6, Sampling Distributions, 174. Englewood Cliffs, New Jersey 07632: Prentice-Hall, Inc, 1977.

167. Morse J. N. "Reducing the Size of the Nondominated Set: Pruning By Clustering," *Computers and Operations Research*, *7*(1-2):55–66 (1980).

168. Myers D. C. "A Simulated Annealing Algorithm for Scheduling Military Training," *invited presentation at the 15th International Symposium on Mathematical Programming* (August 1994). Ann Arbor, Michigan.

169. Neumaier A. "Molecular Modeling of Proteins and Mathematical Prediction of Protein Structure," *SIAM Review*, *39*(3):407–460 (1997).

170. Norris S. R. and Crossley W. A. "Pareto-Optimal Controller Gains Generated by a Genetic Algorithm." *AIAA 36th Aerospace Sciences Meeting and Exhibit*. January 1998.

171. Nott D. J., Dunsmuir W. T., Kohn R., and Woodcock F. "Statistical Correction of a Deterministic Numerical Weather Prediction Model," *Journal of the American Statistical Association*, *96*(455):pp. 794–804(11) (1 September 2001). Publisher: American Statistical Association.

172. Ocenasek J. and Pelikan M. "Parallel Mixed Bayesian Optimization Algorithm: A Scaleup Analysis. Optimization by Building and Using Probabilistic Models," *OBUPM-2004, Seatttle, WA.* (2004).

173. Oei C., Goldberg D. E., and Chang S.-J. *Tournament Selection, Niching and the Preservation of Diversity*. Technical Report 91011, Illinois Genetic Algorithms Laboratory (IlliGAL), December 1991.

174. Pacheco P. *Parallel Programming with MPI*. Morgan Kaufmann Publishers, 1996.

175. Paquete L., Chiarandini M., and Stützle T. "Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study." *Metaheuristics for Multiobjective Optimisation 535*. Lecture Notes in Economics and Mathematical Systems, edited by X. Gandibleux, et al., Springer Verlag, 2004. (©Springer Verlag).

176. Paquete L., Chiarandini M., and Stützle T. "Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study." *Metaheuristics for Multiobjective Optimisation*, edited by Xavier Gandibleux, et al. 177–199. Berlin: Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, 2004.

177. Paquete L. and Stützle T. "A Two-Phase Local Search for the Biobjective Traveling Salesman Problem." *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, edited by Carlos M. Fonseca, et al. 479–493. Faro, Portugal: Springer. Lecture Notes in Computer Science. Volume 2632, April 2003.

178. Paquete L. F. and Fonseca C. M. "A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms." *Proceedings of the 4th Metaheuristics International Conference (MIC'2001)* edited by Jorge Pinho de Sousa, 149–153, Porto, Portugal: Program Operational Ciencia, Tecnologia, Inovaçao do Quadro Comunitário de Apoio III de Fundaçao para a Ciencia e Tecnologia, July 16–20 2001.

179. Pardalos P. M. and Wolkowicz H. "Quadratic Assignment and Related Problems." *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems*, edited by Panos M. Pardalos and Henry Wolkowicz. 1994.

180. Park Y.-K. and Lee G. "Intelligent Congestion Control in ATM Networks," *Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pgs 369–375 (28-30 Aug 1995).

181. Pelikan M. *Bayesian Optimization Algorithm: From Single Level to Hierarchy*. Thesis, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, July 2002. IlliGAL Report No. 2002023.

182. Pelikan M., Goldberg D. E., and Cantú-Paz E. "BOA: The Bayesian Optimization Algorithm." *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99I*, edited by Wolfgang Banzhaf, et al. 525–532. Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA, 13-17 1999.

183. Pelikan M. and Mühlenbein H. "The Bivariate Marginal Distribution Algorithm," *Advances in Soft Computing-Engineering Design and Manufacturing*, 521–535 (1999).

184. Piccolboni A. and Mauri G. "Application of Evolutionary Algorithms to Protein Folding Prediction." *Artificial Evolution*. 123–136. 1997.

185. Plackett R. L. and Burman J. P. *The Design of Optimum Multifactorial Experiments*, *33*, 305–325. Biometrika, 1946.

186. Rao S. S. *Engineering Optimization, Theory and Practice* (3rd Edition). School of Mechanical Engineering, Purdue University, West Lafayette, Indiana: Wiley-Interscience, 1996.

187. Rechenberg I. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution* (1st Edition). 1973.

188. Reeves C. and Wright C. "An Experimental Design Perspective on Genetic Algorithms." *Foundations of Genetic Algorithms 3* edited by L. Darrell Whitley and Michael D. Vose, 7–22, San Francisco, CA: Morgan Kaufmann, 1995.

189. Resarch G. A. and at Michigan State University A. G. "Loss of Explorative Capability (LEC): A New Explanation of Premature Convergence." HFC-the Hierarchical Fair Competition Model for Scalable, Sustainable and Robust Evolutionary Computation, April 2005.

190. Ripoll D. R. "Electrostatically Driven Monte Carlo (EDMC) Protein Folding." Baker Lab. of Chemistry.

191. Sastry K. and Goldberg D. E. "On Extended Compact Genetic Algorithm." *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, edited by Darrell Whitley. 352–359. 8 2000.

192. Saven J. G. "Designing Protein Energy Landscapes," *American Chemical Society*, *101*:3113=3130 (2001).

193. Sawaragi Y., Nakayama H., and Tanino T. *Theory of Multiobjective Optimization*, *176*. Mathematics in Science and Engineering. Department of Mech. Eng. II, Tohoku University, Sendai Japan: Academic Press, Inc. (Harcourt Brace Jovanovich), 1985.

194. Schaffer J. D. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD dissertation, Vanderbilt University, 1984.

195. Schaffer J. D. "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms." *Proceedings of the First International Conference on Genetic Algorithms*. 93–100. Hillsdale, New Jersey: Lawrence Erlbaum, 1985.

196. Schott J. R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. Master thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, Massachusetts, May 1995.

197. Schwefel H.-P. "Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie," (1977).

198. Schwefel H.-P. "Collective Phenomena in Evolutionary Systems," *31st Annu. Meet. Inter'l Soc. for General System Research*, 1025–1033 (1987).

199. Schwefel H.-P. "Natural Evolution and Collective Optimum Seeking." *Computational Systems Analysis – Topics and Trends* edited by A. Sydow, 5–14, Amsterdam: Elsevier, 1992.

200. Schwefel H.-P. *Evolution and Optimum Seeking*. New York: Wiley and Sons, 1995.

201. Schwefel H.-P. P. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., 1993.

202. Sebag M. and Ducoulombier A. "Extending Population–Based Incremental Learning To Continuous Search Spaces." *Parallel Problem Solving from Nature – PPSN V 1498*, edited by A.E. Eiben, et al. 418–427. Springer, 1998.

203. Shakya S. K., "Probabilistic model building Genetic Algorithm (PMBGA): A survey." Tech Report, August 2003. Computational Intelligence Group, School of computing, The Robert Gordon University, Aberdeen, Scotland, UK.

204. Shanmugan K. S. and Breipohl A. M. *Random Signals Detection, Estimation and Data Analysis*, chapter 3, 126. Wiley, 1988.

205. Sklar B. *Digital Communications* (1st Edition), *I*. Engle Cliffs, New Jersey: Prentice Hall, 1988.

206. Spiegel M. R. and Stephens L. J. *Theory and Problems of Statistics*, *1*. 3rd. McGraw-Hill, 1999.

207. Srinivas N. and Deb K. *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*. Technical Report, Kanpur, India: Department of Mechanical Engineering, Indian Institute of Technology, 1993.

208. Srinivas N. and Deb K. "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, *2*(3):221–248 (Fall 1994).

209. Sttzle T. "Iterated Local Search for the Quadratic Assignment Problem," *Technical Report AIDA-99-03* (1999).

210. Syswerda G. "Simulated Crossover in Genetic Algorithms." *Foundations of Genetic Algorithms 2* edited by L. Darrell Whitley, 239–255, San Mateo, CA: Morgan Kaufmann, 1993.

211. T. T. G. and Wallace D. "Multi-Modal Optimization Using Genetic Algorithms," *CADlab, MIT, Cambridge*, (Technical Report 96.02) (1996).

212. Taillard E. D. "Comparison Of Iterative Searches for the Quadratic Assignment Problem," *Location Science*, *3*:87–105 (1995).

213. Thomas C. M., Weidner M. Y., and Durrani S. H. "Digital Amplitude-Phase Keying with M-ary Alphabets," *IEEE Transactions on Communcations*, *Com-22*(2):pg 168–180 (Feb 1974).

214. Thompson M. "Application of Multi Objective Evolutionary Algorithms to Analogue Filter Tuning," *First International Conference on Evolutionary Multi-Criterion Optimization*, 546–559 (2001). Springer-Verlag. Lecture Notes in Computer Science No. 1993.

215. USC C. B. *New American Bible*, chapter Ecclesiastes, verse:12. 3211 4th Street, N.E., Washington, DC 20017-1194 (202) 541-3000: Confraternity of Christian Doctrine, Inc., November, 11 2002.

216. Van Veldhuizen D. A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD dissertation, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

217. Van Veldhuizen D. A. and Lamont G. B. "Evolutionary Computation and Convergence to a Pareto Front." *Late Breaking Papers at the Genetic Programming 1998 Conference*, edited by John R. Koza. 221–228. Stanford University, California: Stanford University Bookstore, July 1998.

218. Van Veldhuizen D. A. and Lamont G. B. *Multiobjective Evolutionary Algorithm Research: History and Analysis*. Technical Report TR-98-03, Wright-Patterson AFB, Ohio: Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, 1998.

219. Van Veldhuizen D. A. and Lamont G. B. “MOEA Test Suite Generation, Design & Use.” *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, edited by Annie S. Wu. 113–114. July 1999.

220. Van Veldhuizen D. A. and Lamont G. B. “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art,” *Evolutionary Computation*, *8*(2):125–147 (2000). Air Force Research Labortory.

221. Van Veldhuizen D. A., Sandlin B. S., , Marmelstein R. M., and Lamont G. B. “Finding Improved Wire-Antenna Geometries with Genetic Algorithms.” *Proceedings of the 1998 International Conference on Evolutionary Computation*, edited by David B. Fogel. 102–107. Piscataway, New Jersey: IEEE, 1998.

222. Veldhuizen D. A. V. and Lamont G. B. “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art,” *Evolutionary Computation*, *8*(2):125–147 (2000).

223. Veldhuizen V. D. A., Zydallis J. B., and Lamont G. B. “Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms,” *IEEE Transactions on Evolutionary Computation*, *7*:144–173 (April 2003).

224. Venkatraman J., Shankaramma S. C., and Balaram P. “Design fo Folded Peptides,” *American Chemical Society*, *101*:3131=3152 (2001).

225. Weisstein E. W. “Markov Chain.” From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/MarkovChain.html`, Feb 2005.

226. Weisstein E. W. “NP-Problem,” *From MathWorld–A Wolfram Web Resource* (2005). `http://mathworld.wolfram.com/NP-Problem.html`.

227. Whitley D. L. “A Genetic Algorithm Tutorial,” *Statistics and Computing*, *4*:65–85 (1994).

228. Whitley L. D. “Fundamental Principles of Deception in Genetic Search.” *Foundations of genetic algorithms* edited by Gregory J. Rawlins, 221–241, San Mateo, CA: Morgan Kaufmann, 1991.

229. Wilson B. E. *An Introduction to Scientific Research*. McGraw-Hill, 1952.

230. Wolpert D. H. and Macready W. G. “No Free Lunch Theorems for Optimization,” *IEEE Transactions on Evolutionary Computation*, *1*(1):67–82 (April 1997).

231. Wu A. S. *Non-Coding Segments and Floating Building Blocks for the Genetic Algorithm*. PhD dissertation, University of Michigan, December 1995. Available via anonymous ftp from: `ftp.eecs.umich.edu` in the directory `people/aswu/thesis`.

232. Zeng S., Ding L., Chen Y., and Kang L. “A New Multiobjective Evolutionary Algorithm: OMOEA.” *Proceedings of the 2003 Congress on Evolutionary Com-*

putation (CEC'2003)2. 898–905. Canberra, Australia: IEEE Press, December 2003.

233. Zeng S. Y., Yao S., Kang L., and Liu Y. "An Efficient Multi-objective Evolutionary Algorithm: OMOEA-II.." *EMO*. 108–119. 2005.

234. Zhang Q. and Leung Y.-W. "An Orthogonal Genetic Algorithm for Multimedia Multicast Routing," *IEEE Transactions Evolutionary Computation*, *3*(1):53–62 (1999).

235. Zitzler E. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

236. Zitzler E., Deb K., and Thiele L. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, *8*(2):173–195 (Summer 2000).

237. Zitzler E., Laumanns M., and Thiele L. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, edited by K. Giannakoglou, et al. September 2001.

238. Zitzler E., Laumanns M., and Thiele L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, May 2001.

239. Zitzler E., Laumanns M., Thiele L., Foneseca C. M., and da Fonseca V. G. "Why Quality Assessment Of Multiobjective Optimizers Is Difficult." *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, edited by W. B. Langdon, et al. 666–674. New York: Morgan Kaufmann Publishers, 9-13 July 2002.

240. Zitzler E., Laumanns M., Thiele L., Fonseca C. M., and Grunert da Fonseca V. "Why Quality Assessment of Multiobjective Optimizers Is Difficult." *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, edited by W.B. Langdon, et al. 666–673. San Francisco, California: Morgan Kaufmann Publishers, July 2002.

241. Zitzler E. and Thiele L. "Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study." *Parallel Problem Solving from Nature V*, edited by A. E. Eiben. 292–301. Amsterdam: Springer-Verlag, September 1998.

242. Zitzler E. and Thiele L. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, *3*(4):257–271 (November 1999).

243. Zydallis J. B. *Explicit Building-Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development*. Dissertation, Air Force Institute of Technology, AFIT/ENG, Bldg 642, 2950 Hobson way, WPAFB (Dayton) OH 45433-7765, Feb 2002.

244. Zydallis J. B. *Explicit Building-Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development*. PhD dissertation, Air Force Institute of Technology, Department of the Air Force, Air University, Wright-Patterson, Airforce Base, Ohio, USA, March 2003.

245. Zydallis J. B. and Lamont G. B. "Explicit Building-Block Multiobjective Evolutionary Algorithms for NPC Problems." *Congress on Evolutionary Computation (CEC'2003)4*. 2685–2695. Piscataway, New Jersey: IEEE Service Center, December 2003.

246. Zydallis J. B., Veldhuizen D. A. V., and Lamont G. B. "A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA–II." *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 226–240, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

Vita

Captain Richard Orison Day enlisted in the United States Air Force on December 7th, 1988. He was assigned to the 611th Aerial Port Squadron, Osan Air Base in Korea where he began taking classes through the University of Maryland's Asian Division. In May of 1993 he had a permanent change of duty station and was assigned to the 1st Communications Squadron, Langley Air Force Base, where he continued to take classes at Christopher Newport University, Newport News, Virginia. In 1995, he applied and was accepted into the Airman Education and Commissioning Program (AECP). Directly following his acceptance into this competitive program he attended Clemson University earning a Bachelors of Science in Computer Engineering in August of 1997. He was commissioned through Officer Training School on December 5th of that same year. Following commissioning he completed an introductory Acquisition Officer course at Lackland Air Force Base, Texas and was assigned to the National Air Intelligence Center at Wright Patterson Air Force Base (WPAFB), Ohio where he was the first officer assigned to the Open Skies Treaty Media Processing Facility (OSMPF). Then Lt Day, planned, organized, and executed the placement of media tracking computer systems and media processing computer support systems and software for the entire operation at the OSMPF. He then moved from the Air Intelligence Command to attend the Air Force Institute of Technology where he graduated with a Masters of Science in Computer Engineering, March 2002. Captain Day continued his educational pursuits of a Doctorial of Philosophy degree in Electrical Engineering and expects to graduate in September 2005.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 16–06–2005	**2. REPORT TYPE** PhD Dissertation	**3. DATES COVERED** *(From — To)* Mar 2002 – Jun 2005

4. TITLE AND SUBTITLE Explicit Building Block Multiobjective Evolutionary Computation: Methods and Applications	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Day, Richard O., Captain, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management 2950 Hobson way, Building 641 WPAFB OH 45434	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DS/ENG/05-03
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Capt Abel S. Nunez, Air Force Research Laboratory (AFRL), Sensors Directorate, Electronic Warfare Technology Branch (Abel.Nunez@wpafb.af.mil, WPAFB OH 45433), Dr. Robert L. Ewing, AFRL, Embedded Information Systems Engineering and Technologies Branch (Robert.Ewing@wpafb.af.mil, 2241 Avionics Circle, WPAFB, OH 45422), and Dr. Ruth Pachter, AFRL, Materials Directorate (Ruth.Pachter@wpafb.af.mil, WPAFB OH 45433).	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/IFTA, AFRL/SNRW, AFRL/MLPJE
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for Public Release; Distribution Unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This dissertation presents principles, techniques, and performance of evolutionary computation optimization methods. Evolutionary computation concepts examined are algorithm convergence, population diversity and sizing, genotype and phenotype partitioning, archiving, BB concepts, parallel evolutionary algorithm (EA) models, robustness, visualization of evolutionary process, and performance in terms of effectiveness and efficiency. Additional contributions include the extension of explicit BB definitions to clarify the meanings for good single and multiobjective BBs and a new visualization technique is developed for viewing genotype, phenotype, and the evolutionary process in finding Pareto front vectors. The culmination of this research is explicit BB state-of-the-art MOEA technology based on the MOEA design, BB classifier type assessment, solution evolution visualization, and insight into MOEA test metric validation and usage as applied to the following: test suite, deception, bioinformatics, unmanned vehicle flight pattern, and digital symbol set design MOPs.

15. SUBJECT TERMS

Evolutionary Computation, Multiobjective Evolutionary Algorithms, Multiobjective Building Blocks, Building Block Builder, Building Blocks, Building Block Evolution Visualization, MOMGA-IIa, Protein Structure Prediction, PSP, Multiobjective Quadratic Assignment Problem, mQAP, M-ary Digital Symbol Set Design, QAM, UAV, OAV

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Gary B. Lamont, AD-24, USAF (ENG)
U	U	U	UU	498	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255–3636, ext 4718