

1-1-2011

Multi-objective cultural algorithms

Dapeng Liu
Wayne State University,

Follow this and additional works at: http://digitalcommons.wayne.edu/oa_dissertations

Recommended Citation

Liu, Dapeng, "Multi-objective cultural algorithms" (2011). *Wayne State University Dissertations*. Paper 318.

This Open Access Dissertation is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Dissertations by an authorized administrator of DigitalCommons@WayneState.

MULTI-OBJECTIVE CULTURAL ALGORITHMS

by

DAPENG LIU

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2011

MAJOR: COMPUTER SCIENCE

Approved by:

Advisor

Date

DEDICATION

To my family I dedicate this work.

ACKNOWLEDGEMENTS

First I would like to thank the members of my committee for their interest and insights. My thanks especially go to Dr. Robert G. Reynolds, my advisor, for all of his efforts and guidance along my road to completing the thesis. I would also like to thank Dr. Farshad Fotouhi, who has supported me for many years with his trust. Also, Dr. Weisong Shi, has always helped me without hesitation. Finally, thanks to Dr. Ratna Babu Chinnam who has always been kind and considerate even though we had never met in person before.

Secondly I would like to thank current and previous members of the Artificial Intelligence Laboratory for their efforts. They include Dr. Xiangdong Che, Chris Best, Dr. Bin Peng, and Dr. Mostafa Ali. My work is based on their previous achievements.

TABLE OF CONTENTS

Dedication	ii
Acknowledgements.....	iii
List of Tables	vii
List of Figures	xii
CHAPTER 1 Introduction and Motivation.....	1
1.1 Particle Swarm Optimization (PSO) and Multi-objective PSO	3
1.2 Ant Colony Optimization (ACO) and Multi-objective ACO (MOACO)	5
1.3 Predator-prey co-evolution	6
1.4 Cultural Algorithms (CA) and Multi-objective CAs.....	7
1.5 Organization of the dissertation	10
CHAPTER 2 Literature view of MOEAs	11
2.1 Non-socially-motivated Optimization algorithms.....	12
2.2 Socially motivated optimization algorithms	13
CHAPTER 3 Introduction to the Cultural Algorithm	29
3.1 History of the CA Implementation.....	30
3.2 The Cultural Algorithm Framework	34
3.3 Effectiveness of MOCA and weakness.....	49
3.4 Expansion of the CA for Multi-objective Optimization.....	54
3.5 Conclusions	57
CHAPTER 4 The MOCAT 1.0 System Design	59
4.1 System Framework	60
4.2 Knowledge sources	65
4.3 The accept() function.....	71
4.4 The update() function	73
4.5 The population space.....	75
4.6 The influence() function.....	81
4.7 Summary	88
CHAPTER 5 An Example MOCAT Run	90
5.1 Start MOCAT 1.0	90

5.2	Control the evolution.....	95
5.3	Collected data	98
CHAPTER 6 Experimental Framework.....		102
6.1	ZDT 1	103
6.2	ZDT 2	104
6.3	ZDT 3	106
6.4	ZDT 4	107
6.5	ZDT 5	109
6.6	ZDT 6	110
6.7	Summaries	112
6.8	Hardware and software configurations.....	114
6.9	MOCAT initialization	114
6.10	Definition of fitness error	115
6.11	Hypotheses	116
6.12	Collection of data.....	117
CHAPTER 7 Solving Problems with Convex Pareto fronts using MOCAT: ZDT1.....		118
7.1	The ZDT 1 Function Specification.....	119
7.2	Performance of the Cultural Algorithm Using a Spread Metric on ZDT1 with 30 Dimensions.....	120
7.3	Using Hyper Volume to Guide MOCAT for ZDT1 in 30 dimensions	129
7.4	Solving ZDT1 with dimension 30 using the combined metrics	133
7.5	Summary of ZDT1.....	141
CHAPTER 8 The performance of mocat on problems with concave pareto fronts: dtz2		143
8.1	Using MOCAT To Find Optimal Pareto Front for Concave Problems: ZDT 2 with 30 dimensions.....	143
8.2	Solving ZDT 2 30 with Homogeneous Hyper-Volume Metric Performance Function	154
8.3	ZDT 2 30 with combined metrics	158
8.4	Summary of ZDT2.....	165
CHAPTER 9 solving Experiment Results of ZDT3		167
9.1	ZDT 3 30 with homogeneous Spread metrics.....	167
9.2	Performance of MOCAT using the spread metric for zdt3 30	169
9.3	ZDT 3 30 with homogeneous Hyper-volume metrics	179

9.4	ZDT 3 30 with combined metrics	184
9.5	Summary of ZDT3.....	190
CHAPTER 10 Using MOCAT to Solve Multi-Modal Problems: ZDT4.....		192
10.1	ZDT 4 10 with homogeneous Spread metrics.....	192
10.2	ZDT 4 10 with homogeneous Hyper-volume metric.....	204
10.3	ZDT 4 10 with combined metrics	208
10.4	Summary of ZDT4.....	215
CHAPTER 11 Experimental Results for mocat on deceptive problems: ZDT5		217
11.1	An Example of the Evolution of ZDT5 Optimal Curve	218
11.2	ZDT 511 with homogeneous Hyper-volume metrics	229
11.3	ZDT 511 with combined metrics	234
11.4	Summary of the MOCAT performance for ZDT5	241
CHAPTER 12 Experiment Results of ZDT6		243
12.1	ZDT 6	243
12.2	ZDT 610 with homogeneous Spread metrics.....	252
12.3	ZDT 610 with homogeneous Hyper-volume metrics.....	256
12.4	ZDT 610 with combined metrics	261
12.5	Summary of ZDT6.....	267
CHAPTER 13 Conclusion And Future Work		268
Appendix—System Implementation.....		272
References		307
Abstract.....		320
Autobiographical Statement.....		323

LIST OF TABLES

Table 3-1 Pareto fronts at year 40 for DTLZ1	50
Table 3-2 Pareto fronts at the end of evolution	52
Table 4-1 The Pseudo Code and Execution Flow Chart of MOCAT	61
Table 4-2 Pseudo code of non-domination Pareto ranking	72
Table 4-3 Pseudo code of function accept()	72
Table 4-4 Pseudo code of selecting a knowledge source from the roulette wheel	74
Table 4-5 Pseudo code of selecting a topology from the roulette wheel.....	83
Table 4-6 neighbors voting for knowledge source to use on an individual Ind	84
Table 4-7 Pseudo code of influence of normative knowledge source.....	84
Table 4-8 Pseudo code of influence of topographical knowledge source	85
Table 4-9 Pseudo code of influence of situational knowledge source	86
Table 4-10 Pseudo code of influence of historical knowledge source.....	86
Table 4-11 Pseudo code of influence of domain knowledge source	87
Table 6-1 Existing benchmark results I	112
Table 6-2 Existing benchmark results II	112
Table 7-1 Statistics for the fitness errors of ending solutions	124
Table 7-2 Use Count of Topologies of each run.....	125
Table 7-3 The number of Individuals influenced by KS using the Spread Metric	126
Table 7-4 The total number of non-dominated solutions generated over all 20 runs for each KS Topology pairing.	128
Table 7-5 Variability of Topology-Knowledge tuple Production.	128
Table 7-6 Statistics for the fitness errors of ending solutions	129
Table 7-7 Use Count of Topologies of each run.....	130
Table 7-8 Using Hyper-volume metric #Individuals influenced by KS	131
Table 7-9 Overall Statistics of Topology-Knowledge tuple	133

Table 7-10	Variability of Non-Dominated Solutions by Topology-Knowledge tuple	133
Table 7-11	Statistics for the fitness errors of ending solutions	134
Table 7-12	Use Count of Topologies using spread metric of each run	135
Table 7-13	Use Count of Topologies using hyper-volume metric of each run	136
Table 7-14	Using Spread metric #Individuals influenced by KS	137
Table 7-15	Using Hyper-volume metric #Individuals influenced by KS	138
Table 7-16	Overall Statistics of Topology-Knowledge tuple	139
Table 7-17	Randomness of Topology-Knowledge tuple	140
Table 8-1	Statistics for the fitness errors of ending solutions	148
Table 8-2	Use Count of Topologies of each run	150
Table 8-3	The Number of Individuals Influenced by the Spread Metric for Each Knowledge Source.	152
Table 8-4	Overall Statistics of Topology-Knowledge tuple	153
Table 8-5	Randomness of Topology-Knowledge tuple, Standard deviation divided by the mean	153
Table 8-6	Statistics for the fitness errors of ending solutions	154
Table 8-7	Use Count of Topologies of each run for the Hyper-Volume.....	155
Table 8-8	Using Spread metric #Individuals influenced by KS	156
Table 8-9	Overall Statistics of Topology-Knowledge tuple	157
Table 8-10	Randomness of Topology-Knowledge tuple	158
Table 8-11	Statistics for the fitness errors of ending solutions	158
Table 8-12	Use Count of Topologies using spread metric of each run	159
Table 8-13	Use Count of Topologies using hyper-volume metric of each run	160
Table 8-14	Using Spread metric #Individuals influenced by KS	162
Table 8-15	Using Hyper-volume metric #Individuals influenced by KS	162
Table 8-16	Overall Statistics of Topology-Knowledge tuple	164
Table 8-17	Randomness of Topology-Knowledge tuple	164
Table 9-1	Statistics for the fitness errors of ending solutions	176
Table 9-2	Use Count of Topologies of each run	176

Table 9-3 Using Spread metric #Individuals influenced by KS	178
Table 9-4 Overall Statistics of Topology-Knowledge tuple	179
Table 9-5 Randomness of Topology-Knowledge tuple	179
Table 9-6 Statistics for the fitness errors of ending solutions	180
Table 9-7 Use Count of Topologies of each run	180
Table 9-8 Using Hyper-volume metric #Individuals influenced by KS	181
Table 9-9 Overall Statistics of Topology-Knowledge tuple	183
Table 9-10 Randomness of Topology-Knowledge tuple	183
Table 9-11 Ratios on the Belief Space roulette wheel before generation 20.....	185
Table 9-12 Use Count of Topologies using spread metric of each run	186
Table 9-13 Use Count of Topologies using hyper-volume metric of each run	186
Table 9-14 Using Spread metric #Individuals influenced by KS	187
Table 9-15 Using Hyper-volume metric #Individuals influenced by KS	188
Table 9-16 Overall Statistics of Topology-Knowledge tuple	189
Table 9-17 Randomness of Topology-Knowledge tuple	190
Table 10-1 Statistics for the fitness errors of ending solutions	199
Table 10-2 Use Count of Topologies of each run.....	200
Table 10-3 Using Spread metric #Individuals influenced by KS	201
Table 10-4 Overall Statistics of Topology-Knowledge tuple	202
Table 10-5 Randomness of Topology-Knowledge tuple	203
Table 10-6 Statistics for the fitness errors of ending solutions	204
Table 10-7 Use Count of Topologies of each run.....	205
Table 10-8 Using hyper-volume metric #Individuals influenced by KS.....	206
Table 10-9 Number of non-dominated solutions generated by Topology-Knowledge tuples	207
Table 10-10 Randomness of Topology-Knowledge tuple	208
Table 10-11 Statistics for the fitness errors of ending solutions	209
Table 10-12 Use Count of Topologies using spread metric of each run	210

Table 10-13 Use Count of Topologies using hyper-volume metric of each run	211
Table 10-14 using Spread metric #Individuals influenced by KS.....	212
Table 10-15 Using Hyper-volume metric #Individuals influenced by KS	213
Table 10-16 generation of Non-Dominated Solutions using the Topology-Knowledge tuple.....	214
Table 10-17 Randomness of Topology-Knowledge tuple	214
Table 11-1 Statistics for the fitness errors of ending solutions	223
Table 11-2 Use Count of Topologies of each run.....	224
Table 11-3 Using Spread metric #Individuals influenced by KS	225
Table 11-4 The Generation of non-dominated solutions by Topology-Knowledge tuples.....	227
Table 11-5 Randomness of Topology-Knowledge tuple	227
Table 11-6 Statistics for the fitness errors of ending solutions	229
Table 11-7 Use Count of Topologies of each run.....	230
Table 11-8 Using Hyper-volume metric #Individuals influenced by KS	231
Table 11-9 Production of non-dominated solution by Topology-Knowledge tuple	232
Table 11-10 Randomness of Topology-Knowledge tuple	233
Table 11-11 Statistics for the fitness errors of ending solutions	234
Table 11-12 Use Count of Topologies using spread metric of each run	235
Table 11-13 Use Count of Topologies using hyper-volume metric of each run	236
Table 11-14 Using Spread metric #Individuals influenced by KS	237
Table 11-15 Using Hyper-volume metric #Individuals influenced by KS	238
Table 11-16 Generation of non-dominated solutions by Topology-Knowledge tuples.....	239
Table 11-17 Randomness of Topology-Knowledge tuple	239
Table 12-1 Statistics for the fitness errors of ending solutions	252
Table 12-2 Use Count of Topologies of each run.....	253
Table 12-3 Using Spread metric #Individuals influenced by KS	254
Table 12-4 Number of non-dominated solutions produced by Topology-Knowledge tuples	255
Table 12-5 Randomness of Topology-Knowledge tuple	256

Table 12-6 Statistics for the fitness errors of ending solutions	256
Table 12-7 Use Count of Topologies of each run.....	257
Table 12-8 Using Hyper-volume metric #Individuals influenced by KS	258
Table 12-9 Generation of non-dominated individuals by Topology-Knowledge tuples.....	259
Table 12-10 Randomness of Topology-Knowledge tuple	260
Table 12-11 Statistics for the fitness errors of ending solutions	261
Table 12-12 Use Count of Topologies using spread metric of each run	262
Table 12-13 Use Count of Topologies using hyper-volume metric of each run	263
Table 12-14 Using Spread metric #Individuals influenced by KS	264
Table 12-15 Using Hyper-volume metric #Individuals influenced by KS	264
Table 12-16 Non-dominated solutions produced by Topology-Knowledge tuple.....	266
Table 12-17 Randomness of Topology-Knowledge tuple	266

LIST OF FIGURES

Figure 1.1 Temporal and Spatial Scales of MOEAs.....	3
Figure 1.2 Basic Framework of the Cultural Algorithm.....	8
Figure 2.1 Pseudo code of the classic PSO.....	15
Figure 2.2 Graphical representation of the hyper-plane distribution of EMOPSO.....	21
Figure 2.3 Pseudo code of classic ACO.....	25
Figure 3.1 Structure of the Belief Space of CADE.....	31
Figure 3.2 Pseudo code of classic CA.....	35
Figure 3.3 Topology of lbest.....	39
Figure 3.4 Topology of global.....	40
Figure 3.5 Topology of square.....	41
Figure 3.6 Topology of star.....	42
Figure 3.7 Topology of circle.....	43
Figure 3.8 Structure of Topographical Knowledge: different space volumes.....	46
Figure 3.9 Normative Knowledge.....	47
Figure 3.10 Data structure of Historical Knowledge.....	49
Figure 3.11 Representative illustration of partial aggregation in population.....	53
Figure 3.12 Illustration of calculation of performance of knowledge sources.....	56
Figure 4.1 MOCAT System.....	61
Figure 4.2 Object diagram of MOCAT.....	64
Figure 4.3 The structure of Normative Knowledge.....	66
Figure 4.4 The structure of situational knowledge.....	66
Figure 4.5 The structure of historical knowledge.....	68
Figure 4.6 Topographical knowledge source provides a mosaic view of Pareto front.....	69
Figure 4.7 The structure of topographical knowledge.....	70

Figure 4.8 Network of Knowledge sources	71
Figure 4.9 Class of Agent.....	76
Figure 4.10 Cuboid metric for Pareto front	78
Figure 4.11 Illustration of calculation of IGD	79
Figure 4.12 Hyper-volume of Pareto fronts.....	81
Figure 4.13 Execution of MOCAT	89
Figure 5.1 To run MOCAT model.....	91
Figure 5.2 Startup of MOCAT	92
Figure 5.3 Parameters of ZDT1	94
Figure 5.4 MOCAT is ready to run	95
Figure 5.5 Step run of MOCAT	96
Figure 5.6 Run of MOCAT.....	97
Figure 5.7 MOCAT finishes on run	98
Figure 5.8 Data collected for one single run	100
Figure 5.9 Data files after several experiment runs.....	101
Figure 6.1 ZDT1 Pareto front.....	104
Figure 6.2 ZDT2 Pareto front.....	105
Figure 6.3 ZDT3 Pareto front.....	107
Figure 6.4 ZDT4 Pareto front.....	108
Figure 6.5 ZDT5 Pareto front	110
Figure 6.6 ZDT6 Pareto front.....	111
Figure 6.7 Fitness error vs. minimal distance to real Pareto front	116
Figure 7.7.1 ZDT1 Pareto front.....	120
Figure 7.2 ZDT1 population at generation 4.	121
Figure 7.3 ZDT1 at generation 10.....	122
Figure 7.4 ZDT1 at generation 20.....	123
Figure 7.5 Overall found Pareto front.....	124

Figure 7.6 Overall found Pareto front.....	140
Figure 8.1 ZDT2 Pareto front.....	144
Figure 8.2 (a-d). A series of screen shot s showing how the Pareto front is constructed over time.....	148
Figure 8.3 Overall Computed Pareto front	150
Figure 8.4 Overall found Pareto front.....	165
Figure 9.1 ZDT3 Pareto front.....	169
Figure 9.2 (a,b,c,d,e) A series of screen copies showing the evolution of the front under the guidance of the spread metric.....	174
Figure 9.3 Overall found Pareto front.....	175
Figure 9.4 Overall found Pareto front.....	184
Figure 9.5 Overall found Pareto front.....	190
Figure 10.1 ZDT4 Pareto front.....	193
Figure 10.2 A series of screen copies along evolution (a-e)	199
Figure 10.3 Overall found Pareto front.....	203
Figure 10.4 Overall found Pareto front.....	208
Figure 10.5 Overall found Pareto front.....	215
Figure 11.1 The sequence for Pareto front produced through cultural evolution.	223
Figure 11.2 Overall found Pareto front.....	228
Figure 11.3 Overall found Pareto front.....	233
Figure 11.4 Overall found Pareto front.....	241
Figure 12.1 ZDT6 Pareto front.....	244
Figure 12.2 A series of screen copies along evolution (a-g).....	251
Figure 12.3 Overall found Pareto front.....	252
Figure 12.4 Overall found Pareto front.....	260
Figure 12.5 Overall found Pareto front.....	266
Figure 0.1 Screen copies of MOCAT’s 3D visualization	274
Figure 0.2 Context, Projection, and agents in Repast Symphony.....	278
Figure 0.3 Splash window of the Repast Symphony.....	280

Figure 0.4 IDE of Repast Simphony.....	280
Figure 0.5 Model score	284
Figure 0.6 Setup Context loader in the runtime environment	289
Figure 0.7 Creating a Data set by extracting fields from agent class.....	290
Figure 0.8 Creating Outputter upon Data Set.....	293
Figure 0.9 Creating a 3D display (Projection) for agents	296
Figure 0.10 Double click an agent and show its properties	301
Figure 0.11 Choose Chart type.....	302
Figure 0.12 Select Data Serial for X and Y Axes.....	303
Figure 0.13 Configuration of the Chart	304
Figure 0.14 Chart is created completely	305
Figure 0.15 Real-time display of the Chart	306

CHAPTER 1 INTRODUCTION AND MOTIVATION

Optimization is defined as the process of *“finding and comparing feasible solutions until no better solution can be found”* (Deb and Kalyanmoy 2001). Solutions are evaluated based upon solution objectives and other criteria such as constraints. Evolutionary algorithms, including the Cultural Algorithm (Reynolds 1986; Reynolds 1994), and other bio-inspired approaches are frequently used to solve problems that are not tractable for traditional approaches. The application of these new computation genres are motivated by the following facts: they can find multiple solutions in one simulation run because of their population based approach; and they can identify satisfying solutions using limited resources for problems that traditional approaches take prohibitive amounts of effort to solve, etc. .

Previously, research in the field of evolutionary optimization has focused on single-objective problems. On the contrary, most real-world problems involve more than one objective where these objectives may conflict with each other. For example in manufacturing the criteria of lowering production costs may be at odds with a goal of improving product quality. If multiple objectives can be unified into one for a given problem, such a problem can be viewed as single-objective in nature. However, this can be problematic if the goals are conflicting.

The first real application of evolutionary algorithms to find multiple trade-off solutions in one single simulation run was presented by Schaffer in his doctoral dissertation (Schaffer 1984). In this work, a simply modified single-objective genetic algorithm was able to capture multiple weighted solutions; however, after a large number of iterations the algorithm tended to converge to individual optimal solutions. The next work on MOEAs came from Goldberg (Goldberg 1989) who introduced the concept

of domination. Domination is a partial order relationship in which a dominating solution is superior to dominated solutions in terms of all objectives. Ever since then, a number of researchers have developed different implementations of MOEAs using this important concept.

Fonseca and Fleming (Fonseca and Fleming 1995) proposed a multi-objective GA and identified some issues that were raised by MOEAs, such as how they affected the fitness landscape. Srinivas and Deb proposed a non-dominated sorting genetic algorithm (NSGA) (Srinivas and Deb 1994). Later, Deb and others proposed a fast and elitist MOEVA, NSGA II (Deb et al. 2000). Horn, Nafpliotis and Goldberg also proposed a niched Pareto-GA (Horn, Nafpliotis, and Goldberg 1994).

All of the approaches described above assumed a population-based approach where there was little or no interaction between individual solutions in the population. Since then, several socially motivated approaches have been proposed to solve multi-objective optimization problems. In these approaches knowledge can be explicitly exchanged between problem solvers in the solution of an optimization problem. Since multi-objective problems are particularly characteristic of complex social systems these approaches may provide insights into how such problems are dealt with in social situation. While the socially motivated approaches will be discussed in the next section in detail, several of the most popular ones are presented in the concise Figure 1.1 below in terms of the scales (spatial and temporal) of the original social system from which the approach was taken. We can see that while Ant Colony and Particle Swarm approaches focus on evolution over a short period in a limited space, the Cultural Algorithm is inherently able to address large-scale problems that spread large temporal and spatial scales, such as the evolution of human civilization. We will briefly discuss the work done on the multi-objective version for each.

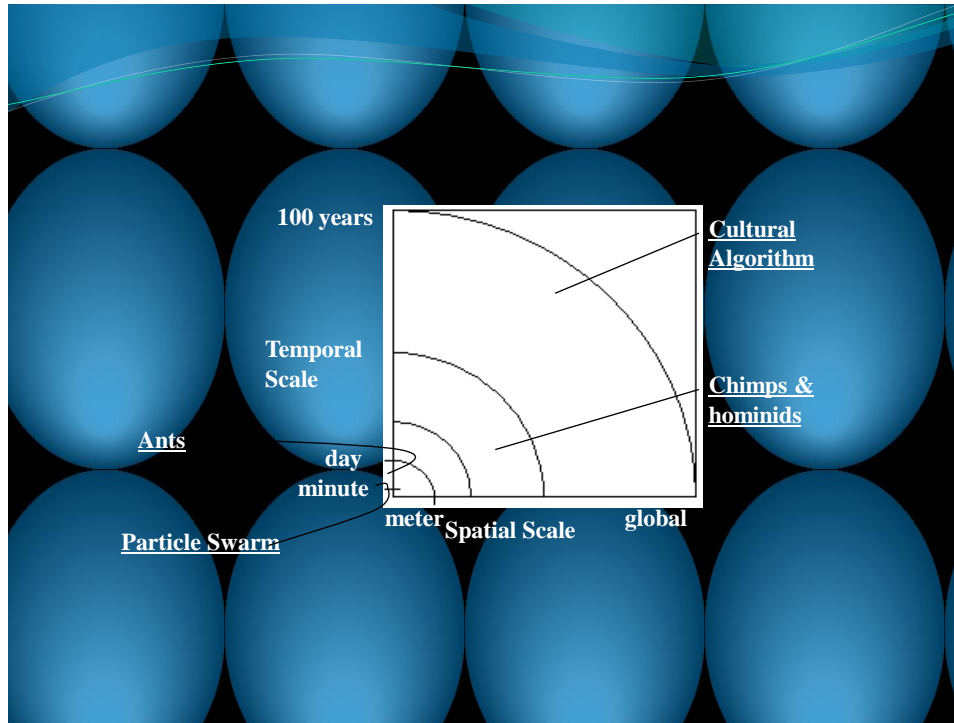


Figure 1.1 Temporal and Spatial Scales of MOEAs

1.1 Particle Swarm Optimization (PSO) and Multi-objective PSO

At the smallest conceptual scale is Particle Swarm Optimization. Particle Swarm Optimization (PSO) is a population-based socially motivated optimization technique (Eberhart and Kennedy, 1995), inspired by social behavior of bird flocking or fish schooling (Kennedy, 1998). Similar to genetic algorithms, PSO is initialized with a population of random individuals and searches for optima by updating generations. However, unlike GA, PSO has no evolutionary operators such as crossover and mutation. Instead, in each generation, each particle may change its velocity and move to a local best solution or a global one. Particles move through the problem space by following the information derived from the best of their neighbor particles. There are just a few parameters to adjust for PSO to

work. Particle swarm optimization can be used across a wide range of applications. In some cases it consumed a fraction of resources that traditional approaches might use (Shi and Eberhart, 1998).

Adopting PSO for multi-objective problems follows the success of its original version. One of the efforts was MOPSO (Coell-Coello and Salazar 2002) which borrowed the concept of Pareto dominance and used a secondary population to store the non-dominated solutions that had been found so far which helped the flock to aggregate around good solutions. In a sense, the secondary population serves the same role as of Situational knowledge in the Cultural Algorithm as we will discuss later. Later, in an improved version EMOPSO (Toscano-Pulido etc, 2007) a turbulence operator was added to further spread the flock and prevent premature convergence. Its experiments showed EMOPSO was able to depict reasonably good approximations of the Pareto front of problems with up to 30 decision variables with the by-then the least computing load. A thorough survey of multi-objective PSOs can be found in (Reyes-Sierra 2006), which suggested that in order to solve multi-objective problems, PSOs have to “1. *Maximize the number of elements of the Pareto optimal set found.* 2. *Minimize the distance of the Pareto front produced by our algorithm with respect to the true (global) Pareto front (assuming we know its location).* 3. *Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible*”, which actually represent the pursuing of all MOEAs at large.

Multi-objective PSOs have been widely used to solve various problems, such as designing planar multilayered electromagnetic absorbers in engineering (Chammani 2007); reducing harmonic current and mitigating noise on electrical utility grid (Sharaf 2009); and designing automobile transmission (Wu 2010), among others.

1.2 Ant Colony Optimization (ACO) and Multi-objective ACO (MOACO)

Another socially motivated evolutionary system is Ant Colony Optimization (ACO) (Dorigo, 1992). It is based on stigmergy, a mechanism of indirect coordination between agents or actions, where the trace left in the environment by an action stimulates, positively or negatively, the performance of a subsequent action, either by the same or a different agent.

While stigmergy occurs in many social animal societies, a typical example is that ants exchange information indirectly by depositing pheromones during their food exploitation. In this scenario, ants initially wander randomly and will return to their colony while laying down pheromone trails right after finding food. If other ants find such a path, they prefer following it to travelling at random directions. As new ants find food at the end of the trail, they will return to colony and deposit pheromone along their traces too.

If there are multiple trails to the food (which was found by multiple ants), the shortest one has greater chance to become the preference of all ants because of the phenomenon explained below. Over time, the pheromone trail starts to evaporate and its attractive strength is reduced. The more time it takes for an ant to travel down the path and back again, the more time the pheromones has to evaporate. A shortest path, by comparison, gets marched over more frequently, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there was no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea

of the ant colony algorithm is to mimic this behavior by simulating ants generating paths through a graph from an origin to a destination.

There are many variations of ACOs (Dorigo and Gambardella, 1997, Stützle and Hoos, 2000) which share the general framework of optimization by canonical ant colonies (COA). In general, any algorithm containing an exchange of information between agents via the environment, i.e., stigmergy, can be deemed an ACO. A thorough survey of ACOs can be found in (Dorigo 2006).

The first approaches to using ACO to handle multiple objectives were based on the ordering or weighting the objectives according to their relative importance (Gambardella et al 1999, T'kindt et al 2002, Doerner 2003). However, when preferences or weights cannot be given, the goal is to find a set of non-dominated solutions that are optimal in the Pareto sense. The first ACO algorithm for finding non-dominated solutions was proposed in (Iredi et al, 2001) for the bi-objective scheduling problem, in which heterogeneous multiple colonies were used. The ants differ in their preferences to either of the two criteria; and every colony uses two pheromone matrices, each suitable for one optimization criterion, so that ants were able to find different solutions along the Pareto front. Later, this approach evolved into a hybrid one (Häckel 2008) by merging Dynamic Programming and Look-Ahead Heuristic. Multi-objective ACOs have had successful applications such as portfolio selection (Doerner 2006), and the quadratic assignment problem (Ibanez 2004).

1.3 Predator-prey co-evolution

Another socially motivated approach is predator prey models. They take their cue from population biology where predator movement is based on information that is not only dynamic temporarily as in ant colony optimization but spatially as well. It is based upon the assumption that a

predator will remain in an area until the calories extracted from prey drop below a certain point relative to the calories that will be expended travelling to another location (Charnov, 1976). Adoption of predator-prey co-evolutionary models in solving MOPs is sparse. However, in one existing work Drezewski (2007) concluded that *“the tendency to lose population diversity appeared”* during the solution process. This often led to premature convergence.

1.4 Cultural Algorithms (CA) and Multi-objective CAs

The next socially motivated evolutionary learning system is the Cultural Algorithms (CA) (Reynolds 1979, Reynolds 1994), which may be popularly considered as an extension of Genetic Algorithms (GA) (Goldberg 1989). However, the CA includes much more than the GA. While the former simulates the process of population evolution, additionally, the CA recognizes interaction between individuals form a culture. In other words, the CA is a functional model of the process by which human fitness is improved through knowledge sharing in a population.

Culture is dynamic and always evolving. Human activities continuously reshape the culture, inject new material into it, and promote new stages while we are reversely defined and constrained by the extra-natural "culture" or "society" (Schwimmer 1996). This bidirectional interaction that pushes forward the cultural evolution process has a straightforward mapping in the CA. The major components of a Cultural Algorithm are the population space, the belief space, and the communication protocols, including acceptance and influence functions, through which the first two components interact. The population space can support any population-based computational model, such as Genetic Algorithms, Evolutionary Programming, etc. The primitive framework is shown in Figure 1.2.

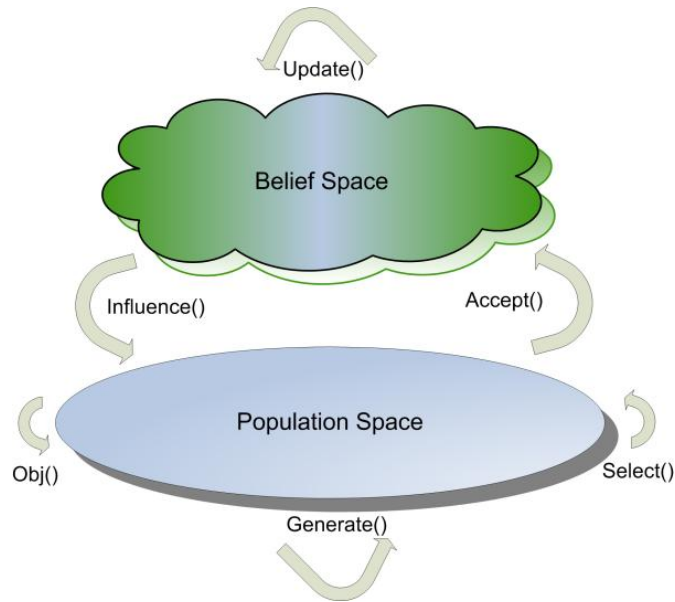


Figure 1.2 Basic Framework of the Cultural Algorithm

The Cultural Algorithm has been applied to solve a variety of problems in business, science, industry, and society. With its ability to step across a larger range of time and space, the CA was used to model the evolution of agriculture (Reynolds 1979) and to assess how humans have solved MOPs relative to the different criteria for ancient site location in the Valley of Oaxaca, Mexico (Reynolds and Nazzal 1997). These criteria related to defensive location on the one hand to accessing resources and trade routes on the other hand. As a system characterized by human activities, it provides a flexible framework (Reynolds 1994) in which to study the emergence of organizational complexity in a multi-agent system (MAS). Additionally, CA has been used to solve benchmark optimization problems (Chung and Reynolds 1998) and to solve real-valued function optimization (Jin and Reynolds 1999, Reynolds and Saleem 2005). Recently the Cultural Algorithm was applied to the study of optimization problems of varying complexity, from simple to chaotic in a “*cones world environment*” (Peng 2004, Che 2009).

As a result of its successful applications to single objective problems, CA has been adopted in solving MOPs. The first application was done by Nazzal (1997). His approach echoed that of Schaeffer in

that the different objectives were weighted as part of a single objective function. The goal was to determine what objectives were most important in driving the settlement observed in an ancient civilization in the Valley of Oaxaca, Mexico (Reynolds and Nazzari 1997). More recently, Coello-Coello (Coello-Coello and Becerra 2003) developed a CA for multi-objective problems in which they saved the non-dominated agents into an external memory, which was similar to the Cultural Algorithm's Situational knowledge and considered as a belief space. They located these non-dominated agents in the objective space, and if the external memory was full but new non-dominated agents were found, one individual who had a crowded neighborhood would be replaced. . This was thought to reduce the likelihood of premature convergence. Most recently, the original CA implementation has been expanded for MOPs (Best et al. 2009) using a wider variety of Cultural Knowledge and it was concluded that *"Cultural Algorithms are a promising technique for solving multi-objective problems"* (Best, 2009).

Even with this promising early-stage success, the multi-objective CA has still many more opportunities for improvement. Here we have developed a design for a full-fledged multi-objective Cultural Algorithm optimization engine, MOCA. The key is to investigate how the different knowledge sources present in the Cultural Algorithm can be used to guide the solution process. Specifically we will inquire about what additional information needs to be added to the knowledge sources to enable efficient searches to take place.

In this thesis, we will investigate the extent to which the Cultural Evolution process can fully support the multi-criteria optimization process. Since the spatial and temporal scale of CA encompasses all of the other approaches, multi-objective techniques used with them can in fact be applied within the cultural algorithm framework. We begin in chapter 2 by summarizing the existing achievements of multi-objective CAs' and analyzing how to take advantage of merits of the features of other MOEAs in

order to enhance the performance of a CA for multi-objective problems. Some of the features used by other MOEAs that we will exploit here will be spread and volume. We will examine in our experiments how these different features are exploited by the CA during the problem solving process.

1.5 Organization of the dissertation

The organization of the dissertation is as follows. Chapter 2 explores the literature of multi-objective algorithms, including a very brief introduction to CA. Chapter 3 then enumerates the details of previous enrichments to CAT, including the latest version, MOCAT, which is able to handle MOPs. Chapter 4 describes the design of the improved MOCAT, which is to a large extent self-adapting, by choosing a specific network configuration that most effectively helps propagating knowledge among populations. Chapter 5 presents the concrete implementation of MOCAT which can also serve as a simple user manual. Chapter 6 introduces the experimental framework with demonstrations of various system output. Chapter 7 discusses and analyzes the experiment results on ZDT1, which has a continuous concave Pareto front and may be considered as easy for some MOEAs. Chapter 8 focuses on experiment results on ZDT2, which is similar to ZDT1 but has a convex Pareto front. Chapter 9 analyzes the experiment results on ZDT3, whose Pareto front consists of five separate segments. Chapter 10 discusses the experiment results on ZDT4, whose Pareto front has a similar shape with ZDT1 but is discrete. Chapter 11 presents experiment results on ZDT5, which has string encoding for individuals instead of normal numeric representation and was missed in some related literature. Chapter 12 discusses about experiment results on ZDT6 whose Pareto front does not cross the whole objective space. Each of the ZDT benchmark problems has special characteristics which are discussed in details in corresponding chapter. Finally, Chapter 13 summarizes all experiment results and concludes the dissertation.

CHAPTER 2 LITERATURE VIEW OF MOEAS

Most initial research in the optimization field focused on single-objective problems. Since there is only one objective, all solutions can be compared based on one criterion, naturally, the solutions together compose a totally ordered set. As a consequence, making a choice between two solutions is generally uncontested.

If multiple objectives can be unified into one objective function, i.e., the objectives are not conflicting with each other, such a problem is a single-objective one in nature because the minimum solution corresponding to any objective function is the same (Deb, 2001). And, certainly single-objective algorithms can solve them by combining multiple objectives into one. Those problems that contain no solutions that are optimal for all objectives are commonly named well-formed (wiki 2010, Beheshti and Rahmani 2009) and are the focus of interest in this study. Traditional single-objective optimization algorithms cannot be adopted for well-formed MOPs without some modification. In other words, single-objective optimization is not a degenerate case of multi-objective optimization but rather the latter is not merely a simple extension of the former.

Among optimization algorithms, Evolutionary Algorithms (EAs) (Deb, 2001) mimic nature's evolutionary process in order to direct its search towards optimal solutions: reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness is determined in the environment within which the solutions live. Evolution of the population takes place after the repeated application of the above operations. Since in each round of evolution a population exists and is processed, the outcome of an EA is the existence of a set of solutions. The ability of EAs to produce multiple optimal solutions in one single

simulation run makes them ideal in solving MOPs, especially while classical optimization methods can at best find one solution in one simulation run. Such EAs are called Multi-Objective Evolutionary Algorithms (MOEAs).

Based upon the type of interaction between agents, MOEAs are divided into two genres, non-socially-motivated, and socially-motivated. In the latter genre, knowledge can be explicitly exchanged, either directly or indirectly, or both, between problem solvers in the solution of an optimization problem. It is a common occurrence within social systems to have agents deciding between numerous conflicting objectives. In fact, we argue that for cultures to survive they must support mechanisms for the solution of such multiple objective problems. As a result, socially-motivated MOEAs are prominently pervasive and successful in solving MOPs, and thereafter in this thesis they are of our special focus. However, to complete this section, non-socially-motivated MOEAs are introduced as well.

There are a few popular MOEA approaches, among which are PSO, ACO, and the Cultural Algorithms. In the rest of this chapter we will go through each optimization approach by describing their unique characteristics and how they are proposed to handle MOPs.

2.1 Non-socially-motivated Optimization algorithms

The first real application of evolutionary algorithm was a non-socially motivated one which tried to find multiple trade-off solutions in one single simulation run (Schaffer 1984). In fact, this work was a modified single-objective genetic algorithm that had the ability to capture multiple weighted solutions. As the first exploration into multi-objective optimization, the algorithm tended to converge to individual optimal solutions after a large number of iterations. The next work on MOEAs (Goldberg 1989) successfully brought up the concept of domination in its simple 10-line code draft. Ever since then, a

number of researchers have developed different implementations of MOEAs using this important concept. Domination is a partial order relationship in which a dominating solution is superior to dominated solutions in terms of all objectives. In his book Goldberg (Goldberg 1989) pointed out that while traditional numerical calculation algorithms were efficient to some specific problems, they were not as robust as evolutionary algorithms.

Fonseca and Fleming (Fonseca and Fleming 1995) proposed a multi-objective GA and identified some issues that were raised by MOEAs, such as how they affected the fitness landscape. Srinivas and Deb proposed a non-dominated sorting genetic algorithm (NSGA) (Srinivas and Deb 1994). Later, Deb and other proposed a fast and elitist MOEVA, NSGA II (Deb et al. 2000). Horn, Nafpliotis and Goldberg presented niched Pareto-GA (Horn, Nafpliotis, and Goldberg 1994). It was very soon verified that domination-based MOEAs could be reliably used to find good solutions. At the same time, other different versions of EAs were successful in solving MOPs as well. For example, Kursawe used diploidy approach (Kursawe 1991) which tried to gain insight into the structure of the Pareto set by computing a finite number of efficient solutions. Osyczka and Kundu proposed a distance-based GA (Osyczka and Kundu 1995). Hajela and Lin proposed weight-based approach (Hajela and Lin 1992) for designs of structural systems with a mix of continuous, integer and discrete design variables.

2.2 Socially motivated optimization algorithms

Socially motivated evolutionary approaches support the explicit interaction of individuals. These approaches are a good fit with the multi-objective approach since they will allow sub-groups of the population to work on different objectives. In the Figure 1.1 several of the most popular socially motivated approaches are presented in terms of the scales (spatial and temporal) of the original social

system from which the approach was taken. We can see that various socially motivated algorithms scale remarkably different in the two dimensions. Admittedly, the difference originated from the social system that they tried to emulate. Each of these original versions was extended to deal with multi-objective problems. In the rest of this chapter we will discuss the original work, MOP versions, and improvements done for the change for PSO, Any Colony Optimization, and Predator-Prey models. We will discuss Cultural algorithms separately in the next chapter.

2.2.1 PSOs and MO PSOs

PSO (Kennedy and Eberhart, 1995), a population based stochastic optimization technique, was first inspired by the social behavior of bird flocking or fish schooling (Kennedy, 1998) with some add-ons to simulate human social behavior.

To emulate intelligent behaviors, a small piece of memory of each agent —the best position it has been—is retained and implemented in code as an array of PBEST. Conceptually, each individual remembers its own experience and the velocity adjustment associated with PBEST has been called “*simply nostalgia*” since the individual tends to return to the place that most satisfied it in the past.

While birds and fish adjust their physical movement to avoid predators and seek food, humans adjust not only their physical movement, but also cognitive or experiential factors as well. To emulate intelligent behaviors at this level, each agent senses the globally best position, marked as GBEST, that one member of the flock had found. Obviously, GBEST is conceptually similar to publicized knowledge which all individuals seek to attain.

The pseudo code of classic PSO taken from (Eberhart et al, 2001) is shown below:

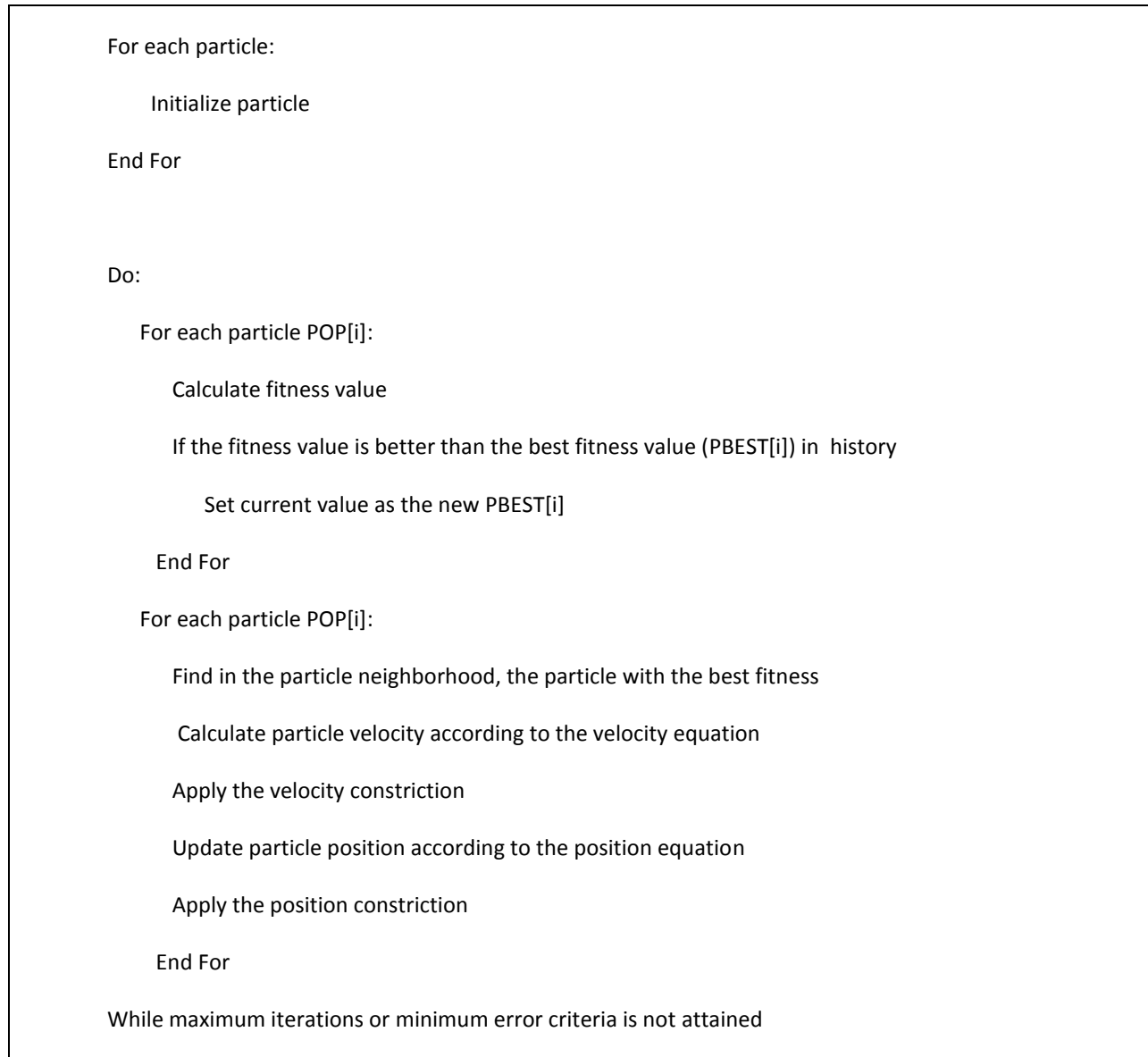


Figure 2.1 Pseudo code of the classic PSO

In general PSO is similar to GA in that it is initialized with a population of random individuals and searches for the optima by updating generations. However, unlike the GA, PSO has no evolutionary operators such as crossover and mutation. Instead, in each generation every particle may change its velocity and move towards a local best solution or a global one. As a result, the PSO requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory

requirements and speed. The side effect is that it is mainly used for problems in which only limited communication is needed between individuals; which is why in Figure 1.1 it is located near to origin point. Another problem is that in emulation, the flock can quickly settle on a unanimous, unchanging direction. Therefore, a stochastic variable called *craziness* was introduced (Eberhart et al, 2001), which was reflected as the random coefficients in the formula calculating the individual velocity. The formula used in the original PSO is described as:

$$POP^{T+1}[i] = POP^T[i] + rand() \times P_INCREMENT \times (PBEST[i] - POP^T[i]) + rand() \\ \times P_INCREMENT \times (GBEST - POP^T[i])$$

In this formula, POP represents the population. Therefore, POP[i] is an agent while superscripts T and T+1 indicate the generation number and P_INCREMENT is a constant that specifies how agile an agent is when moved to a position that is guided by both its own memory and the global optimum. In the original paper P_INCREMENT was set to 2.

Among multi-objective versions of PSO, one of the first efforts was MOPSO (Coello-Coello and Salazar 2002) which borrowed the concept of Pareto dominance to better rank the fitness of individuals and used a secondary population to store the non-dominated solutions that had been found so far to guide the flock to good solutions. The motivation of proposing MOPSO was described as *“The use of global attraction mechanism combined with a historical archive of previously found nondominated vectors would motivate convergence towards globally nondominated solutions.”* In some sense, the secondary population served the same role as Situational knowledge does in Cultural Algorithm as we will discuss later. As in the original PSO, in MOPSO each agent still retains its own memory of the best position that it has explored by-far. However, the MOPSO maintains multiple non-dominated agents in its global memory and adopted a complex calculating process to choose one of them to affect the agent

during evolution. Such global information became effective through a geographically-based approach with the intention to maintain population diversity. This corresponds to topographic knowledge in the basic cultural algorithm. The whole evolution process will be explained in detail below.

At the beginning of the evolution of MOPSO, every particle is initialized to be stationary. For each particle $POP[i]$ a memory is maintained in $PBESTS[i]$. During the evolution of $T+1^{th}$ round, the following formula was used:

$$Velocity^{T+1}[i] = W \times Velocity^T[i] + R_1 \times (PBESTS[i] - POP^T[i]) + R_2 \times (REP[h] - POP^T[i])$$

In this formula which mainly derives from the original single-objective one, W was the inertia weight and takes a value of 0.4 (while in PSO 1 was taken for granted). R_1 and R_2 are two random numbers in the range $[0, 1]$ that depict the *craziness* of this individual. $PBESTS$ is an array saving the best memory for each agent, which may be updated only when a new non-dominated agent is found,

While the whole formula is straightforward to understand, $REP[h]$ which replaced the single-value $GBEST$ in the original formula, needs extra explanation: It is a value take from the repository and is selected in the following way in each generation:

Divide the objective space into hyper-cubes (which is a multiple dimensional analog of a cube) and assign values to them that are calculated by the underneath formula:

Hyper-cubes containing no particles are assigned 0;

Hyper-cubes containing one particle are assigned the particle's fitness value;

Hyper-cubes containing multiple particles are assigned a fitness equal to the result of

$\frac{x}{\text{number of particles in this hypercube}}$, where $x > 1$ and is an any number, in the original paper, $x=10$.

This is named *fitness sharing* by giving crowded hyper-cubes small values proportional to its particle density

Apply roulette-wheel selection on these fitness values to select one hyper-cube;

Select one particle (randomly if there are more than one particle) from the selected hyper-cube, its position will be $REP[h]$.

During this step of evolution, particles will be updated while avoiding moving particles out of the search space:

$$POP^{T+1}[i] = POP^T[i] + Velocity^{T+1}[i]$$

Then, for each agent, $PBESTS[i]$, will be updated if its new position dominates the previous best ones in the memory.

By doing this MOPSO tried to improve the speed in which particles are moved towards the Pareto fronts and enlarge the diversity by fitness sharing. Later, an improved version EMOPSO (Toscano-Pulido etc, 2007) used a list of methods to improve its original algorithm by distributing non-dominated solutions.

First, EMOPSO used an adaptive grid (Knowles and Corne, 2000) which was a space formed by adjacent hyper-cubes that have as many coordinates as objective functions. Each hyper-cube represents a geographical region that contains some number of individuals. The adaptive grid allows us to store non-dominated solutions and to redistribute them when the maximum capacity of the hyper-cube is reached. When each solution is generated, its grid location in objective space is determined. As the individual appears in one hyper-cube, the edges of this hyper-cube may be bisected in the middle so

that a smaller hyper-cube containing the individual is produced. One normal cause for this further division is that the number of agents in the hyper-cube exceeds some threshold. Creating a classic hyper-cube grid can be memory prohibitive because, there are totally d dimensions and if we repeatedly bisect the range in each objective l times, there will be 2^{ld} hyper-cubes generated. For example, for a 4-objective problem with $l=5$, the objective space is divided into 1,048,576 hyper-cubes. To remedy this exponential worst-case memory usage, the adaptive grid just divides the crowded hyper-cube as needed. Two other benefits come with this adaptation. First, there is a smaller computational cost at the beginning and second, there is no pre-defined cube size is needed. The concept is exactly the same with Topographic knowledge in the Cultural Algorithms (Jin and Reynolds, 1999).

In EMOPSO, to both prevent being crowded in one hyper-cube and further reduce the load of calculation, a note of ϵ -dominance was used. ϵ -dominance is a relaxed form of dominance in which the so-called ϵ -Pareto set is an archiving strategy that maintains a subset of generated solutions. The general idea of this mechanism is to divide the objective function space into subspaces of size ϵ but no further. In other words, no edge of any hyper-cube can be smaller than this fixed value. Inside each hyper-cube, the contained agents can be interpreted as a geographical region that contains a single solution. The approach accepts a new solution into the ϵ -Pareto set if: 1) it is the only solution in the box which it belongs to; 2) it dominates other(s) solution(s); or 3) it competes against other non-dominated solutions inside the subspace, but it is closer to the origin vertex of the box. In both theory and practice, it guarantees convergence and diversity according to the value of the ϵ parameter, which defines the resolution of the grid to be adopted for the secondary population. This mechanism prevents the infinite division of the adaptive grid when the fitness is evaluated in terms of real numbers. One

practical challenge of using ϵ -dominance is the unknown box size, i.e., the ϵ parameter which is problem-dependent and normally not known before executing a MOEA.

Additionally, EMOPSO tried to distribute non-dominated solutions using so called Hyper-plane distributions. The motivation was to create a series of individuals that are well distributed and a reasonable representation of the objective hyperspace and use them to guide evolution. The algorithm divided hyperspace into $n-1$ regions. A perpendicular line to the hyper-plane was computed for each regional vertex. Then only the solutions which were closest to each line could be selected for population during evolution. While the above description is abstract, the concrete example that was given in the proposal of EMOPSO is actually easy to follow, which is copied here in Figure 2.2. In those figures, black dots represent agents that are located at the current Pareto front while the dotted line depicts the real Pareto front, shown in the top right figure. In order to efficiently guide the search to evenly cover the real Pareto front, five representative solutions, including both end points of the Pareto front, and their corresponding normal lines (lines perpendicular to the tangent planes) are marked out, as shown in the bottom left figure. An agent with the closest distance to any normal lines has priority to be propagated into the next generation over others with larger distance, which is shown in the bottom right figure. There, three agents are selected in this case while others are eliminated according to this policy.

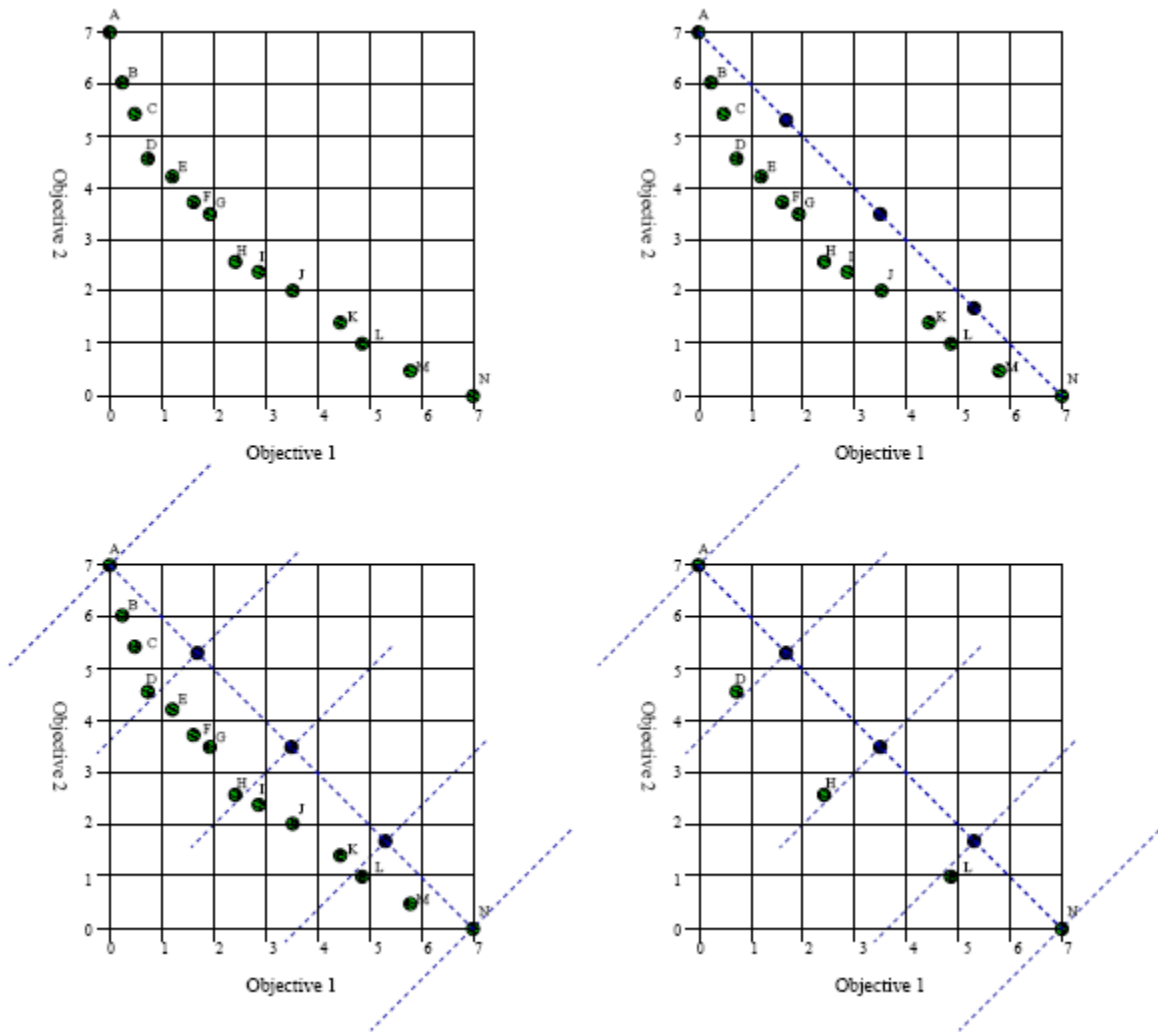


Figure 2.2 Graphical representation of the hyper-plane distribution of EMOPSO

Experiments comparing EMOPSO and NSGA-II showed that EMOPSO was able to depict reasonably good approximations of the Pareto front of problems with up to 30 decision variables with less computing load than NSGA-II. With limited computation, EMOPSO was superior to NSGA-II regarding spread on Pareto front. However, the paper also indicated that *“if allowed to perform a higher number of fitness function evaluations, the NSGA-II would be able to converge to the true Pareto front of most of these test functions”*; in other words, EMOPSO performs better only with limited computing.

In addition to the above approaches to maintaining a set of well-distributed non-dominated solutions, the turbulence operator was introduced that avoids premature convergence, which was not reflected in the above formula.

$$Probability_{Turbulence} = \left(\frac{current_generation}{total_generations} \right)^{1.7} - 2.0 \times \frac{current_generation}{total_generations} + 1.0$$

This turbulence operator alters the flight velocity of a particle in all dimensions so that the particle can move to completely isolated regions. From the formula we can see that the probability of turbulence becomes smaller along generations, i.e., the extent to which the modification caused by turbulence attenuated along with the evolution progress. This change is desired because at the beginning of the search there should be higher probability to perturb the flight of the particles so that they intend to explore unknown spaces, and at the end the particles should move slowly so that they are easy to converge to Pareto front.

As to constraint handling, EMOPSO inherited the approach proposed in (Pulido and Coello, 2004). The idea was to punish the particle that was infeasible under the constraints in selecting a leader. When comparing two particles, if both of them are feasible, the one with higher fitness value wins. If one is feasible while the other is not, the feasible one wins. If neither is feasible, the particle that has the lowest value in terms of its total violation of constraints (normalized with respect to the largest violation of each constraint achieved by any particle in the current population) wins.

So, in total, EMOPSO has the following improvements MOPSO: 1. Using norms of the evenly distributed positions along the real Pareto front to help distribute solutions obtained by the algorithm, 2. Adding a turbulence operator to spread the flock and prevent premature convergence especially at the

early stage of evolution, 3. Controlling the number of sub-swarms by using ϵ -dominance; 4. Handling constraints by punishing violating agents.

A summary of multi-objective PSOs can be found in (Reyes-Sierra 2006). There, the authors suggested that to solve multi-objective problems, PSOs have to “1. Maximize the number of elements of the Pareto optimal set found. 2. Minimize the distance of the Pareto front produced by our algorithm with respect to the true (global) Pareto front (assuming we know its location). 3. Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible” (page291). One thing worth pointing out is that in different variations of multi-objective PSOs different neighborhood topologies are used, among which there are two popular ones: fully connected and ring; which have been applied in the CAT system as well.

Recently, metrics, such as spacing and maximum spread combined, have been used to evaluate the evolution of Pareto fronts. During evolution the metrics were used to determine the switching rules between the operation modes and it was found that the combination of multiple metrics outperformed other algorithms that used only one metric (Bastos-Filho and Miranda, 2011).

2.2.2 ACO and MO ACOs

It has been long known that ants were capable of finding the shortest path from a food source to the nest without using visual cues (Hölldobler and Wilson, 1990). Ants were also observed to be capable of adapting to changes in the environment, for example, finding a new shortest path once the old one is no longer feasible due to a new obstacle. Ant Colony Optimization (ACO) (Dorigo et. Al, 1996) was inspired by the observation of laboratory ant colonies (Beckers, Deneubourg, and Goss, 1992).

It is well known that the primary means for ants to form and maintain the line is a pheromone trail. In the Ant Colony Optimization Algorithms, an ant is defined to be a simple computational agent, which iteratively constructs a solution to a problem, frequently a path planning problem. In the model, ants deposit a certain amount of pheromone, a chemical substance, while walking, and each ant probabilistically prefers to follow a direction rich in pheromone. Thus, the pheromone and the density of the pheromone along the trail is the knowledge that the ACO shares among its individual ants. Partial problem solutions are seen as states and each ant moves from a state to another one corresponding to a more complete partial solution. At each step, each ant computes a set of feasible expansions to its current state, and moves to one of these according to a probability distribution.

Since the first ACO emerged in the 1990s, several other ACO algorithms have been proposed. One thing worth pointing out is that there are many variations of ACOs (Dorigo and Gambardella, 1997, Stützle and Hoos, 2000). In general, any algorithm containing the exchange of information between agents via the environment, i.e., stigmergy, can be deemed an ACO. A thorough survey of ACOs can be found in (Dorigo 2006), which observed that ACOs had been applied for NP-hard problems, dynamic optimization problems, stochastic optimization problems, continuous-variable optimization problems, etc.

The pseudo code of the classic ACO is available from (Mariezzo and Carbonaro, 1999).

Step 1. (Initialization)

Initialize $\tau_{i\psi}, \forall i, \psi$

Step 2. (Construction)


```

For each ant  $k$  do
  repeat
    compute  $\eta_{i\psi}, \forall i, \psi$ 
    choose the state to move into
    append the chosen move to the  $k^{\text{th}}$  ant's set  $tabu_k$ 
  until ant  $k$  has completed its solution
  carry the solution to its local optimum
End For

Step 3. (Trail update)
For each ant move  $(i\psi)$  do
  computer  $\Delta \tau_{i\psi}$ 
  update the trail matrix
End For

Step 4. (Terminating condition)
If not (end condition) go to step 2.

```

Figure 2.3 Pseudo code of classic ACO

The first approaches to using ACO to handle multiple objectives were based on the ordering or weighting the objectives according to their relative importance (Mariano 1999, Gambardella et al 1999, T'kindt et al 2002, Doerner 2003); their approach suggested that they were not working on well-formed MOPs so that they are out of our concern in this thesis.

The first ACO algorithm for finding non-dominated solutions was proposed by Iredi et al (Iredi et al, 2001) for the bi-objective scheduling problem. This paper pointed out one important fact that multiple colony ant algorithms have been proposed before as parallel ACO algorithms, which were

different from multi-objective ACOs. Iredi worked on the Single Machine Total Tardiness Problem (SMTTP), which is known to be NP-complete. In this problem there are two optimization objectives: one is minimizing the sum of the waiting times of all jobs, while the other one is minimizing the changeover costs that happened when the machine switched its service target. Both objectives needed to be minimized and were intuitively conflicting with each other.

Heterogeneous multiple colonies were used Iredi. i.e., the ants differed in their preferences to either of the two criteria; and every colony used two pheromone matrices while each suitable for one optimization criterion, so that ants were able to find different solutions along the Pareto front. In the extreme case, an ant considered only one objective while totally ignoring the other one. Obviously, following this permutation patten, for n objectives, there are totally n^2 configurations, in one of the n colonies n different pheromone matrices are used. To update the pheromones, all ants in the non-dominated front of the generation were allowed to update. An ant that updated would update both pheromone matrices corresponding to the two objectives. However, Iredi pointed out that this strategy made sense only when there were not too few ants in a colony, because otherwise no real competition about best solutions could be expected to occur since the sparse ants would search in different regions of the nondominated front. Nonetheless, it was not explicitly defined how many agents were enough to avoid this problem.

Later, this approach evolved into a hybrid one (Häckel 2008) which combined dynamic programming and a Look-Ahead Heuristic (LAH) into the implementation.

Dynamic Programming (Bellman, 1957) is a method for solving complex problems by dividing them into simpler sub-steps. For example, Dijkstra's shortest path algorithm in a graph can be viewed as an instance of Dynamic Programming because in it the shortest path is the sum of the distances of the

source node to its neighbors and their neighbors to the target node, so that if we find the shortest path from the neighbors to the target node, the original problem is solved.

Generally, when Dynamic Programming is applicable, the method is expected to take less time than classical methods because solutions to the complex problem are built upon solutions to smaller sub-problems which are easier to get and cached for reuse. Because Häckel chose a problem in which all objective functions were separable and fulfilled the terms of Dynamic Programming, the algorithm was applied. The intermediate results created by the Dynamic Programming were saved into a look-up table to facilitate future search, which is called Look-Ahead Heuristic (LAH).

The test problem is a classic acyclic directed graph with three added objectives to be minimized: 1. Monetary value of cost, which was the result of summing up all weights along the certain path, 2. An aggregated probability value, which was the product of all weights along the path that described the possibility of local hazard events, 3. A value subject to the Min-Max aggregation method, which could be used as a threshold for acceptability decisions.

For an ant at a specific location, heuristic information about local status is provided to depict the length of a section or the attractiveness of a single alternative, respectively. The pheromone information includes a global memory of the ants and therewith an experience about the global attractiveness of alternatives. For the Traveling Salesman Problem (TSP) the available alternatives were built from the total number of edges that go out from the current node without the edges leading to already visited nodes. To reach higher performance, the known shortest distance from the current node to the end node was used in calculating the heuristic value so that it was named the Look Ahead Heuristic.

2.2.3 Prey-predator

Predator prey models were inspired from population biology where a predator is assumed to remain in current area until the calories extracted from prey drop below a certain point relative to the calories that will be expended travelling to another location (Charnov, 1976). Another term to describe this scenario is co-evolution.

Adoption of predator-prey co-evolution model in solving MOPs is sparse. There is one existing work (Drezewski 2007) that used the predator-prey approach without explaining why the formulas were particularly suitable for MOPs in detail. Two problems were used to test the proposed approach, CoEMAS. It turned out that on the easy problem CoEMAS was slightly better than the other two algorithms, Predator Prey Evolutionary Strategy (PPES) (Laumanns 1998) and the Niche Pareto Genetic Algorithm (NPGA) (Zitzler 1999), but on the hard problem CoEMAS clearly won out. Finally the paper concluded that *“the tendency to lose population diversity appeared”* and extra effort was needed *“especially when we consider the stable maintaining of useful population diversity”*.

In general, all such socially-motivated optimization algorithms have simple social structures and strict evolutionary mechanisms. Nonetheless, they lack competitions between different evolutionary propulsion and dynamics inside evolution. The Cultural Algorithms have such merits.

Add a conclusion section here saying what you did.

CHAPTER 3 INTRODUCTION TO THE CULTURAL ALGORITHM

In every society, some social structures have emerged as the efficient solution of problems that the society must deal with. Miraglia (Miraglia 1999) stated that, *“culture, as a body of learned behaviors common to a given human society, acts rather like a template (i.e. it has predictable form and content), shaping behavior and consciousness within a human society from generation to generation”*. From this perspective, culture is useful in guiding the problem solving activities and social interaction of individuals in the population (Reynolds 1994). Since in reality people are always facing multiple conflicting objectives and making decisions upon them, it has been conjectured that *“Culture is an optimizing process”*. Based on principles of human social evolution, Reynolds (Reynolds 1978) developed the Cultural Algorithm to model the evolution of cultural systems.

Culture is not static but evolving. The culture we have, even just a few years ago, is much different from what we have now. Human activities continuously reshape it, inject new material into it, and promote new stages. At the same time, we often find ourselves defined and constrained by the extra-natural "culture" or "society"(Schwimmer 1997). This bi-directional interaction forms the cultural evolution process which is reflected in the Cultural Algorithms.

At large, cultural evolution can be seen as an inheritance process that occurs at two levels: the micro-evolutionary level—in the population, and the macro-evolutionary level—in the culture; both at the same time and reciprocally (Melin etc, 2007). As a result of emulating cultural evolution, similarly, the CA can also be viewed as running on two levels; in other words, it is deemed as a dual inheritance system that characterizes evolution in human culture at both the macro-evolutionary level, which takes place within the belief space, and at the micro-evolutionary level, which occurs in the population space.

Knowledge produced in the population space at the micro-evolutionary level is selectively accepted or passed to the belief space and used to adjust the knowledge structures there. This knowledge can then be used to influence the changes made by the population in the next generation.

3.1 History of the CA Implementation

Elaborations to the Cultural Algorithms were made in two areas: with the belief space and within the population space. First in terms of the Belief Space Chung and Reynolds (Chung and Reynolds, 1998) began by using Situational and Normative knowledge. Situational knowledge reflects specific examples of performance for a problem. Normative knowledge reflects ranges of acceptable behavior by agents. Jin (Jin and Reynolds, 1999) added in topographic knowledge that describes aspects of the performance landscape for the problem solvers.

Reynolds and Saleem (Reynolds and Saleem 2001, Saleem 2001) then amended these Knowledge sources by introducing Domain and History Knowledge; Saleem first started integration of all knowledge sources into the same Cultural Algorithm Framework while at the beginning knowledge sources were randomly selected to affect the population. Saleem developed the Cultural Algorithms for Dynamic Environment (CADE) to investigate how knowledge structures in Belief Space affect in tracking changes in dynamic environments. The internal structure of the Belief Space is depicted in the figure below.

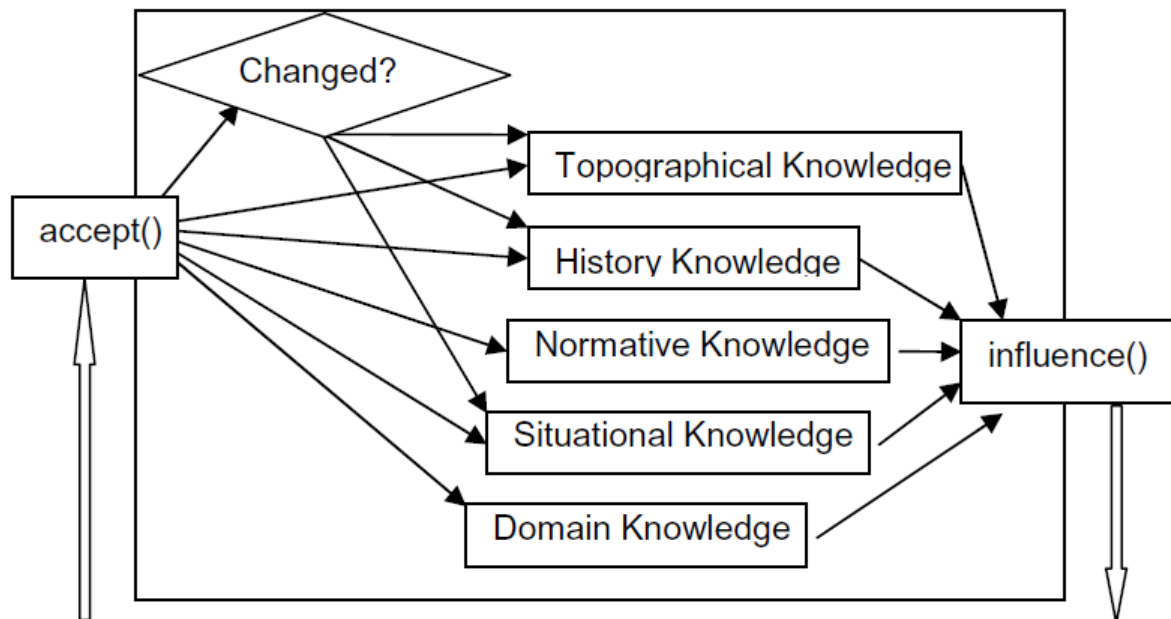


Figure 3.1 Structure of the Belief Space of CADE

It is shown in Figure 3.1 that the information extracted from the population can update multiple knowledge structures simultaneously; and two knowledge sources Historic and Situational will be additionally updated if a different evolution trend is found. In their implementation, knowledge sources were selected randomly to influence individuals. Saleem noticed that even with random selection knowledge sources performed better in some contexts.

Peng (Peng 2004, Peng 2005) found that the similarities in social structures that emerged in similar cultures were a result of the integration of different knowledge sources in the solving process. So that she proposed a biologically motivated method to integrate the application of these knowledge sources based upon the Marginal Value Theorem (Charnov 1976) to drive the problem solving process. She demonstrated that certain social structures emerged from the Cultural System as a result of this approach. Her system was implemented with Java and MATLAB was used for visualizing the experimental results. In her system, each individual agent in the population was associated with a single

knowledge source that influenced it at each time step and there was no exchange of information between agents in the population.

Up to this point individuals in the population were viewed as independent entities whose communication was indirect through their contributions to the Belief Space. Next, Ali (Ali 2008) expanded the ability of knowledge sources to influence a population through the notion of a “*social fabric*”, i.e., networks connecting individuals in the population. He embedded the Cultural Algorithms framework within the Recursive Porous Agent Simulation Tool (Repast) (North and Howe 2005) and produced a toolkit called Cultural Algorithms Simulation Toolkit (CAT) (Reynolds and Ali 2008). The network was called the “*social fabric*” because the connections between individuals were viewed to be easily modified during problem solving. The term was derived from work at IBM in which they were interested in building software to dynamically support interconnections between individuals in a software development team (Cheng, etc. 2005). The social fabric enabled the spread of knowledge through a population. In his system, knowledge sources in the Belief Space select individuals from the population to directly influence. The actual decision as to which knowledge source was to influence individual agents in the population was made by a majority vote based upon the direct influence for the agent and that of their directly connected neighbors. While the network itself was static, he allows the actual connections of the nodes in the network to be changed randomly at each time step, because Ali’s purpose was to “*investigate whether just having access to the Social Fabric is sufficient for the Knowledge sources to improve the performance of the influence function as opposed to not having a network to distribute their influence at all*”. In these initial experiments, individuals were randomly connected with others at each time step. So there was no persistent connection between any pair of individuals. However, just having the ability to distribute successful influences through the population

space was sufficient to produce statistically significant improvement in performance for a set of benchmark engineering problems.

Next, Che (2009) further investigated the effectiveness of various kinds of social fabrics in complex systems in his thesis. In his research, networks of individuals had a memory, i.e., the network organization was established before the evolution began and its topology was retained all along: the connections between individuals remained fixed over the problem solving process while the individuals move to various different positions in the domain space. Additionally, Che upgraded CAT into version 2.0. CAT 2.0 embedding the Cultural Algorithm within an agent-based problem solving framework, Repast 2.0. The agents make up the population space in the CAT systems and are interconnected with each other through a social network. With the consideration of the nature of MOPs, multiple agents are often used. A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents that can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. In the CAT system, the population component is made up of a population of Repast agents.

Most recently, the CA has been expanded to handle MOPS (Best et al 2009) by introducing non-domination sort into the implementation, which was named MOCA. Best compared MOCA to NSGA-II using two performance metrics: hyper volume (HV) (Zitzler and Thiele 1999) and generational distance (GD) (Valenzuela-Rendon, Uresti-charre, and Monterrey 1997) on the benchmark known as DTLZ (Deb, et al, 2002). With comparably good experiment results, Best concluded that the Cultural Algorithms were a promising technique for solving multi-objective problems. Meanwhile, some behaviors were observed and correspondent future work was proposed, such as the gradient approximation that was used in domain knowledge source is not always effective and more heuristics might be used; the situational and

historical knowledge sources can achieve an acceptable spread on simple problems but not on problems with very uneven density and overall spread over the Pareto front sometimes can be improved. Such proposals will be addressed in this thesis.

In the rest of this section, most introduced features of the CA have been implemented in CAT 2.0 and all multi-objective parts come from MOCA.

3.2 The Cultural Algorithm Framework

The basic CA framework is shown in Figure 1.2, from which it can be seen that the CA has three major components: a population space, a belief space, and a protocol that describes how knowledge is exchanged between the first two components, including accept process which extracts knowledge from population space into belief space, an update process that adjusts the knowledge sources based upon this new information, and the influence function which enables specific knowledge sources to influence the behavior of problem solvers in the population space.

While the CA is often viewed as an extension of GA, Cultural algorithms are additionally based on some theories proposed in sociology and archaeology to model cultural evolution, but not only GA. As a result, the population space of CA can support any population-based computational model, not exclusively Genetic Algorithms, such as Evolutionary Programming. Similarly, there are no inherent restraints for the concrete implementation of the belief space. The communication protocol can be developed independently of both the Belief Space and the Population Space as well. Thus, over time additional knowledge sources, population structures, update processes, and communication protocols have been added into various versions of the system. Some example technologies that have been added are swarm-based, predator-prey based among others.

The pseudo code of the classic CA implementation is shown in Figure 3.2 below.



Figure 3.2 Pseudo code of classic CA

In the pseudo code, P_t represents the Population at time t , and B_t for the Belief Space at the same time. When the CA begins, both the Population and the Belief Space are initialized. In normal implementations, population is initialized with randomly created individuals that distribute in the domain space and belief space is initialized without knowledge or memory.

At each generation, individuals in the Population space are first evaluated with the objective function so that the good-performing individuals are found. The acceptance function, *accept()*, is then used to determine which individuals will be allowed to update the Belief space, which normally is an elitist activity. Knowledge carried by those chosen individuals is then added to the Belief space via function *update()*. Next, knowledge from the Belief Space is instilled into some chosen individuals through the *influence()* function to produce new decision makers. By merging the old generation and new individuals and filtering them to create the next generation, whose effect is expected to be transferred into the Belief Space in the next generation, we expect the product to have better performance than its precedents as a whole. The Cultural Algorithm repeats this process for each generation until either a solution—for single-objective problems, or a good Pareto front—for multi-objective problems is identified or a predefined iteration number is reached. From this description we can see that the population and the Belief space interact with each other reciprocally in a way that is analogous to the evolution of human culture (Barkow et al 1995, Johnson and Earle 2000, Richerson and Boyd 2004).

In the rest of this section, details of these components of the basic CA will be introduced.

3.2.1 The Population Space

The Population space is the framework in which individuals contribute to the belief space, assimilate new knowledge, and may be replaced by some descendants over time. Theoretically any population model can be used for the population; however, Che selected an agent-based framework for the Cultural Algorithm Toolkit. The agent based model was based on the Repast (North and Collier 2005, Anon 2008a), an agent-based simulation environment. Agents can interact with other agents and with the knowledge sources through networks.

Frequently we need to find individuals in the population space that perform better than other peers. In single-objective problems, it is unambiguous to find the winner in terms of performance and locate the top m individuals; however, in multi-objective problems, the concept of non-dominance takes the place of simple ranking. As a general rule, if the CA wants a single good individual, it will be selected randomly from the Pareto front; if the CA wants a list of good individuals for the situational knowledge in the belief space then multiple agents are selected from the Pareto front; however, if there are not enough agents, then agents on the next Pareto front will be taken into consideration. , This process will be run until enough agents have been chosen.

3.2.1.1 Agents

There are multiple definitions of the concept of agent. A general view is that any type of independent component can be viewed as an agent (Bonabeau 2001), no matter its behavior is a primitive reaction or complex intelligent system. On the other hand, others require very specialized behavior for an agent. Casti (1997) insisted that a component's behavior must be adaptive in order for it to be considered an agent. In other words, agents have to be able to learn from their environments and change their behaviors in response. Some (Dowty 1991) claimed that an agent is only a proto-agent if it maintains a set of properties and behaviors but does not exhibit learning behavior.

Though the agents in a multi-agent system could be robots, humans or human teams, here we are mainly interested in software agents. One computer-science view (Jennings 2000) of agents emphasized the essential characteristic of autonomous behavior. In other words, an agent is the capability of the component to make independent decisions. This requires agents to be active rather than purely passive.

Under this definition, the agents in a multi-agent system have several important characteristics: Autonomy, the agents are at least partially autonomous; Local views, no agent have a full global view of the system, or the global knowledge is too complex for an agent to make practical use; Decentralization, there is no designated controlling agent.

3.2.1.2 Network Configuration (Social Fabrics)

In addition to being autonomous, individuals in CA are impacting each other through a network of relations. The interconnections among individuals in the population are viewed as a social fabric in which human culture which is created by the interactions between people. After a knowledge source in the belief space selects individual from the population to influence, the candidate agent is affected by its peers that are connected by it through the social fabric. As in human culture, the impact of knowledge is distributed into the population through human interactions. Each individual is connected to a set of neighbors and each individual is influence by a knowledge source. A given individual decides which knowledge source to use in order to direct its decisions based on a weighted majority vote over itself and its neighbors.

In order to emulate various social fabrics in human society multiple geometrical shapes of different complexity are used. In this way, each agent has a particular position on the predefined and fixed network. If we consider that the social fabric represents paths connecting agents, a specific organization of social fabric is a configuration of the links in one network. Here, we use network configuration to indicate a specific instance of the social fabric. The term that was used in previous CA literature was that of a graph, which emphasized the geometric meaning other than the human-society metaphor.

The network configurations that have been used so far are *lbest* (local best with connection degree of two for each agent), square (degree of 4 for each agent), *gbest* (all individuals, degree of $n-1$ nodes for each agent), star (one hub agent is connected to all other agents which have no other neighbors), and circle (agents are metaphorically arranged on circle and an arc centered at one agent covers its neighbors). Such topologies are shown below.

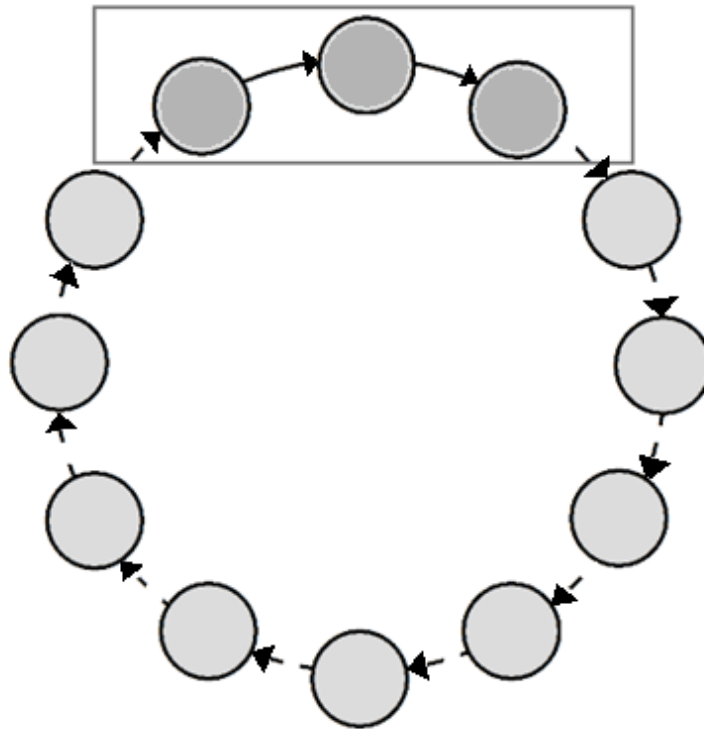


Figure 3.3 Topology of *lbest*

For the *lbest* network configuration, an individual is only aware of its closest neighbors, here *l* indicates local. As shown in the figure, the top agent only knows the neighbors that are painted in dark. In *lbest* the information or influence flows in only one direction, left or right for a given structure. Ring structure is less likely to converge to a sub-optimal point and require fewer computations than the others.

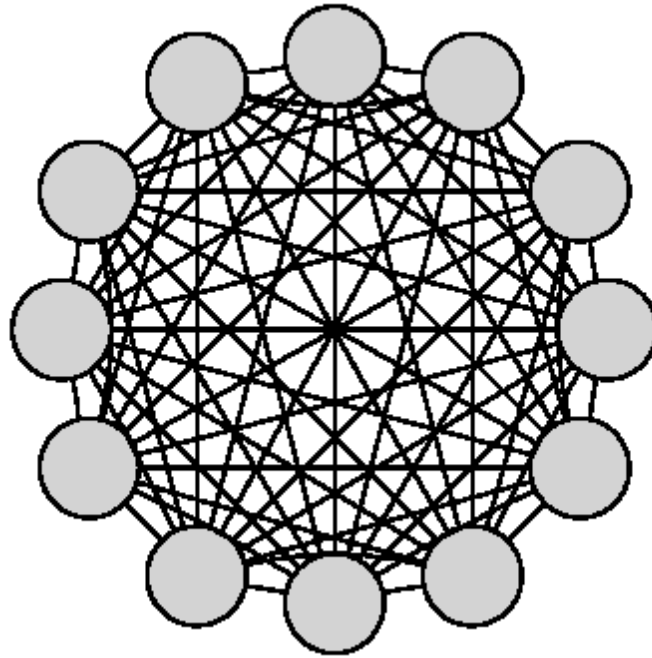


Figure 3.4 Topology of global

For global topology, every individual is connected to all the rest. Admittedly, such a topology required the most calculation during influence because each agent has the maximum number of neighbors. Such topologies can produce premature convergence during the search process.

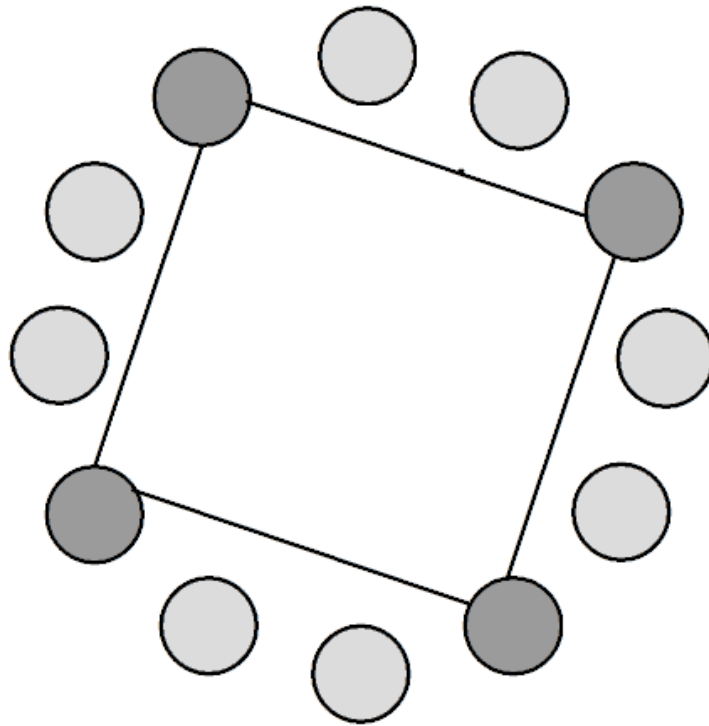


Figure 3.5 Topology of square

For a square topology, an individual is connected to a square. No the agent is connect to four other neighbors. I have no idea what the thing above.

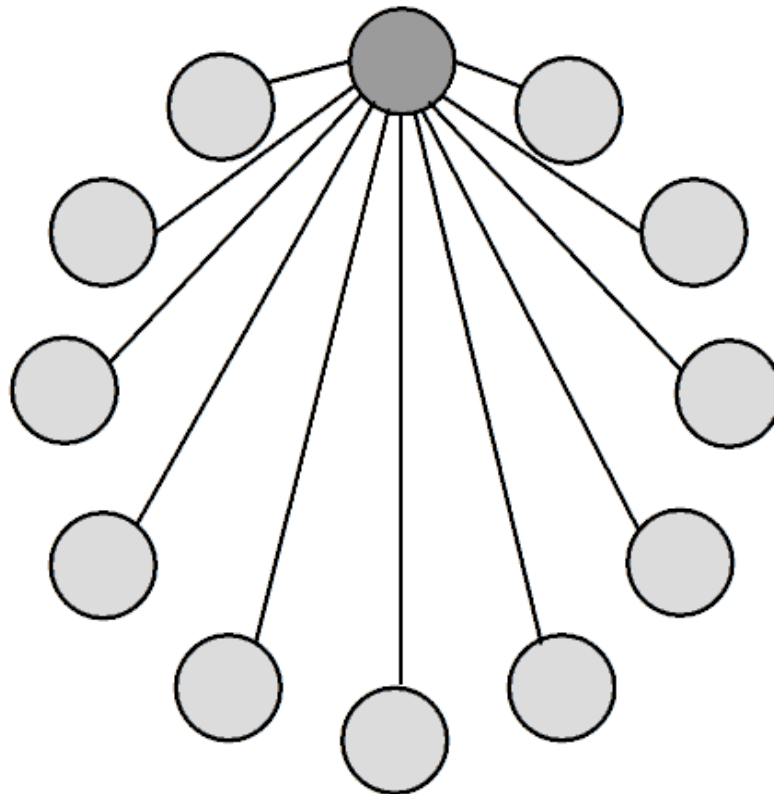


Figure 3.6 Topology of star

In the Star topology, the hub individual is connected to all others individuals, while every other individual is only connected to the central one. The number of neighbors is then either 1 or $n-1$, depending on the position of the individual.

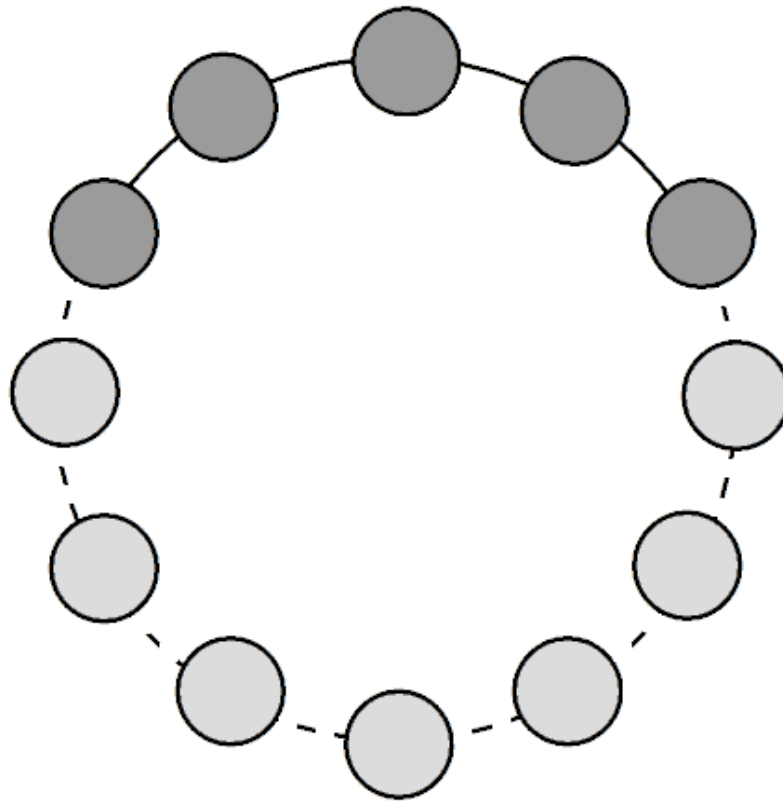


Figure 3.7 Topology of circle

The topology Circle can be deemed as an extension of lbest. In the circle the information flows in both directions from an individual. So each individual has two neighbors who can influence them while in a ring it is just one.

In addition to those topologies that have been used in previous implementations, more ways of constructing a neighborhood topology can be found in (Jin, Girvan, and Newman 2001, Pattison and Robins 2002, Ebel, Davidsen, and Bornholdt 2002).

3.2.2 The Belief Space

Culture, as defined by Tylor in Primitive Culture(Tylor 1920), as *“culture, or civilization, taken in its broad, ethnographic sense, is that complex whole which includes knowledge, belief, art, law, morals,*

custom, and any other capabilities and habits acquired by man as a member of society.” Certainly, a culture is overwhelmingly complex for existing modeling approaches..

In order to deal with that, anthropologists such as Durham (1991) suggested that cultural evolution can occur on both of two levels, the population level and the belief or knowledge level. In the previous section we examined aspects of the population space. Here, we describe the knowledge sources in the belief space and their interconnections.

While there are other socially-motivated evolutionary algorithms, Cultural Algorithms distinguishes itself from the PSO and ACO algorithms in that the CA uses five basic knowledge types in the problem solving process rather than just one or two locally transmitted value. It has been observed in the field of cognitive science that each of these knowledge types is supported by various animal species (Wynne 2001; Clayton, Griffiths etc, 2000) and it is assumed that human social systems at least support each of these knowledge types as well. The knowledge sources include: normative knowledge (ranges of acceptable behaviors); situational knowledge (exemplars or memories of successful and unsuccessful solutions etc.); domain knowledge (knowledge of domain objects, their relationships, and interactions); history knowledge (temporal patterns of behavior); and topographical knowledge (spatial patterns of behavior). This set of categories is viewed as being complete for a given domain in the sense that all available knowledge can be expressed in terms of a combination of one of these classifications.

In the rest of this section, all knowledge sources will be introduced in detail. Each knowledge source will be described in terms of a simple data structure in order to describe the major relationships supported within it. However, in any given application the exact implementation of each will vary.

3.2.2.1 Situational Knowledge

Situational Knowledge maintains a set of non-dominated individuals, which can be viewed as exemplary cases, with the expectation of facilitating individual experiences. Obviously, situational knowledge leads individuals to “move toward the exemplars”. This was the earliest knowledge source used with Cultural Algorithms and was inspired by elitist approaches in Genetic Algorithm. This mechanism has been extensively adopted by various MOEAs, such as the GBEST in PSO.

There is no specific data structure for this knowledge source. For example in MOCA during update, non-dominated individuals are promoted into this knowledge source as exemplars. In single-objective application, the CA will keep a list of elitists in situational knowledge. In multi-objective optimization, the CA will randomly select from the Pareto front; and will go to the next Pareto front if there are not enough individuals here until enough exemplars are found to fill the list.

3.2.2.2 Topographical Knowledge

Topographical knowledge was originally named “*regional schema*” (Jin and Reynolds 1999). It is represented in terms of a multi-dimensional grid if there are more than one objectives or an array if there is only one objective. While in CA the term grid cell has been used, some other literature uses hyper-cube to depict the same concept. This mechanism has been used by other MOEAs such as the hyper-cubes in MOPSO.

Topographical knowledge was motivated in conjunction with data mining problems where the problem space was so large that a systematic way of partitioning the space during the search process was needed so that search could focus on the promising areas. If we view a state as associated with a

region in the functional landscape then the topographic knowledge source is looking for new states. Thus, the state space may vary dynamically as new sub-regions are discovered and added to the mix.

The topographical knowledge source is initialized by sampling a solution in every grid cell and creating a list of best cells. The update occurs when a cell is divided into sub-cells when an accepted individual's fitness value is better than the best solution in that cell, or if the fitness value of the cell's best solution has increased after a change event is detected. In reality, topological knowledge is used to distribute individuals potentially over the entire landscape.

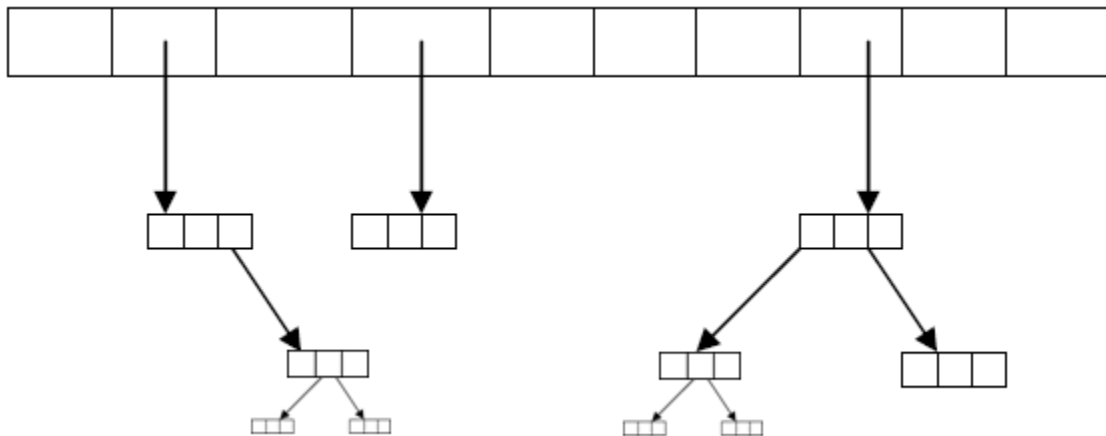


Figure 3.8 Structure of Topographical Knowledge: different space volumes

Figure 3.8 presents a division of the grid, some cells are large, while some other have been divided into smaller ones recursively because more non-dominated individuals have been found in them. Conceptually, topographical knowledge is similar to adaptive grid that had been used in EMOPSO.

One of the important changes is that Topographical knowledge is implemented upon the idea of belief cell [98] to answer the challenge given by high dimension. Previously, MOCA cut the whole objective space into sub space and continue the process along with evolution. This method was similar to adaptive grid [97] that had been used by many other MOEAs. However, one mere division on each of

30 dimensions will produce 230 sub spaces. This large number exerts heavy and unnecessary computing burden to MOCA. To solve the explorative growth of power function, MOCA borrowed the concept of belief cell in that the whole space is virtually divided into sub spaces in advance; and only those who containing non-dominated individuals and thereafter look promising are deposited into the Topographic knowledge. For example for a problem having 30 dimensions, if each dimension is divided into 10 segments, there are totally 1030 sub spaces.

At the beginning of evolution, population is randomly created. Since then on, the population evolves in the same way of MOCA.

3.2.2.3 Normative Knowledge

Normative Knowledge retains a set of intervals for objectives that provide promising variable ranges for good solutions, metaphorically, define standards for social individual behaviors. Such intervals are named *beacon*. While use of norms is the key of human social intelligence (Clayton et al 2000), normative knowledge is expected to guide an individual to jump into a good range if it is not already there. This mechanism is rarely seen in related literature.

The normative knowledge data structure for n objectives is shown underneath.

Object ₁	Object ₂	Object _n
[L ₁ , U ₁]	[L ₂ , U ₂]	[L _n , U _n]

Figure 3.9 Normative Knowledge

For each objective, there is a pair of number defining the lower and upper bond of the range in which good individuals are expected to be found. During update procedure, beacons are updated

according to the current non-dominated individuals. During influence procedure, if the chosen individual is outside of the good range, then one that is randomly produced in the good range will replace it.

3.2.2.4 Domain Knowledge

As its name suggests, domain knowledge takes advantage of the knowledge of the problem space to guide search into good areas in objective space efficiently. Say, for a continuously derivable solution surface, we can use the maximum gradient of a give location to guide the individual to move upwards quickly. When the CA dealt with the “*Cones World*” (Che 2009) to produce arbitrary real-valued landscapes, such a strategy was used. This mechanism was adopted by some MOEAS and deemed as a hybrid approach of accelerating search in local areas, such as the hill climbing using local differentials in a hybrid evolutionary algorithm (Gong etc 2010).

3.2.2.5 Historical Knowledge

Historical knowledge, also known as temporal knowledge, monitors the search process and records good Individuals that have been found historically. In order to reason about global dynamics and to facilitate backtracking or the retracing of actions, it contains information about sequences of environmental changes in terms of shifts in the distance and direction of the optimum in the search space. This knowledge source does not only save the locations of the good individuals—which has been done by situational knowledge—but also the direction for finding them. Therefore, history knowledge can consult those recorded events for guidance in predicting a good move direction. While it is easy to see avatars of situational knowledge in other MOEAs, historical knowledge is rarely seen in related literature.

Its cognitive origin comes from episodic memory (both in humans and animals), which is a type of event-based memory. It stores information about events, and temporal-spatial relations among those (Heyes and Huber 2000).

History knowledge is expected to provide a global perspective regarding the change in solutions. It computes the average change in parameter values within a region, the window size, and predicts the direction of the shift in the optimum from the previous position. The knowledge data structure representation is shown underneath.

Individual1 location: (x_1, x_2, \dots, x_n) direction (d_1, d_2, \dots, d_n)	Individual1 location: (x_1, x_2, \dots, x_n) direction (d_1, d_2, \dots, d_n)	Individual1 location: (x_1, x_2, \dots, x_n) direction (d_1, d_2, \dots, d_n)
---	---	--------	---

Figure 3.10 Data structure of Historical Knowledge

To help guide the optimization process, knowledge sources have been selected in order to influence members of the population by sampling a dynamic probability distribution, which was implemented as a roulette wheel in source code. As the optimization runs, we adjust the distribution of knowledge sources in order to encourage knowledge sources that are producing promising individuals by increasing their probability. Additionally, a mechanism for the preventing the starvation of any knowledge source was implemented by giving each knowledge source a minimum quota of influencing, with the intention of ensuring that all sources have the opportunity to affect the evolution.

3.3 Effectiveness of MOCA and weakness

Best's MOCA confirmed CA's ability for solving MOPs while bringing to light the fact that there are still some aspects that can be improved. Best has pointed out that in his implementation the

situational and historical knowledge sources can achieve an acceptable spread on simple problems but not on problems with very uneven density. In addition, the topographical knowledge source, which was effective in previous CAT version to solve the Cones World problem, is less productive when considering only one of the objective functions in the problem at a time. Best suggested modifying the topographical influence function to divide the search space based on the Pareto rank of the individuals in the population. Among all the concerns of improving MOCA, such as improving the computation efficiency by adopting a more heuristic gradient calculation in domain knowledge, the assessment of spread appear to the most prominent problem.

The insufficiency of population spread was caused by the fact that in SOPs, which CAT had been designed for, there is no such a criterion for evaluating and guiding the evolutionary process. As an initial extension of CAT to hand MOPs, MOCA can identify good individuals quickly. The following table describes the Pareto fronts after non-domination sorting for the population of year 40 that was produced by MOCA on DTLZ.

Table 3-1 Pareto fronts at year 40 for DTLZ1

Pareto#	X	Y	Z			
0	0.059093	0.144503	0.296429	2	2.57E-05	max 0.022237 min 1.84E-06 ave 0.005329
0	0.088483	0.186583	0.228858	3	0.003924	
0	0.090181	0.023987	0.387014	3	0.001182	
0	0.104606	0.33921	0.057196	3	0.001013	
0	0.082192	0.155827	0.263213	4	0.001233	
0	0.121097	0.071882	0.324418	2	0.017397	
0	0.12114	0.147176	0.233667	2	0.001983	
0	0.110408	0.174008	0.215586	2	1.84E-06	
0	0.261906	0.194125	0.066206	4	0.022237	
0	0.25402	0.118836	0.130343	4	0.003198	
1	0.160917	0.106704	0.234819	2	0.00244	max 0.025982 min 6.08E-05
1	6.56E-04	0.021131	0.48082	2	0.002608	

1	0.286969	0.188181	0.044658	4	0.019808	ave 0.010122
1	0.098354	0.16674	0.239664	4	0.004757	
1	0.023299	0.034007	0.452065	5	0.009371	
1	0.034693	0.130959	0.334456	3	0.000108	
1	0.037244	0.066256	0.398099	2	0.001599	
1	0.12833	0.143864	0.237547	4	0.00974	
1	0.109934	0.107252	0.282874	1	6.08E-05	
2	0.023484	0.244979	0.251207	4	0.01967	max 8.851882 min 0.000138 ave 0.50207
2	0.113673	0.31913	0.093179	4	0.025982	
2	0.059151	0.207388	0.233876	2	0.000415	
2	0.038543	0.133362	0.333718	2	0.005623	
2	0.036419	0.226744	0.238054	4	0.001218	
2	0.313544	0.1382	0.048394	2	0.000138	
2	0.157233	0.029503	0.318062	2	0.004799	
2	0.060426	0.107375	0.342032	4	0.009833	
2	0.571029	3.95E-04	0.039593	2	0.111017	
2	8.62E-04	0.186234	0.327298	3	0.014393	
2	0.202133	0.183341	0.115779	2	0.001253	
2	0.148676	0.048386	0.303354	4	0.000415	
2	8.83E-05	0.003727	0.498384	2	0.002199	
2	0.046358	0.064087	0.395163	4	0.005608	
2	0.117267	0.290411	0.113012	2	0.020691	
2	0.057488	2.36E-04	9.294158	5	8.851882	
2	0.228741	0.023326	0.249054	3	0.001121	
2	0.291634	0.142923	0.066475	4	0.001032	
2	0.024447	0.071008	0.406476	2	0.001932	
2	0.034435	0.400285	0.068967	2	0.003687	

We can see clearly in the table that even those individuals that lie on rear fronts still have good fitness values. This confirms the conclusion of insufficient Pareto front coverage because otherwise solutions with good fitness value won't be put on secondary fronts. In other words, aggregation of individuals reduces the system's computation power.

The statistics of Pareto fronts at the end of evolution 100 generations is shown in Table 3-2. We can see that the phenomenon discussed in the above paragraph still presents. In addition, fitness of

best individuals doesn't get considerably improved; which, on the other side, proved the efficiency of CA to find good solutions.

Table 3-2 Pareto fronts at the end of evolution

	Max	Min	Average
Pareto 0	0.507109	1.31E-06	0.056808
Pareto 1	3.40E-02	7.49E-07	4.77E-03
Pareto 2	0.039243	1.75E-07	0.003916
Pareto 3	0.016857	2.51E-07	0.003903
Pareto 4	0.010412	1.82E-05	0.002896
Pareto 5	2.48492	2.16E-08	0.141828
Pareto 6	0.006206	0.000216	0.002372

The insufficiency of spread cannot be solved with the fitness evaluation alone; there must be additional approaches. For example, in the popular Multi-Objective algorithm Non-dominant Sorting Genetic Algorithm (NSGA), agents don't know the general layout of the population, and they just try to reach the Pareto front and stay far away from neighbors. They don't care where the whole population should go as a team, they don't know (or care) whether the Pareto varies along time, neither do they know how crowded or spare one special sub area is. This strategy, though simple, had been proved to be effective. However, in Cultural Algorithms we should be able to coordinate our mining of the Pareto Front more efficiently.

The insufficiency of spread cannot be solved without amending other CA components either. In the previous CAT version, the belief space has been tuned to speed up the search process, such as that Domain knowledge was used to guide the search trend in some specific directions. While the populations aggregate into small areas, even though some individuals have good fitness values, it is hard to evaluate the solutions as good as a whole. Even though random deviation in evolutionary algorithms cannot be fully eradicated because they are population-based, populations are guided by various

knowledge sources to aggregate to some small areas. Our investigation figured that after population aggregated to some parts of the Pareto front, they had inertia to stay around but not energy to explore new areas.

Figure 3.11 represents the X-Y projection of the domain space for DTLZ1 when year==60 for previous MOCA, which is typical for our test. With the consideration that the optimal solution is a line of $x+y=1$, we can see that most individuals lie close to the real solution line. However, those individuals are aggregated into three sections while leaving two big empty areas unexplored. After our observation of this figure, even when we extend evolution to year 200, most individuals still stay in the three areas.

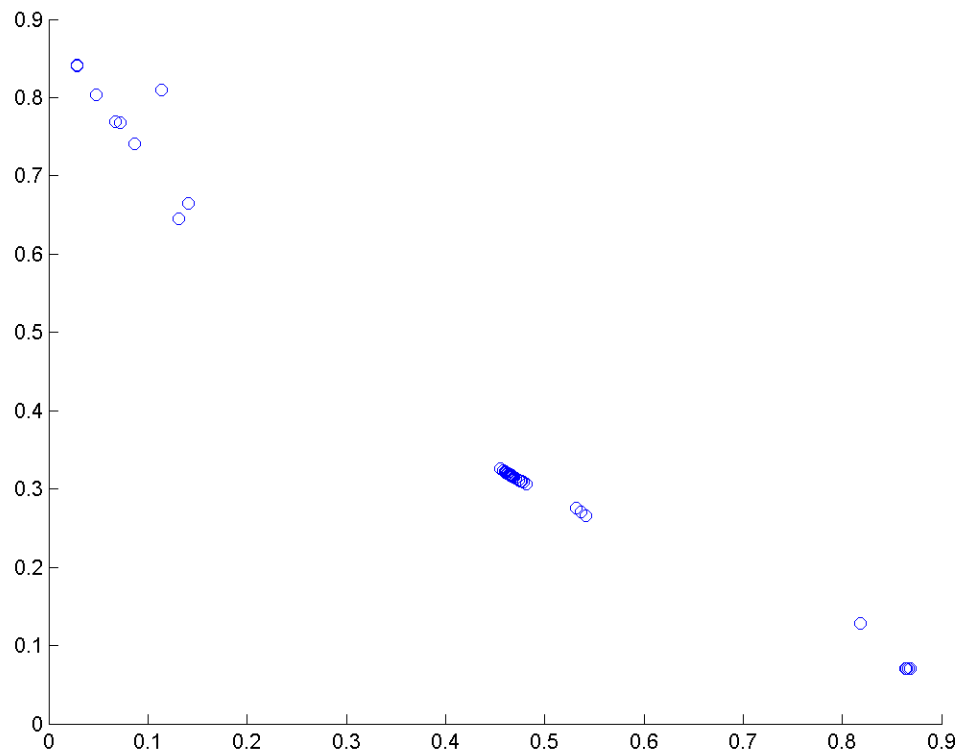


Figure 3.11 Representative illustration of partial aggregation in population

Certainly, due to randomness of evolution, the aggregation pattern shown in the figure is not the only one, though it is the representative one that appears frequently. In a few simulations we have observed different aggregation patterns, such as aggregating at only the ends of the Pareto front. Best had not pointed out the cause of such aggregations. Similarly, we are not interested in figuring out the exact cause of such phenomenon though we primitively assume it was due to the dominance of situational knowledge source over the other knowledge sources in the current implementation. Instead, we generalize the phenomenon as the problem that the current implementation of CA cannot cover sufficient spread of Pareto front. In this way, we hope that we can eradicate this problem.

One way of doing this is to make all knowledge sources stronger participants in the problem solving process as we suggested earlier. As Che (2008) has confirmed in this thesis, different knowledge sources fit different problems. As a reasonable extension, in front of a specific implementation of MOCA, for different problems, there may be different knowledge sources that can contribute to the generation of a more evenly distributed front.

In summary, CA is mature in increasing individual's fitness, we can see how fast and efficiently it finds a few good individual solutions; in addition, it has been confirmed that network configuration plays an important role in distributing knowledge in a population and therefore increase the evolution efficiency. In this thesis, we will add the consideration of spread of the whole generation into our implementation and identify its usefulness with quantitative analysis.

3.4 Expansion of the CA for Multi-objective Optimization

During the expansion of CA into MOCA, the same framework, including the five knowledge sources and the communication protocol between the population space and the belief space that has

been used in CAT were inherited. Certainly, there have been some modifications that are necessary for tuning CAT to fit MOPs, and, some modifications are critical for turning the CA into MOCA and are worth special attention. Most of the work that is introduced in this section was done by Best (Best, 2009);

First, as has been mentioned above, in the CA the highest performing individuals are frequently needed. For a multi-objective problem, in which generally there is no such single individual because populations are not a completely ordered set, MOCA used the Goldberg ranking scheme (Goldberg 1989), where all non-dominated solutions in the population are given rank 1; these solutions are removed, and the non-dominated solutions in the remaining population are given rank 2, and so on. Individuals laying at the same front are considered as being equally good. The total ordering on solutions is replaced by a partial ordering of individuals into Pareto ranks. As a result, the concept of *“the best performing individual”* in single-objective optimization is replaced by *“an individual chosen from the set of non-dominated individuals in the current population.”* for multi-objective problems. This choice is currently random, though for specific problems a heuristic can be used to suggest an auspicious individual.

Another change in MOCA was related to how knowledge sources are chosen to influence individuals. To help guide the optimization process, knowledge sources have been selected in order to influence members of the population by sampling a dynamic probability distribution. Each knowledge source, KS_i , has an associated probability P_i of being chosen to influence a given individual. In the classic CA, P_i is calculated by summing the fitness values of the individuals that were influenced by it. Here for multi-objective optimization, P_i is derived from the Pareto ranks of the individuals in the current population for which KS_i is generated. The formula used to calculate the score of each knowledge source to be used in roulette wheel selection is listed as follows.

$$\text{Score}(KS_i) = \frac{\sum_j \frac{1}{\text{PR}(\text{IND}_{i,j})}}{|\text{IND}_i|}$$

In this formula, IND_i represents the set of individuals in a population that were created by KS_i , and $\text{IND}_{i,j}$ is the j th individual in it; $\text{PR}()$ is the function to retrieve the Pareto ranking, and operation $||$ gets the cardinality of the set.

For example, we see a population for a two-objective space in Figure 3.12. It shows eight individuals from three arbitrary knowledge source marked by Δ , \circ , and \square . The two axes represent the two objectives, three curves marked by numbers are three Pareto fronts while the numbers are their Pareto ranking.

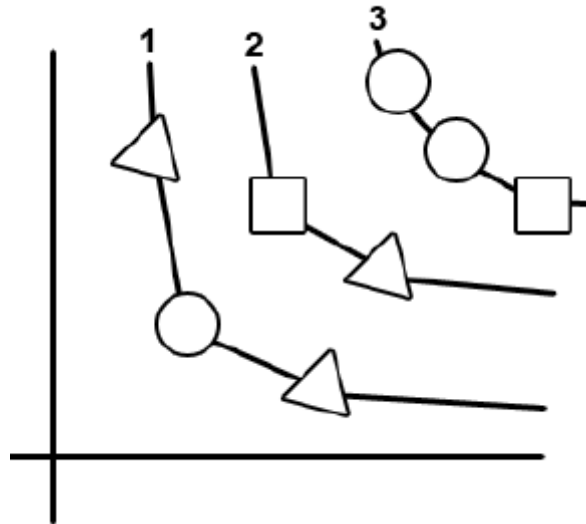


Figure 3.12 Illustration of calculation of performance of knowledge sources

The individuals form three Pareto ranks. The scores for each knowledge source are as follows:

$$\text{Score}(KS_{\Delta}) = \frac{\frac{1}{1} + \frac{1}{1} + \frac{1}{2}}{3} = \frac{\frac{5}{2}}{3} = \frac{5}{6}$$

$$\text{Score}(KS_{\circ}) = \frac{\frac{1}{1} + \frac{1}{3} + \frac{1}{3}}{3} = \frac{\frac{5}{3}}{3} = \frac{5}{9}$$

$$\text{Score}(KS_{\square}) = \frac{\frac{1}{2} + \frac{1}{3}}{2} = \frac{\frac{5}{6}}{2} = \frac{5}{12}$$

Except for the new evaluation mechanism, the random selection of a knowledge source in the belief space during the influence procedure is the same with CAT 2.0.

In addition, since the benchmark problem set DTLZ on which Best worked can easily scale to have high dimensions, such as 10, Topographical knowledge source splits the domain space only as needed. In his implementation, each hyper cube is called a cell; when individuals in the cell got crowded the cell will be split in half along each dimension. At the end of evolution, some cells have tiny volume because they are the results of multiple divisions while some others are big hyper cubes since there are not many individuals contained inside.

After such modifications, Best was able to apply the Cultural Algorithms on DTLZ benchmark. He compared MOCA to NSGA-II using two performance metrics: hypervolume (HV) (Zitzler and Thiele 1999) and generational distance (GD) (Valenzuela-Rendon, Uresti-charre, and Monterrey 1997). With comparably good experiment results, Best concluded that the Cultural Algorithms were a promising technique for solving multi-objective problems.

3.5 Conclusions

In this section we began by providing an overview of the Cultural Algorithm framework with an eye towards how they might be modified for Multi-Objective problem solving. We then discussed the

major changes made by Best in adopting the CAT 2.0 system to multi-objective problem solving. In his approach he focused on only a reduced set of features available in the Cultural Algorithm.

In general, the changes made to it to become MOCA were not complicated enough to hinder those who have studied CA from understanding the system. This suggests that there is a natural fit between Cultural Systems as modeled by the Cultural Algorithms and multi-objective problem solving.

CHAPTER 4 THE MOCAT 1.0 SYSTEM DESIGN

Previously MOCA has demonstrated the ability of Cultural Algorithms to handle MOPs. However, the MOCA implementation does not take advantage of all of the available features in the Cultural Algorithm Toolkit. In particular, as discussed earlier, it makes limited use of the available knowledge sources to influence the problem solving. As Best indicated (Best 2009), the gradient approximation that was used in domain knowledge source was not always effective and more heuristics might be useful. Also, the situational and historical knowledge sources can achieve an acceptable spread on simple problems but not on problems with very uneven density. As a result, overall spread over the Pareto front sometimes can be improved by augmenting the data structures and connectivity of existing knowledge sources in order to address a more general class of multi-objective optimization problems. In other words, the system does not produce enough coverage on the Pareto front.

In addition, we need to take into consideration spread metrics used to guide the system. In this thesis we employ modified versions of two different spread metrics, spread and hyper volume, in order to provide more information to the knowledge sources. Neither metric had been used in conjunction with each other in the past. It was felt that if they provided complementary information on spread then more information would be available for the knowledge sources to generate solutions.

A third modification relates to how the influence of successful knowledge sources is spread through the population. The spread of knowledge through the population takes place within the social fabric. As the problem solving process progresses, especially for complex problems, it was felt that the appropriate network structure would need to change as well.

In addition to the theoretical changes, to unify CAT 2.0 and MOCA and to take advantage of the visualization power of Repast Symphony, a new multi-objective CA execution framework is designed and implemented. This chapter discusses how each of the suggestions mentioned above are implemented and proposes an improved toolkit framework. The concrete implementation, named MOCAT 1.0 is presented in details in the next sub section.

Section 4.1 gives the overall architecture of the MOCAT system. Section 4.2 describes the internal implementation of knowledge sources. Section 4.3 describes the acceptance function, including how voters are selected from the population. Section 4.4 explains the update functions for the belief space, especially how the slices in the roulette wheel are updated. Section 4.5 focuses on the influence function which has two parts: One part describes using pseudo code how to select the topologies for a generation and the other part of the section discusses how the knowledge sources are selected.

4.1 System Framework

MOCAT is implemented in Java as a stand-alone system as shown in Figure 4.1. Input is the classes that implement the problem formulas. The CA engine will control the evolution in which a set of individuals hopefully move to the Pareto front. Output of the MOCAT system includes both a visual representation of the population and files that include information about the evolution. While the real-time display give us a good chance of observing the progress of evolution, the files that are save on hard disk contain all necessary information that is used to do detailed statistical analysis of the optimization.

MOCAT can easily work on other problems that have not been coded in the current implementation if only the new problem is wrapped in an algorithm class following a specific format. At

this stage, MOCAT only handles numeric problems. In other words, the domain space and the objective space of the problem have to be described in numbers.

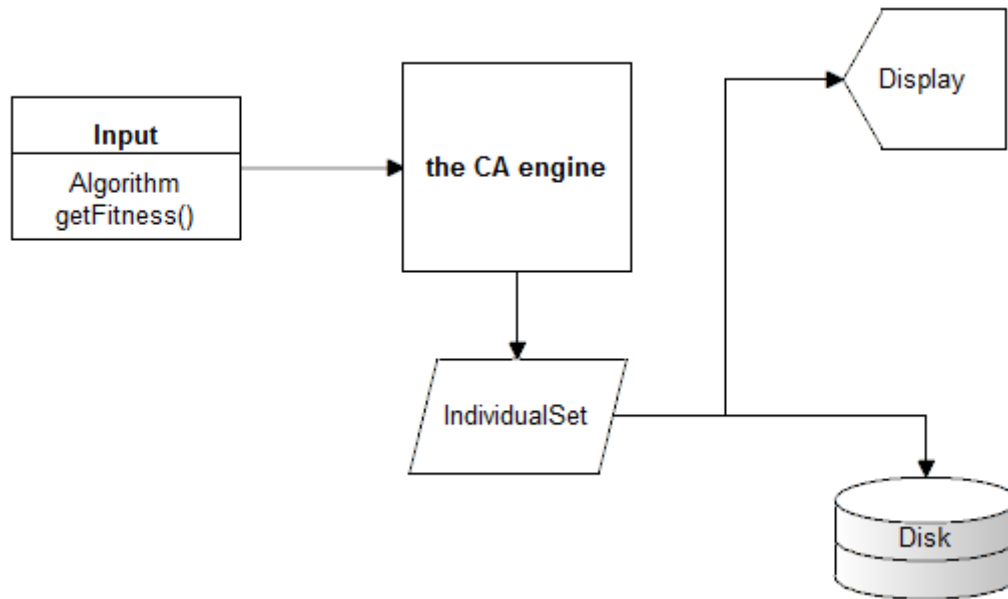


Figure 4.1 MOCAT System

To render a brief image of the MOCAT, the pseudo code which shows the basic framework of the implementation is given in Table 4-1.

Table 4-1 The Pseudo Code and Execution Flow Chart of MOCAT

<p>Begin</p> <p>t = 0;</p> <p>initialize Bt, Pt</p> <p>initialize roulette wheel for network configurations</p> <p>repeat</p>

```

accept()

    non-dominance sorting

    voting

    updating the belief space, including the roulette wheel and knowledge sources

influence()

    choosing knowledge sources based on their performance

        guarantee no knowledge source die out: N/20 quota

    choosing a network configuration

        currently support: lbest, square, ring, star, global

    t = t + 1;

    select Pt from Pt- 1 by starting from better Pareto fronts

    keep record of the performance of the recently used network configuration

    updating the roulette wheel that is used to select network configurations

until (termination condition achieved)

End

```

The pseudo code reveals a similar frame that has existed in previous CAs. While the theory of the Cultural Algorithms remains—therefore, we still claim that MOCAT 1.0 is a CA, but not an application based on CA, though a lot of changes have been made into MOCAT 1.0 so that it encourages

beneficial competitions among knowledge sources and competitions among social fabrics, is able to spread found solutions over real Pareto front, and is able to scale easily.

MOCAT was implemented upon object-oriented programming techniques and was designed to embrace new algorithms without need of structural changes. Figure 4.2 depicts the system structure of MCOAT using a format that is similar to UML class diagrams.

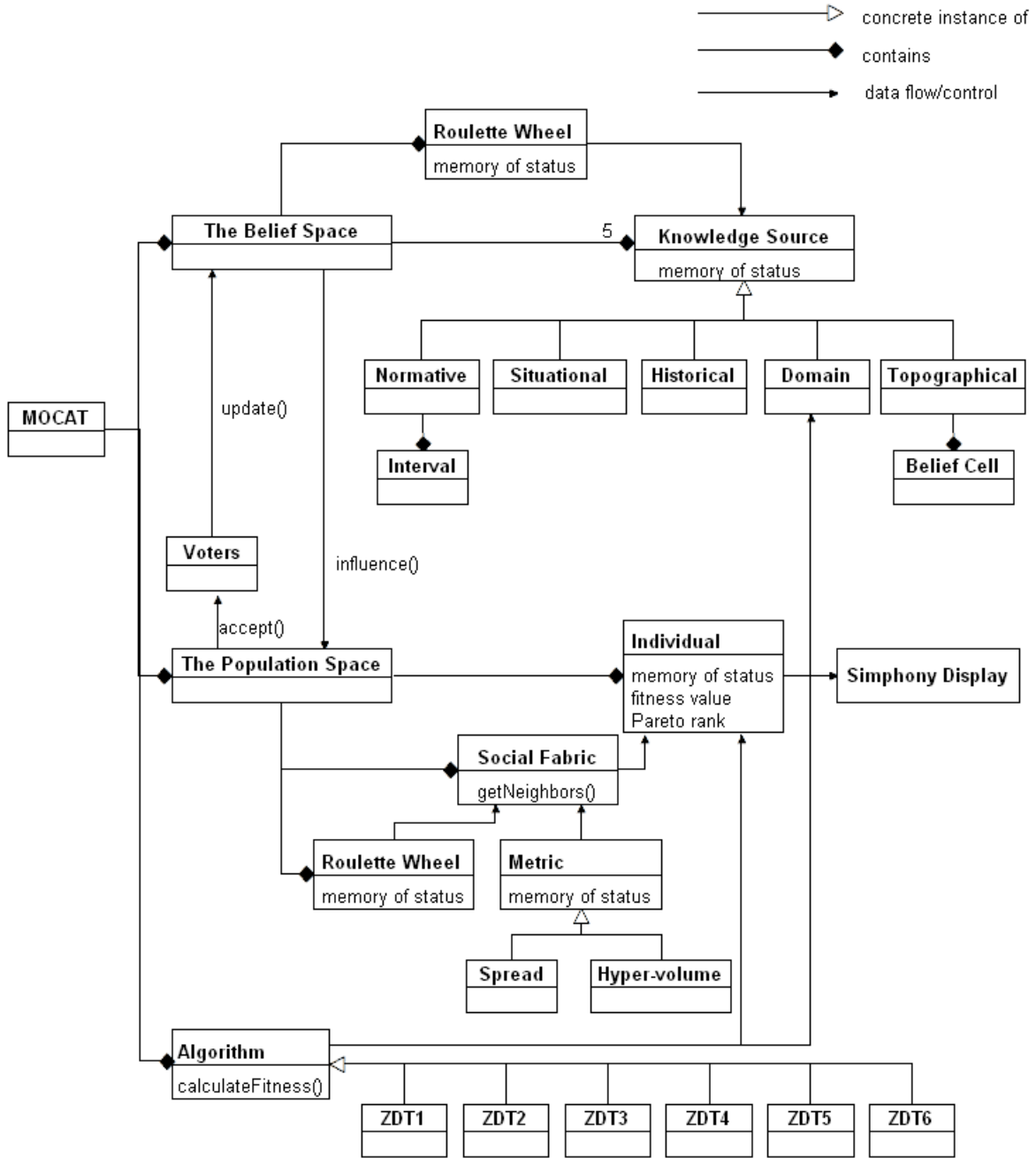


Figure 4.2 Object diagram of MOCAT

Figure 4.2 shows that there are clearly belief space and population space in MOCAT and they interact through function `accept()` and `influence()`; those are the intrinsic characteristics of the CA. We can see from Figure 4.2 that MOCAT 1.0 has specially tuned up for multi-objective optimization by

encouraging competition between knowledge sources and competition between social fabrics. While parts of Figure 4.2 have been introduced in Chapter 3, in this chapter we specially focus on the improvements that MOCAT brings to the traditional CA.

One programming progress is that each test problem is implemented as a separate class that is used to describe the problem, calculate objective values, and explore derivatives around neighborhood. While for any individual of given location, MOCAT is able to calculate its fitness, measure its fitness error, and figure out the slopes in local area. However, MOCAT has no clue of what shape the real Pareto front is or what kind of search strategy will be effective in exploring the objective space.

4.2 Knowledge sources

In Figure 4.2, we can tell that the belief space contains five concrete knowledge sources, normative, situational, domain, historical, and topographical. Knowledge sources are updated by voters, which are in turn selected from the population with the help of non-domination sort. This sub section will discuss the internal data structure of every knowledge source, how they are updated by voters, and what kind of effect through their influence.

4.2.1 Normative knowledge source

Normative knowledge source contains intervals which are used to mark the boundary of the promising space. In MOCAT, there are separate intervals for each dimension. Figure 4.3 shows that for a n-dimension problem, the interval for the second dimension defines the high and low values of this dimension.

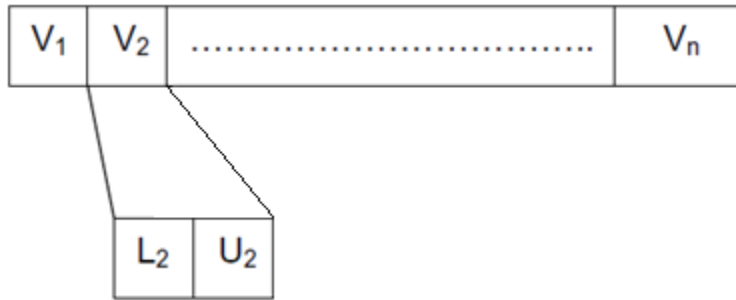


Figure 4.3 The structure of Normative Knowledge

It is updated according to the following formula when a voter is on the found Pareto front.

$$\begin{cases} L_j^{t+1} = \begin{cases} f(x_i) & \text{if } x_{i,j} \leq L_j^t \\ L_j^t & \text{otherwise} \end{cases} \\ U_j^{t+1} = \begin{cases} f(x_i) & \text{if } x_{i,j} \geq U_j^t \\ U_j^t & \text{otherwise} \end{cases} \end{cases}$$

As a result, intervals for all dimensions delimit boundaries of one hyper cube in which the real Pareto front exists. In function influence(), an individual is randomly created inside the hyper cube. It provides coarse heuristics while allow big flexibility to evolution.

4.2.2 Situational knowledge source

Situational knowledge source records all exemplars that have been identified up to now.

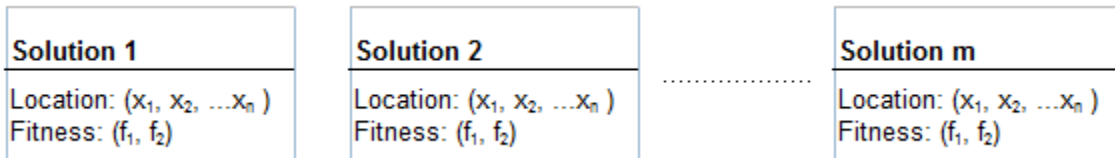


Figure 4.4 The structure of situational knowledge

It is updated by merging the exiting exemplars with voters and then selecting the elitists among them. In this procedure, some previous elitists may be discarded since they are dominated by some new solutions.

$$S^{t+1} = \text{ParetoFront}(S^t + \text{Voters}^t)$$

In function influence(), an exemplar is randomly selected and a new individual is created by dragging the parent individual to it. Obviously, it provides strong guidance to evolution while leaves flexibility. However, when the objective space fluctuates violently, the new individual that is created by situational knowledge source may have inferior fitness values because short distance in domain space does not guarantee similar objective values.

4.2.3 Domain knowledge source

Different from other knowledge sources, domain knowledge source does not work upon its persistent memory. Instead, it takes advantage of the limited but precise information that problem class can give to any specific location in the problem space.

Since there is no way to exhaustively detect the maximum slope from a given location, and in order to save computing resource, only the derivatives along axes are calculated. If one move produces a new individual that is not dominated by the parent individual, this move is counted. Finally, the vector sum of all counted move points to the direction that the new individual should be moved.

After the maximum slope is identified, we will move along that direction 20 times with equal step length to find out 20 candidates of the new individual; then non-domination sort is used to select one from the non-domination set. Now the step length is a random number smaller than 0.005.

4.2.4 Historical knowledge source

Historical knowledge source is used to record the evolution history. It consists of separate memories for each objective dimension.

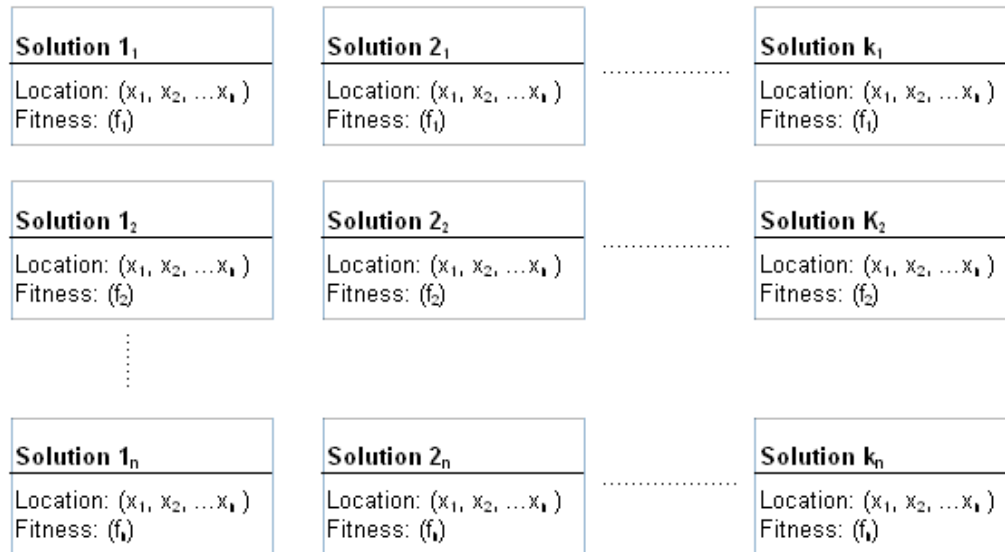


Figure 4.5 The structure of historical knowledge

The historical knowledge source is updated based on each objective dimension without interference with each other:

$$\{e_1, e_2, \dots, e_k\}^{t+1} = \begin{cases} \{e_1, e_2, \dots, e_k\}^t + e_{new} & \text{if } k < l \\ \{e_2, \dots, e_k\}^t + e_{new} & \text{otherwise} \end{cases}$$

where l is predefined size limit of saved solutions and e_{new} is an individual in the voters which has the highest fitness value on dimension t . In function $influence()$, one memory is randomly selected and then a saved individual is randomly selected out of it to sprout a new individual around its neighborhood.

In this way, while we encourage the evolution to forage area that is only certainly good for one objective, we will practically spread the found Pareto front along every dimension.

4.2.5 Topographical knowledge source

Topographical knowledge source contains belief cells (Jin 1999) which are used to indicate hyper grid in problem space. Since we will handle high-dimension problems, traditional division of problem space along each dimension will create large amount of child hyper cubes. For example, for 30 dimensions, one single division will split the current hyper cube into 2^{30} ones. This exponential increase makes topographical knowledge source stop working. Thereafter, we conceptually split the whole problem space into hyper cubes in advance and only record those in which promising individuals exist.

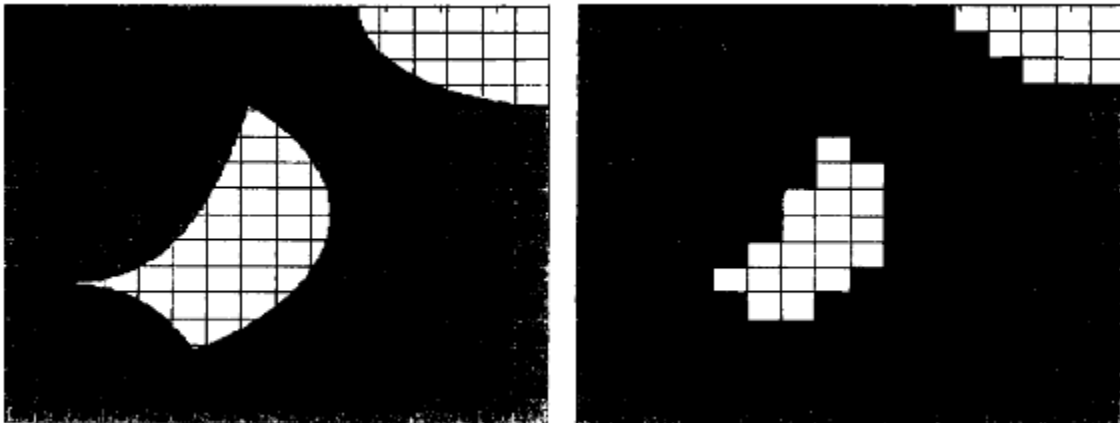


Figure 4.6 Topographical knowledge source provides a mosaic view of Pareto front

From Figure 4.6, we can see that topographical knowledge marks a coarse-granularity record of the found Pareto front. At this point, the domain space is cut to 10 segments along each dimension. In this way, topographical knowledge source works like a group of normative knowledge sources. Its data structure is shown in Figure 4.7.

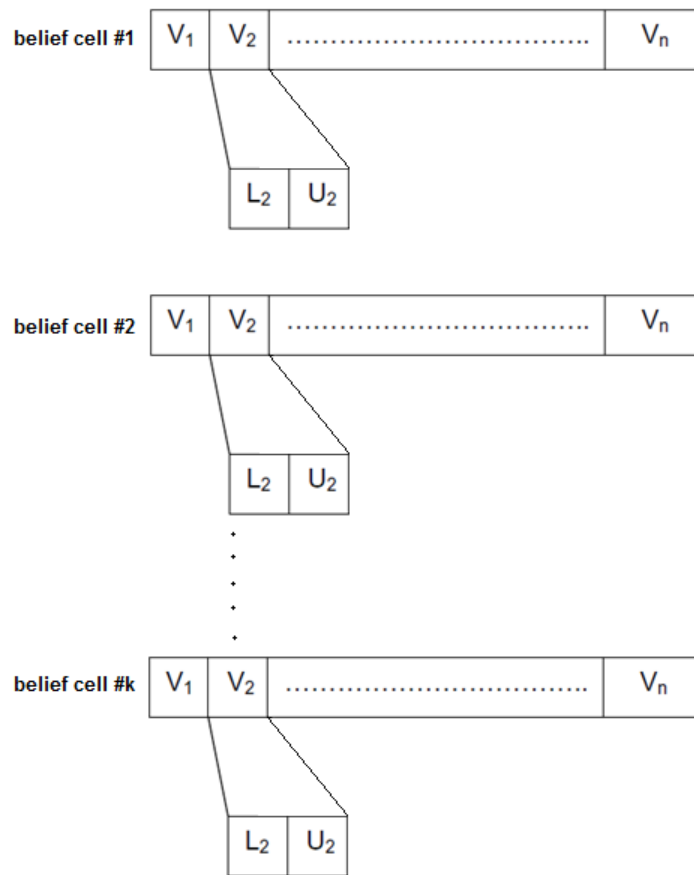


Figure 4.7 The structure of topographical knowledge

A belief cell is added into topographical knowledge source if a non-dominated solution in the voters appears in it. Topographical knowledge source provides guidance to evolution in a way that is more flexible than domain knowledge source but stricter than normative knowledge source.

4.2.6 Interactions among Knowledge sources

All of the knowledge sources contribute to the optimization process. As in the real world, different objectives can be more effectively viewed by different knowledge sources. The key is that results elicited by individuals generated by one knowledge source can be distributed to other knowledge sources in the belief space since both the population and the belief space have a network structure.

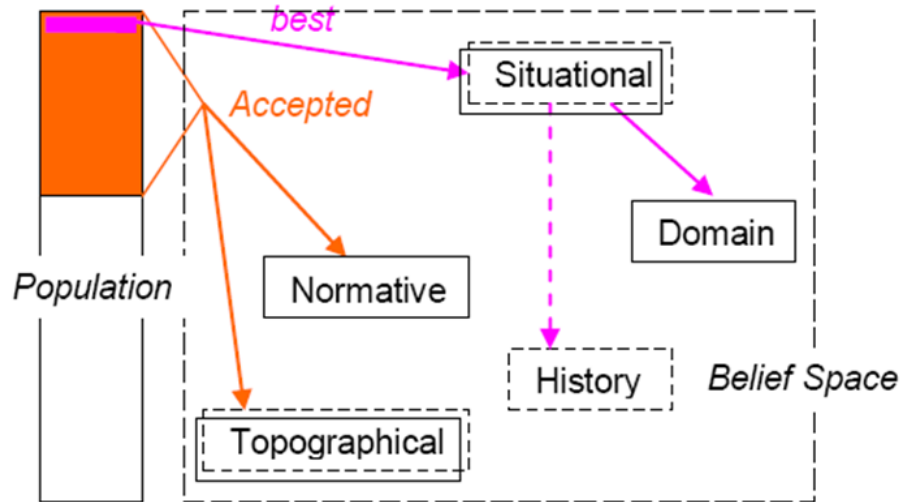


Figure 4.8 Network of Knowledge sources

In Figure 4.8 the network for the knowledge sources in the belief space is given. By allowing the knowledge sources not only to compete but to share results of their explorations, the system is able to exploit the dual inheritance feature of cultural algorithms to hone in on the solution quickly.

4.3 The `accept()` function

Not all of the populations are used to update the belief space, instead, only those who perform not worse than others can be elected as voters to update the belief space. Non-domination sort is used to evaluate and rank individuals.

4.3.1 Non-domination sorting solutions

Domination (Goldberg, 1989) is a partial order relationship in which a dominating solution is superior to dominated solutions in terms of all objectives. Two solutions that do not dominate each other are on the same Pareto front.

Table 4-2 Pseudo code of non-domination Pareto ranking

<pre> Set pr = 1 While(population is not empty) Identify the solutions that are not dominated by others, mark their Pareto rank = pr Remove the identified solutions Set pr = pr + 1 End While </pre>
--

4.3.2 Accept as voters

Based on non-domination sort, the function accept() prefers the solutions on anterior fronts.

Table 4-3 Pseudo code of function accept()

<pre> Set pr = 1 Set returnSet = empty set While(wantedSolution# > 0) If #solutions on front pr <= wantedSolution# add front pr to returnSet Set wantedSolution# = wantedSolution# - #solutions on front pr Else Randomly select wantedSolution# solutions from front pr, put into returnSet Set wantedSolution# = 0 End If Set pr = pr + 1 End While </pre>
--

return returnSet

In other words, the solutions that are able to push the found Pareto front forward will be preferred to update the knowledge saved in the belief space. In this way, we expect the belief space to have more efficient information to guide evolution.

4.4 The update() function

This function updates the knowledge saved in the belief space. How the five knowledge sources are updated has been introduced in 4.3. Nonetheless, function update() additionally update the roulette wheel in the belief space.

In the belief space, there is a roulette wheel to partially randomly pick up a knowledge source to influence an individual. However, on the roulette wheel, the size of the slices for knowledge sources is proportional to their moving average performance. The performance of a knowledge sources is defined as average of the evaluations of all the individuals that have been influenced by it:

$$Performance(KS_i) = Performance(IND_i) = \frac{\sum_j \frac{1}{PR(IND_{i,j})}}{|IND_i|}$$

This evaluation formula was defined by Goldberg (1989). The roulette wheel remembers performance values for each knowledge source; during the function of update(), performances of knowledge sources that are calculated from voters will be taken into consideration to create mathematical averages, which serve as the new performance values for the knowledge sources.

$$Slice_i^{t+1} = \frac{Slice_i^t + Performance(Votes_i)}{2}$$

Then, sum of slices is normalized to 1.0 and the portion of each knowledge source is calculated.

Now, we can rotate the roulette wheel to partially randomly select a knowledge source.

Table 4-4 Pseudo code of selecting a knowledge source from the roulette wheel

<pre> Set sum = sum of slices Set $ns_i = \text{Slice}_i / \text{sum}$ create random number $r \in [0, 1.0)$; Set splitter = 0.0 Set ksResult = 1 While(ksResult <= #Knowledge sources) Set splitter = splitter + $ns_{ksResult}$ If (splitter > r) Return $KS_{ksResult}$ Else Set ksResult = ksResult + 1 End If End While return $KS_{ksResult-1}$ </pre>

In this way, when a knowledge source fails to contribute to the evolution, other knowledge sources have the change to take its position; but still every knowledge source has opportunities to influence individuals.

In summary, the function update() updates both the knowledge sources and the roulette wheel which is used to select a knowledge source for influence() with the consideration of their performances.

4.5 The population space

The population space accommodates the individuals and various topologies that are used to connect the social fabric. While the topologies in the population has been introduced in 3.2.1, this sub we will discuss about the data structure of individuals and the two metrics that are implemented in the population space to evaluate the spread of the solutions over the real Pareto front.

4.5.1 Individuals

Individuals in socially motivated evolutionary algorithms represent locations in the domain space. While their coordinates determine the objective values, individuals are also named solutions. Additionally, since MOCAT uses Repast Symphony, individuals are also called agents according to the terminology of Symphony. Thereafter, in this thesis, individuals, solutions, and agents indicate the same concept and are inter-exchangeable.

An individual mainly holds its own information, such as its location, fitness values, and the knowledge source under which it was created—the operator. At the beginning of evolution, individuals have unknown operators. An individual object also provides storage for its Pareto rank after non-domination sort for future use. To facilitate non-domination sort, two functions are provided to decide whether another individual dominates or is dominated by this individual. The UML class diagram of an individual is shown in Figure 4.9.

Agent
location[1...n] fitness[1...m] operator paretoRank
gt(Agent) lt(Agent)

Figure 4.9 Class of Agent

4.5.2 Metrics to evaluate the spread over the Pareto front

As shown in Chapter 3.3, sometimes MOCA could not spread found solutions evenly over the whole Pareto front. Thereafter, there is a need to encourage individuals to explore areas that are not crowded. This sub section introduces existing spread metrics and then chooses two in our implementation and explains why they are preferred to others.

4.5.2.1 Existing Metrics

There are a few ways to evaluate the performance of MOEAs, including the spread over Pareto front. First, graphical plots serve as an intuitive way to compare the outcomes of MOEAs, especially when different results have prominent discrepancies, such as in (Zitzler et al, 2000). The advantage is that it is easy to evaluate the outcome and most times very helpful; the disadvantage is that it is hard to quantify the visual evaluations and therefore incorporate into the heuristics in the CAT system.

Second, the distance from the known Pareto front to the true can be calculated. As shown in Veldhuizen etc (Van Veldhuizen and G. B Lamont 2000), the distance metric is a value representing how “far” the unknown Pareto front (PF_{known}) is from the true Pareto front (PF_{true}) and is defined as:

$$G = \frac{\left(\sum_{n=1}^n d_i^p\right)^{1/p}}{n}$$

where n is the number of vectors in PF_{known} , $p = 2$, and d_i is the Euclidean distance (in objective space) between each vector and the nearest member of PF_{true} . The idea is to calculate the shortest distance of the points of the known Pareto front to the true one; the sum of all the distances shows how good the known Pareto front is as a whole. The theoretical minimum result of 0 indicates $PF_{\text{true}} = PF_{\text{known}}$; any other value indicates PF_{known} deviates from PF_{true} . In their paper, an example with a discrete Pareto front is given as Figure 1 for illustration of the formula.

Third, the spread of the known Pareto front can be calculated. One representative example is published by Deb etc (Deb et al. 2000). The idea is to get an evaluation of the density of solutions surrounding a particular solution in the population by calculating the average distance of two points on either side of this point along each of the objectives, which was called “*crowding distance*”.

As shown in Figure 4.10, for each solution located at i , this value roughly estimates the size of the largest box, or cuboid, enclosing the point i without merely touching any other point in the population. Here, the crowding distance of i -th solution in its front (represented as solid dots) is the average side-length of the cuboid (shown as the dashed box).

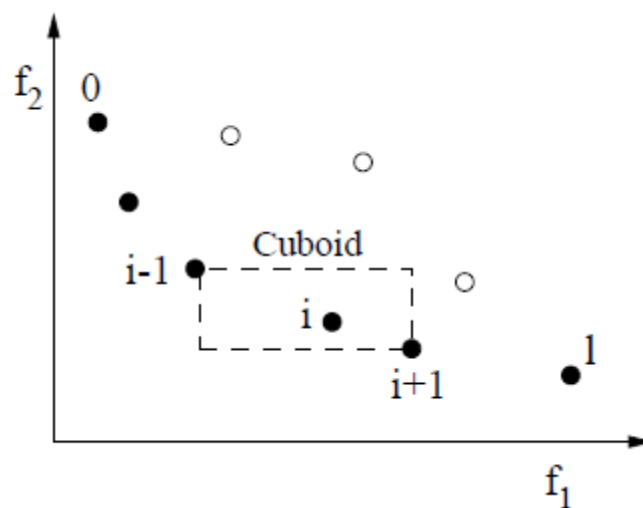


Figure 4.10 Cuboid metric for Pareto front

Because they calculated the perimeters instead of the volumes of the cuboids, they were able to design a linear algorithm to accumulate the distances along each domain after sorting solutions along it. If we arrange all individuals in a linear way and then walk along either the x or y axes from one individual to next, the walk length is a quarter of the crowding distance. While the true Pareto front is unknown, such a metric cannot really describe the coverage of the known Pareto front over the real one. For example, if we shift the known Pareto front in Figure 4.10 down, the calculated metric stays the same, while this generation becomes farther away from the true Pareto front and intuitively it covers the true Pareto front in a worse way. In this approach, the spread of the current Pareto front is calculated, while the real coverage over the true Pareto front is unknown. To overcome this shortcoming, end points of the problems were added in the result set (Deb et al, 2000) and the sum of the distances between them was calculated.

Another such metric is (Schott 1995) in which the metric is a value measuring the spread (distribution) of vectors throughout PF_{known} . While PF_{known} 's beginning and end are known by sorting the solutions along dimensions, Schott proposed the following formula to calculate the range variance of neighboring vectors in PF_{known} . In this metric, a value of zero indicates all members of PF_{known} are equally distributed. However, the true Pareto front is not necessarily uniformly spaced, similar to the above metric. Certainly, adding end points into the solution set may improve the situation.

Forth, IGD (Inverted Generational Distance) (Van Veldhuizen and Gary B Lamont 1998) combines both the second and the third metrics. In this approach, a single value tries to describe both the fitness and the spread. Let P^* be a set of uniformly distributed points along the Pareto front in the objective space. Let A be an approximate set to the PF, the average distance from P^* to A is defined as:

$$IGD = \frac{1}{|P^*|} \sum_{i=1}^{|P^*|} d(v, A)$$

where $d(v, A)$ is the minimum Euclidean distance between v and the points in A . If P^* is large enough to represent the Pareto front very well, $IGD(A; P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $D(A; P^*)$, The set A must be very close to the PF and cannot miss any part of the whole PF. In the following figure, f_1 and f_2 indicate two coordinates in the objective space.

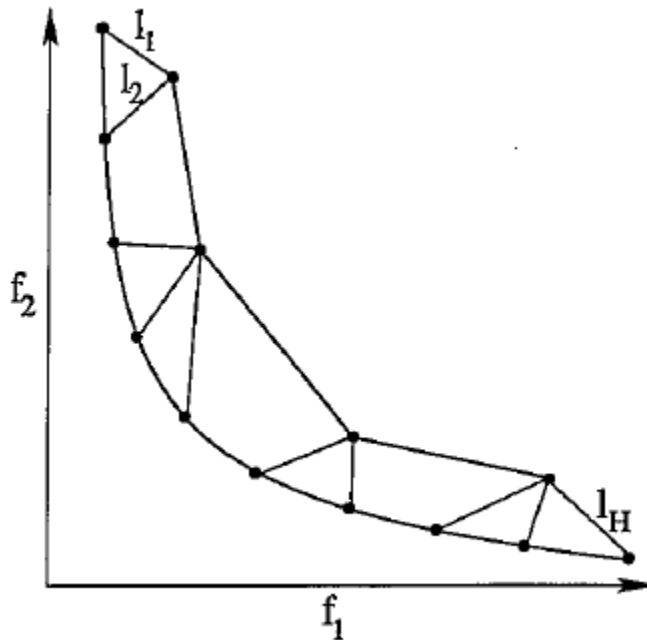


Figure 4.11 Illustration of calculation of IGD

For CEC test problems, the data file and source code of computing IGD (Zhang n.d.) was created in C and Matlab by Zhang and can be downloaded from online.

All the later three metrics have only one real number indicating the quality of the result, while the later two are of our interest.

4.5.2.2 Chosen spread metric

Schott (1995) in his Ph.D thesis proposed spacing metric which estimates the diversity of the found Pareto front. The metric value is evaluated by computing the relative distance between adjacent solutions as:

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (\bar{d} - d_i)^2}$$

where n is the number of non-dominated solutions, d_i is the distance between adjacent solutions to the solution i , \bar{d} is the average distance between the adjacent solutions. Certainly, this metric does not reflect complete information of whether the found Pareto fronts cover the real one evenly; for example, if the individuals on the first Pareto front evenly cover only a small portion of the real Pareto front, the metric values is zero, which is the best it achieves. Nonetheless, the metric is chosen by us because of its merit that it does not require any knowledge about the real Pareto front.

In addition to the spread metric, one more metric is selected by us to increase competition and dynamics in evolution, which will be introduced in the next sub section.

4.5.2.3 Chosen hyper-volume metrics

Zitzler (1999) in his Ph.D thesis proposed hyper-volume metric which was defined by the hyper volume in the objective space covered by the found Pareto front. The formula is:

$$HV = \left\{ \bigcup_i a_i | x_i \in P^* \right\}$$

where \bar{x}_l ($l = 1, 2, \dots$) is a non-dominated solution of the Pareto front P^* .

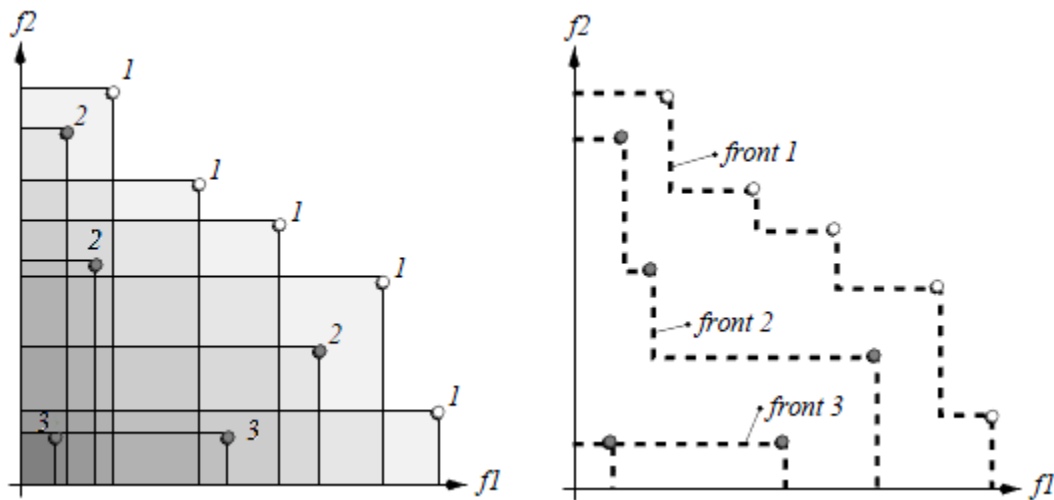


Figure 4.12 Hyper-volume of Pareto fronts

For each Pareto front, the metric value is the union of hyper volumes that are defined by the individuals that are on this front. The meaning can be easily seen in Figure 4.12 which uses a two-objective space to illustrate hyper-volume metric. There are three Pareto fronts identified in the population and all individuals are marked with their Pareto ranking numbers, as shown in the left part. The hyper volume covered by any single individual is the rectangle that is limited by axes f_1 and f_2 , and the two lines starting from the individual and perpendicular to either axis. Nonetheless, the overlapped areas are not taken into account multiple times. Finally, the hyper-volume metric values are the areas that are constrained by the serrated Pareto fronts—which is also named aliasing Pareto front since in Computer Graphics describing a curve using serrated vertical and horizontal lines is called aliasing.

4.6 The influence() function

There are two steps during influence of each generation, first, metric roulette wheels are used to choose a network configuration and then individuals under the influence will be affected by the

neighbors defined by this network. This former step of social fabric selection is transparent to the knowledge sources.

4.6.1 Selection of topologies

The topologies in MOCAT include lbest (local best), square, octal, hex, and global. However, for any given generation, there is only one social fabric in effect. The choice is made by a partially random selection with the consideration of the performances of the topologies.

The performance of a social fabric is evaluated as the metric value of the new population that has been created while this social fabric is enabled.

$$Performance(SF_i^t) = \begin{cases} Metric(population^t) & \text{if } SF \text{ is used in generation } t \\ N/A & \text{otherwise} \end{cases}$$

Since there are two metrics that are used in MOCAT, spread metric and hyper-volume metric, practically there are two different sets of performances, depending on which metric is being enabled. This implies that for each given generation only one metric is in effect.

There are two roulette wheels corresponding to the two metrics. On each of them, there are five slices that are corresponding to the five topologies; and the size of a slice is represented as the performance value of the topology.

For any new generation, when only one metric and one social fabric are enabled, the corresponding roulette wheel will be updated for that used social fabric:

$$Slice_i^{t+1} = \frac{Slice_i^t + Performance(SF^t)}{2}$$

Then, sum of slices is normalized to 1.0 and the portion of each topology is calculated. Now, we can rotate the roulette wheel to partially randomly select a topology for a new generation of evolution.

Table 4-5 Pseudo code of selecting a topology from the roulette wheel

<pre> Set sum = sum of slices Set ns_i = Slice_i / sum create random number r ∈ [0, 1.0); Set splitter = 0.0 Set topoResult = 1 While(topoResult ≤ #Topologies) Set splitter = splitter + ns_{topoResult} If (splitter > r) Return Topology_{topoResult} Else Set topoResult = topoResult + 1 End If End While return Topology_{topoResult-1} </pre>

This social fabric selection procedure is transparent to the knowledge sources. However, when a social fabric is selected for a generation, the influence of knowledge sources will infiltrate through them to more efficiently distribute knowledge and enhance evolution (Ali 2008).

After a topology is selected, every individual has defined neighbors. Then, majority voting is used to find out which knowledge source should be used for any given individual.

Table 4-6 neighbors voting for knowledge source to use on an individual Ind

Set IndSet = [Ind] Add neighbors of Ind into IndSet Create an array ksCount[#knowledgeSources] Initialize ksCount to all 0s For each individual idv in IndSet inc ksCount[knowledge source that created inv into IndSet] End For Find max value M of ksCount Create a set S containing knowledge sources ks if ksCount[ks] ==M randomly select a knowledge source R from S Return R.
--

4.6.2 Influence of knowledge sources

First, a knowledge source should be selected to influence the population. The selection algorithm has been presented in Table 4-4. After

Normative knowledge source tries to create a new individual with random location in its hyper cube. It provides coarsest heuristics while allow big flexibility to evolution.

Table 4-7 Pseudo code of influence of normative knowledge source

Create a new individual Ind For each dimension d Create a random number $r \in [0, 1]$
--

```

        Set Ind.location[d] = Interval[d].L + r * (Interval[d].H - Interval[d].L)
    End For
    If (parent individual dominates Ind)
        return parent individual
    Else
        return Ind
    End If

```

Topographical knowledge source influences a new individual with random location in one of the belief spaces that it maintains.

Table 4-8 Pseudo code of influence of topographical knowledge source

```

Randomly select a belief cell C
Create a new individual Ind
For each dimension d
    Create a random number  $r \in [0, 1]$ 
    Set Ind.location[d] = C.location[d].L + r * (C.location[d].H - C.location[d].L)
End For
If (parent individual dominates Ind)
    return parent individual
Else
    return Ind
End If

```

Situational knowledge source maintains exemplars that have been found up to now. It creates new individual by dragging the parent individual to it.

Table 4-9 Pseudo code of influence of situational knowledge source

<pre> Create a new individual Ind For each dimension d Create a random number $r \in [0, 1]$ Set $\text{Ind.location}[d] = \text{Interval}[d].L + r * (\text{Interval}[d].H - \text{Interval}[d].L)$ End For If (parent individual dominates Ind) return parent individual Else return Ind End If </pre>
--

Historical knowledge source maintains memories for every objective domain. During influence(), one memory is randomly selected and then a saved individual is randomly selected out of it to sprout a new individual around its neighborhood.

Table 4-10 Pseudo code of influence of historical knowledge source

<pre> Randomly select one individual Idv Create a new individual Ind For each dimension d Create a random number $r \in [0, 0.1]$ </pre>

```

        Set Ind.location[d] = Idv.location[d] + r
    End For
    return Ind

```

Domain knowledge explores the objective space by move along each dimension to detect the slope of it. After the maximum slop is identified, we will move along that direction 20 times with equal step length to find out 20 candidates of the new individual; then non-domination sort is used to select one from the non-domination set. Now the step length is a random number smaller than 0.005. In function influence(), one memory is randomly selected and then a saved individual is randomly selected out of it to sprout a new individual around its neighborhood.

Table 4-11 Pseudo code of influence of domain knowledge source

```

Set delta = 0.1

Create array maxSlop of #dimension elements

Initialize maxSlop to all 0s

For each dimension d

    Clone new individual Ind from Parent

    Set Ind.location[d] = Parent.lcoation[d] + delta

    If(Ind is not dominated by Parent)

        add delta to maxSlop[d]

    End If

    Set Ind.location[d] = Parent.lcoation[d] - delta

    If(Ind is not dominated by Parent)

        add -delta to maxSlop[d]

```

```
        End If
    End For
    Create random number step  $\in [0, 0.005]$ 
    Create newIndSet = empty set
    For Iteration =1 to 20
        Create individual nId
        nId.location = Parent.location + step * maxSlop * Iteration
        add nId into newIndSet
    End For
    Non-domination sort newIndSet
    return one individual from the first Pareto front
```

4.7 Summary

The execution of MOCAT is described in Figure 4.13, in which the top left element MOCAT is the control class that drives the evolution and stops it when evolution is complete. The evolution is completed upon the close collaboration of the population space and the belief space.

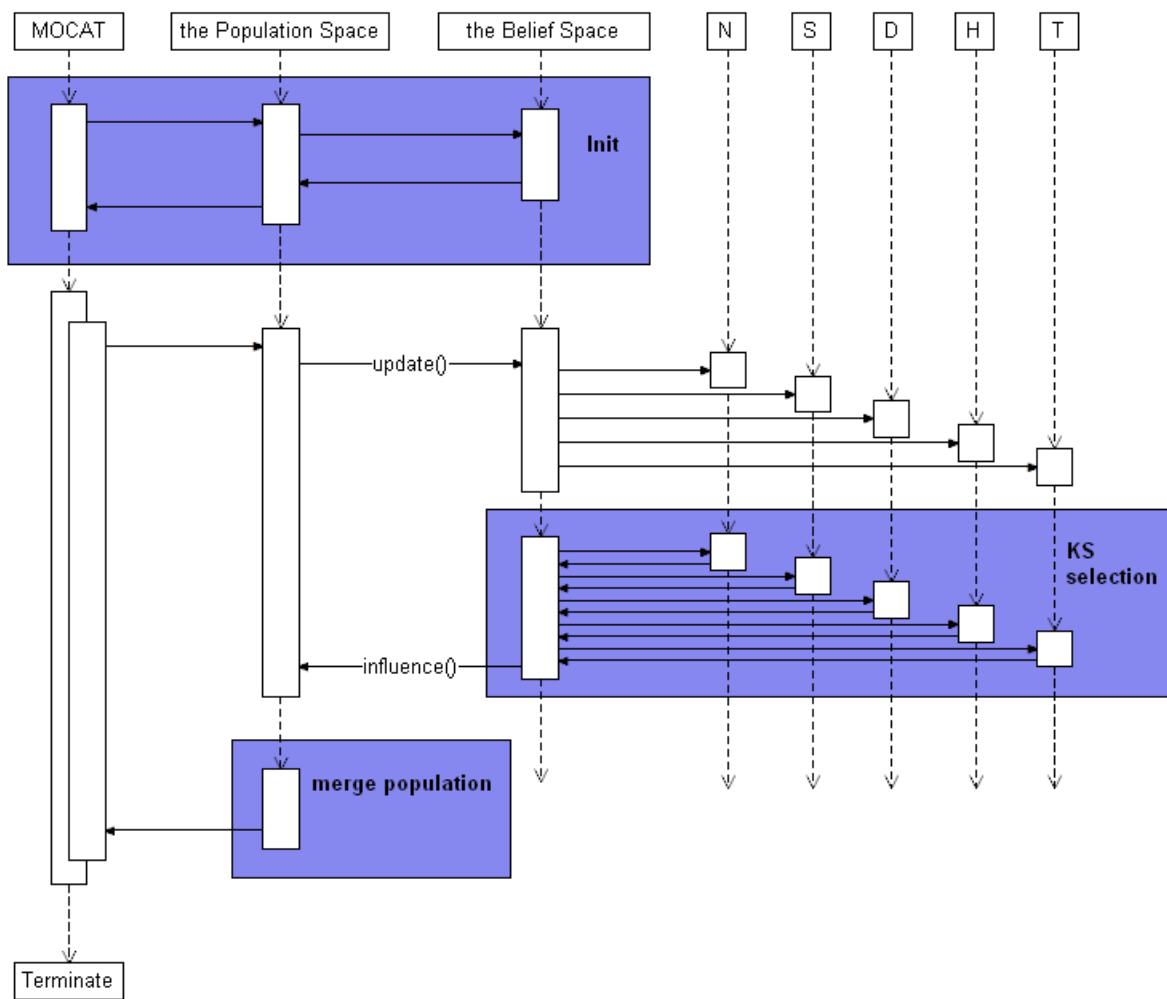


Figure 4.13 Execution of MOCAT


CHAPTER 5 AN EXAMPLE MOCAT RUN

In the previous chapter we described the MOCAT 1.0 system. In this chapter we focus on the information needed to generate a MOCAT 1.0 run and the information produced by the system as a result. We will select the ZDT1 problem described in chapter 7.0 as our example here.

This section will introduce a complete run of MOCAT on ZDT1, explain what data are created, and how they are displayed. Since Repast Symphony supports both 2-D and 3-D real-time display of evolutionary process, observing how MOCAT works is straightforward.

5.1 Start MOCAT 1.0

At the very beginning, when a project is created in Symphony, the integrated development environment (IDE) automatically creates an executable program framework and an execution configuration for it, so that both adding system behaviors and executing the evolution are easy. Additionally, Symphony automatically creates a builder which will produce a single-file installer. After the installation is done, MOCAT can run in the future without the Symphony's IDE. Certainly, programming details are not concern of this chapter while some of them can be found in Appendix I. In this chapter, we will focus on how to run MOCAT 1.0, control the evolution procedure, understand its user interface, and learn about the data that are produced along evolution.

To start running MOCAT, we click the button , Initially a menu is provided with the different execution options presented. Here, we click on "Run MOCAT model", as shown in Figure 5.1.

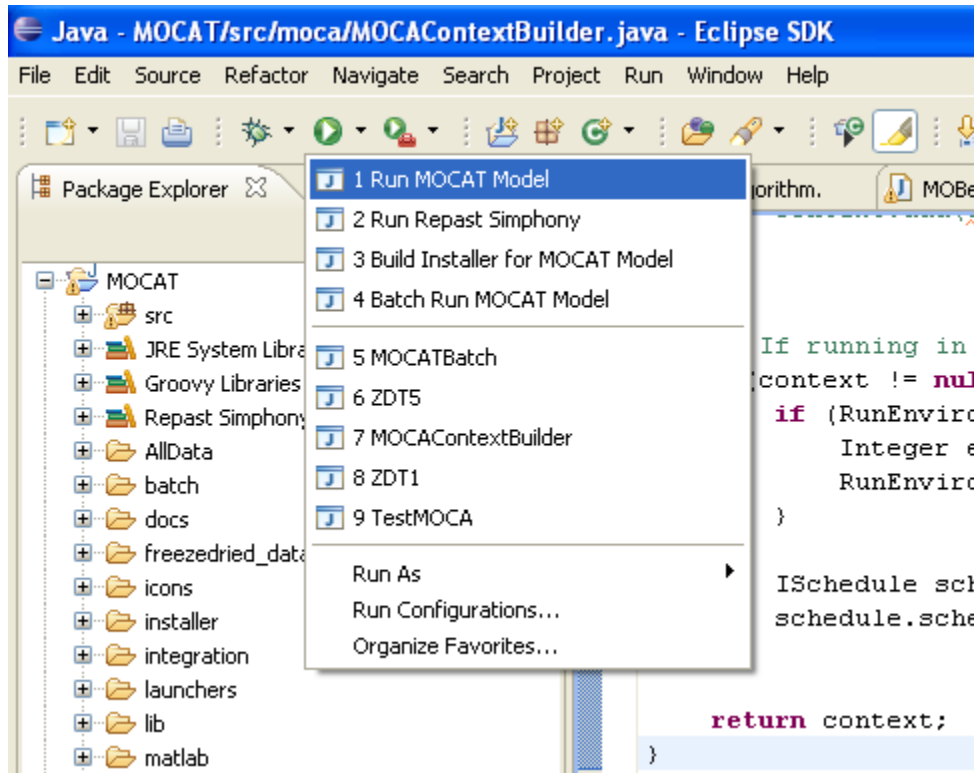



Figure 5.1 To run MOCAT model

Presentation of a Symphony project is configured at runtime. Features such as what kind of display, 2D or 3D can be configured and setup. After setup is done, we can click the disk button  which is on the toolbar or select the menu “File | Save” to remember the configuration so that future runs will automatically load the same configuration and represent the same interface to users in the future. Again, in this chapter we only care about the interface that is ready to be sued. Figure 5.2 represents the interface that has been prepared for the experiments through these activities.

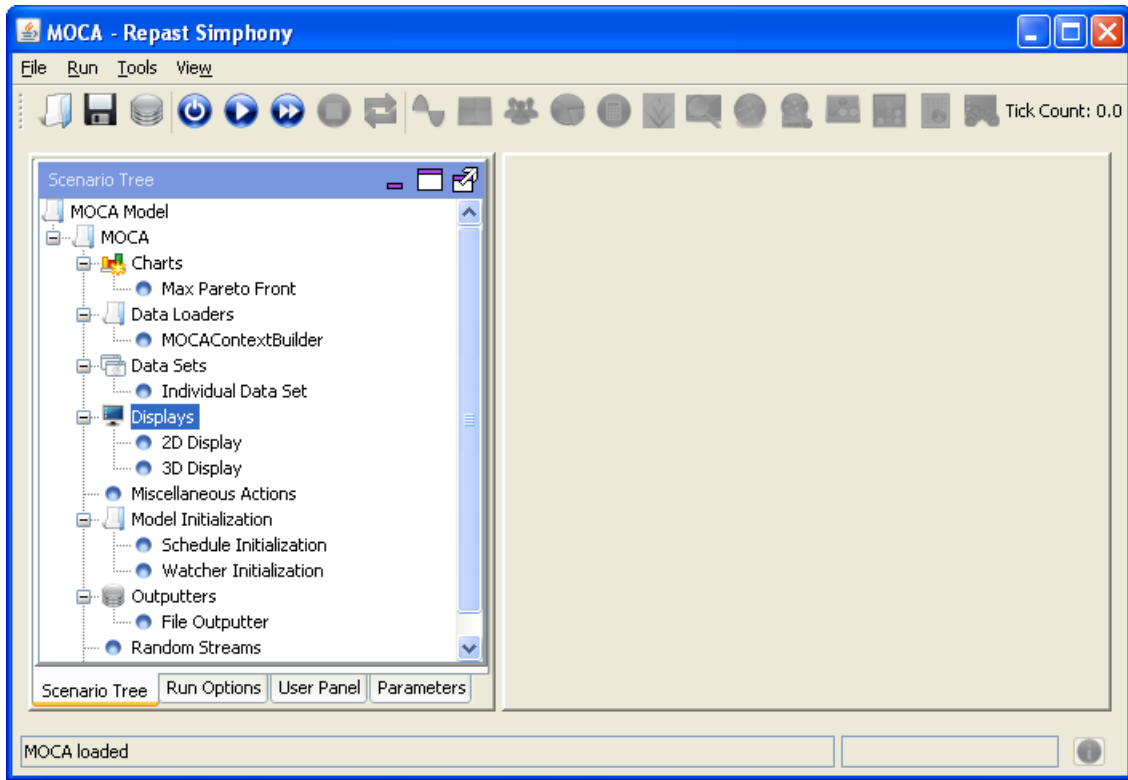


Figure 5.2 Startup of MOCAT

The parameters of MOCAT can be dynamically passed into the algorithm before evolution starts. The parameters are shown in Figure 5.3. Some parameters are related to the graphical representation or used by Symphony internally so that they are out of our interest.

- Continuous Space X Extent
- Continuous Space Y Extent
- Continuous Space Z Extent
- Default Random Seed

The parameters that are specified and used by MOCAT are:

- Dimension, dimension of the problem space, for ZDT1 it is 30; in real experiment we additionally increase it to 60
- Elitist Ratio, defined the percentage of the individuals that will be selected as voters
- Generation To Switch Metric, if spread metric and hyper-volume metric are combined, after how many generations the metric is alternated
- Number of Generation, after how many generations the evolution stops
- Number of Population, how many individuals are there in each generation
- Problem Name, one of ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, and ZDT6

At this point, MOCAT is prepared to run ZDT1 with domain dimension 30, generation 100, population 100, and every 25 generations the spread or hyper-volume metrics will be switched.

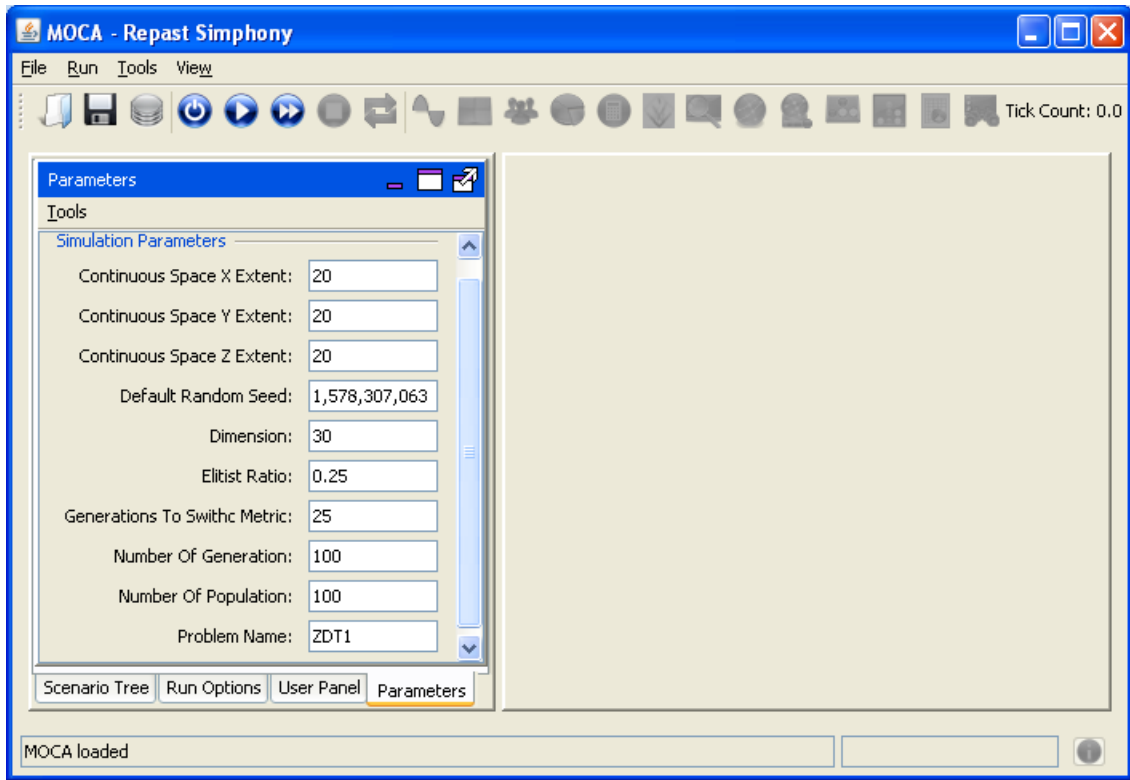



Figure 5.3 Parameters of ZDT1

To start evolution, we click the power button  on the toolbar or select the menu Run | init, to make Simphony be ready to run. We can see from Figure 5.4 that setup of MOCAT is disabled at this point.

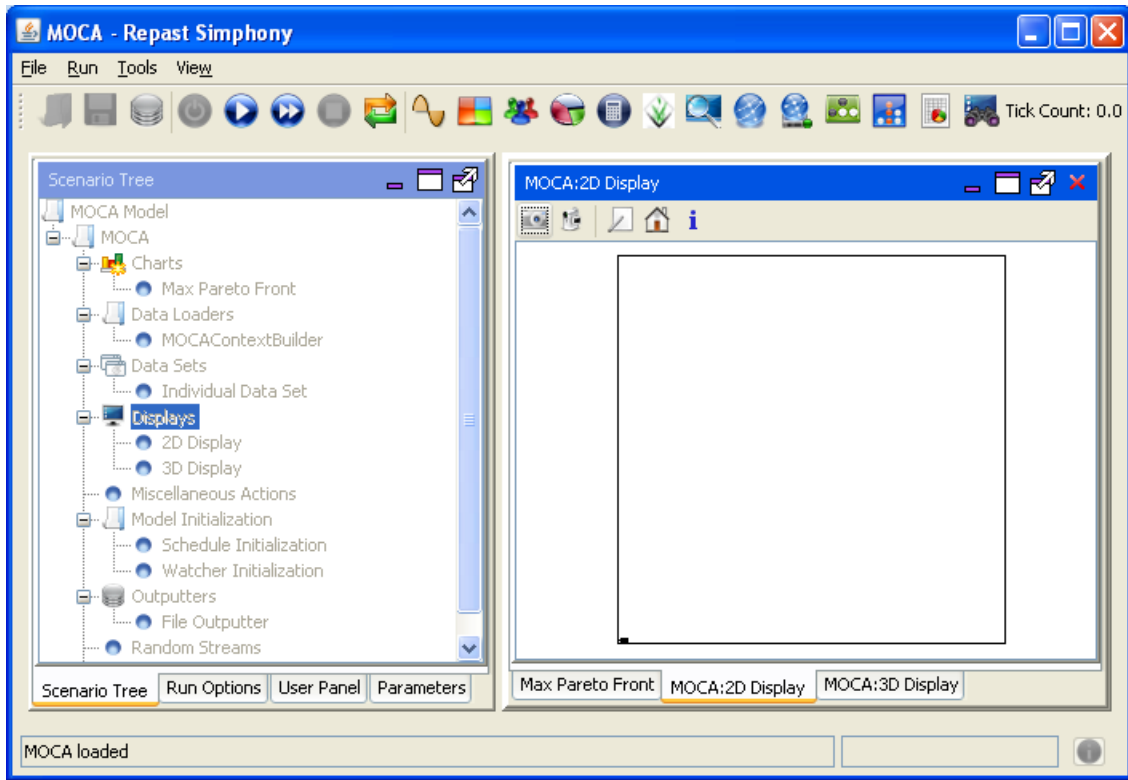




Figure 5.4 MOCAT is ready to run

5.2 Control the evolution

After MOCAT is initialized, we can either step on the evolution by clicking  so that we can observe progress in details, as shown in Figure 5.5, or run MOCAT non-stop by clicking  as shown in Figure 5.6. Corresponding control buttons are disabled in either situation.

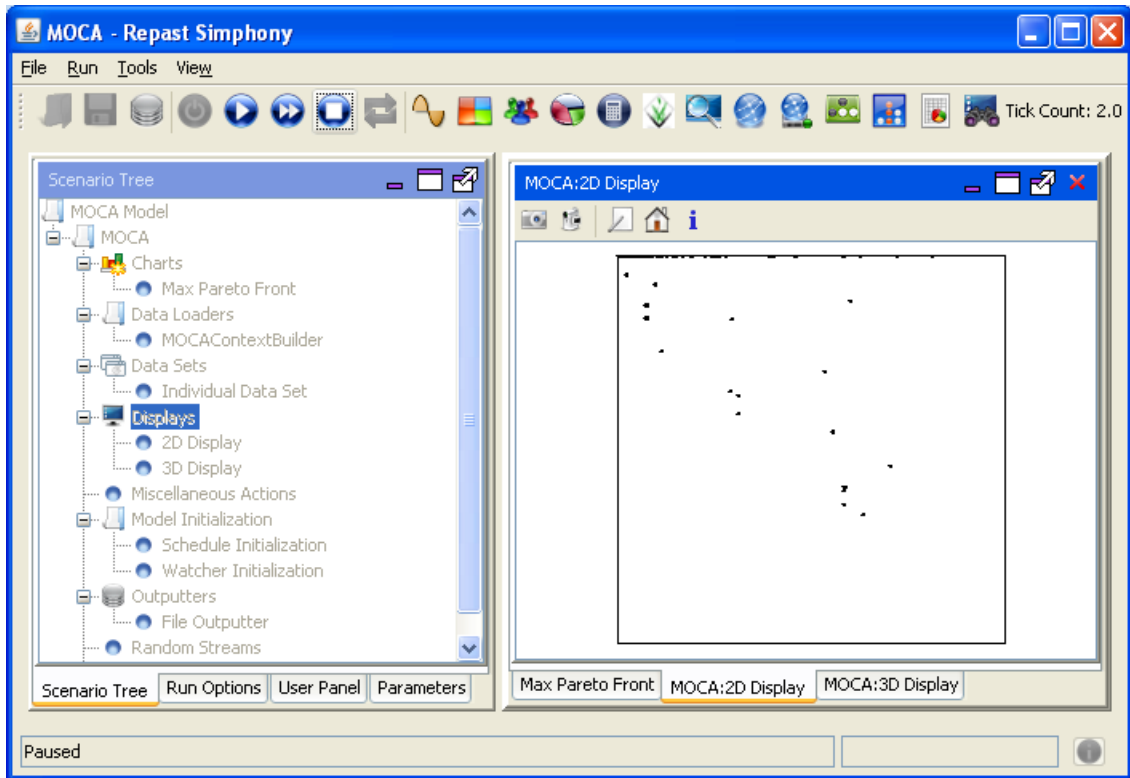


Figure 5.5 Step run of MOCAT

Along with the evolution, the generation number is told by the number shown in the top right corner, such as **Tick Count: 2.0**.

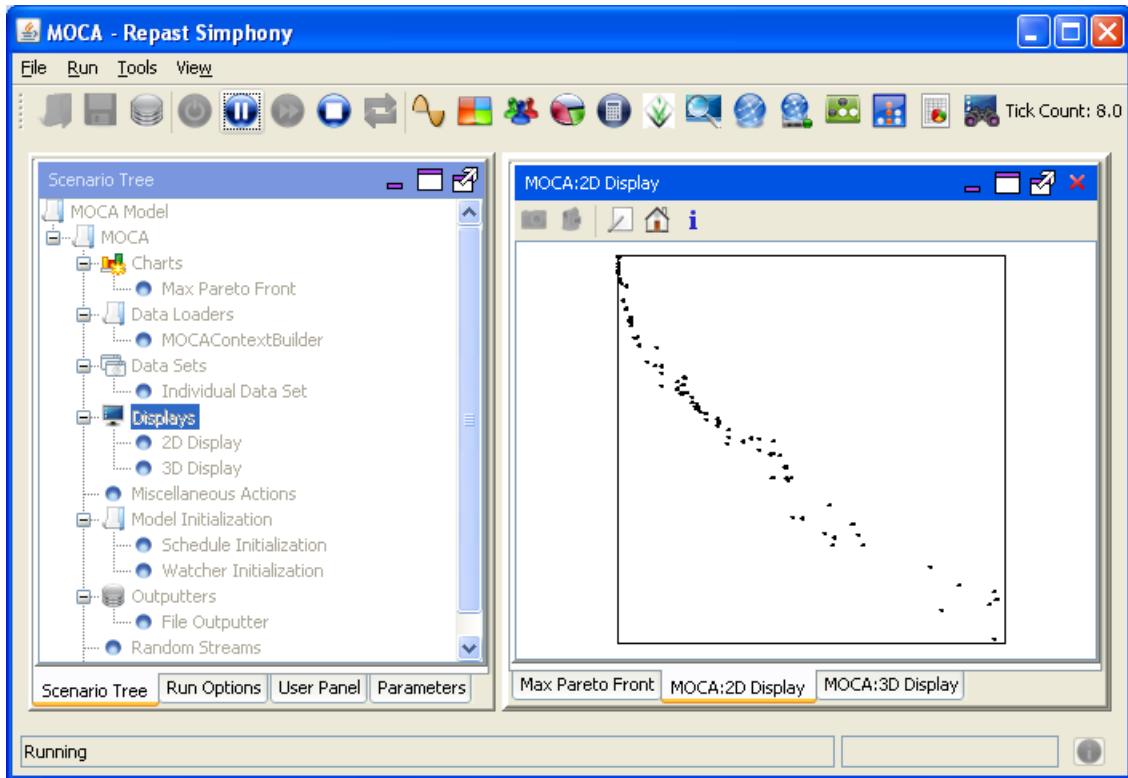


Figure 5.6 Run of MOCAT

Along with evolution, the most interesting interface component is the 2D display, in which the solutions in the objective space are shown. The displays dynamically refresh along with the evolution and reflect the up-to-date information.

When the maximum generation number is reached, execution ceases as shown in Figure 5.7. Termination of the evolution can be told by the still picture and the unchanged Tick Count that is located at the top right corner.

At this point, we can visually tell how good this evolution run is by observing the figure shown in the 2D display.

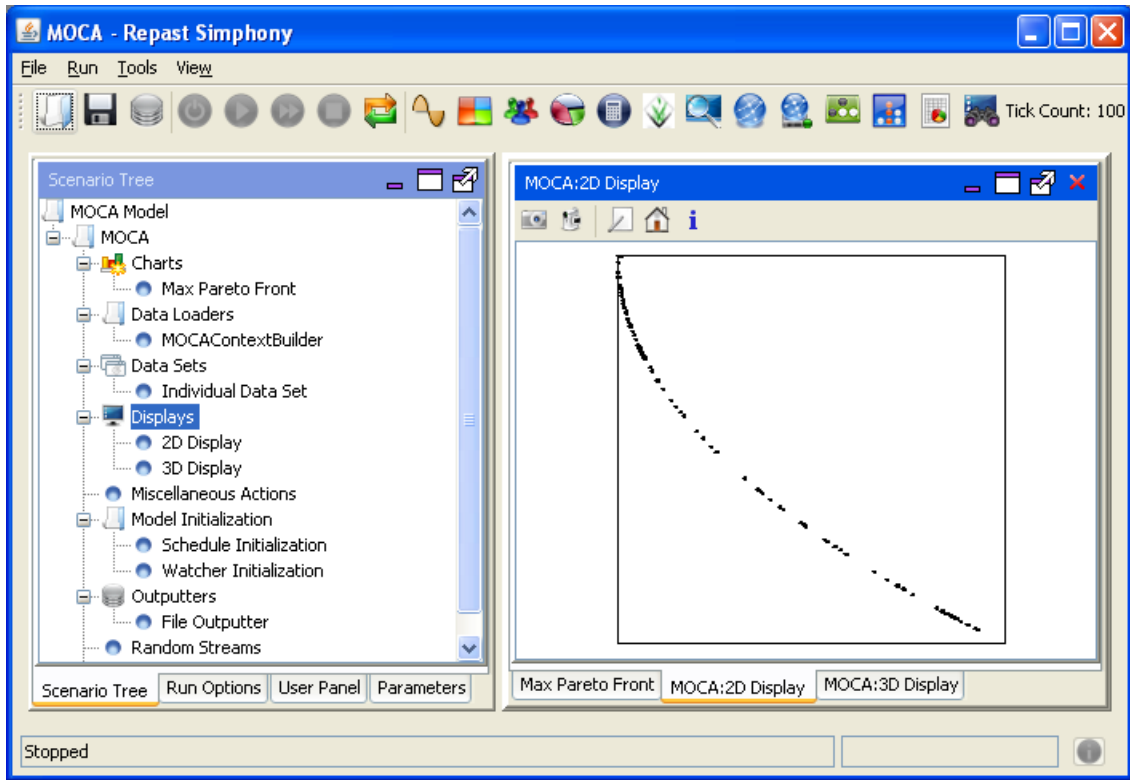


Figure 5.7 MOCAT finishes on run

5.3 Collected data

5.3.1 Information about a single run

Along with evolution, details about individual solutions from each generation are recorded in files; and when evolution is finished, statistical information about this run is sent to summary files. To facilitate the automation of data collecting and analyzing, such data files are specially named and organized. Each run is assigned an exclusive id, *RunID*, which is generated according to the experiment time and is different from other ids.

Evolutionary details are saved in a sub-folder in the data collection folder with the name format *ProblemName-GenerationNumber-PopulationNumber-Dimension-GenerationsToSwitchMetric_RunID*. A

sample name is “ZDT1-100-100-30-200.43204”. This long name reveals how we organize saved data. While the naming convention is easy to understand, *GenerationToSwitchMetric* needs special explanation: number 200 indicates that the evolution only uses spread metric, number 300 indicates that the evolution only uses hyper-volume metric, and all other numbers specify after how many generations the metric will be switched in MOCAT.

Coordinates of individuals are recorded in a file named *GenerationCounter.coord.txt* and the locations in objective space are saved in a file named *GenerationCounter.obj.txt*.

When the evolution is done, five other files are additionally saved. Files *ProblemName-GenerationNumber-PopulationNumber-Dimension-RunID-topology*.txt* remember metric values and their quotations on the roulette wheel of all generation in time order. Files *ProblemName-GenerationNumber-PopulationNumber-Dimension-RunID-knowledge*.txt* record how every knowledge source behaves along evolution under the influence of different metrics. The last file *ProblemName-GenerationNumber-PopulationNumber-Dimension-RunID-summaries.txt* contains textual information which is ready to be copied as an introduction to this test run, the text reads like,

“On ZDT1 with dimension 30 run #ZDT1-100-100-30-25.49768 finished in 35 (s).

In the last generation, the absolute range of fitness error is [0.000000000, 5.119815513].

In evolution, every 25 generations the metric for network configuration is switched.

In Spread roulette wheel, each network configuration (Topology) is used : 18 1 8 8 15 times

In Volume roulette wheel, each network configuration (Topology) is used : 12 9 8 14 7 times.”

Such information is high-level and intended to be read by human readers. Statistical information, such as how knowledge sources change, how social fabrics are used, etc, is saved in summary files which will be introduced next.

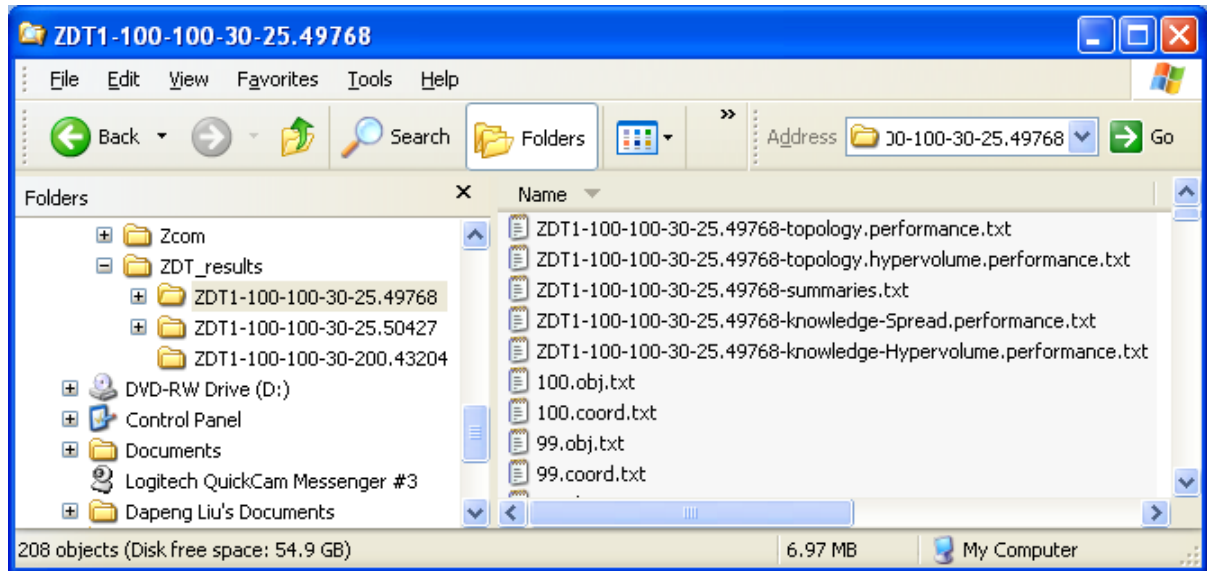


Figure 5.8 Data collected for one single run

5.3.2 Information about all runs on the same problem

In the data collection folder, there are a few summary files containing information about evolution of all test runs that have been done up to now. Every test that has been done on the same problem with the same generation, population, and dimension is collected in a single total summary file, such as ZDT1-100-100-30-totalSummary.csv. These files record the status of the roulette wheel in the belief space, performances of topologies, and the statistical information of the found solutions.

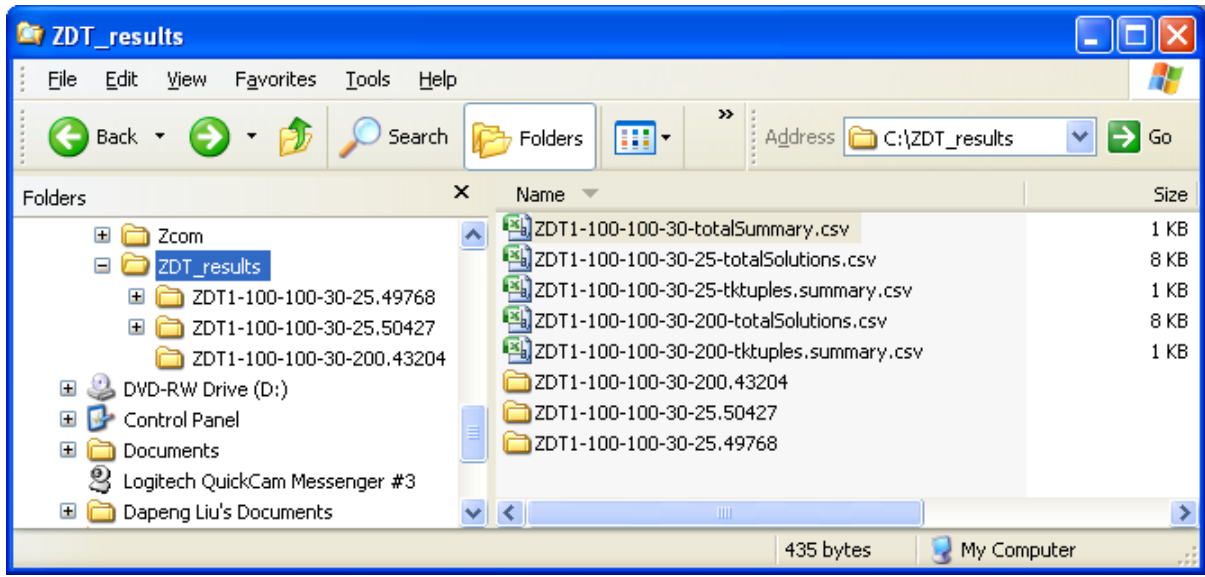


Figure 5.9 Data files after several experiment runs

Depending on how spread metric, hyper-volume metric, or combined metrics are used in evolution, the other two sets of summary files record all solutions that have been found in test runs and all relationships between knowledge sources and topologies.

Information saved in summary files will be used after test runs are completed to produce various statistics about the experiments. To complete this thesis, Microsoft Excel is used because of its general availability.

CHAPTER 6 EXPERIMENTAL FRAMEWORK

We want to check whether MOCAT can get competitive results fast enough, whether the evolution is robust and stable, and whether it can handle scalable problems with stable performance. In other words, we want to find out whether MOCAT can compete with other capable MOEAs. In addition, in complex evolution, we want to observe interactions of metrics and problems—whether one metric may work better on some kinds of problems than on others; interactions of metrics and dimensionality—whether metrics behave differently while dimensionality rises up; how metrics select specific topologies or whether they prefer any one; etc. Thereafter, we need a benchmark that contains a variety of different kinds of optimization problems and has been popularly adopted in related publications.

Based on the considerations listed above, ZDT test suite (Zitzler et al, 2000) was selected for our experiments.

Realizing that *“Two major problems must be addressed when an evolutionary algorithm is applied to multi-objective optimization: 1. How to accomplish fitness assignment and selection, respectively, in order to guide the search towards the Pareto-optimal set. 2. How to maintain a diverse population in order to prevent premature convergence and achieve a well distributed trade-off front.”*, Zitzler etc. designed the benchmark problems with special consideration in mind.

Concerning the first issue, multi-modality, deception, and isolated optima—which indicates that the Pareto front can be divided into separate areas each of which seems continuous—are normally obstacles to evolution towards real Pareto front. As to the second issue, convex or non-convex Pareto

front shape, discreteness of Pareto front, and non-uniformity of density of Pareto front are obvious challenges to MOEAs. Thereafter, Zitzler et al. designed six problems each of which was corresponding to different considered factors. However, in fact, none of the problems has even density of Pareto front, which is understandable since all of them are curves. In addition, they restricted the problems to only two objectives in order to observe evolution easily while two objectives were still *“sufficient to reflect essential aspects of multi-objective optimization.”*

Finally, following DTLZ benchmark problems that had been proposed earlier (Deb etc, 1999), all test problems share the same mathematical formula format. Such a format allows domain modality to be easily scaled up. Similar to DTLZ problems as well, ZDT benchmark was composed with the consideration of including challenges to MOEAs so that they are abstraction of practical problems but not direct derivatives of them. The motivation for composing a set of testing problems is to facilitate the observation on the same problem while its modality changes but not compare the efficiency of MOEAs upon different problems (Deb et al, 1999).

ZDT benchmark has been generally used in recent optimization venues ().

6.1 ZDT 1

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

$$f_2 = g * h$$

where $m=30$ and $x_i \in (0,1]$. The Pareto-optimal front is formed with $g(x) = 1$.

The Pareto front is:

$$x_2 = 1 - \sqrt{x_1}$$

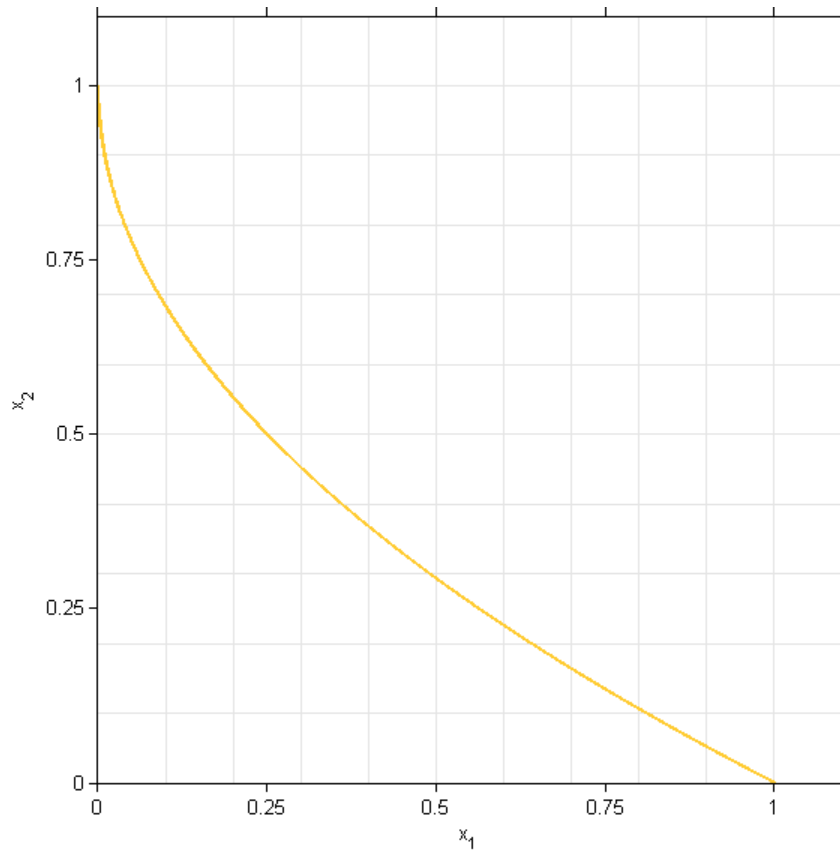


Figure 6.1 ZDT1 Pareto front

ZDT1 has a concave and continuous Pareto front and may be handled by most MOEAs efficiently.

6.2 ZDT 2

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g}\right)^2$$

$$f_2 = g * h$$

where $m=30$ and $x_i \in (0,1]$. The Pareto-optimal front is formed with $g(x) = 1$.

The Pareto front is:

$$x_2 = 1 - x_1^2$$

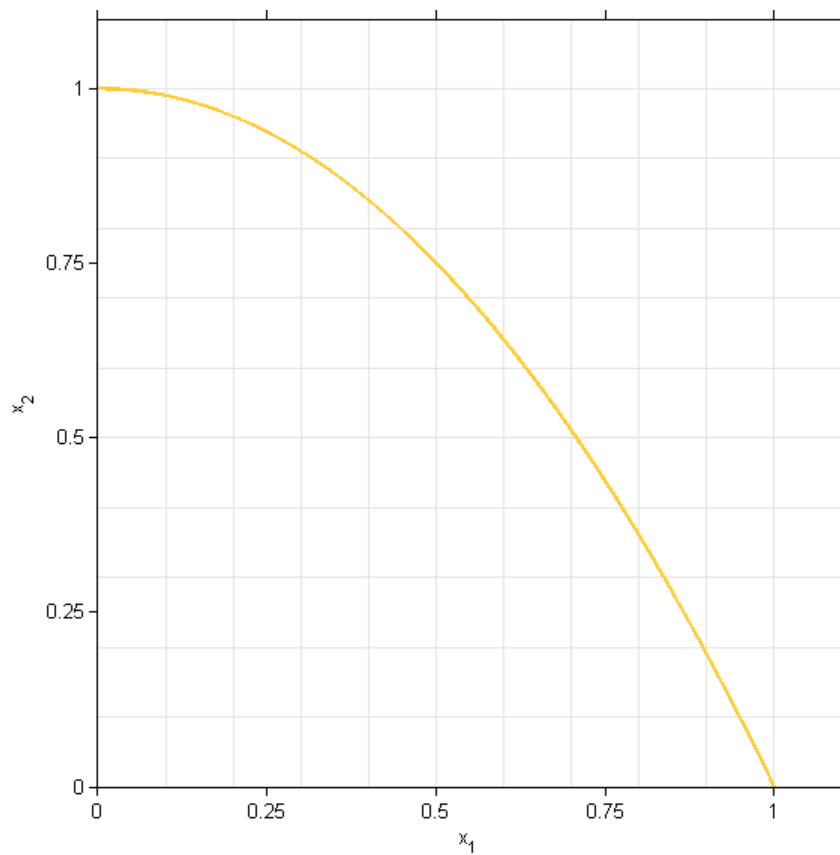


Figure 6.2 ZDT2 Pareto front

ZDT2 has a continuous Pareto front but it is convex; which may be a challenge to some MOEs (Deb, 2001) because they tried to reduce the area encircled by the found Pareto front and X and Y axes.

6.3 ZDT 3

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \sqrt{x_1/g} - \frac{x_1}{g} * \sin(10 * \pi * x_1)$$

$$f_2 = g * h$$

The Pareto front is:

$$x_2 = 1 - \sqrt{x_1} - x_1 * \sin(10 * \pi * x_1) \text{ where}$$

$$x_1 \text{ in } [0, 0.0830015349] \cup (0.1822287280, 0.2577623634] \cup$$

$$(0.4093136748, 0.4538821041] \cup (0.6183967944, 0.6525117038] \cup$$

$$(0.8233317983, 0.8518328654]$$

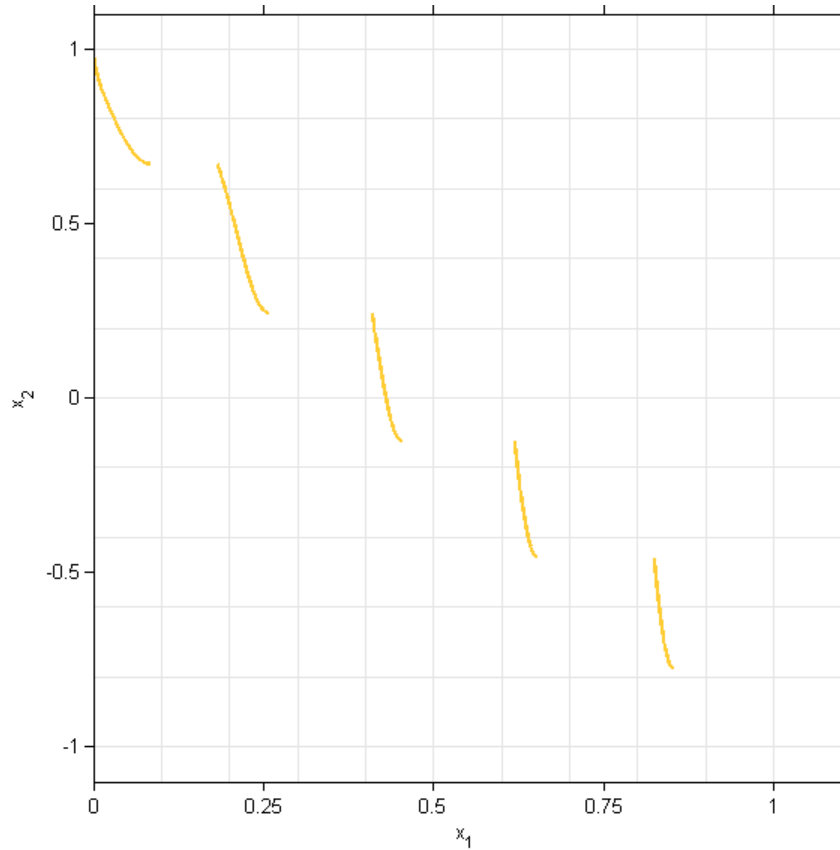


Figure 6.3 ZDT3 Pareto front

Clearly, ZDT3 has an isolated Pareto front, which can be deemed as five separate sub-problems. Intuitively, many spread metrics that evaluate whether the density of Pareto front is even will be challenged by ZDT3.

6.4 ZDT 4

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 10 * (n - 1) + \sum_{i=2}^m (x_i^2 - 10 * \cos(4 * \pi * x_i))$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g}\right)^2$$

$$f_2 = g * h$$

where $m=30$ and $0 \leq x_1 \leq 1$ and $x_i \in [-5, 5]$ for $i = 2, \dots, m$.

The Pareto front contains 21^9 local Pareto fronts and the overall curve is described as:

$$x_2 = 1 - \sqrt{x_1}$$

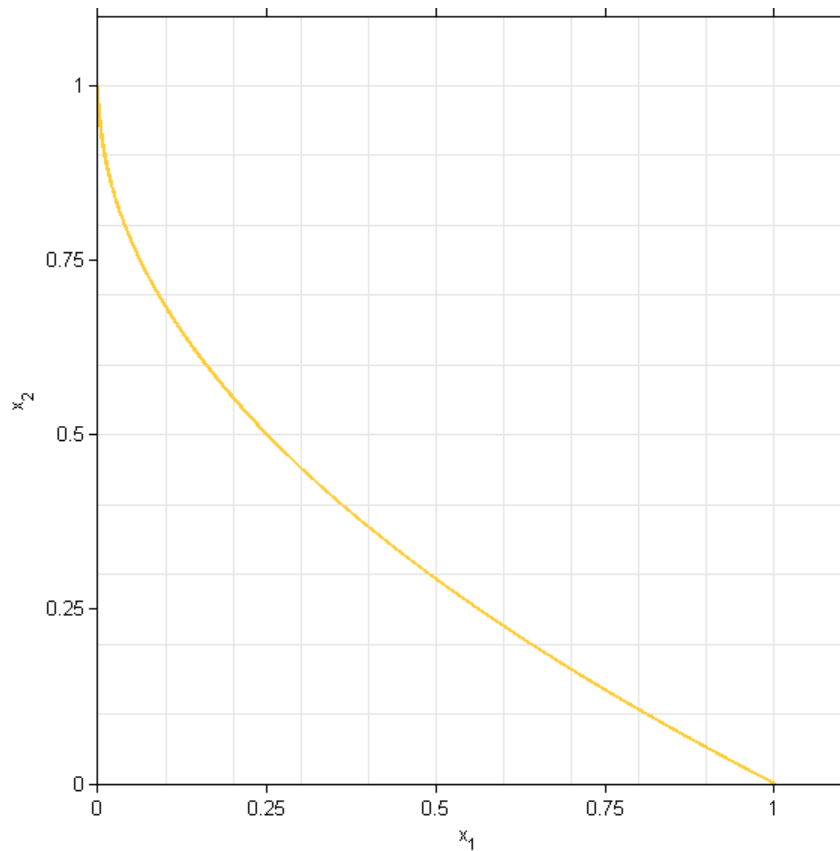


Figure 6.4 ZDT4 Pareto front

The Pareto front of ZDT4 looks like the one of ZDT1, however, they are fundamentally different: while the Pareto front of ZDT1 is continuous, there the Pareto front of ZDT4 contains 2^{19} local Pareto

fronts. This means that if an algorithm cannot jump out of local neighborhood it will be stuck in a small area.

6.5 ZDT 5

$$f_1 = 1 + \mu(x_1)$$

$$g(x_2, \dots, x_m) = \sum_{i=2}^m v(\mu(x_i))$$

$$h(f_1, g) = 1 - \frac{1}{f_1}$$

where $\mu(x_1)$ gives the number of ones in the bit vector x_1 (unitation).

$$v(\mu(x_i)) = \begin{cases} 2 + \mu(x_i) & \text{if } \mu(x_i) < 5 \\ 1 & \text{if } \mu(x_i) = 5 \end{cases}$$

and $m=11$, $x_1 \in (0, 1)^{30}$, and $x_j \in (0, 1)^5$ for $j=2, \dots, m$.

The Pareto front is formed with $g(x)=10$ while the best deceptive Pareto front is represented by the solutions for which $g(x)=11$.

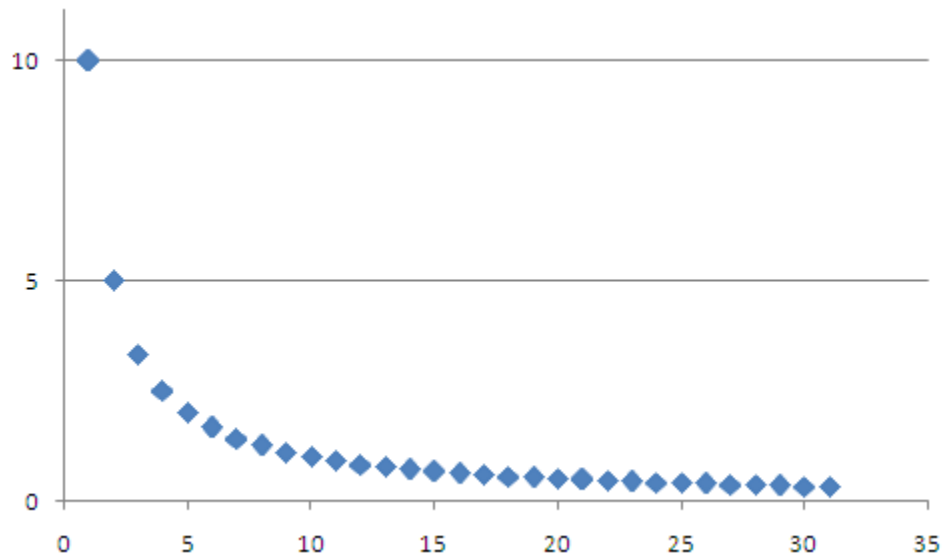


Figure 6.5 ZDT5 Pareto front

ZDT5 stands out with several special characteristics; first, its problem description does not use mathematic formula directly; second, its Pareto front is not continuous; third, the area of domain space that contributes to the Pareto front is a tiny portion of the whole domain space thereafter ZDT5 will beat many MOEAs that solely depend on gradient exploration.

6.6 ZDT 6

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \left[\sum_{i=2}^m x_i / (m - 1) \right]^{0.25}$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g} \right)^2$$

$$f_2 = g * h$$

where $m=30$ and $0 \leq x_1 \leq 1$ and $x_i \in (0,1]$ for $i = 2, \dots, m$.

The Pareto front is:

$$x_2 = 1 - x_1^2 \text{ while } x_1 \in [0.2807753191, 1].$$

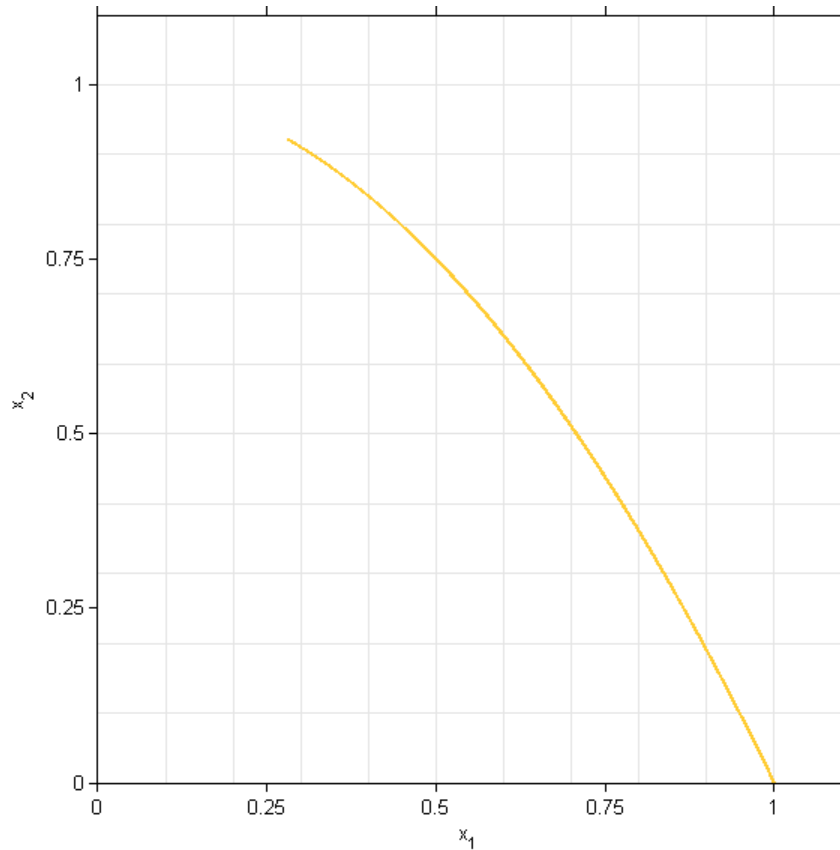


Figure 6.6 ZDT6 Pareto front

The Pareto front of ZDT6 does cross the whole objective space so that it is an extremity of having uneven density of Pareto front.

For selected test problems, we will run MOCAT 20 times with population of 100, generation of 100, and may use Spread metric, Hyper-volume metric, and both of them alternatively after 25 generations. All results will be presented and summarized with statistical analyses.

6.7 Summaries

MOCAT runs on each problem of ZDT benchmark and execution processes are detailed recorded. During one run, maybe only spread metric is effective, or only hyper-volume metric is effective, or both; we try each configuration 20 runs. At the end of each run, statistics are made to present the characteristics of the run. Finally, statistics of experiments are made to sum up. Additionally, to test the scalability of MOCAT, we will double the standard dimensions of ZDT problems and redo the experiments. Admittedly, doubling dimensions increase computation load and challenges to MOCAT.

A couple of result sets are extracted from related literature and used as a baseline to evaluate the performance of MOCAT. The following table was taken from (Bastos-Filho and Miranda, 2011) which only contained the performance values.

Table 6-1 Existing benchmark results I

Algorithms	Metrics	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MOPSO	Spread	0.0046/5E-4	0.006/0.001	0.005/4E-4	0.006/0.0014	0.129/0.122
	Hyper-volume	0.36/0.002	0.69/0.001	0.950/0.004	0.631/0.526	1.261/0.386
m-DNPSO	Spread	0.0457/0.014	0.054/0.017	0.045/0.016	0.04/0.037	0.126/0.108
	Hyper-volume	0.713/0.053	0.94/0.06	1.296/0.088	2.157/0.935	1.279/0.506
MOPSO	Spread	0.0042/6E-4	0.006/0.001	0.006/9E-4	0.005/9E-4	0.186/0.145
CDLS	Hyper-volume	0.39/0.003	0.716/0.0035	1.006/0.009	4.82/2.174	1.717/0.519
CSS	Spread	0.0023/1E-4	0.0035/7E-4	0.003/7E-4	0.005/0.1112	0.234/0.153
MOPSO	Hyper-volume	0.34/0.002	0.674/0.001	0.953/0.008	5.38/2.54	2.051/0.697
MOPSO	Spread	0.0033/2E-4	0.0032/1E-4	0.003/2E-4	0.003/3E-4	0.088/0.056
CDR	Hyper-volume	0.33/3E-5	0.66/3E-5	0.92/1E-4	0.57/0.26	1.670/0.300
MOPSO	Spread	0.0027/1E-4	0.0029/2E-5	0.0025/0.0	0.0025/2E-4	0.078/2E-3
CDRS	Hyper-volume	0.31/2E-5	0.656/3E-5	0.94/6E-5	0.56/0.012	1.345/0.46

Tables borrowed from (Li et al, 2008). The performance values are listed in Table 6-2:

Table 6-2 Existing benchmark results II

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4
-----------	------	------	------	------

NSGA2	mean	0.03348	0.07239	0.11450	0.51305
	stdev	0.06892	0.177989	0.089107	0.34418
PESA2	mean	0.00105	0.00074	0.00789	9.98254
	stdev	0.0000	0.00000	0.010488	4.487093
MOPSO	mean	0.00133	0.00089	0.00418	7.37429
	stdev	0.00000	0.00000	0.00000	2.341551
AMOPSO	mean	0.00099	0.00074	0.00391	0.40311
	stdev	0.00000	0.00000	0.00000	0.112205
ANMOPSO	mean	0.00005	0.00000	0.00006	0.04083
	stdev	0.00000	0.00000	0.00000	0.039749

The original paper used variance in its tables, which have been calculated into standard deviation, i.e., square root of their values. Calculation of our fitness error makes our error data look bigger than the real values. For concave curve, we use the vertical distance from the individual to the curve as the fitness values, for concave curve, we use the horizontal distance from the individual to the curve, which can be much bigger than the distance from the individual to the curve. MOCAT runs in this way without any problems.

We use double precision numbers in our emulation and use scientific representation in our tables for them to avoid over large length.

At the end of evolution, only the individuals that have comparatively good performance and stay at the first Pareto front will be evaluated for statistics. However, in the middle of evolution, all individuals will be taken into count to calculate statistics. This is a reasonable computation method because we want to introduce randomness into the evolutionary procedure.

For topologies, in each run we will use one concrete example to represent how they behave during the whole evolution, whilst the statistical table renders a general picture of their behavior.

6.8 Hardware and software configurations

MOCAT was implemented in Java on top of Repast Symphony. The Java version is 1.6_20 and had the default configuration.

The experiments were done on a computer with 2G memory, 5400 rpm 3.5-inch hard disk, and a CPU of Intel Core 2 Duo T5270 that is running at 1.40GHz. This computer has an on-board video card. The operating system is Windows XP Home edition with Service Pack 3 installed.

During experiments, only the evolution time is counted while the graphical interface initialization time is ignored. As a fact we find that execution time is very stable for the same problem in different runs. There were no performance data in terms of evolution time in related literature. Here we recorded the average execution time in numbers with two decimal digits.

6.9 MOCAT initialization

To prepare MOCAT to run, the population solution is filled with randomly created individuals. Then we calculate the performances for all social fabrics and update the population space roulette wheel accordingly. Otherwise, after the first generation one social fabric on the wheel has non-zero values while all others have zeros; that means the social fabric for the whole evolution has been fixed.

In the belief space, the roulette wheel for knowledge sources is initialized with performance values of the knowledge sources after they temporarily influence the population. However, the new created individuals are simply discarded instead of merged into the population and all knowledge sources will clear their memory after the initialization. The motivation for us to test run one step is to figure out the value range of the performances of the knowledge sources. For example, for one

problem the ranges may be [0.1, 1] while for another problem the ranges may be [0.001, 0.1]; excessively large or smaller initial values may lead evolution to wrong directions at the beginning. For example, if we give over big initial values, after function update() is executed, the knowledge sources that contribute to pushing the found Pareto front are punished because their performance values are actually decreased.

6.10 Definition of fitness error

To evaluate how close the solutions approach the real Pareto front, the minimal distance of the solution to the Pareto front is often used. In MOCAT, to save computation power, fitness error was defined and used in measuring the closeness of the solutions to the real Pareto front.

$$FitnessError(f_1, f_2) = |f_2 - f_2^*|$$

Where f_2^* is the fitness value on the second objective while f_1 is fixed. The graphical representation of the meaning of the fitness error is shown in Figure 6.7, which the yellow curve depicts the real Pareto front. As to ZDT problems, the fitness error for an individual is generally bigger than the minimal distance from the individual to the real Pareto front.

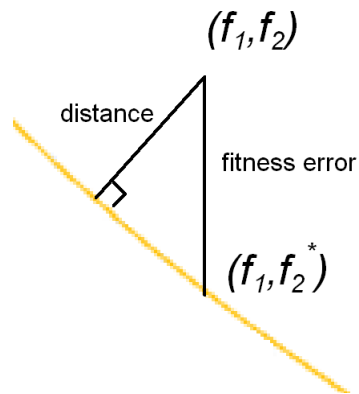


Figure 6.7 Fitness error vs. minimal distance to real Pareto front

6.11 Hypotheses

At first, we want to test whether MOCAT can efficiently handle multi-objective optimization. Second, we want to test whether MOCAT is scalable, i.e., whether it can handle higher-dimension problems. In addition to these, we want to observe internal behaviors inside MOCAT; to be precise, how knowledge sources and topologies interact with each to contribute to evolution.

There are a series of hypotheses that we want to test, and these null hypotheses are:

1. All knowledge sources contribute equally to solution finding;
2. All topologies contribute equally to solution finding;
3. Specific knowledge sources have no preference of topologies, and vice versa.

We will use statistical approaches to validate these null hypotheses.

6.12 Collection of data

During the experiment, some data are routinely collected, which include:

1. The sequence of screen copies of the first run, so that the evolution can be visually presented;
2. For each run, the fitness errors of the ending solutions and statistics upon them for all runs; at the end of evolution, only the non-dominated individuals are picked as solutions while dominated individuals are discarded;
3. Under a specific metric, how many times the topologies are used for each run, and the statistics upon them for all runs;
4. Under a specific metric, how many times the knowledge sources influence non-dominated individuals for each run, and the statistics upon them for all runs;
5. Under a specific metric, how many times the knowledge sources influence non-dominated individuals through different topologies for each run, and the statistics upon them for all runs.

CHAPTER 7 SOLVING PROBLEMS WITH CONVEX PARETO FRONTS USING MOCAT: ZDT1

In this chapter we begin a looking at how MOCAT deals with problems that generate a traditional convex Pareto front. In terms of such problems we will investigate the following questions:

1. What is the relative effectiveness of the Homogeneous Spread metric, Homogeneous Hyper-volume metric, and the combined metrics in directing the generation of the Pareto Front using the Cultural Algorithm?
2. What are the relative contributions of the various knowledge sources to the generation of convex Pareto fronts?
3. What social fabric topologies are most effective in spreading information through the population during the generation process?
4. How are these relationships affected as the dimensionality of the problem is increased from 30 dimensions to 60?
5. How do these different configurations compare to the results produced for convex Pareto fronts by other MOEAs?

We begin in section 7.1 by describing the ZDT1 problem.

7.1 The ZDT 1 Function Specification

There are several features that can cause problems for Evolutionary Algorithms in terms of generating an optimal Pareto Front and maintaining diversity within the population.

One such situation can occur when the optimal front exhibits a convex curve which means that the neighbors of a given non-dominated solution are non-linearly related to each other.

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

$$f_2 = g * h$$

where $m=30$ and $x_i \in (0,1]$. The Pareto-optimal front is formed with $g(x) = 1$.

The Pareto front is:

$$x_2 = 1 - \sqrt{x_1}$$

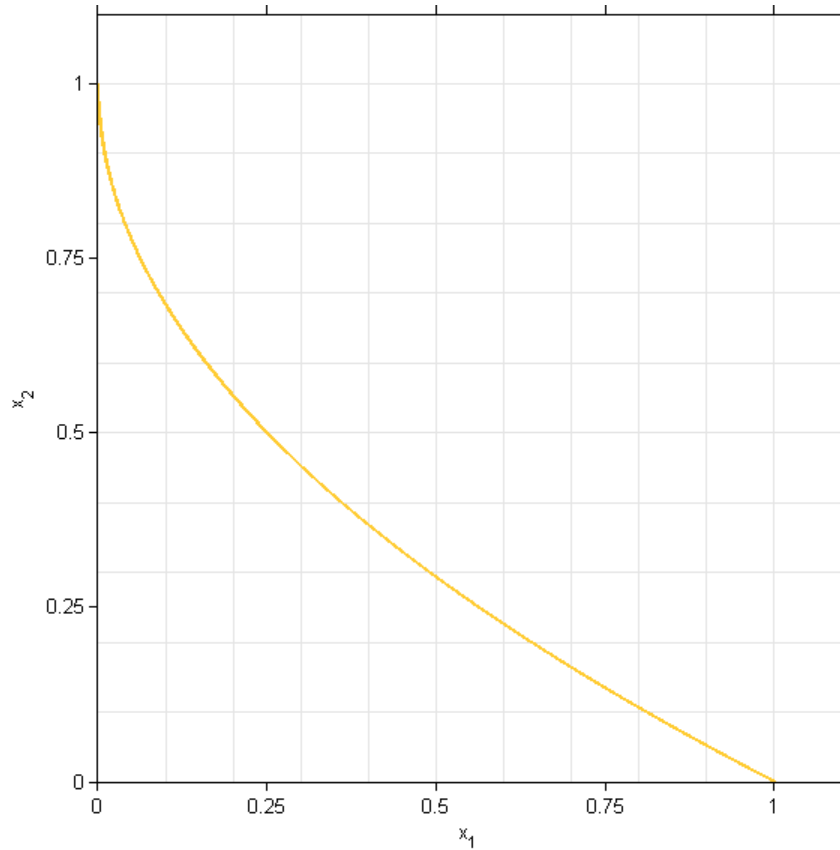


Figure 7.7.1 ZDT1 Pareto front

Figure 7.1 gives the optimal Pareto front for this problem. ZDT1 has a convex and continuous Pareto front and may be handled by most MOEAs efficiently. The continuity of the front will make it relatively easy to generate diversity, whereas the non-linearity of the optimal front will provide a challenge to MOEAs. Most MOEAs have been developed to handle this category of problems.

7.2 Performance of the Cultural Algorithm Using a Spread Metric on ZDT1 with 30 Dimensions

The runs were complete in an average of 33.54 seconds for 100 generation and 100 individuals in each population.

7.2.1 The Evolution of the Found Pareto Front

In this section we provide a series of screen shots that describe the evolution of the Pareto front over time using just the spread metric as a performance guide.

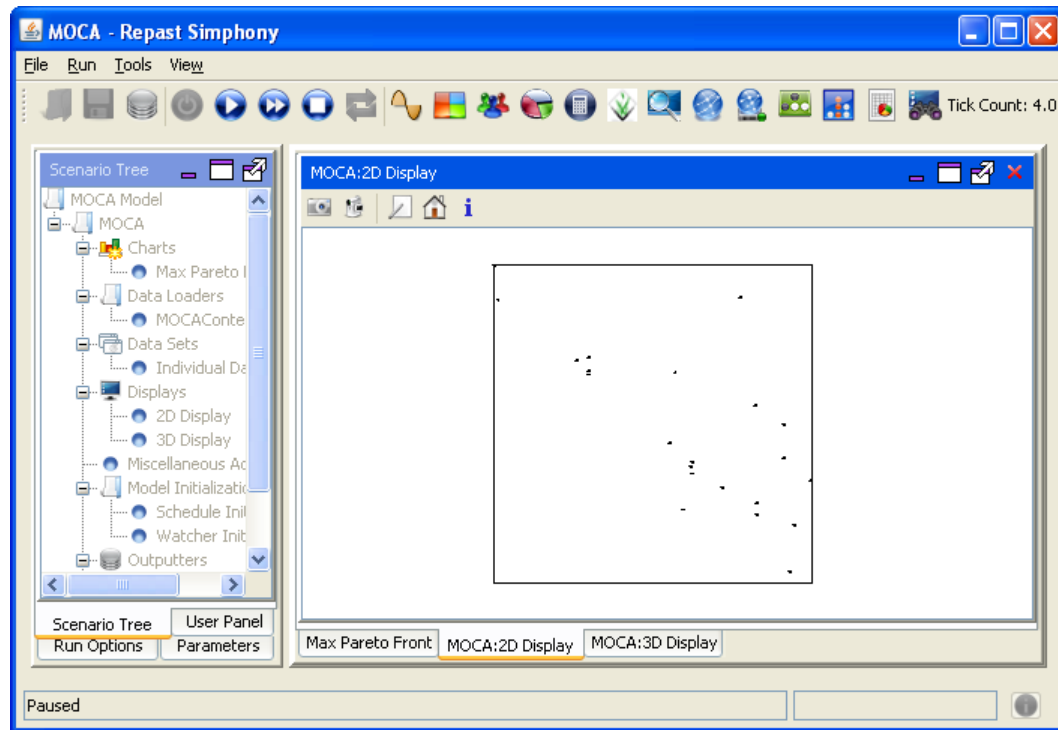


Figure 7.2 ZDT1 population at generation 4.

Figure 7.2 shows the population of solutions generated after just four generations. Notice that the system has produced outcomes primarily on a secondary front which exhibits a fairly linear patterns —patterns that are easy for EAs to pick up. There are a couple of points that appear to on or very near the optimal front. Those points and the knowledge sources that generated them will be attractors for the population in subsequent generations. As we will observe later, both normative and topographical knowledge have a marked influence in these early stages, while all five knowledge sources play some

role. Traditionally these two are known for their exploratory activities. Once they have been able to find attractive points, the other knowledge sources swarm to those areas and exploit them.

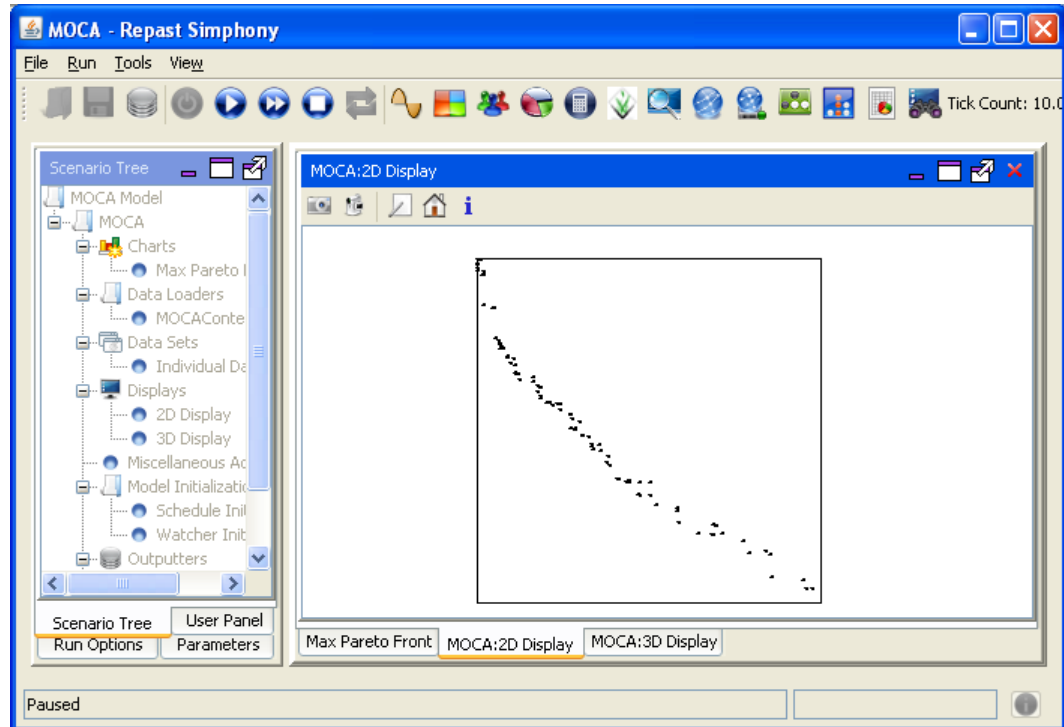


Figure 7.3 ZDT1 at generation 10

At generation 10, Figure 7.3, the population has produced a linear approximation of the optimal Pareto front. In other words, the Normative and Topographic knowledge sources that typically are exploratory in nature gradually make way for the Situational, Domain, and History knowledge sources that exploit the detail around the linear curve. This detail will produce nonlinearities seen in the final result. These relative roles have been observed in every version of the CAT system from Peng (2007) through Che (2009).

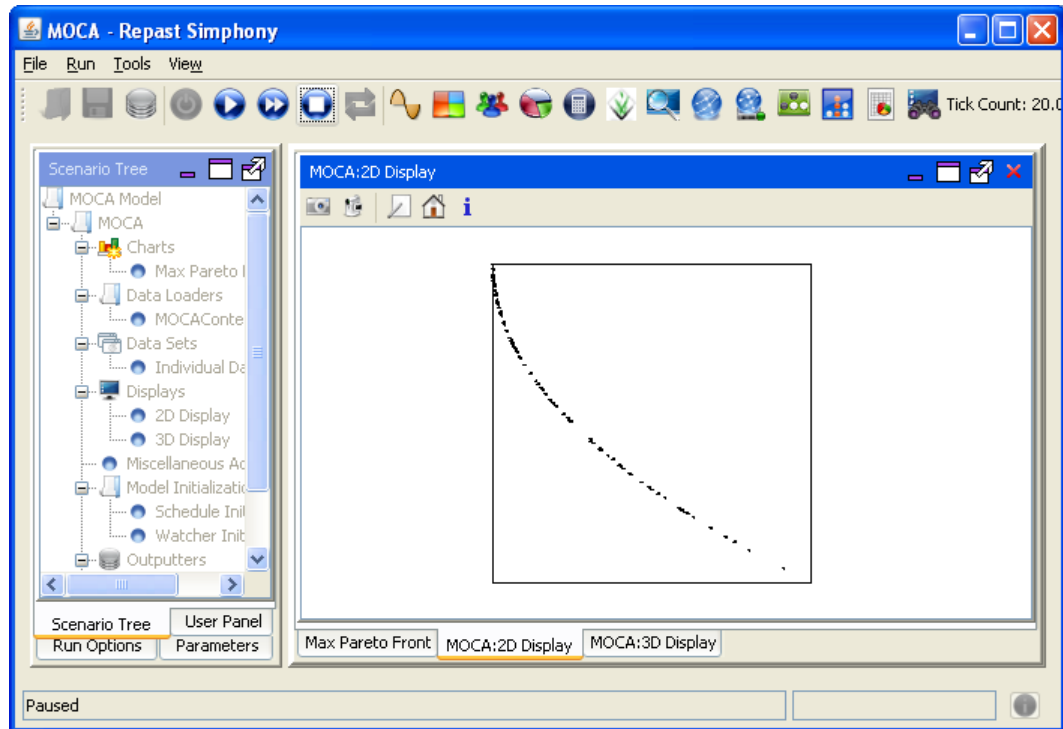


Figure 7.4 ZDT1 at generation 20

By generation 20, Figure 7.4, the population has produced a more convex approximation of the optimal Pareto front. What remains to be done is to fill in the gaps along the continuous curve. By this time the exploitative knowledge sources, situational, Domain and History, are dominating the search while the exploratory knowledge sources have moved on to search other less productive parts of the space. While this might seem inefficient, if something happens to cause a shift in the Pareto front then the exploratory knowledge sources will be in a good position to exploit them.

We can see that while each run left some gaps along the real Pareto front which is unavoidable because there are only limited number of individuals in the population (100) at each time step in our configuration. Still the gaps are not uniformly spread out which suggests that additional information may be useful in guiding the system. Note however, that when we combine the points produced over all 20 runs we get the continuous curve shown in Figure 7.5 below.

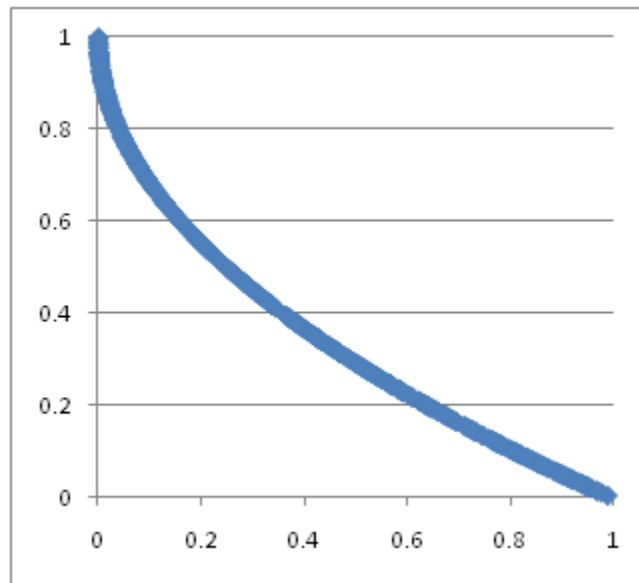


Figure 7.5 Overall found Pareto front

7.2.2 Performance of MOCAT

In table 7.1 we give the statistics for the 20 runs that were conducted. The errors were produced by comparing each point with the actual value on the known curve. The system was run for just 100 generations and not until it converged so the errors are non-zero overall.

Table 7-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending solutions					
	median	mean	min	max	stdev
Run #1	3.35E-05	8.76E-05	0.00E+00	6.31E-04	1.34E-04
Run #2	9.54E-06	2.59E-05	0.00E+00	2.36E-04	4.56E-05
Run #3	9.54E-06	2.59E-05	0.00E+00	2.36E-04	4.56E-05
Run #4	1.17E-05	1.39E-05	0.00E+00	7.28E-05	1.22E-05
Run #5	3.39E-05	4.30E-05	0.00E+00	1.53E-04	3.94E-05
Run #6	5.28E-05	9.12E-05	1.00E-09	5.63E-04	1.03E-04
Run #7	6.81E-05	6.86E-04	0.00E+00	3.69E-02	3.84E-03
Run #8	3.04E-05	4.02E-05	0.00E+00	1.69E-04	3.53E-05
Run #9	2.49E-05	3.69E-05	0.00E+00	2.39E-04	4.31E-05
Run #10	5.79E-06	7.52E-06	0.00E+00	2.74E-05	6.55E-06

Run #11	7.06E-05	1.24E-04	1.80E-08	7.51E-04	1.44E-04
Run #12	4.62E-05	5.16E-05	0.00E+00	1.83E-04	4.43E-05
Run #13	3.34E-05	3.39E-05	0.00E+00	1.37E-04	2.75E-05
Run #14	1.24E-05	2.38E-04	0.00E+00	1.40E-02	1.60E-03
Run #15	1.68E-04	2.40E-04	2.00E-09	1.11E-03	2.30E-04
Run #16	7.73E-06	8.66E-06	0.00E+00	2.76E-05	6.37E-06
Run #17	3.37E-05	2.67E-04	0.00E+00	2.02E-02	2.07E-03
Run #18	5.94E-05	1.33E-04	1.00E-09	2.78E-03	3.31E-04
Run #19	5.42E-06	7.72E-06	0.00E+00	4.20E-05	9.18E-06
Run #20	2.76E-04	3.37E-04	1.00E-09	1.69E-03	3.55E-04
mean	4.97E-05	1.25E-04	1.15E-09	4.01E-03	4.56E-04
stdev	6.33E-05	1.61E-04	3.90E-09	9.10E-03	9.43E-04

The performances are good; its median and standard deviation are both better than NSGA-II shown in 6.7 More importantly, we can see that the performance of MOCAT is very stable and reliable.

7.2.3 Statistics of Topologies using Spread metric

In Figure 7.2 examines how the five homogeneous topologies affected the search process here. It shows the number of times that each topology was used over the 100 generations used in each run. T-Tests were conducted and appended to the table that tested whether a topology was used more frequently for this convex problem than other topologies. The statistical significance of the tests is not sufficiently high to reject the null hypothesis that there is no difference in usage here.

Table 7-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	13	20	26	15	26	0.847027419
Run #2	23	12	30	17	18	1.111389115
Run #3	10	19	21	21	29	1.119421394
Run #4	21	16	36	16	11	0.978107524
Run #5	28	14	13	19	25	1.086135314
Run #6	14	19	23	21	23	6.666230388

Run #7	21	23	16	21	19	1.287356055
Run #8	16	20	18	14	32	1.098285051
Run #9	21	23	17	27	12	0.964309758
Run #10	22	26	17	16	19	1.168181793
Run #11	23	20	14	19	24	6.535200529
Run #12	22	26	23	17	12	8.853087069
Run #13	20	14	14	24	28	6.822412532
Run #14	11	19	27	14	29	1.251946181
Run #15	22	22	19	19	18	1.077883337
Run #16	29	25	14	18	14	4.076536355
Run #17	15	13	23	27	22	1.129818909
Run #18	12	22	17	28	21	8.931720072
Run #19	19	25	12	27	17	1.030404928
Run #20	36	14	13	11	26	1.211594209
mean	19.9	19.6	19.65	19.55	21.25	
stdev	6.512336	4.5002924	6.425893	4.925498	6.19741	
t-test	Lbest vs.	0.433213	0.451691	0.424538	0.252964	
	Square vs.		0.488714	0.48672	0.170998	
	Octal vs.			0.47813	0.213915	
	Hex vs.				0.171623	

7.2.4 Behaviors of Knowledge sources using the Spread Metric

In table 7-3 we see the number of individuals influenced by each knowledge source over time. Notice that each of the two exploratory knowledge sources, normative and topographic, are both relatively low in the number of agents controlled in comparison with the exploitative knowledge sources. The t-tests given in the table show that there are indeed statistically significant differences, at $\alpha = .05$, between the influence of the different knowledge sources. Clearly the system spends most of its time exploiting the early results of search over all runs.

Table 7-3 The number of Individuals influenced by KS using the Spread Metric

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	346	1453	1489	1075	187

Run #2	356	1395	1535	1159	168
Run #3	350	1464	1422	1167	172
Run #4	310	1381	1615	919	130
Run #5	335	1399	1523	1033	151
Run #6	324	1459	1503	1039	126
Run #7	317	1496	1537	805	81
Run #8	291	1434	1519	1003	122
Run #9	289	1402	1508	1051	160
Run #10	311	1400	1516	977	109
Run #11	344	1443	1548	1122	128
Run #12	292	1401	1557	1044	107
Run #13	279	1481	1415	1059	124
Run #14	330	1386	1526	1162	163
Run #15	317	1373	1617	872	128
Run #16	326	1415	1548	1008	171
Run #17	346	1475	1523	1120	168
Run #18	301	1428	1542	880	167
Run #19	343	1399	1532	1068	168
Run #20	282	1362	1456	1101	146
mean	319.45	1422.3	1521.55	1033.2	143.8
stdev	24.4	38.8	50.6	101.0	27.9
t-test	KS-N vs.	8.22E-43	2.05E-36	2.25E-19	1.01E-22
	KS-S vs.		1.96E-08	7.94E-15	4.64E-47
	KS-D vs.			4.99E-18	4.33E-40
	KS-H vs.				9.09E-22

7.2.5 Statistics of Topology-Knowledge tuple

In Table 7-4 the total number of non-dominated solutions that are generated by each knowledge source for a given topology on average over the 20 runs is given. The exploratory knowledge sources, Normative and Topographic, generate the fewest non-dominated individuals per run but they are critical to generating the first approximations that the three exploitative knowledge sources (history, situational and domain) exploit. Notice that the hexagonal topology produces the most non-dominated

solutions over all of the topologies on average. In fact the most complex topologies, hex and global, generate the most non-dominated solutions.

In Table 7-5 the ratio of the standard deviation to the mean is given. This ratio gives an estimate of the variability in production of non-dominated solutions. Notice that the topologies that produce the most non-dominated solutions are the most consistent. On the other hand, LBEST, which is the simplest topology, is the most variable in the generation of non-dominated solutions from run to run. This suggests that it is used in a more opportunistic sense than the other topologies here..

Table 7-4 The total number of non-dominated solutions generated over all 20 runs for each KS Topology pairing.

		N	S	D	H	T
Lbest	mean	59.8	267.55	272.45	198.2	27.65
	stdev	36.77327	127.2964	130.2578	95.18326	19.75981
Square	mean	65.95	290.25	305.75	216	29.3
	stdev	33.80202	86.61401	98.94968	67.4412	12.86816
Octal	mean	67.45	284.95	291.8	217.95	32.45
	stdev	24.7439	100.2625	101.2753	80.25058	15.54442
Hex	mean	66.4	303.9	319.1	229.1	30.9
	stdev	23.27954	85.46769	85.82044	57.93363	12.0739
Global	mean	65.9	286.3	302.9	210.05	33.15
	stdev	23.38218	102.2999	100.3326	78.35377	18.28222

Table 7-5 Variability of Topology-Knowledge tuple Production.

stdev/mean	N	S	D	H	T
Lbest	0.614938	0.475785	0.478098	0.480238	0.71464
Square	0.51254	0.298412	0.323629	0.312228	0.439186
Octal	0.366848	0.35186	0.347071	0.368206	0.479027
Hex	0.350595	0.281236	0.268945	0.252875	0.390741
Global	0.354813	0.357317	0.33124	0.373024	0.5515

7.3 Using Hyper Volume to Guide MOCAT for ZDT1 in 30 dimensions

In this section we use the other spread metric, hyper-volume, to guide the search. The overall Pareto front produced is similar to that for the spread metric and will not be given here. Each run was completed in an average of 34.42 seconds.

7.3.1 Performance of MOCAT

The statistics shown in Table 7-6 give the overall error produced using the hyper-volume for the 20 runs. In general, the mean error and standard deviation is slightly higher than that for the spread metric. This suggests that it is focused on a larger portion of the search space than for the spread metric on average.

Table 7-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of endingsolutions					
	median	Mean	Min	max	stdev
Run #1	3.79E-05	5.41E-05	1.00E-09	2.89E-04	5.50E-05
Run #2	3.74E-05	5.12E-05	0.00E+00	3.24E-04	5.23E-05
Run #3	2.46E-04	8.07E-04	2.00E-09	4.80E-02	4.93E-03
Run #4	4.60E-05	5.23E-05	0.00E+00	1.80E-04	4.17E-05
Run #5	1.43E-04	1.98E-04	0.00E+00	8.21E-04	1.74E-04
Run #6	2.23E-05	1.97E-04	0.00E+00	3.55E-03	5.92E-04
Run #7	9.99E-05	1.30E-04	0.00E+00	5.30E-04	1.15E-04
Run #8	2.29E-05	3.62E-05	0.00E+00	2.05E-04	4.12E-05
Run #9	1.25E-05	1.74E-05	0.00E+00	6.14E-05	1.66E-05
Run #10	1.70E-04	1.93E-04	0.00E+00	5.97E-04	1.25E-04
Run #11	1.97E-05	2.50E-05	1.00E-09	1.04E-04	2.02E-05
Run #12	1.37E-05	2.19E-05	0.00E+00	8.62E-05	2.20E-05
Run #13	1.11E-05	1.45E-05	0.00E+00	6.07E-05	1.37E-05
Run #14	1.23E-04	1.88E-04	1.00E-09	9.72E-04	2.23E-04
Run #15	2.81E-05	7.41E-05	0.00E+00	9.55E-04	1.50E-04
Run #16	2.33E-05	2.58E-05	0.00E+00	1.09E-04	2.07E-05
Run #17	5.56E-05	6.09E-05	0.00E+00	2.35E-04	5.60E-05

Run #18	2.57E-05	2.99E-05	0.00E+00	1.96E-04	2.73E-05
Run #19	8.19E-05	4.10E-04	0.00E+00	4.65E-03	8.00E-04
Run #20	7.18E-05	8.54E-05	0.00E+00	3.01E-04	6.69E-05
mean	6.46E-05	1.34E-04	2.50E-10	3.11E-03	3.77E-04
stdev	6.15E-05	1.81E-04	5.36E-10	1.04E-02	1.06E-03

7.3.2 Statistics of Topologies using Hyper-volume metric

In table 7-7 the number of times that each topology was selected during a given run is presented. The t-test given in the table suggests that as with the spread metric there is no evidence for a difference in the use of topologies to direct the search process for the convex problem here.

Table 7-7 Use Count of Topologies of each run

	Using Hyper-volume metric					Hyper-volume Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	28	17	14	19	22	0.737209538
Run #2	22	19	22	18	19	0.680990974
Run #3	16	23	17	19	25	0.532614128
Run #4	22	19	16	26	17	0.555647676
Run #5	18	30	19	23	10	0.626039696
Run #6	19	18	20	21	22	0.65271006
Run #7	17	16	27	17	23	0.694732545
Run #8	24	20	20	16	20	0.621438754
Run #9	17	28	16	23	16	0.682875636
Run #10	25	13	25	14	23	0.505261666
Run #11	22	18	14	26	20	0.574701695
Run #12	22	25	18	13	22	0.493580557
Run #13	15	20	24	17	24	0.582580069
Run #14	17	25	17	22	19	1.20564987
Run #15	19	22	18	20	21	0.725434879
Run #16	20	12	16	27	25	0.701219191
Run #17	25	18	25	23	9	0.675484116
Run #18	20	14	27	18	21	0.634895268
Run #19	21	19	20	19	21	0.938095568
Run #20	24	22	20	16	18	0.582703204
mean	20.65	19.9	19.75	19.85	19.85	

stdev	3.468277	4.700504	4.063639	3.963983	4.295346
t-test	Lbest vs.	0.284761	0.227989	0.250581	0.260512
	Square vs.		0.457303	0.485593	0.486086
	Octal vs.			0.468811	0.470055
	Hex vs.				0.5

7.3.3 Behaviors of Knowledge sources using Hyper-volume Metric

In table 7-8 we see the extent to which each of the knowledge sources can influence the individuals in the population. As with the spread metric the exploratory knowledge sources are used less frequently than the exploitative knowledge sources. The t-tests in the table indicate a significance difference in the influence of the knowledge sources over the courses of the runs. This result is again similar to that for the spread metric.

Table 7-8 Using Hyper-volume metric #Individuals influenced by KS

Using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	366	1416	1507	993	188
Run #2	331	1444	1518	907	176
Run #3	333	1432	1518	1146	155
Run #4	319	1454	1443	1179	144
Run #5	347	1425	1553	976	181
Run #6	332	1468	1541	929	142
Run #7	315	1471	1465	1181	169
Run #8	320	1451	1450	1059	155
Run #9	299	1389	1492	1026	133
Run #10	311	1406	1430	1231	127
Run #11	337	1464	1477	1127	180
Run #12	291	1475	1479	879	118
Run #13	318	1452	1513	963	142
Run #14	332	1456	1501	898	126
Run #15	284	1432	1492	928	90
Run #16	319	1417	1540	899	117
Run #17	332	1434	1509	1058	149
Run #18	276	1418	1548	934	128

Run #19	320	1464	1549	1093	175
Run #20	353	1439	1455	1068	183
mean	321.75	1440.35	1499	1023.7	148.9
stdev	22.4	23.6	37.5	109.0	27.0
t-test	KS-N vs.	6.84E-55	2.7E-43	3.04E-18	4.4E-23
	KS-S vs.		6.85E-07	8.01E-14	5.53E-55
	KS-D vs.			9.35E-16	2.05E-48
	KS-H vs.				1.38E-20

7.3.4 Generation of non-dominated individuals by Knowledge Topology Tuple Using the Hyper-Cube Performance Function

Table 7-9 gives the number of non-dominated solutions produced by each topology-knowledge source pair. Notice that the global topology was the most effective at generating non-dominated solutions in each of the knowledge sources for this problem. In terms of the knowledge sources, the exploitative knowledge sources dominated the generation of non-dominated solutions, especially the domain knowledge source. The latter is influential in driving the hill climbing activity of the Cultural Algorithms. These results are similar to the spread metric version.

Table 7-10 examines the how each knowledge source topology pair varies in its production of non-dominated solutions over the ten runs. The statistic is produced by taking the ratio of the standard deviation for the runs and dividing by the mean of the runs. The values are taken from Table 7-9. The larger the ratio the more diversity there is in the generation of the non-dominated solutions for each pair. It is clear that the exploitative knowledge sources (history, domain, and situational) are more consistent generators of non-dominated solutions across all runs. On the other hand, the exploitative knowledge sources show more variability in the generation of the non-dominated solutions over the set of runs. This indicates that more exploration may be needed in one run versus another based upon how the search process begins.

Table 7-9 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	66.3	296.65	307.8	204.05	34.3
	stdev	21.53357	70.33998	64.90852	50.27448	13.7462
Square	mean	61.65	268.15	281.6	191.5	25.85
	stdev	22.47168	57.56852	63.87109	41.99311	9.560197
Octal	mean	58.7	265.7	281.7	200.65	29.25
	stdev	19.855	70.57217	75.61962	59.27037	11.76021
Hex	mean	64.2	279.25	294.45	198.15	28.4
	stdev	18.70997	53.16298	58.6304	46.66485	12.62579
Global	mean	71.15	318.15	337.85	223.65	34.6
	stdev	21.0595	69.41278	72.35459	52.80578	14.8054

Table 7-10 Variability in the Production of Non-Dominated Solutions by Topology-Knowledge tuple as given by the ratio of the standard deviation to the mean for each knowledge topology pair.

stdev/mean	N	S	D	H	T
Lbest	0.32479	0.237114	0.210879	0.246383	0.400764
Square	0.364504	0.214688	0.226815	0.219285	0.369834
Octal	0.338245	0.265608	0.26844	0.295392	0.402059
Hex	0.291433	0.190378	0.199118	0.235503	0.44457
Global	0.295987	0.218176	0.214162	0.236109	0.427902

7.4 Solving ZDT1 with dimension 30 using the combined metrics

In these 20 runs we combine the two metrics and observe their performance in guiding the MOCAT search here. Each run was complete in an average of 36.18 seconds. Thus, the combined metric version takes about two seconds more on average to generate a solution than the hyper-cube volume metric controlled Cultural Algorithm. Recall that in the combined approach the metric took turns guiding the search process, switching every 25 generations.

7.4.1 Performance of MOCAT with the Combined Metrics

The performance statistics for the MOCAT system guided by the combined metrics are given in Table 7-11. Overall the combined system produces results that are similar to the two independent metrics above. Of the three metric combinations used the spread metric by itself had the smallest mean overall error but the largest standard deviation. The hyper-volume metric on its own produced a slightly increased mean error term but reduced the standard deviation from that of the spread metric. However, the overall mean error for the combined metrics is the highest of the three configurations, but it exhibits the lowest standard deviation. This suggests that the combined metric controlled solution is more focused than either of the two metric controlled Cultural Algorithms on their own.

Table 7-11 Statistics for the fitness errors of ending solutions

Statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	3.39E-05	4.30E-05	0.00E+00	1.53E-04	3.94E-05
Run #2	1.86E-04	4.09E-04	0.00E+00	5.98E-03	7.71E-04
Run #3	2.16E-04	5.53E-04	0.00E+00	4.30E-03	7.43E-04
Run #4	1.61E-05	2.51E-05	1.00E-09	1.05E-04	2.56E-05
Run #5	5.25E-05	9.02E-05	0.00E+00	8.92E-04	1.24E-04
Run #6	1.42E-04	1.85E-04	1.10E-08	8.40E-04	1.67E-04
Run #7	9.99E-06	3.31E-05	0.00E+00	1.72E-03	1.75E-04
Run #8	2.06E-06	2.52E-06	3.00E-09	2.30E-05	3.31E-06
Run #9	9.75E-05	1.33E-04	9.10E-08	1.03E-03	1.54E-04
Run #10	4.84E-05	6.06E-05	1.00E-09	3.92E-04	6.30E-05
Run #11	2.97E-04	4.16E-04	3.00E-09	4.35E-03	5.45E-04
Run #12	1.15E-05	1.73E-05	1.00E-09	1.01E-04	1.76E-05
Run #13	9.24E-05	1.36E-04	4.00E-09	9.72E-04	1.48E-04
Run #14	3.83E-05	4.33E-05	9.00E-09	2.00E-04	3.70E-05
Run #15	9.65E-05	1.13E-04	0.00E+00	3.67E-04	8.48E-05
Run #16	2.73E-05	2.90E-04	0.00E+00	2.04E-03	4.60E-04
Run #17	2.56E-05	2.47E-05	0.00E+00	9.31E-05	1.83E-05
Run #18	1.15E-05	9.74E-05	4.30E-08	5.12E-03	5.58E-04

Run #19	3.32E-05	4.58E-05	0.00E+00	1.50E-04	3.89E-05
Run #20	2.29E-05	3.33E-04	0.00E+00	3.99E-03	6.85E-04
mean	7.30E-05	1.53E-04	8.35E-09	1.64E-03	2.43E-04
stdev	7.79E-05	1.56E-04	2.12E-08	1.90E-03	2.64E-04

7.4.2 Statistics of Topologies for both metrics

Given that the control was switched every 25 generations, each metric was observed for 50 generations in each run. Tables 7-12 and 7-13 give the topology usage for each of the two metric components respectively over the 50 generations that they were used.

Table 7-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	9	18	10	9	4	1.030436212
Run #2	9	10	11	12	8	1.317797209
Run #3	8	10	9	8	15	1.02599956
Run #4	7	6	8	11	18	1.038701676
Run #5	6	18	6	14	6	7.19100245
Run #6	7	10	14	11	8	1.029884084
Run #7	7	3	15	11	14	1.068474789
Run #8	10	22	6	2	10	0.964449467
Run #9	13	10	8	6	13	4.142557897
Run #10	4	9	16	12	9	1.318180782
Run #11	13	7	13	4	13	1.038507953
Run #12	10	10	7	9	14	7.978955659
Run #13	8	12	14	10	6	1.17575409
Run #14	13	6	15	4	12	1.600382192
Run #15	4	6	9	21	10	1.115927844
Run #16	12	12	7	7	12	4.758465232
Run #17	8	4	12	20	6	1.178250946
Run #18	8	12	7	10	13	5.903779215
Run #19	7	11	9	12	11	6.744121779
Run #20	8	9	14	13	6	1.034601066
mean	8.666667	10.277778	10.38889	10.05556	10.61111	
stdev	2.786522	5.0035935	3.415172	4.940496	3.775134	

t-test	Lbest vs.	0.086345	0.024465	0.080109	0.039653
	Square vs.		0.424235	0.486784	0.45601
	Octal vs.			0.439177	0.464632
	Hex vs.				0.470606

For the spread metric in Table7-12 there are t-tests showing whether there are statistical differences between the pair of use of topologies when. Here we can see that topology Lbest has obviously lower values of usage than the others for the spread metric. In fact, it is significantly lower than Octal and Global topologies in use. This suggests that Lbest was not as effective in distributing knowledge as the other more connected topologies with the spread metric in the combined case.

Table 7-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric					Hyper-volume metric value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	10	7	14	11	8	0.487763434
Run #2	10	1	9	14	16	0.554713793
Run #3	8	17	6	10	9	0.580528462
Run #4	13	9	13	9	6	0.598461602
Run #5	7	8	13	12	10	0.600307753
Run #6	14	8	9	13	6	0.617521337
Run #7	11	12	9	8	10	0.581744843
Run #8	9	8	11	14	8	0.664396647
Run #9	13	11	12	5	9	0.710420946
Run #10	7	9	10	12	12	0.556012493
Run #11	5	9	10	13	13	0.515702789
Run #12	8	11	15	7	9	0.913295602
Run #13	11	9	12	11	7	0.853775975
Run #14	7	14	13	7	9	0.531787556
Run #15	10	9	12	11	8	0.716315639
Run #16	11	7	12	8	12	0.672850736
Run #17	15	8	5	11	11	0.659040005
Run #18	9	9	11	10	11	0.663649704
Run #19	12	7	7	8	16	0.669755857
Run #20	7	10	11	9	13	0.520502392
mean	9.85	9.15	10.7	10.15	10.15	
stdev	2.680828623	3.133435982	2.617753155	2.49789385	2.870448342	

t-test	Lbest vs.	0.226282	0.158378	0.358147	0.367275
	Square vs.		0.048998	0.135884	0.149656
	Octal vs.			0.250384	0.265232
	Hex vs.				0.5

As shown in Table 7-13 the Octal topology is used most frequently on average under the control of the hyper-cube metric, just as with the spread metric. It is used significantly more often than the LBEST topology, similar to the spread metric. While both metrics did not exhibit statistical difference in their usage of topologies on their own, when combined they make similar usage of the topologies. Octal is used the most in both, and its usage is significantly more than that of LBEST for both the Spread metric and Hyper-Volume phases of the combined approach.

7.4.3 Behaviors of Knowledge sources with the two metric phases.

Tables 7-14 and 7-15 give the relative use of the knowledge sources by both the spread metric and the hyper-volume phases of the combined system respectively. As was exhibited by each metric separately, the most influential knowledge source was the domain knowledge source which controlled the hill climbing activities. It was used significantly more often than the other statistics in both phases of the problem solving process. Also, we can see that knowledge sources N and T, the exploratory knowledge sources, were chosen significantly less frequently than other knowledge sources.

Table 7-14 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	209	692	746	541	109
Run #2	216	726	766	490	124
Run #3	264	682	761	450	91
Run #4	229	723	743	438	76

Run #5	219	742	788	422	100
Run #6	220	702	766	494	109
Run #7	194	730	785	380	71
Run #8	205	735	787	515	124
Run #9	229	719	798	443	99
Run #10	215	717	747	584	94
Run #11	190	727	748	554	81
Run #12	182	759	709	467	76
Run #13	191	702	773	499	48
Run #14	234	697	808	439	79
Run #15	185	722	754	548	102
Run #16	215	708	802	380	67
Run #17	215	745	747	471	90
Run #18	208	722	754	420	54
Run #19	201	685	794	424	104
Run #20	230	696	748	487	68
mean	212.55	716.55	766.2	472.3	88.3
stdev	19.6	20.7	25.2	56.9	21.2
t-test	KS-N vs.	6.1E-44	8.04E-42	3.33E-16	2.07E-21
	KS-S vs.		2.73E-08	9.37E-16	5.07E-47
	KS-D vs.			2.81E-18	1.57E-45
	KS-H vs.				2.38E-20

Table 7-15 Using Hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	145	711	781	603	95
Run #2	119	723	734	546	70
Run #3	120	714	759	559	64
Run #4	106	747	721	533	49
Run #5	124	696	761	566	52
Run #6	136	703	755	588	68
Run #7	146	742	739	437	47
Run #8	122	731	743	580	85
Run #9	106	678	750	505	56
Run #10	116	665	726	629	80
Run #11	117	708	730	638	70
Run #12	113	744	730	524	63

Run #13	127	709	744	509	60
Run #14	123	727	736	600	53
Run #15	97	702	724	535	50
Run #16	134	685	772	580	69
Run #17	120	731	696	584	59
Run #18	107	721	754	516	40
Run #19	131	773	734	497	48
Run #20	105	715	713	553	60
mean	120.7	716.25	740.1	554.1	61.9
stdev	13.2	25.4	20.3	48.9	13.8
t-test	KS-N vs.	2.12E-37	1.9E-44	8.99E-22	1.27E-16
	KS-S vs.		0.001138	5.85E-14	3.47E-39
	KS-D vs.			6.84E-15	2.06E-46
	KS-H vs.				4.13E-23

7.4.4 The Generation of non-dominated solutions by Topology-Knowledge tuples.

Tables 7-18 and 7-19 give the productivity of each topology-knowledge source combination over all of the runs in terms of non-dominated solutions. Whereas the most productive pairings were octal-Domain and Hex-Domain for the spread metric and Global-Domain for the hyper-volume metric, the combined metric system employs the hex-domain combination to produce the most non-dominated solutions. This appears to be a blending of influences for the two different metric phases.

Also, in Table 7-16 we note that now the system is using the exploitative knowledge sources in a more focused way, with less variation. This is the reason for a reduction in the overall variability in performance over the 20 runs.

Table 7-16 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	60.2	283.85	303.75	211.6	26.1
	stdev	22.7286	55.47429	68.24482	48.89559	7.05542
Square	mean	58.75	280.15	297.65	201.95	27.6
	stdev	19.24325	54.02658	58.86941	39.5494	7.111148

Octal	mean	64.3	287.15	305.95	209.4	30.65
	stdev	20.68842	49.95501	51.15968	45.22971	12.68764
Hex	mean	74	298.5	314.5	216.9	31.95
	stdev	25.21487	72.89249	76.26236	53.4956	13.2207
Global	mean	61.25	274.6	295.25	203.8	28.1
	stdev	27.44924	84.78418	90.13375	58.35337	12.61119

Table 7-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.377552	0.195435	0.224674	0.231076	0.270323
Square	0.327545	0.192849	0.197781	0.195838	0.25765
Octal	0.321748	0.173968	0.167216	0.215997	0.413953
Hex	0.340741	0.244196	0.242488	0.246637	0.413793
Global	0.448151	0.308755	0.305279	0.286327	0.448797

7.4.5 Analysis of found Pareto front

The Optimal Pareto front produced by the combination of the results from the 20 runs is given below. The coverage of the optimal front is quite complete.

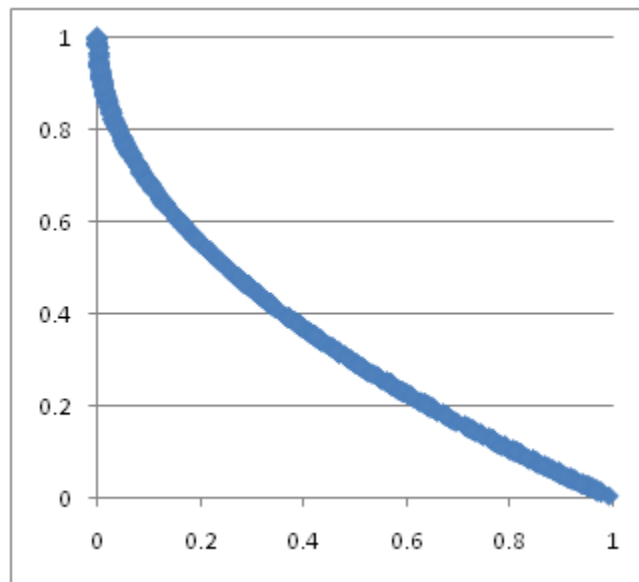


Figure 7.6 Overall found Pareto front

7.5 Summary of ZDT1

In summary, based upon the 20 runs the MOCAT system is able to perform better than other existing systems such as NSGA_II on the convex problem .It's performance is affected by the spread metrics employed. Some conclusions that can be drawn from our experiments are:

1. MOCAT solves the problem by first generating a linear approximation to the curve via exploration and then pushes out on the line using exploitation.
2. Exploratory knowledge sources are used less frequently than exploitation knowledge sources to generate non-dominated solutions.
3. For 30 dimensions there is no statistically significance preference for one topology over another for either of the metrics individually or combined. The combined use of the metrics produces a system in 30 dimensions with a slightly increased performance error but decreased standard deviation. This suggests a synergy between the two metrics since the spread metric is good at supporting search and the hyper-volume metric good at exploitation.
4. There is also an apparent blending of behaviors between the two approaches when they are used together. They focus in on a shared topology=knowledge source that is different that is slightly different from the ones that were favored by each alone.
5. The complex topologies, hex and global, generate more non-dominated solutions here with the simpler topologies generating fewer.
6. The exploratory generate fewer non-dominated solutions than the exploitative ones but their contribution is more volatile reflecting the need to jump start the search process in certain situations.

7. The combined metric system produced more focused search which means that there is less variability in the productivity of the knowledge-source-topology pairs.
8. While this problem is relatively straightforward and can be handled by either metric on its own with an emphasis on hill-climbing, the combined system does provide a more focused and blended environment for problem solving. This will become more important as the problems to be solved become more difficult.

CHAPTER 8 THE PERFORMANCE OF MOCAT ON PROBLEMS WITH CONCAVE PARETO FRONTS: DTZ2

At the first glance, ZDT2 is very similar to ZDT1, especially in terms of the definition of the problems. However, ZDT1 is a concave problem whereas ZDT2 is a convex. It will be interesting to observe how the spread metrics perform in this case. Recall that the system first developed a linear approximation and then proceeded to shift points to produce a curve. Here the direction of the curve, and therefore movement of points will be in the opposite direction. The ability of MOCAT to move individuals in both directions will be an indicator of its versatility. Thereafter, in this section we will observe whether MOCAT can still perform well in presence of disturbance.

We will use homogeneous spread metric, homogeneous hyper-volume metric, and combined two metrics to conduct experiments and try to observe and summarize the correspondence among knowledge sources and topologies.

8.1 Using MOCAT To Find Optimal Pareto Front for Concave Problems: ZDT 2 with 30 dimensions

ZDT2 is a continuous function with an optimal Pareto Front that is concave. The mathematical description of the problem is shown below and the optimal front is shown in Figure 8.1

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g}\right)^2$$

$$f_2 = g * h$$

where $m=30$ and $x_i \in (0,1]$. The Pareto-optimal front is formed with $g(x) = 1$.

The Pareto front is:

$$x_2 = 1 - x_1^2$$

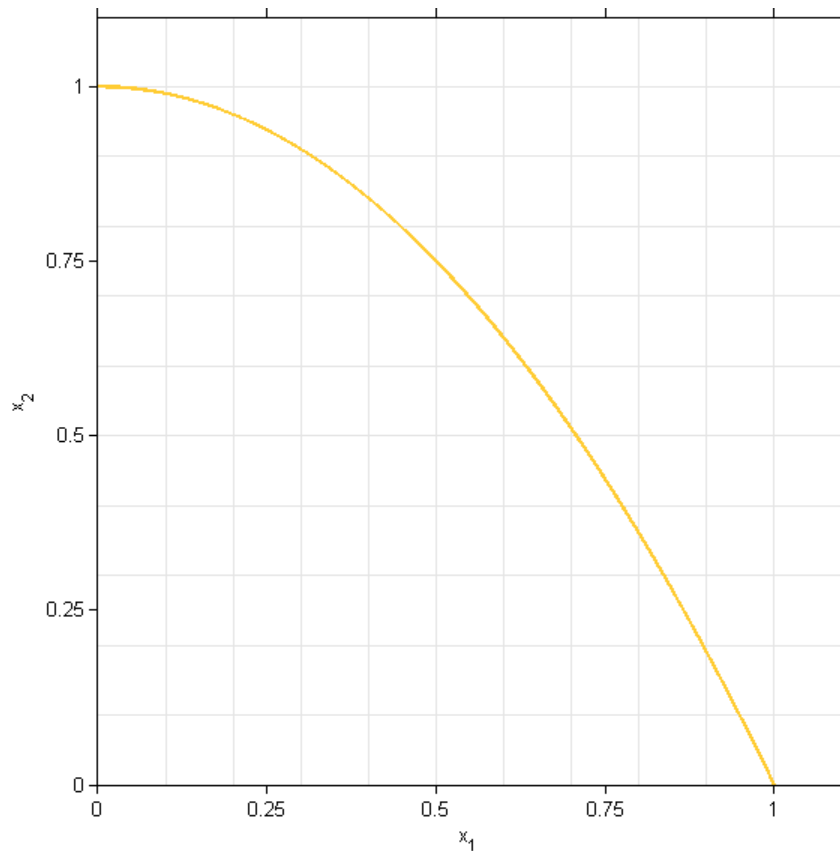


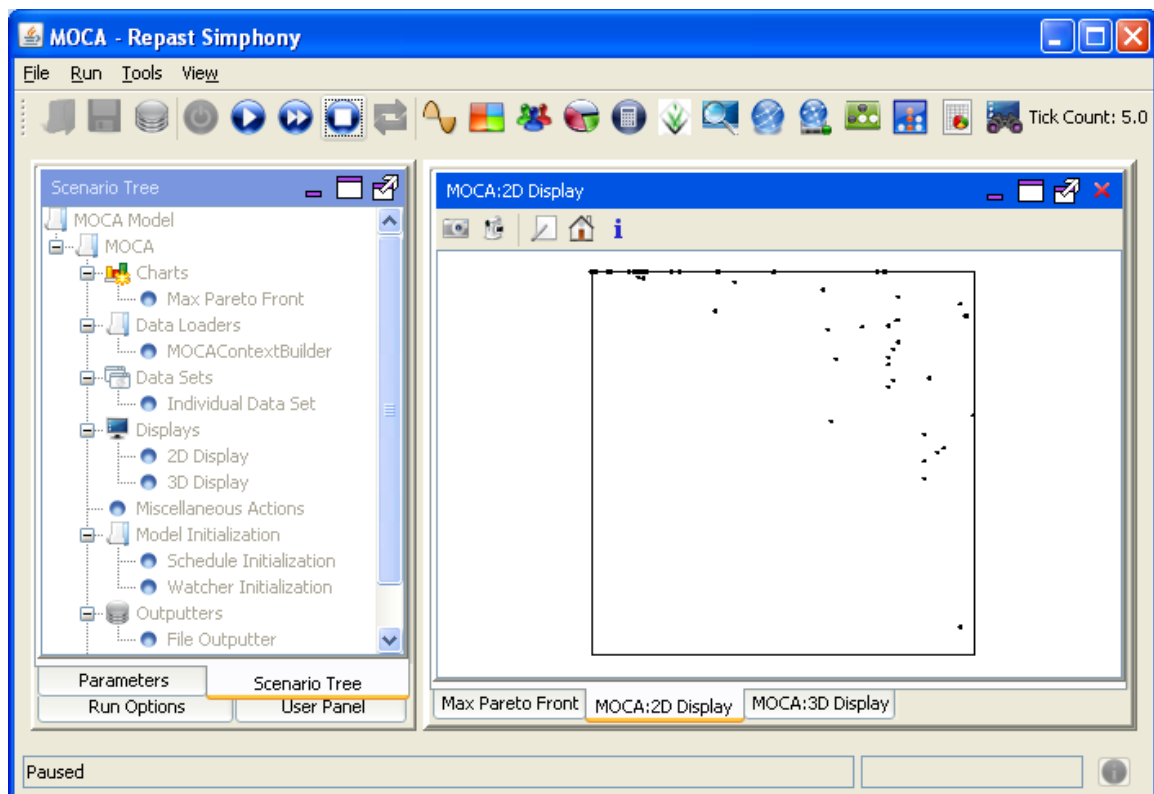
Figure 8.1 ZDT2 Pareto front

ZDT2 has a continuous Pareto front but it is convex; which may be a challenge to some MOEs (Deb, 2001) because they tried to reduce the area encircled by the found Pareto front and X and Y axes.

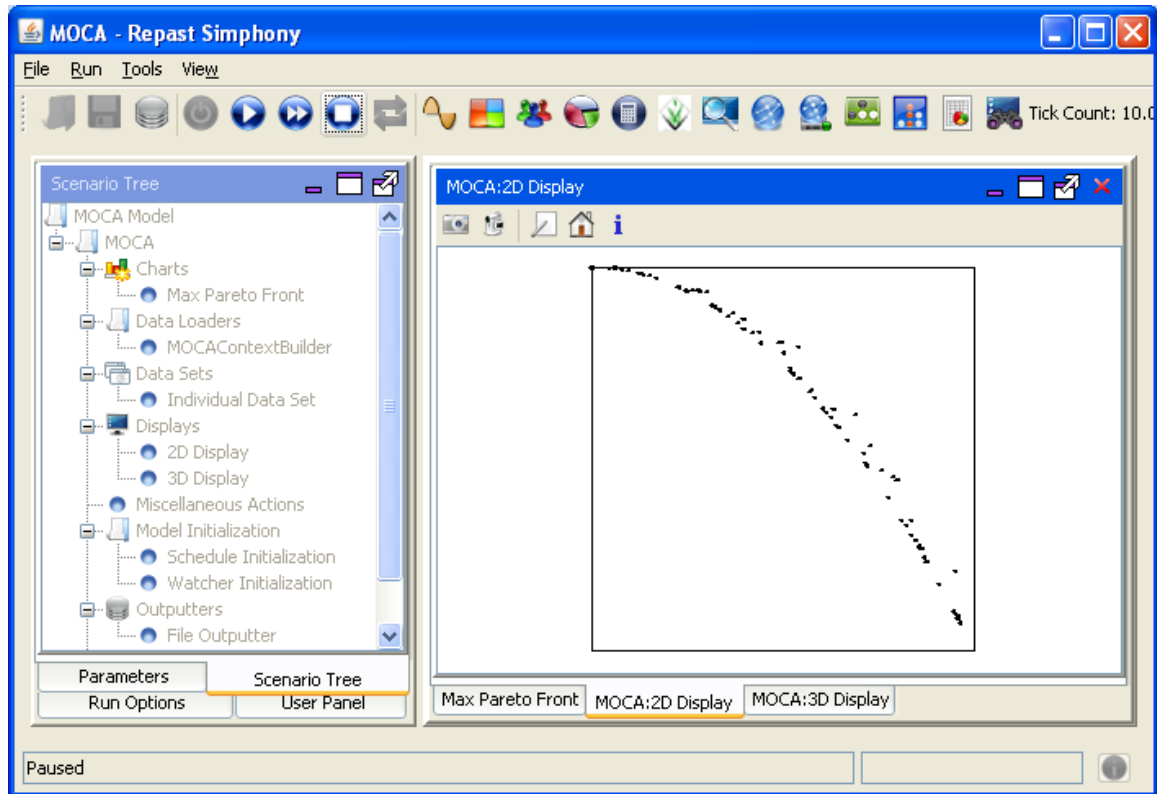
The runs were complete in an average of 33.78 seconds for 100 generation and 100 individuals in each population.

8.1.1 Performance of MOCAT Guided by the Spread metric

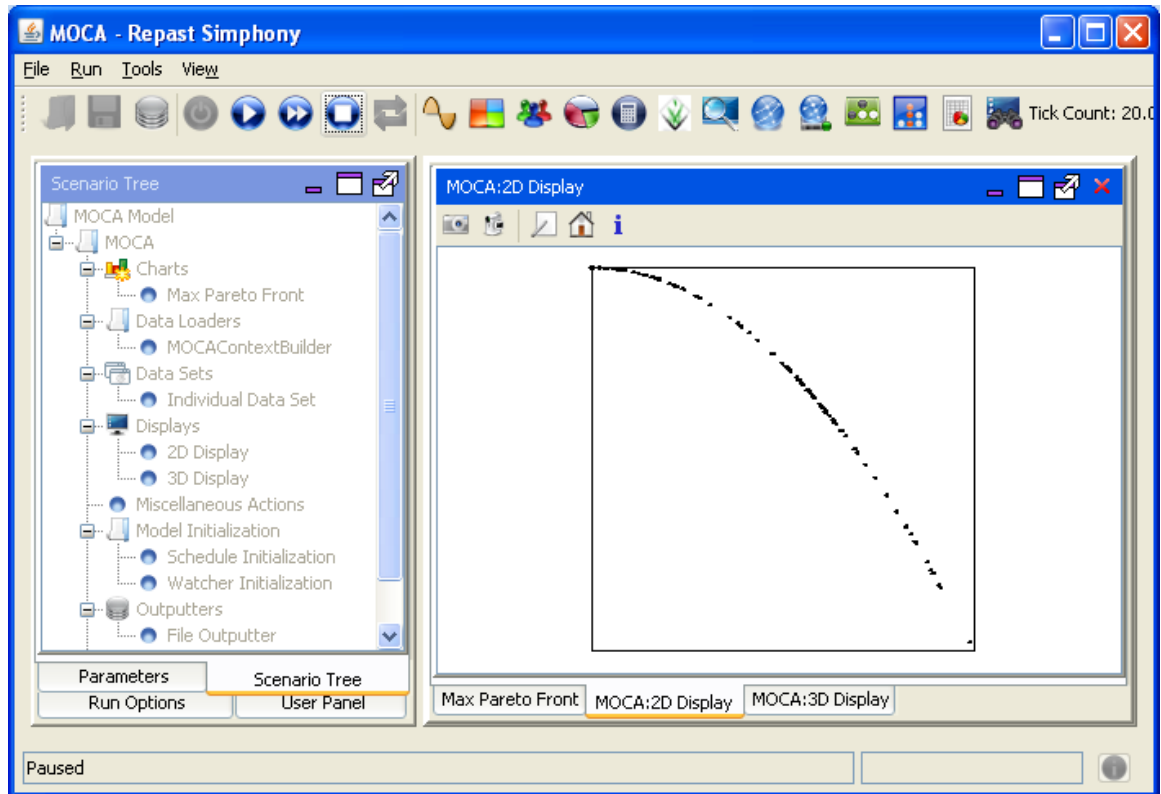
In Figure 8.2 (a-d) the evolution of the Pareto front over the 100 generations of an example run is given. Notice that the optimal front is represented by a small but relatively linear sequence of points that serve as attractors for the knowledge sources that exploit that pattern to create a more convex curve. Since we use only 100 individuals and run for only 100 generations there will undoubtedly be gaps in the curve for any run. At the end of the section, the individual points generated over each of the 20 runs are merged to produce a more refined curve.



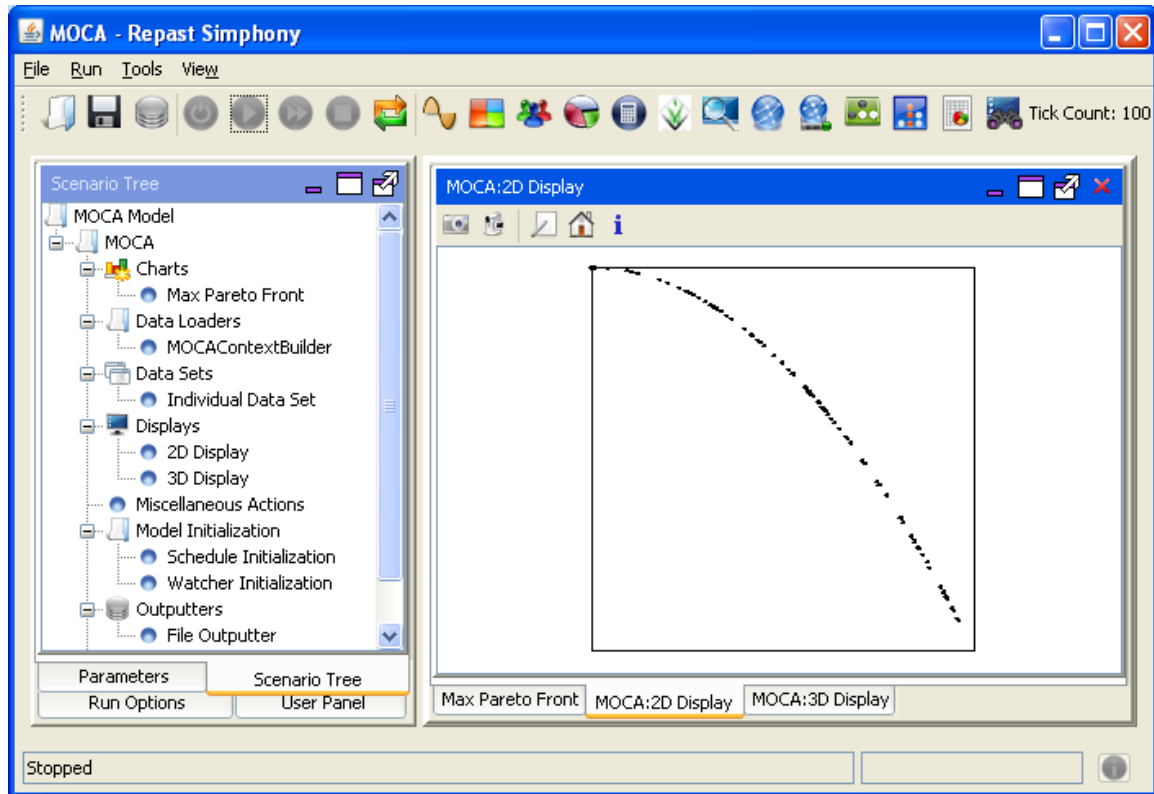
(a) Generation 5.



Generation 10



(c)Generation 20



(d)Generation 100

Figure 8.2 (a-d). A series of screen shots showing how the Pareto front is constructed over time.

Table 8-1 gives the performance for the MOCAT system using the spread metric performance function. What is interesting is that while the mean error is higher than that for the convex problem ($3.63\text{E-}4$ vs. $1.25\text{E-}04$) the standard deviation is now lower ($4.56\text{E-}03$ vs. $1.15\text{E-}03$). These results outperform all of the other benchmarked systems given in table 6.2.

Table 8-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	$1.85\text{E-}05$	$5.61\text{E-}05$	$0.00\text{E+}00$	$5.06\text{E-}04$	$6.80\text{E-}05$
Run #2	$4.01\text{E-}05$	$2.21\text{E-}04$	$-5.55\text{E-}17$	$1.39\text{E-}03$	$3.00\text{E-}04$

Run #3	4.80E-07	1.80E-05	4.07E-11	6.86E-04	9.62E-05
Run #4	2.95E-06	4.47E-04	9.96E-12	4.12E-02	4.20E-03
Run #5	2.29E-05	1.04E-04	-5.55E-17	6.86E-04	1.36E-04
Run #6	1.94E-04	6.35E-04	-5.55E-17	3.07E-03	6.40E-04
Run #7	1.46E-04	1.04E-03	3.55E-12	3.68E-02	3.76E-03
Run #8	3.10E-06	1.67E-05	4.97E-10	7.60E-05	2.09E-05
Run #9	1.70E-05	9.77E-05	1.20E-10	8.74E-04	1.36E-04
Run #10	6.54E-04	7.80E-04	1.08E-09	2.85E-03	6.42E-04
Run #11	2.41E-06	5.60E-06	-3.28E-17	1.90E-05	4.48E-06
Run #12	1.01E-04	3.11E-04	-5.55E-17	1.32E-03	3.04E-04
Run #13	1.31E-04	1.75E-04	3.46E-12	7.00E-04	1.69E-04
Run #14	2.71E-05	1.27E-04	8.98E-09	9.91E-04	1.67E-04
Run #15	3.20E-04	1.18E-03	3.51E-13	7.98E-03	1.39E-03
Run #16	8.50E-05	4.87E-04	8.89E-11	2.50E-03	6.02E-04
Run #17	3.27E-06	4.94E-05	-3.64E-17	3.00E-03	3.05E-04
Run #18	6.48E-06	1.41E-05	3.83E-13	4.94E-05	1.06E-05
Run #19	1.05E-06	7.31E-06	-2.78E-17	1.65E-04	2.33E-05
Run #20	4.91E-06	1.49E-03	2.47E-11	7.70E-02	1.00E-02
mean	8.90E-05	3.63E-04	5.43E-10	9.09E-03	1.15E-03
stdev	1.53E-04	4.31E-04	1.95E-09	1.93E-02	2.34E-03

Since we ran the system for only 100 generation, we pooled the results of the 20 runs to get the Pareto front shown in Figure 8.3 below.

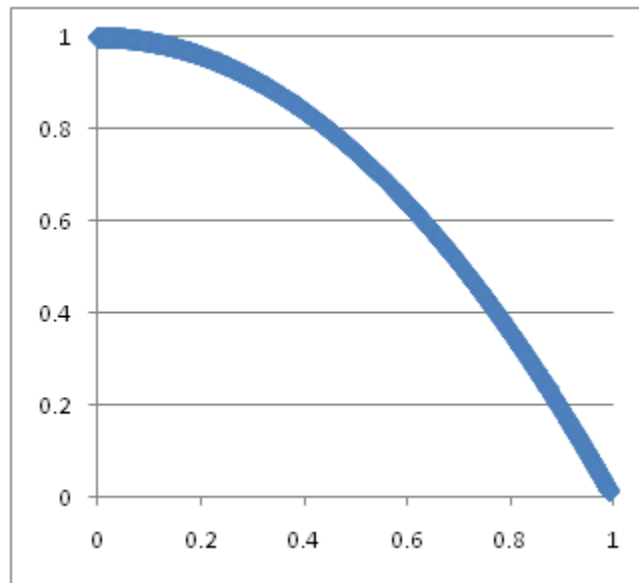


Figure 8.3 Overall Computed Pareto front

8.1.2 The Use of the Topologies in MOCAT with the Spread Metric

Table 8-2 gives the count of the topologies used under the spread performance metric. For the Convex problem there were no significant differences in usage of the topologies by MOCAT with the Spread Metric. Here, the use of the GLOBAL topology is significantly less than the others with the exception of the Hexagonal topology.

Table 8-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	13	20	22	28	17	1.075076
Run #2	30	17	20	15	18	7.713442
Run #3	27	17	19	19	18	6.858578
Run #4	19	19	21	18	23	6.617002
Run #5	23	29	28	9	11	1.131607
Run #6	24	25	15	18	18	8.642619
Run #7	12	23	23	18	24	3.38353
Run #8	24	19	28	11	18	6.399448

Run #9	23	25	15	20	17	7.085921
Run #10	11	21	30	22	16	3.380326
Run #11	21	14	17	24	24	1.41518
Run #12	25	17	19	24	15	7.526634
Run #13	30	12	13	30	15	5.384621
Run #14	28	19	17	20	16	1.131752
Run #15	28	22	24	10	16	5.427405
Run #16	26	24	12	17	21	7.559107
Run #17	15	15	35	18	17	1.409467
Run #18	17	22	22	20	19	6.90993
Run #19	21	14	25	21	19	7.021774
Run #20	24	27	16	21	12	7.132664
mean	22.05	20.05	21.05	19.15	17.70	
stdev	5.71	4.50	5.87	5.20	3.35	
t-test	Lbest vs.	0.119021	0.298726	0.054902	0.003716	
	Square vs.		0.279661	0.285862	0.038157	
	Octal vs.			0.148802	0.019375	
	Hex vs.				0.157143	

In the convex case the more complex topologies such as global were more active in the problem solving process. For the concave case it is the other way around, LBEST is now used most frequently on average. This suggests that too much communication with other individuals may have adverse effects on the evolution for ZDT2. It may be that too much communication results in more individuals being moved in the wrong direction from the concave front. Additionally, it suggests that the system is able to adjust its search process to reflect changes in the Pareto front for a given problem.

8.1.3 Knowledge Source Usage with the Spread Metrics

In table 8-3 the number of individuals influenced by each knowledge source for each run is given. Notice that as with the convex problem each knowledge source is used to a statistically different extent. The system here has reduced the emphasis on exploration by reducing the usage of normative and topographic knowledge and increased its use of domain and situational knowledge source, ones that

focus on exploitation in an area. This may be a way of reducing the number of attempts to move individuals to the left of the linear approximation, so as to generate a concave one.

Table 8-3 The Number of Individuals Influenced by the Spread Metric for Each Knowledge Source.

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	260	1487	1476	1020	99
Run #2	261	1337	1550	892	96
Run #3	295	1362	1616	863	93
Run #4	318	1344	1467	973	73
Run #5	277	1358	1501	983	101
Run #6	298	1375	1602	704	87
Run #7	285	1317	1662	674	101
Run #8	309	1442	1479	1026	89
Run #9	308	1390	1540	880	73
Run #10	252	1388	1514	925	60
Run #11	295	1444	1528	1013	92
Run #12	287	1410	1547	832	74
Run #13	299	1389	1523	918	75
Run #14	258	1398	1533	1033	91
Run #15	311	1309	1560	946	99
Run #16	310	1395	1562	918	86
Run #17	269	1396	1534	770	73
Run #18	266	1474	1485	945	64
Run #19	294	1390	1480	996	69
Run #20	291	1475	1519	807	85
mean	287.15	1394.00	1533.90	905.90	84.00
stdev	19.73	49.18	48.85	102.07	12.55
t-test	KS-N vs.	2.58E-33	9.48E-35	2.07E-17	1.4E-28
	KS-S vs.		5.29E-11	1.81E-17	1.54E-31
	KS-D vs.			2.9E-20	1.39E-32
	KS-H vs.				2.2E-19

8.1.4 Combined Usage Frequency of Topology-Knowledge Tuple with the Spread

Performance Metric

In Table 8-4 the average number of non-dominated solutions produced by each knowledge source topology pair is given. Although the most complex topologies dominated the generation of non-dominated solutions for the convex problem, here the simplest topology dominates the generation of non-dominated solutions. This suggests that in the concave problem much more local search is used. In Table 8-5 the variability in production is now associated with the exploratory knowledge sources regardless of topology. This means that exploratory knowledge sources will be invoked more frequently when they have found a good region in the search space.

Table 8-4 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	65.65	308.3	343.1	196.95	21.2
	stdev	31.01829	76.42065	98.05847	63.95432	11.53302
Square	mean	52.25	278.75	306.65	182.85	15.35
	stdev	21.036	68.70139	75.46228	42.44907	8.36833
Octal	mean	64.5	288.05	325.5	190.4	17.55
	stdev	29.18273	76.6856	87.49707	60.22624	10.41494
Hex	mean	55.3	268.4	291.5	177	17.95
	stdev	22.06474	76.5736	76.91246	55.05595	10.92787
Global	mean	49.45	250.5	267.15	158.7	11.95
	stdev	21.69034	52.93243	55.47429	35.55885	7.830272

Table 8-5 Randomness of Topology-Knowledge tuple, Standard deviation divided by the mean

stdev/mean	N	S	D	H	T
Lbest	0.47248	0.247878	0.285801	0.324724	0.54401
Square	0.402603	0.246462	0.246086	0.232152	0.545168
Octal	0.452445	0.266223	0.268808	0.316314	0.593444
Hex	0.399001	0.285297	0.263851	0.311051	0.608795
Global	0.438632	0.211307	0.207652	0.224063	0.655253

8.2 Solving ZDT 2 30 with Homogeneous Hyper-Volume Metric Performance

Function

For the convex problem the spread metric by itself was more effective in solving the problem than the hyper- volume metric. For the concave problem it is the situation is reversed. The Hyper- volume guided system outperforms the spread metric based system in terms of mean error and standard deviation. In fact, it outperformed all of the other systems based upon their statistics in tables 6-2 and 6-3 using just 100 individuals over 100 generations.

The runs were complete in an average of 36.78 seconds.

8.2.1 Performance of MOCAT on ZDT2 Using the Hyper-Volume Metric

In Table 8-6 the mean error and standard deviation of the error is given. These values are both improvements over that of the spread metric and better than any of the other benchmarked system values given in Chapter 6. This suggests that while the spread metric is more effective in guiding MOCAT search in convex situations, the Hyper-Volume metric does better in the concave situation.

Table 8-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	Mean	min	max	stdev
Run #1	6.13E-06	1.95E-04	1.68E-12	1.55E-02	1.58E-03
Run #2	8.51E-05	2.39E-04	1.48E-09	9.13E-04	2.21E-04
Run #3	2.48E-04	2.86E-04	4.22E-15	1.02E-03	2.33E-04
Run #4	1.33E-05	3.21E-05	2.08E-09	1.92E-04	2.89E-05
Run #5	8.67E-06	3.65E-05	-5.55E-17	3.23E-04	5.27E-05
Run #6	6.74E-06	2.01E-05	-2.11E-17	9.06E-05	1.99E-05
Run #7	7.97E-07	2.41E-06	7.25E-13	1.34E-05	3.49E-06
Run #8	5.26E-08	2.15E-07	1.29E-12	1.55E-06	2.80E-07
Run #9	4.97E-06	2.74E-05	3.54E-09	1.98E-04	3.71E-05

Run #10	3.45E-05	2.64E-04	0.00E+00	2.53E-03	5.05E-04
Run #11	1.81E-04	6.57E-04	1.92E-11	3.61E-03	6.91E-04
Run #12	2.10E-06	5.80E-06	4.15E-11	2.45E-05	5.66E-06
Run #13	1.73E-05	4.39E-05	2.44E-09	2.32E-04	4.15E-05
Run #14	1.32E-05	1.62E-05	-2.78E-17	7.37E-05	1.45E-05
Run #15	3.44E-05	5.92E-05	1.85E-11	2.07E-04	6.15E-05
Run #16	5.67E-08	4.19E-05	2.44E-15	9.79E-04	1.69E-04
Run #17	1.98E-08	5.93E-08	-5.39E-17	3.22E-07	7.54E-08
Run #18	2.53E-05	9.09E-05	-4.16E-17	7.43E-04	1.25E-04
Run #19	2.21E-05	6.49E-05	1.04E-17	2.87E-04	6.53E-05
Run #20	1.06E-04	2.82E-04	1.25E-11	1.72E-03	2.90E-04
mean	4.05E-05	1.18E-04	4.81E-10	1.43E-03	2.07E-04
stdev	6.49E-05	1.58E-04	1.01E-09	3.35E-03	3.61E-04

8.2.2 Statistics of Topologies using Hyper-volume metric

Table 8-7 gives the number of times that each topology was used over the 20 runs. The statistical tests in the extended table suggest that all topologies are used equally over the search. This is in contrast with the Spread metric situation in which the global topology was used significantly less. There was a slight tendency to use the simpler topologies such as square, but not a statistically significant trend.

Table 8-7 Use Count of Topologies of each run for the Hyper-Volume

	Using Hyper-volume metric					Hyper-volume
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	Metric Value
Run #1	23	19	24	16	18	1.011891
Run #2	14	15	23	21	27	1.304145
Run #3	20	21	24	14	21	1.531087
Run #4	27	17	17	20	19	1.185342
Run #5	23	13	21	20	23	1.214848
Run #6	18	30	17	19	16	1.37573
Run #7	24	15	25	16	20	1.138522
Run #8	23	19	15	25	18	0.900597
Run #9	20	22	17	18	23	1.044022

Run #10	24	21	13	28	14	1.734349
Run #11	22	19	20	16	23	1.32526
Run #12	22	19	24	23	12	0.949667
Run #13	18	31	15	20	16	1.17246
Run #14	20	28	18	17	17	1.351572
Run #15	17	18	29	12	24	1.152803
Run #16	24	26	20	14	16	1.055576
Run #17	17	21	24	19	19	0.907154
Run #18	19	12	25	24	20	1.401431
Run #19	7	27	18	25	23	0.963887
Run #20	17	11	17	30	25	1.135178
mean	19.95	20.20	20.30	19.85	19.70	
stdev	4.32	5.64	4.16	4.68	3.82	
t-test	Lbest vs.	0.439501	0.400273	0.472902	0.425572	
	Square vs.		0.475394	0.418181	0.375589	
	Octal vs.			0.377955	0.323044	
	Hex vs.				0.457227	

Table 8-8 gives the number of individuals controlled by the knowledge sources over the 20 runs. All five knowledge sources exhibited statistically different influences over the 20 runs. As with the spread metric the two exploitative knowledge sources, situational and domain, were used the most frequently. The two exploratory knowledge sources were used much less frequently.

Table 8-8 Using Spread metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	292	1353	1499	895	78
Run #2	278	1425	1601	743	89
Run #3	278	1455	1549	793	78
Run #4	334	1363	1561	989	97
Run #5	290	1423	1540	893	83
Run #6	293	1402	1510	822	77
Run #7	259	1480	1556	784	85
Run #8	311	1500	1471	957	72
Run #9	282	1347	1650	822	91
Run #10	254	1377	1568	787	89

Run #11	244	1397	1555	856	86
Run #12	279	1426	1508	992	97
Run #13	300	1401	1557	853	93
Run #14	286	1456	1470	922	90
Run #15	274	1375	1499	948	90
Run #16	346	1431	1464	883	71
Run #17	298	1392	1464	971	65
Run #18	267	1433	1542	861	72
Run #19	273	1441	1543	843	53
Run #20	247	1367	1634	840	73
mean	284.25	1412.20	1537.05	872.70	81.45
stdev	25.31	41.14	51.51	70.68	11.15
t-test	KS-N vs.	1.26E-41	1.17E-36	4.94E-22	9.18E-23
	KS-S vs.		3.81E-10	5.53E-24	1.05E-33
	KS-D vs.			3.62E-28	2.46E-31
	KS-H vs.				2.02E-22

8.2.3 Number of Non-Dominated Solutions generated by Each Topology-Knowledge tuple

In table 8-9 the number of non-dominated solution produced by each topology-knowledge source pair is given. While the Octal Domain tuple generates the most there is no dominant configuration on display here. In Table 8-10 it is clear that while the topographic knowledge source generates the fewest overall; that the number generated can increase if it finds a good region to search.

Table 8-9 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	53.9	283.65	302.05	174.95	14.85
	stdev	14.54901	65.62916	65.20129	50.52928	7.095403
Square	mean	59.35	280.8	308.8	175.5	20.1
	stdev	26.65624	82.59196	87.71161	51.30661	13.61462
Octal	mean	59.95	286.1	317.35	181.4	16.9
	stdev	21.08498	59.855	68.68714	47.33242	10.73067
Hex	mean	56.65	282.85	304.55	171.6	14.85
	stdev	20.23617	71.46974	81.9059	47.14412	8.863141
Global	mean	54.4	278.8	304.3	169.25	14.75
	stdev	16.69352	55.82265	67.81872	37.88886	9.419325

Table 8-10 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.269926	0.231374	0.215863	0.288821	0.477805
Square	0.449136	0.294131	0.28404	0.292345	0.677344
Octal	0.351709	0.20921	0.21644	0.260928	0.634951
Hex	0.357214	0.252677	0.268941	0.274733	0.596844
Global	0.306866	0.200225	0.222868	0.223863	0.638598

8.3 ZDT 2 30 with combined metrics

For the convex problem the combined solution using both metric did not fare as well as the spread metric on its own, although the combined approach had a slightly reduced error. The problem was simple enough that the potential synergy provided by both metrics was not necessary. Here, the combination of the two metrics produced an improvement over either on their own and changed how each participated in the solution. Recall that in these runs the metrics took turns guiding the search for 25 generations each.

The runs were complete in an average of 35.78 seconds.

8.3.1 Performance of MOCAT

In Table 8-11 is clear that the combined system performance is an improvement over both metrics on their own. While the Hyper-Volume metric was the better of the two, the combined system outperforms that as well. The results outperform those for all of the benchmarked systems as described in chapter 6.

Table 8-11 Statistics for the fitness errors of ending solutions

Statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	1.79E-05	4.59E-05	2.50E-12	9.70E-04	1.15E-04

Run #2	7.33E-05	1.80E-04	-5.55E-17	7.07E-04	1.46E-04
Run #3	1.33E-05	1.43E-04	0.00E+00	1.06E-03	2.16E-04
Run #4	1.05E-05	2.57E-05	2.56E-09	7.83E-05	1.85E-05
Run #5	6.79E-09	1.93E-08	3.82E-17	7.29E-08	1.81E-08
Run #6	6.97E-06	3.14E-05	2.62E-12	2.44E-04	4.52E-05
Run #7	8.14E-06	1.17E-05	2.21E-15	5.55E-05	1.10E-05
Run #8	1.06E-05	5.27E-05	3.38E-17	4.29E-04	7.66E-05
Run #9	4.31E-05	5.27E-05	3.40E-11	2.02E-04	4.60E-05
Run #10	1.53E-05	3.92E-05	1.69E-12	2.62E-04	4.05E-05
Run #11	1.35E-06	5.66E-05	2.08E-16	1.98E-03	2.27E-04
Run #12	2.94E-05	7.65E-05	6.28E-13	3.71E-04	8.07E-05
Run #13	2.98E-05	1.04E-03	3.66E-13	8.95E-02	9.12E-03
Run #14	5.82E-05	1.50E-04	3.16E-08	4.70E-04	1.19E-04
Run #15	1.44E-06	5.07E-06	5.84E-13	2.68E-05	5.38E-06
Run #16	3.94E-06	1.24E-05	2.39E-13	5.31E-05	1.28E-05
Run #17	3.36E-06	9.83E-06	1.70E-11	5.70E-05	1.39E-05
Run #18	1.84E-06	4.27E-06	3.02E-09	1.15E-05	3.24E-06
Run #19	4.88E-05	2.38E-04	6.08E-13	1.95E-03	3.57E-04
Run #20	2.71E-05	1.06E-04	2.21E-09	5.88E-04	1.23E-04
mean	2.02E-05	1.14E-04	1.97E-09	4.95E-03	5.39E-04
stdev	2.05E-05	2.22E-04	6.87E-09	1.94E-02	1.97E-03

8.3.2 Statistics of Topologies for both metrics

Tables 8-12 and 8-13 give the number of times that each topology was used in the combined run. Since each was used overall for just 50 generations the counts will have changed but what is important is the differences in usage between the topologies. When used by itself the spread metric used statistically fewer global topologies however there is no difference in usage. On the other hand the spread metric now uses significantly more global topologies than before.

Table 8-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	10	7	8	19	6	6.991931

Run #2	12	9	7	12	10	6.648939
Run #3	9	8	4	21	8	6.661652
Run #4	17	8	7	11	7	1.329185
Run #5	2	10	14	6	18	6.378567
Run #6	12	13	10	9	6	6.691653
Run #7	9	9	6	12	14	4.612961
Run #8	4	7	13	16	10	7.144973
Run #9	9	10	9	14	8	7.232991
Run #10	13	5	14	9	9	5.881953
Run #11	8	8	6	5	23	1.124861
Run #12	12	11	11	9	7	1.352452
Run #13	9	7	11	3	20	1.37916
Run #14	9	9	9	13	10	0.9547
Run #15	12	14	11	6	7	1.152332
Run #16	7	13	9	13	8	4.821896
Run #17	11	10	12	9	8	6.209814
Run #18	11	11	3	9	16	1.275664
Run #19	16	11	10	9	4	1.259173
Run #20	13	5	13	8	11	7.114922
mean	10.25	9.25	9.35	10.65	10.50	
stdev	3.45	2.43	3.09	4.41	4.96	
t-test	Lbest vs.	0.154232	0.200979	0.378583	0.429004	
	Square vs.		0.456108	0.117393	0.166355	
	Octal vs.			0.149865	0.198811	
	Hex vs.				0.461037	

Here we cannot see statistical difference between any pair of topologies.

Table 8-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric				Hyper-volume metric value	
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	8	17	4	10	11	1.46096
Run #2	10	5	11	5	19	1.253579
Run #3	10	9	8	11	12	1.226193
Run #4	7	11	12	8	12	1.516897
Run #5	8	8	9	14	11	1.112439
Run #6	10	6	10	14	10	1.425319
Run #7	11	9	11	6	13	0.983411
Run #8	14	9	10	9	8	0.962264

Run #9	11	16	12	6	5	1.747168
Run #10	7	10	11	12	10	1.326611
Run #11	11	8	8	9	14	1.769225
Run #12	9	7	13	14	7	1.237062
Run #13	8	8	17	8	9	1.345778
Run #14	10	8	13	6	13	1.194311
Run #15	11	5	7	14	13	2.493285
Run #16	9	7	10	11	13	1.112412
Run #17	9	7	11	7	16	1.096588
Run #18	9	1	16	13	11	1.379982
Run #19	11	6	10	12	11	1.128648
Run #20	7	15	4	11	13	1.125405
mean	9.50	8.60	10.35	10.00	11.55	
stdev	1.72	3.75	3.18	2.97	3.01	
t-test	Lbest vs.	0.174889	0.156994	0.264825	0.007496	
	Square vs.		0.064624	0.104895	0.005553	
	Octal vs.			0.363891	0.119852	
	Hex vs.				0.059023	

As shown in the above table, this time Square has a smaller mean than other topologies while Global has the biggest mean. The difference between Lbest and Global and Square and Global are statistically important.

8.3.3 Behaviors of Knowledge sources with the two metrics

Tables 8-14 and 8-15 give the number of individuals controlled by each knowledge source over the 20 runs. As before, each knowledge source exhibits a significantly different control effect from the others. The domain and situational knowledge sources still dominate, but now the history knowledge source plays more of a role. This is because as control shifts between the two metrics the history knowledge source is able to facilitate the shift in terms of the distribution of information from one phase to the next. The results of the spread metric are now available for the Hyper-Volume metric through the history knowledge source.

Table 8-14 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	197	684	771	498	66
Run #2	182	746	698	446	68
Run #3	183	697	758	446	47
Run #4	188	707	820	343	59
Run #5	166	655	763	454	45
Run #6	199	699	779	460	52
Run #7	187	684	767	411	68
Run #8	155	727	749	469	59
Run #9	186	748	794	343	73
Run #10	195	676	829	340	61
Run #11	222	658	756	423	63
Run #12	185	759	757	496	61
Run #13	208	700	766	411	57
Run #14	180	711	760	405	58
Run #15	183	704	799	404	43
Run #16	153	695	731	496	68
Run #17	200	718	787	367	55
Run #18	179	736	769	477	41
Run #19	162	660	753	464	60
Run #20	195	690	798	399	63
mean	185.25	702.70	770.20	427.60	58.35
stdev	16.68	29.07	28.87	50.17	8.69
t-test	KS-N vs.	7.54E-35	1.23E-36	2.12E-16	3.18E-23
	KS-S vs.		7E-09	8.64E-20	1.04E-30
	KS-D vs.			1.68E-22	8.66E-32
	KS-H vs.				6.1E-19

Table 8-15 Using Hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	94	703	728	545	38
Run #2	77	658	732	501	36
Run #3	92	696	726	501	14
Run #4	96	717	780	422	25

Run #5	113	679	789	422	31
Run #6	88	699	784	433	31
Run #7	94	723	722	527	39
Run #8	96	699	754	479	36
Run #9	80	712	757	468	40
Run #10	104	709	776	385	33
Run #11	82	670	766	534	27
Run #12	91	673	734	599	36
Run #13	106	641	766	476	10
Run #14	85	718	794	442	22
Run #15	95	693	772	494	36
Run #16	98	701	754	533	33
Run #17	92	701	747	504	28
Run #18	93	719	762	513	44
Run #19	87	725	761	490	32
Run #20	98	692	783	503	43
mean	93.05	696.40	759.35	488.55	31.70
stdev	8.49	21.85	21.57	48.89	8.60
t-test	KS-N vs.	3.67E-35	1.65E-36	9.32E-20	1.3E-23
	KS-S vs.		3.53E-11	5.86E-16	2.43E-36
	KS-D vs.			9.87E-19	1.31E-37
	KS-H vs.				5.09E-21

8.3.4 The Generation of non-dominated solutions by Topology-Knowledge tuple

Tables 8-16 and 8-17 give the number of non-dominated solutions produced by each tuple on the one hand, and the variability in that production on the other. Recall that for the spread metric the LBEST topology was the most productive. Now the global topology has become the dominant one. In addition, the ability of history knowledge to produce non-dominated solutions has increased markedly as well.

Thus, in the combined system increased performance is produced by increasing the connectivity within the topology on the one hand and between the metric segments using history knowledge on the other.

Table 8-16 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	59.25	272.8	303.25	180.4	14.2
	stdev	23.48544	52.60038	57.56449	34.90498	7.338435
Square	mean	46.35	247.7	272.25	167.15	16.85
	stdev	15.8721	42.45505	55.50901	36.03401	9.033709
Octal	mean	51.05	277.6	303.95	174.4	21.2
	stdev	15.73623	58.36131	65.78712	43.45887	10.90919
Hex	mean	61.9	291.4	311.15	191.85	18.15
	stdev	18.18588	71.47351	75.3918	50.83852	10.40382
Global	mean	59.75	309.6	338.95	202.35	19.65
	stdev	30.69695	82.78978	98.57376	52.70601	10.0644

Table 8-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.396379	0.192817	0.189825	0.193487	0.516791
Square	0.34244	0.171397	0.20389	0.215579	0.536125
Octal	0.308251	0.210235	0.216441	0.249191	0.514585
Hex	0.293795	0.245276	0.242301	0.264991	0.573213
Global	0.513756	0.267409	0.290821	0.26047	0.512183

8.3.5 Analysis of found Pareto front

The resultant Pareto front produced by merging the results of the 20 runs is given below for the combined system.

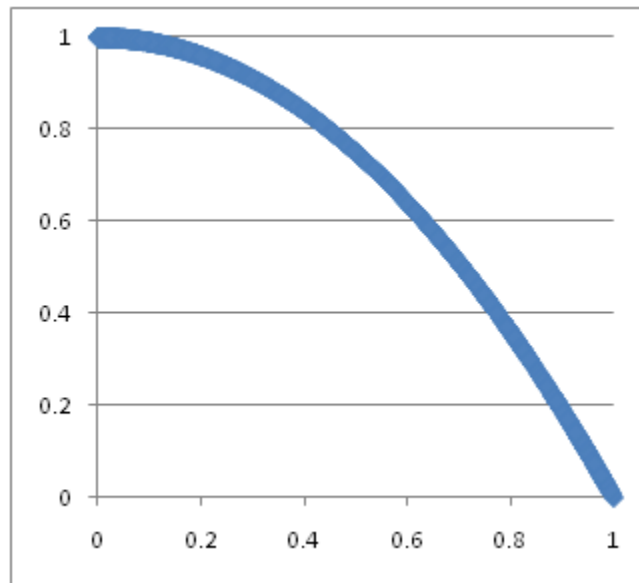


Figure 8.4 Overall found Pareto front

8.4 Summary of ZDT2

ZDT2 has a convex Pareto front which is normally considered as a challenge for hyper-volume metric. However, in our experiments, performance of hyper-volume metric cannot be considered inferior. This has to do with the fact that in our system the hyper-volume is not just the only source of knowledge in the search. Data collected in the knowledge sources is used to guide the search. As a result, the hyper-volume metric outperforms the spread metric here as well as the other benchmarked systems. In summary, the following conclusions can be drawn:

1. Of the two solo metrics, the hyper-volume metric is better as controlling the MOCAT search for the problem than the spread metric.
2. Here the combined system outperforms either metric individually since both types of metric knowledge are integrated together. The integration takes place through the increased use of

the global topology to distribute information and the history knowledge source to convey information produced by the one metric for use by the other metric.

3. Unlike the other MOEP systems, the metrics are not the principle source of information here. They are used as guides for the generation of knowledge and its spread through the social fabric by MOCAT. MOCAT is able to adjust the configuration of the system in order to exploit the different information provided by each.

CHAPTER 9 SOLVING EXPERIMENT RESULTS OF ZDT3

ZDT3 is not an easy problem for MOEAs because its real Pareto front consists of five segments that are far from each other which may impair the effectiveness of both spread metric and hyper-volume metric. In other words the front has been discretized into segments. We will still use homogeneous spread metric, homogeneous hyper-volume metric, and the combined two metrics to conduct experiments and try to observe and summarize the correspondence among knowledge sources and topologies.

While such a problem can be hard for other MOEAs it is important to recall that those approaches only use the metrics to guide the search. In MOCAT there are 5 other knowledge sources that can contribute. It is clear from the discussion of the first two problems that the presence of these five knowledge sources serves to enhance the performance of the MOCAT system over traditional ones.

9.1 ZDT 3 30 with homogeneous Spread metrics

ZDT3 is simply a modified version of ZDT1 where a sine function has been added to the equation which results in the discretization of the front into segments. Although the front is now segmented, the parameter space is still continuous. The optimal Pareto Front is given in Figure 8.3. This can pose a problem to systems that just use spread metrics. However, MOCAT has other knowledge sources than can generate knowledge for use in the search. The question then is, what knowledge sources will be recruited here to deal this discretization issue?

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \sum_{i=2}^m x_i / (m - 1)$$

$$h(f_1, g) = 1 - \sqrt{x_1/g} - \frac{x_1}{g} * \sin(10 * \pi * x_1)$$

$$f_2 = g * h$$

The Pareto front is:

$$x_2 = 1 - \sqrt{x_1} - x_1 * \sin(10 * \pi * x_1) \text{ where}$$

$$x_1 \text{ in } [0, 0.0830015349] \cup (0.1822287280, 0.2577623634] \cup$$

$$(0.4093136748, 0.4538821041] \cup (0.6183967944, 0.6525117038] \cup$$

$$(0.8233317983, 0.8518328654]$$

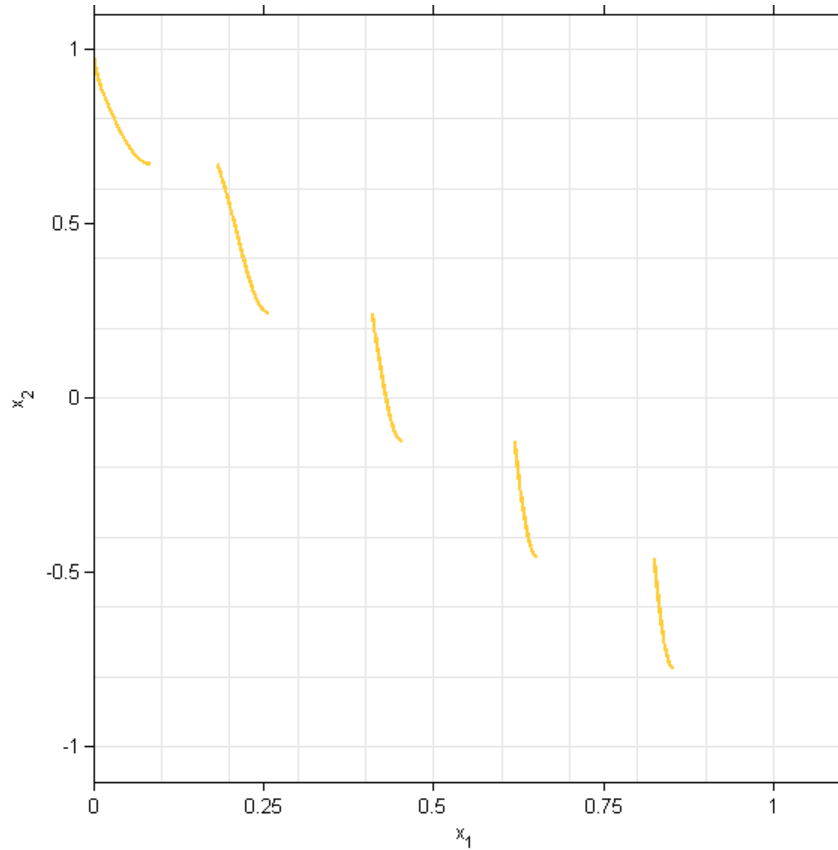


Figure 9.1 ZDT3 Pareto front

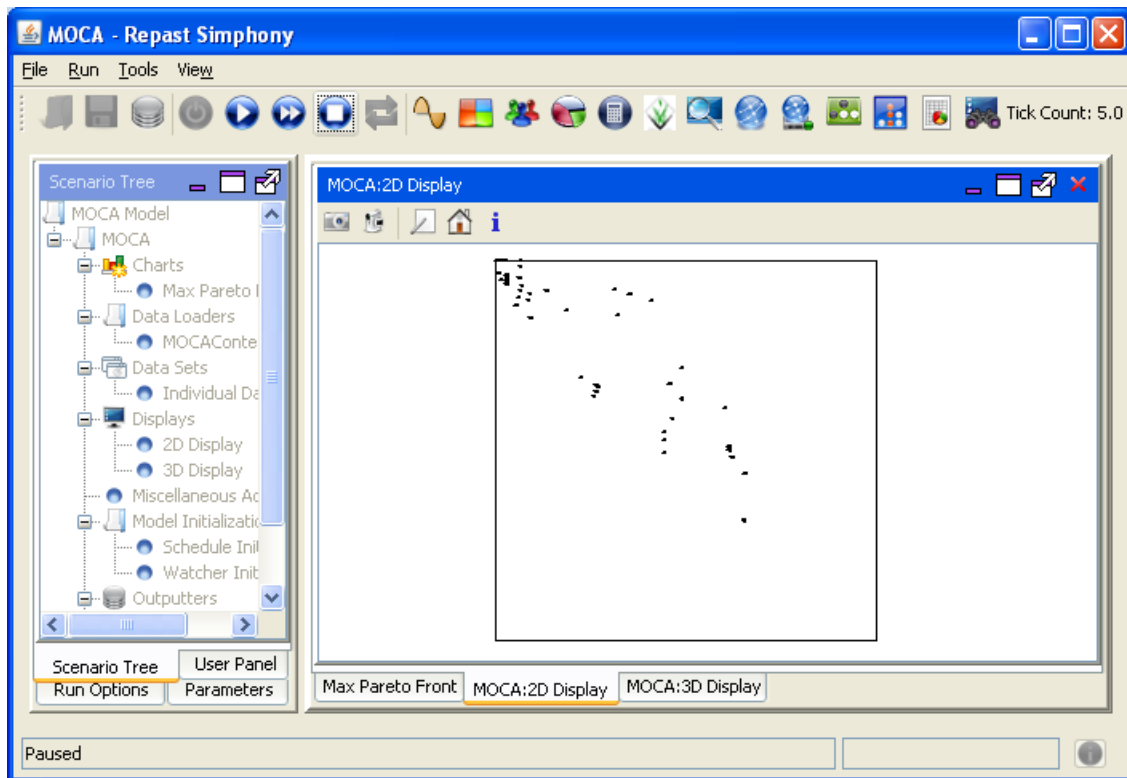
9.2 Performance of MOCAT using the spread metric for zdt3 30

In the previous problems MOCAT started with a basic linear approximation of the curve and then adjusted it using exploitation activities. The system works the same way here. It is illustrated in the following section using the spread metric to guide the process.

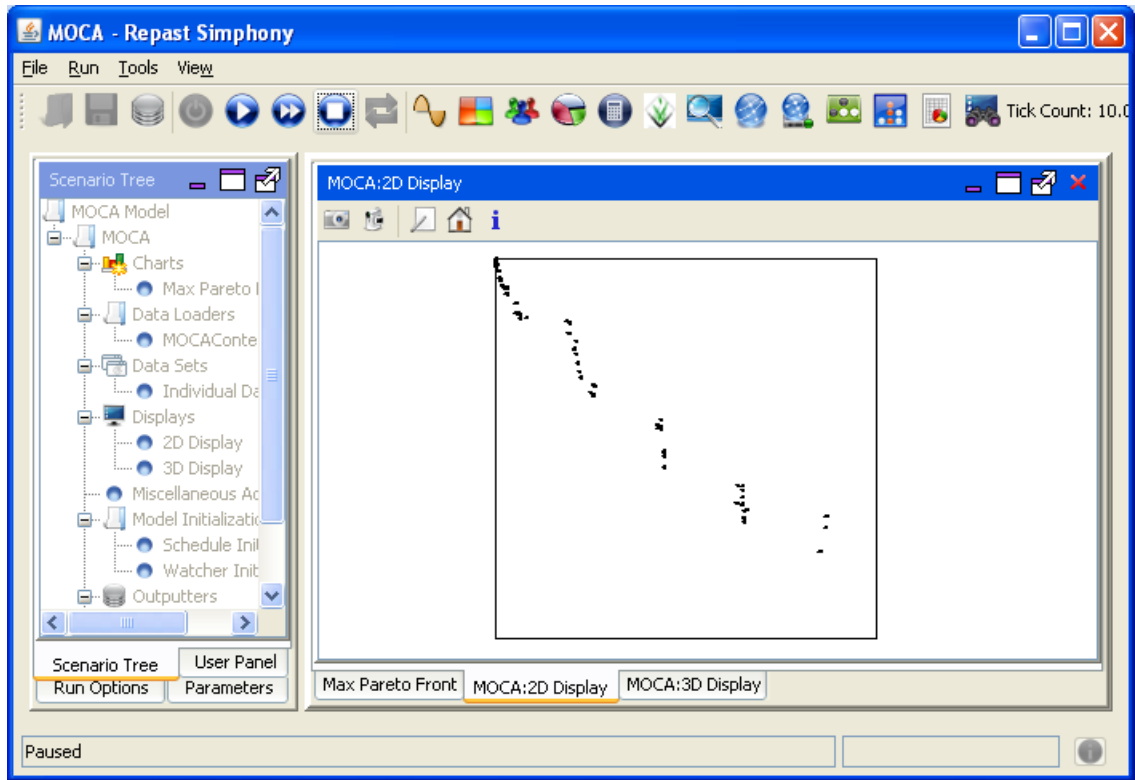
The runs were complete in an average of 32.48 seconds for 100 generation and 100 individuals in each population.

9.2.1 Using the spread metric to guide the solution of ZDT3 in 30 dimensions

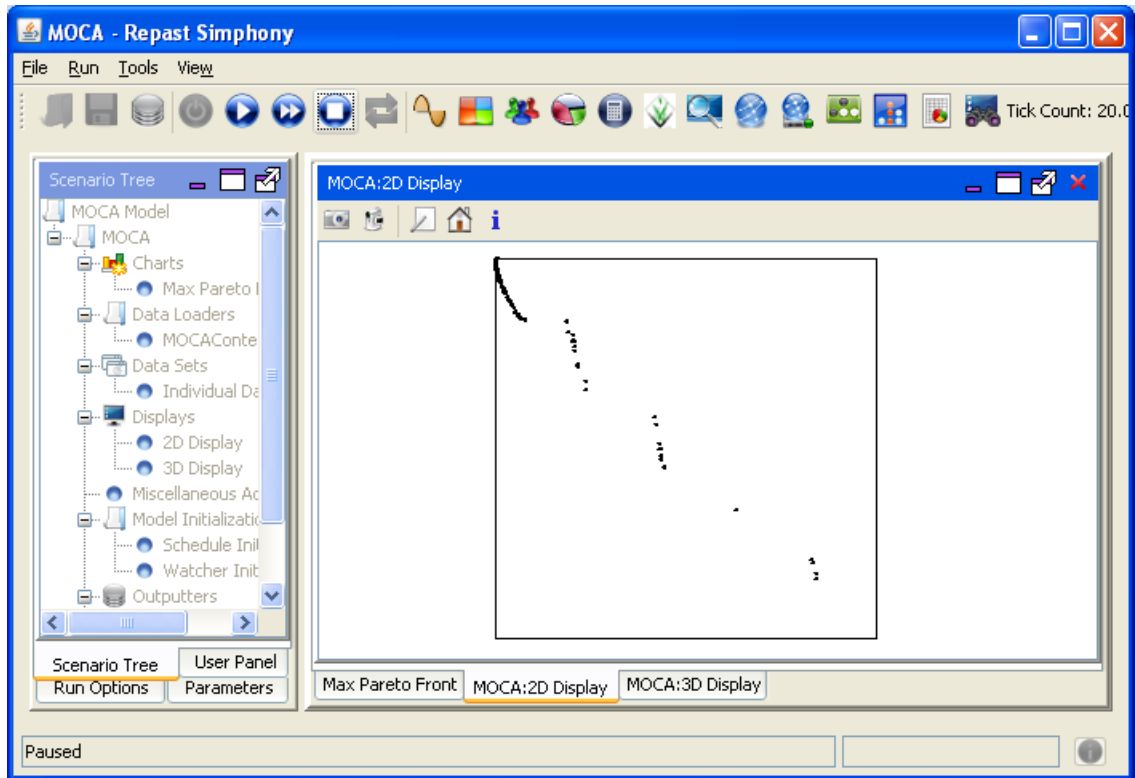
The following sequence of figures 9(a-e) presents in one run how the population moves to the Pareto front under the spread metric. As before, the system generates a linear approximation of the curve and then proceeds to bend each segment successively to produce a more curved shape. There is a clear preference for the system to modify the segments from left to right.



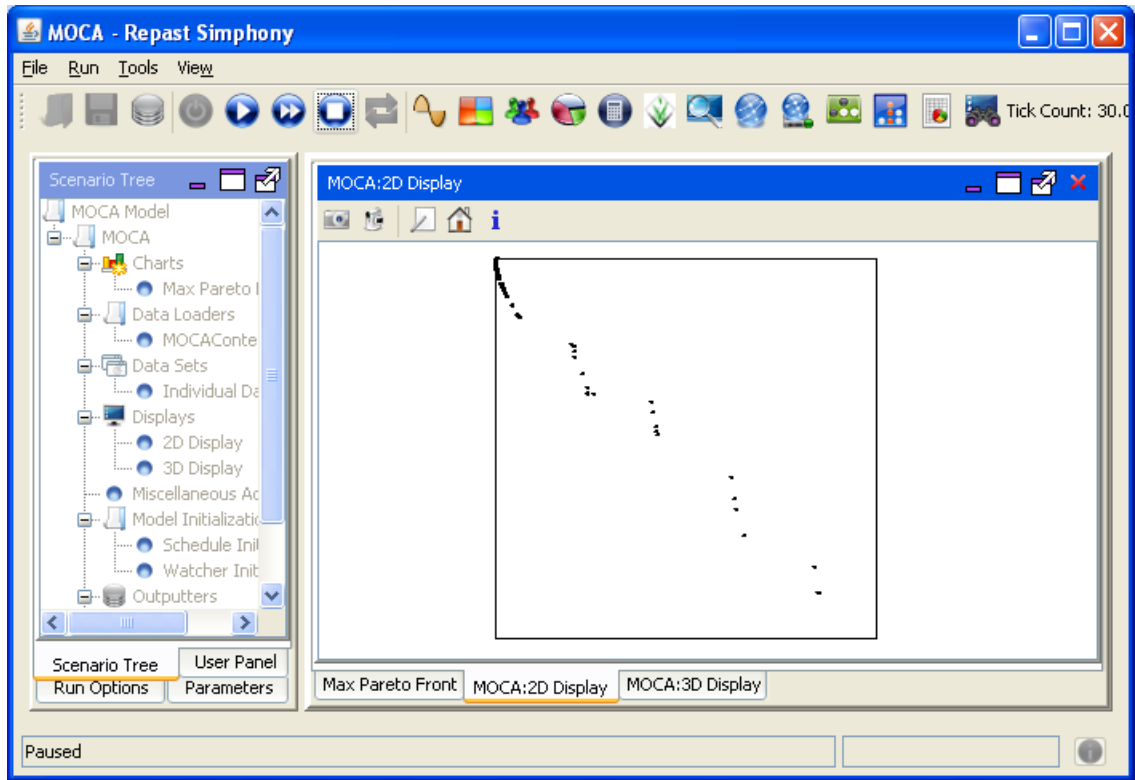
(a)



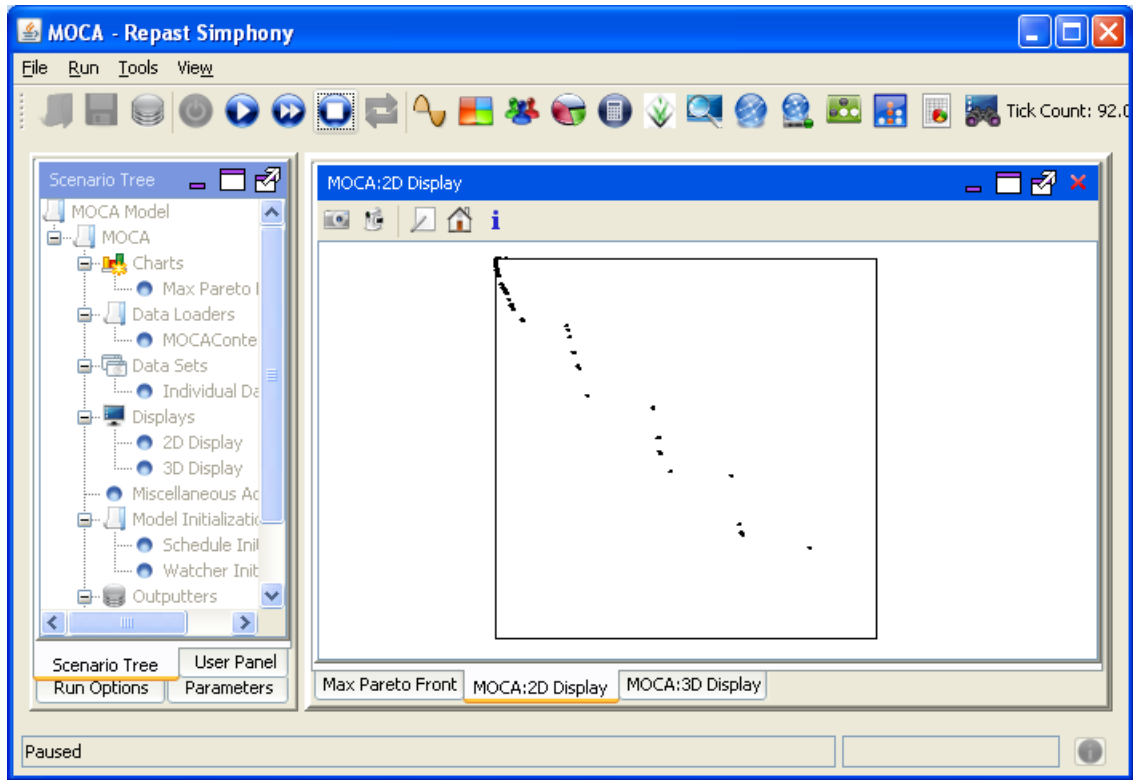
(b)



(c)



(d)



(e)

Figure 9.2 (a,b,c,d,e) A series of screen copies showing the evolution of the front under the guidance of the spread metric.

Finally, in Figure 9.3 the addition of found Pareto front from each run is represented, which is a much better approximation of the real Pareto front, compared to any single run.

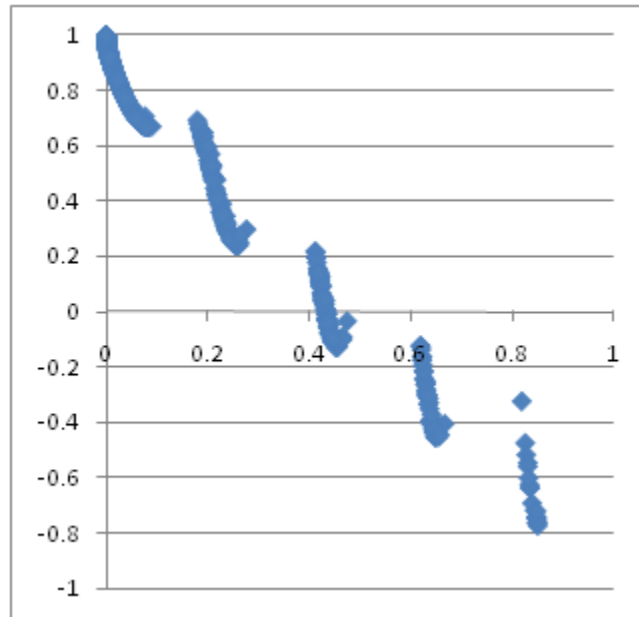


Figure 9.3 Overall found Pareto front

We can see that while each run left some empty spots along the real Pareto front—which is unavoidable because there are always only limited number of individuals and each was run for only 100 generations, in total the 20 runs covers the Pareto front. The overall found Pareto front does not have a perfect shape and there are some outliers which reflect the fact that the last segment at the bottom still needs exploration.

9.2.2 Performance of MOCAT Using the Spread Metric

Table 9-1 gives the run statistics for MOCAT using just the spread metric. The results are not as good as for the first two problems since it is more difficult. On the other hand, the achieved mean error and standard deviation exceeds all of the other benchmarked systems except ANMOPSO. However, this is just with one metric on board. It will be interesting to see how the combined metric will fare.

Table 9-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	4.39E-03	1.25E-02	1.87E-04	8.76E-02	1.76E-02
Run #2	5.06E-03	2.25E-02	3.89E-05	8.98E-02	2.52E-02
Run #3	9.06E-03	2.13E-02	4.14E-05	9.28E-02	2.47E-02
Run #4	1.14E-02	2.02E-02	1.27E-04	9.90E-02	2.26E-02
Run #5	4.08E-03	1.35E-02	3.40E-05	6.18E-02	1.44E-02
Run #6	1.11E-02	1.85E-02	7.00E-04	9.84E-02	2.14E-02
Run #7	4.89E-03	2.26E-02	7.45E-04	8.89E-02	2.40E-02
Run #8	1.06E-02	2.01E-02	6.65E-05	9.48E-02	2.23E-02
Run #9	8.55E-03	1.62E-02	4.34E-04	9.05E-02	2.12E-02
Run #10	4.28E-03	1.39E-02	1.98E-04	8.61E-02	1.68E-02
Run #11	3.84E-03	1.57E-02	9.92E-04	8.28E-02	1.93E-02
Run #12	8.42E-03	1.63E-02	1.09E-04	7.70E-02	1.99E-02
Run #13	4.28E-03	1.66E-02	2.30E-04	9.92E-02	2.01E-02
Run #14	5.11E-03	2.06E-02	2.06E-04	9.52E-02	2.44E-02
Run #15	3.73E-03	1.49E-02	1.60E-04	8.61E-02	1.95E-02
Run #16	6.22E-03	1.68E-02	6.67E-04	9.32E-02	2.29E-02
Run #17	8.95E-03	1.54E-02	2.76E-04	8.63E-02	1.70E-02
Run #18	5.73E-03	1.01E-02	5.38E-04	9.72E-02	1.41E-02
Run #19	5.54E-03	9.78E-03	2.21E-04	6.36E-02	1.33E-02
Run #20	6.85E-03	1.57E-02	2.35E-04	8.40E-02	2.15E-02
mean	6.61E-03	1.67E-02	3.10E-04	8.77E-02	2.01E-02
stdev	2.51E-03	3.65E-03	2.68E-04	1.01E-02	3.52E-03

9.2.3 Statistics of Topologies using Spread metric

Table 9-2 give the usage of the topologies over the 20 runs. It can be seen that there is no preference for one topology over another with the spread metric.

Table 9-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	20	19	21	14	26	1.833499

Run #2	25	23	21	16	15	1.153962
Run #3	1	31	27	19	22	0.952648
Run #4	25	19	10	29	17	1.472176
Run #5	22	14	24	19	21	1.26794
Run #6	8	26	26	27	13	2.036993
Run #7	21	28	21	14	16	1.754554
Run #8	19	16	22	22	21	1.362669
Run #9	27	18	12	28	15	1.363473
Run #10	21	25	16	13	25	1.883721
Run #11	25	19	15	15	26	1.856412
Run #12	28	21	19	21	11	1.766661
Run #13	20	21	21	22	16	1.377467
Run #14	27	25	11	19	18	1.028349
Run #15	16	16	19	28	21	1.503096
Run #16	28	15	15	22	20	1.407871
Run #17	26	17	18	18	21	1.538659
Run #18	16	17	24	24	19	1.926854
Run #19	24	21	17	17	21	1.499136
Run #20	22	18	21	19	20	1.372687
mean	21.05	20.45	19.00	20.30	19.20	
stdev	6.63	4.48	4.65	4.82	3.99	
t-test	Lbest vs.	0.372882	0.138731	0.346197	0.152743	
	Square vs.		0.166798	0.460668	0.184835	
	Octal vs.			0.201287	0.443827	
	Hex vs.				0.224252	

9.2.4 Behaviors of Knowledge sources using Spread Metric

The number of individual influenced by each knowledge source is given in Table 9-3. What is interesting is that while the knowledge sources are still exhibiting statistically significant differences in influence, the role of the Knowledge Sources now different. The exploratory knowledge sources are now being used much more frequently to explore the space. Also, domain knowledge is now the dominant knowledge source for exploitation.

Table 9-3 Using Spread metric #Individuals influenced by KS.

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	463	1314	1588	718	198
Run #2	447	1260	1693	668	186
Run #3	446	1366	1574	793	149
Run #4	442	1280	1677	716	221
Run #5	391	1364	1615	799	175
Run #6	466	1257	1627	737	178
Run #7	428	1377	1585	635	176
Run #8	456	1340	1617	688	138
Run #9	399	1345	1570	745	203
Run #10	458	1247	1651	750	167
Run #11	440	1217	1656	695	205
Run #12	466	1284	1607	725	212
Run #13	478	1282	1614	817	216
Run #14	417	1213	1616	745	137
Run #15	456	1256	1604	805	190
Run #16	409	1352	1567	833	170
Run #17	476	1216	1676	682	177
Run #18	375	1371	1641	855	227
Run #19	475	1262	1589	801	223
Run #20	449	1291	1562	720	206
mean	441.85	1294.70	1616.45	746.35	187.70
stdev	28.92	53.66	37.89	58.04	26.51
t-test	KS-N vs.	1.27E-32	1.54E-46	1.23E-18	2.83E-27
	KS-S vs.		1.22E-21	2.24E-28	9.31E-35
	KS-D vs.			4.86E-34	2.83E-48
	KS-H vs.				4.62E-25

9.2.5 Statistics of Topology-Knowledge tuple

Tables 9-4 and 9-5 give the number of non-dominated solutions generated by each topology knowledge source tuple and the variability in their production respectively. It is clear that regardless of the topology domain knowledge better at generating solutions than the other knowledge sources

overall for this problem. In terms of variability, it appears that the LBEST topology is the most able to adjust to take advantage of new found areas.

Table 9-4 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	91.65	271.95	341.9	154.85	40.5
	stdev	31.87686	90.01197	113.0346	52.70901	15.35372
Square	mean	83.9	265.35	324.8	155.6	40.85
	stdev	23.27388	65.18014	72.44788	37.25926	13.15205
Octal	mean	89.7	245.9	309.95	138.8	34.6
	stdev	29.64722	66.90205	80.58697	38.0714	13.82751
Hex	mean	84.85	260.45	334.95	153.75	36.9
	stdev	25.46675	69.1965	84.01533	44.61045	13.67248
Global	mean	91.75	251.05	304.85	143.35	34.85
	stdev	22.52922	50.1403	66.46826	34.12867	9.062793

Table 9-5 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.347811	0.330987	0.330607	0.340388	0.379104
Square	0.2774	0.245638	0.223054	0.239455	0.32196
Octal	0.330515	0.27207	0.26	0.27429	0.399639
Hex	0.300138	0.265681	0.250829	0.290149	0.370528
Global	0.24555	0.199722	0.218036	0.238079	0.260051

9.3 ZDT 3 30 with homogeneous Hyper-volume metrics

9.3.1 Performance of MOCAT Using the Hyper-Volume Metric

As shown in table 9-6 below, the hyper-volume metric guided system outperforms that of the spread metric slightly in terms of both mean error and standard deviation of the error. In fact, it now outperforms all of the other benchmarked functions including ANMOPSO.

The runs were complete in an average of 39.42 seconds.

Table 9-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	Mean	min	max	stdev
Run #1	4.37E-03	1.84E-02	8.16E-04	8.89E-02	1.94E-02
Run #2	2.81E-03	1.63E-02	6.72E-04	9.89E-02	2.17E-02
Run #3	7.74E-03	1.11E-02	2.97E-05	6.24E-02	1.29E-02
Run #4	3.43E-03	1.46E-02	4.70E-05	6.94E-02	1.82E-02
Run #5	9.09E-03	1.69E-02	7.49E-05	9.61E-02	2.33E-02
Run #6	9.24E-03	1.69E-02	4.16E-06	8.96E-02	2.06E-02
Run #7	4.08E-03	1.29E-02	5.10E-05	7.90E-02	1.43E-02
Run #8	4.21E-03	1.88E-02	2.69E-04	9.99E-02	2.31E-02
Run #9	9.17E-03	1.34E-02	2.25E-04	9.74E-02	1.62E-02
Run #10	2.47E-03	1.06E-02	5.77E-04	8.98E-02	1.63E-02
Run #11	9.89E-03	2.08E-02	4.45E-04	9.82E-02	2.57E-02
Run #12	4.44E-03	1.97E-02	1.54E-04	8.86E-02	2.25E-02
Run #13	4.47E-03	1.49E-02	1.66E-04	9.59E-02	2.03E-02
Run #14	8.34E-03	2.19E-02	1.47E-04	9.70E-02	2.55E-02
Run #15	1.06E-02	1.66E-02	7.81E-04	9.12E-02	1.87E-02
Run #16	4.46E-03	1.79E-02	3.41E-06	9.49E-02	2.23E-02
Run #17	3.55E-03	1.48E-02	9.18E-04	9.34E-02	2.08E-02
Run #18	3.93E-03	1.66E-02	1.35E-04	8.89E-02	2.16E-02
Run #19	7.31E-03	1.36E-02	9.62E-04	8.47E-02	1.54E-02
Run #20	6.53E-03	1.13E-02	1.27E-03	8.35E-02	1.68E-02
mean	6.01E-03	1.59E-02	3.87E-04	8.94E-02	1.98E-02
stdev	2.58E-03	3.10E-03	3.78E-04	9.57E-03	3.52E-03

9.3.2 Statistics of Topologies using Hyper-volume metric

While there was no statistical difference in usage between topologies for the spread metric, table 9-7 indicates that both the simplest (LBEST) and the most complex (Global) are the preferred choices here. The former is to conduct local search and latter is to link the segments together in the search.

Table 9-7 Use Count of Topologies of each run

	Using Hyper-volume metric	Hyper-volume
--	---------------------------	--------------

						Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	20	16	24	17	23	0.817267
Run #2	20	17	20	21	22	0.742551
Run #3	21	17	19	19	24	1.204697
Run #4	24	17	21	19	19	1.061088
Run #5	24	13	19	19	25	0.9809
Run #6	26	21	17	10	26	0.96152
Run #7	23	17	17	24	19	0.94163
Run #8	15	25	21	26	13	0.733853
Run #9	18	24	20	22	16	1.366974
Run #10	31	24	8	18	19	0.850537
Run #11	15	19	27	22	17	0.72768
Run #12	24	17	24	15	20	0.874815
Run #13	18	18	19	21	24	0.905912
Run #14	24	21	16	24	15	0.937269
Run #15	14	15	20	19	32	1.041747
Run #16	22	18	17	16	27	1.049134
Run #17	32	9	21	18	20	1.070652
Run #18	28	18	16	17	21	1.934245
Run #19	22	23	13	27	15	0.757881
Run #20	25	25	21	15	14	0.621279
mean	22.30	18.70	19.00	19.45	20.55	
stdev	4.80	4.05	4.00	3.98	4.78	
t-test	Lbest vs.	0.008506	0.013514	0.026869	0.133521	
	Square vs.		0.409779	0.284139	0.103039	
	Octal vs.			0.365038	0.142704	
	Hex vs.				0.222873	

9.3.3 Behaviors of Knowledge sources using Hyper-volume Metric

As with the spread metric the hyper-volume system uses the exploratory knowledge sources in the search more than in the previous problems. Also, domain knowledge controls the most individuals over the 20 runs as with the spread metric.

Table 9-8 Using Hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS

	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	437	1296	1547	738	189
Run #2	468	1288	1568	788	170
Run #3	423	1302	1571	782	167
Run #4	423	1373	1512	834	192
Run #5	420	1266	1598	840	214
Run #6	474	1318	1500	725	182
Run #7	363	1334	1555	847	166
Run #8	415	1336	1491	797	180
Run #9	500	1237	1605	776	199
Run #10	370	1359	1561	817	192
Run #11	417	1351	1523	834	197
Run #12	470	1275	1557	714	163
Run #13	455	1292	1646	757	225
Run #14	466	1345	1565	724	202
Run #15	447	1384	1526	815	241
Run #16	421	1286	1555	734	178
Run #17	434	1388	1517	697	203
Run #18	410	1346	1546	892	200
Run #19	445	1270	1568	784	191
Run #20	422	1366	1591	817	217
mean	434.00	1320.60	1555.10	785.60	193.40
stdev	32.62	42.47	36.80	50.99	19.99
t-test	KS-N vs.	1.59E-40	2.73E-47	3.48E-23	7.52E-24
	KS-S vs.		2.02E-20	3.27E-30	4.21E-37
	KS-D vs.			4.31E-35	1.93E-43
	KS-H vs.				5.1E-26

The values are similar to those in ZDT3 30 using pure spread metric.

9.3.4 Statistics of Topology-Knowledge tuple

MOCAT using the hyper-volume metric is able to produce more non-dominated solution with exploratory knowledge sources than the spread metric. Since the Pareto front is separated into segments this gives it a strong competitive edge. Domain Knowledge is still the dominant method for

generating non-dominated solutions regardless of topology. As shown in Table 9-10, no one topology exhibits the ability to vary its generation of non-dominated solutions markedly.

Table 9-9 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	94.35	294.55	350.55	177.7	40.1
	stdev	28.19346	72.5574	74.29633	40.95967	11.1256
Square	mean	84.2	245.35	290.95	146.8	39.1
	stdev	27.54441	54.24338	67.61071	33.30497	15.92053
Octal	mean	81.8	255	289.4	151	36.75
	stdev	26.22895	56.43534	65.03311	36.39549	12.16066
Hex	mean	80	258.25	306.3	150.5	36
	stdev	20.38059	58.49145	68.64255	33.63504	13.95104
Global	mean	93.65	267.45	317.9	159.6	41.45
	stdev	26.97811	63.75981	76.07192	42.08563	18.96111

Table 9-10 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.298818	0.246333	0.211942	0.230499	0.277446
Square	0.327131	0.221086	0.232379	0.226873	0.407175
Octal	0.320647	0.221315	0.224717	0.24103	0.330902
Hex	0.254757	0.226492	0.224102	0.223489	0.387529
Global	0.288074	0.238399	0.239295	0.263694	0.457445

9.3.5 Analysis of found Pareto front

The found front below still favors segments to the left as did the spread metric, but now the intervals are much more fleshed out with fewer gaps than before.

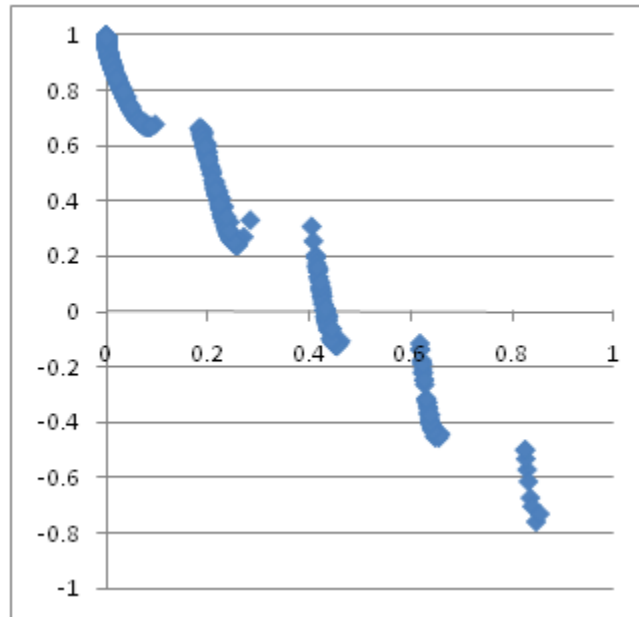


Figure 9.4 Overall found Pareto front

9.4 ZDT 3 30 with combined metrics

Configuration for this part of experiment is: population 100, generation 100, dimension 30, and every 25 generations Topology metrics are switched.

The runs were complete in an average of 33.96 seconds.

9.4.1 Performance of MOCAT with the Combine Metrics

The combined metrics again outperform either one separately as shown in Table 9-11 in terms of both mean error and standard deviation of the error. It again illustrates the synergy of combining the two metrics together in terms of controlling the knowledge sources. The question is how does the combination impact how each knowledge source conducts its segment?

Table 9-11 Ratios on the Belief Space roulette wheel before generation 20

Statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	8.62E-03	1.85E-02	2.90E-06	9.17E-02	2.31E-02
Run #2	4.68E-03	1.71E-02	6.20E-04	9.59E-02	2.44E-02
Run #3	6.03E-03	1.98E-02	2.15E-03	8.63E-02	1.93E-02
Run #4	9.19E-03	1.32E-02	8.50E-04	6.47E-02	1.24E-02
Run #5	1.01E-02	1.63E-02	3.07E-05	8.19E-02	1.91E-02
Run #6	3.50E-03	1.07E-02	2.35E-06	9.05E-02	1.40E-02
Run #7	2.91E-03	8.32E-03	9.56E-05	6.64E-02	9.98E-03
Run #8	4.89E-03	1.72E-02	8.95E-05	9.67E-02	1.84E-02
Run #9	6.12E-03	1.19E-02	1.86E-04	9.07E-02	1.77E-02
Run #10	8.55E-03	2.18E-02	7.73E-05	9.28E-02	2.74E-02
Run #11	6.94E-03	1.50E-02	1.16E-04	9.31E-02	2.14E-02
Run #12	4.61E-03	1.82E-02	1.13E-03	8.49E-02	2.11E-02
Run #13	7.75E-03	1.60E-02	1.45E-04	9.84E-02	1.94E-02
Run #14	4.47E-03	1.52E-02	6.89E-04	8.96E-02	1.68E-02
Run #15	7.47E-03	1.45E-02	4.13E-04	9.38E-02	2.06E-02
Run #16	6.56E-03	9.44E-03	1.58E-04	8.13E-02	1.28E-02
Run #17	7.68E-03	1.41E-02	5.21E-06	9.18E-02	1.80E-02
Run #18	5.76E-03	1.01E-02	5.90E-04	6.87E-02	1.42E-02
Run #19	5.79E-03	1.91E-02	3.06E-04	9.69E-02	2.46E-02
Run #20	3.45E-03	1.32E-02	1.10E-04	8.32E-02	1.69E-02
mean	6.25E-03	1.50E-02	3.88E-04	8.70E-02	1.86E-02
stdev	1.97E-03	3.57E-03	5.09E-04	9.81E-03	4.37E-03

9.4.2 Statistics of Topologies using for Both Metrics

Tables 9-12 and 9-13 give the usage of topologies for both metrics during their 50 generations. Recall that they take turns conducting the search every 26 generations. It is interesting to note that LBEST and Octal are dominant for the spread portion but for the hyper-volume portion it is LBEST and Square. In fact, Octal is significantly worse the LBEST there. This suggests that the two phases are working in complementary ways.

Table 9-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	8	8	19	8	7	1.344107
Run #2	10	9	13	5	13	1.804859
Run #3	12	12	7	8	11	1.631843
Run #4	8	8	14	12	8	1.75174
Run #5	11	8	11	8	12	1.497024
Run #6	13	6	12	13	6	1.464572
Run #7	17	16	5	6	6	1.600356
Run #8	14	7	12	7	10	1.817861
Run #9	11	8	9	8	14	1.396507
Run #10	12	7	11	9	11	0.95383
Run #11	11	11	3	9	16	0.655216
Run #12	14	5	11	12	8	3.33157
Run #13	8	6	9	18	9	1.915242
Run #14	8	6	17	12	7	1.931795
Run #15	4	12	10	12	12	1.768109
Run #16	11	7	14	8	10	1.087823
Run #17	12	11	11	7	9	1.392345
Run #18	8	13	12	10	7	1.713928
Run #19	10	12	9	9	10	1.566155
Run #20	14	9	11	8	8	1.588173
mean	10.80	9.05	11.00	9.45	9.70	
stdev	2.87	2.82	3.56	2.91	2.67	
t-test	Lbest vs.	0.032877	0.42502	0.079079	0.114448	
	Square vs.		0.034765	0.334595	0.234894	
	Octal vs.			0.075171	0.105656	
	Hex vs.				0.391921	

Table 9-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric					Hyper-volume metric value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	10	13	14	7	6	1.449694
Run #2	10	11	6	8	15	1.112144
Run #3	13	6	5	10	16	0.886865
Run #4	12	9	8	10	11	0.860081
Run #5	12	9	10	10	9	2.258497

Run #6	15	9	5	9	12	0.860638
Run #7	10	7	7	14	12	1.5608
Run #8	16	10	12	10	2	0.905942
Run #9	9	10	5	16	10	1.323813
Run #10	10	10	9	14	7	0.705549
Run #11	8	12	11	8	11	0.82653
Run #12	12	8	13	9	8	1.083261
Run #13	15	14	3	10	8	0.75213
Run #14	9	6	11	13	11	0.780946
Run #15	9	17	13	5	6	0.805511
Run #16	10	10	5	12	13	0.811523
Run #17	7	9	10	14	10	1.440028
Run #18	9	10	10	9	12	0.828039
Run #19	11	10	10	8	11	0.995826
Run #20	8	12	9	14	7	0.855525
mean	10.75	10.10	8.80	10.50	9.85	
stdev	2.43	2.57	3.09	2.80	3.23	
t-test	Lbest vs.	0.21375	0.018648	0.385191	0.169022	
	Square vs.		0.083466	0.324494	0.396578	
	Octal vs.			0.041915	0.156223	
	Hex vs.				0.255794	

9.4.3 Behavior of Knowledge sources with the two metrics

Again the complementary nature of the two phases is visible in tables 9-14 and 9-15 below. The spread metric controls more individuals with the normative and topographic exploratory metrics than does the hyper-volume metric. The hyper-volume makes more use of the history knowledge sources to control search. As said earlier, this is a way of transferring learned knowledge from one phase to the next.

Table 9-14 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	286	633	758	395	99
Run #2	237	626	852	355	98

Run #3	301	601	829	355	102
Run #4	254	644	788	357	104
Run #5	237	673	773	398	107
Run #6	261	567	876	301	86
Run #7	270	662	786	353	115
Run #8	296	587	837	290	93
Run #9	218	676	823	345	95
Run #10	263	623	766	344	96
Run #11	268	627	855	371	119
Run #12	298	628	846	307	109
Run #13	259	656	792	422	137
Run #14	307	575	852	310	110
Run #15	251	610	846	266	92
Run #16	254	600	849	409	114
Run #17	310	578	846	301	92
Run #18	249	572	917	283	88
Run #19	274	652	803	324	124
Run #20	251	676	783	367	94
mean	267.20	623.30	823.85	342.65	103.70
stdev	24.82	35.14	40.22	42.94	12.85
t-test	KS-N vs.	4.29E-29	2.45E-32	1.13E-07	1.8E-21
	KS-S vs.		6.43E-19	5.16E-23	4.76E-28
	KS-D vs.			5.03E-31	4.76E-29
	KS-H vs.				1.84E-17

Table 9-15 Using Hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	160	639	783	394	68
Run #2	188	656	788	369	74
Run #3	209	610	771	414	92
Run #4	145	660	769	351	86
Run #5	173	687	747	427	89
Run #6	178	633	764	320	52
Run #7	148	690	739	424	89
Run #8	183	660	765	349	55
Run #9	161	670	782	445	92
Run #10	166	698	775	464	89

Run #11	140	691	762	400	83
Run #12	162	645	798	406	69
Run #13	179	672	782	444	106
Run #14	169	666	815	381	69
Run #15	159	673	779	304	59
Run #16	176	635	781	421	101
Run #17	170	624	824	367	98
Run #18	152	665	836	382	87
Run #19	169	666	790	394	117
Run #20	177	670	709	442	84
mean	168.20	660.50	777.95	394.90	82.95
stdev	15.66	22.88	27.90	41.66	16.66
t-test	KS-N vs.	9.53E-40	3.2E-37	6.19E-18	5.82E-19
	KS-S vs.		9.39E-17	2.14E-21	7.3E-43
	KS-D vs.			2.19E-27	7.34E-40
	KS-H vs.				1.72E-21

9.4.4 Statistics of Topology-Knowledge tuple

As shown in Table 9-16 and 9-17 the dominant topology in producing non-dominated solutions is LBEST. This is to be expected since both metrics on their own favored this topology.

The system demonstrates increased ability to generate new solutions in good areas for both of the exploratory knowledge sources.

Table 9-16 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	91.4	272.45	345.05	160.65	40.65
	stdev	27.51727	49.31368	62.73962	42.19164	11.95727
Square	mean	84.5	247.95	301.15	142.3	36.25
	stdev	26.30889	53.9439	64.45747	32.20183	12.28124
Octal	mean	94.85	252	320.4	143.8	35.75
	stdev	32.80609	63.05303	92.60067	43.33298	13.00152
Hex	mean	87.15	257.2	321	143.3	39.65
	stdev	20.57151	51.1135	56.03007	38.52559	13.58899
Global	mean	77.5	254.2	314.2	147.5	34.35
	stdev	19.50304	62.64738	75.2173	39.29979	8.579504

Table 9-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.301064	0.181001	0.181828	0.262631	0.294152
Square	0.311348	0.21756	0.214038	0.226295	0.338793
Octal	0.345873	0.25021	0.289016	0.301342	0.363679
Hex	0.236047	0.198731	0.174548	0.268846	0.342723
Global	0.251652	0.246449	0.239393	0.266439	0.249767

9.4.5 Analysis of found Pareto front

The found optimal front exhibits small gaps only in the last segment, much improved over either metric separately.

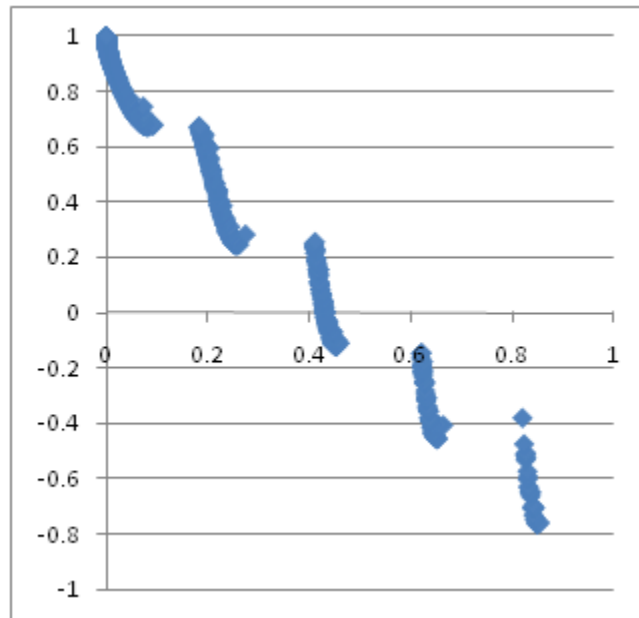


Figure 9.5 Overall found Pareto front

9.5 Summary of ZDT3

ZDT3 has a special Pareto front which consists of five segments which are distant from each other, which does impose big challenges to MOCAT in that fitness errors are much larger than those in

previous two sections. However, both the hyper-volume system and the combined systems outperform all of the other benchmarked systems for this problem. The combined system outperforms the hyper-volume system as well. The curves produced by the combined system exhibit fewer gaps as well.

Some basic conclusions are:

1. For this problem there has emerged a clear distinction between local and global search. Both the global and best topologies are favored by the combined system.
2. Also, there is an increased emphasis on the exploratory knowledge sources on the one hand and the domain knowledge source on the other.
3. History knowledge again plays a fundamental role in connecting the activities of the two metrics together in the combined system.

CHAPTER 10 USING MOCAT TO SOLVE MULTI-MODAL PROBLEMS: ZDT4

At the first glance, ZDT4 is the same as ZDT1. However, they are fundamentally different. While for ZDT1 the real Pareto front is a continuous curve, for ZDT4 the real one contains 21^9 local Pareto fronts. This problem combines aspects of all three previous problems into one problem. It has been observed previously that as the Pareto front has become more complex the MOCAT system has been able to adjust how it uses the knowledge sources and distributes their influences using the topologies. While the other approaches relied primarily on the spread metrics to control search, in MOCAT knowledge from the search is digested into the knowledge sources and used to guide the search as well. This provides a powerful mechanism for generating the Optimal Pareto front. Also, it is clear that the combined metrics can combine their influences in order to produce an overall better result than either metric by itself. It will be interesting to see whether this trend continues for this more complex problem.

We will still use homogeneous spread metric, homogeneous hyper-volume metric, and combined two metrics to conduct experiments and try to observe and summarize the correspondence among knowledge sources and topologies.

10.1 ZDT 4 10 with homogeneous Spread metrics

In ZDT4 the function is similar to ZDT1 but now there are a number of locally optimal fronts. In order to achieve the global optimal front the system will need to connect together the various local fronts.

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 10 * (n - 1) + \sum_{i=2}^m (x_i^2 - 10 * \cos(4 * \pi * x_i))$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g}\right)^2$$

$$f_2 = g * h$$

where $m=30$ and $0 \leq x_1 \leq 1$ and $x_i \in [-5, 5]$ for $i = 2, \dots, m$.

The Pareto front contains 21^9 local Pareto fronts and the overall curve is described as:

$$x_2 = 1 - \sqrt{x_1}$$

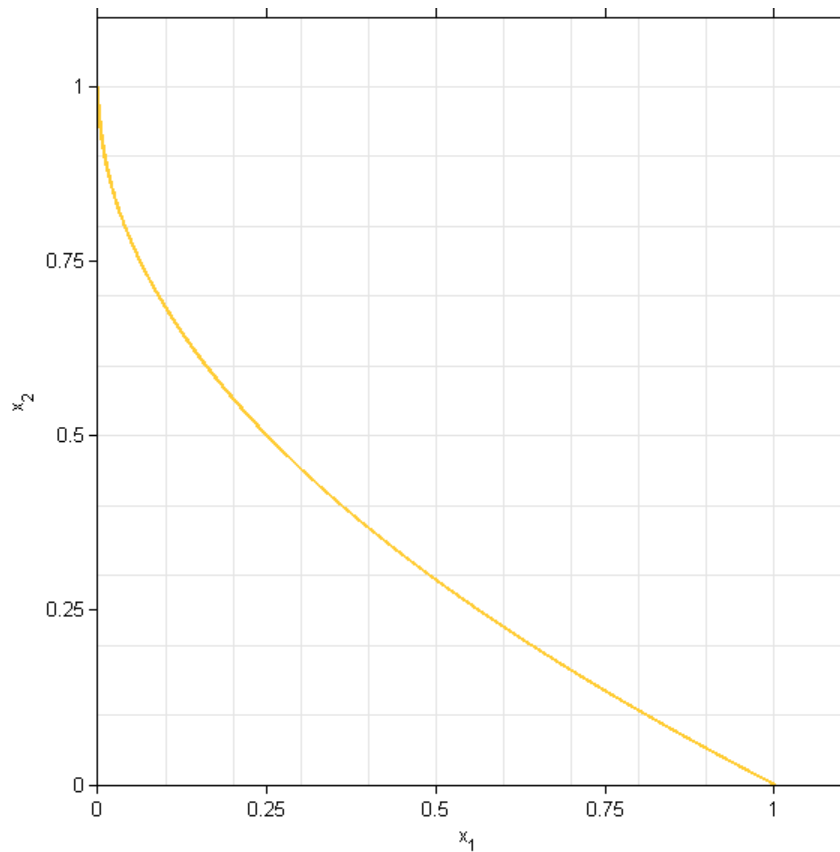


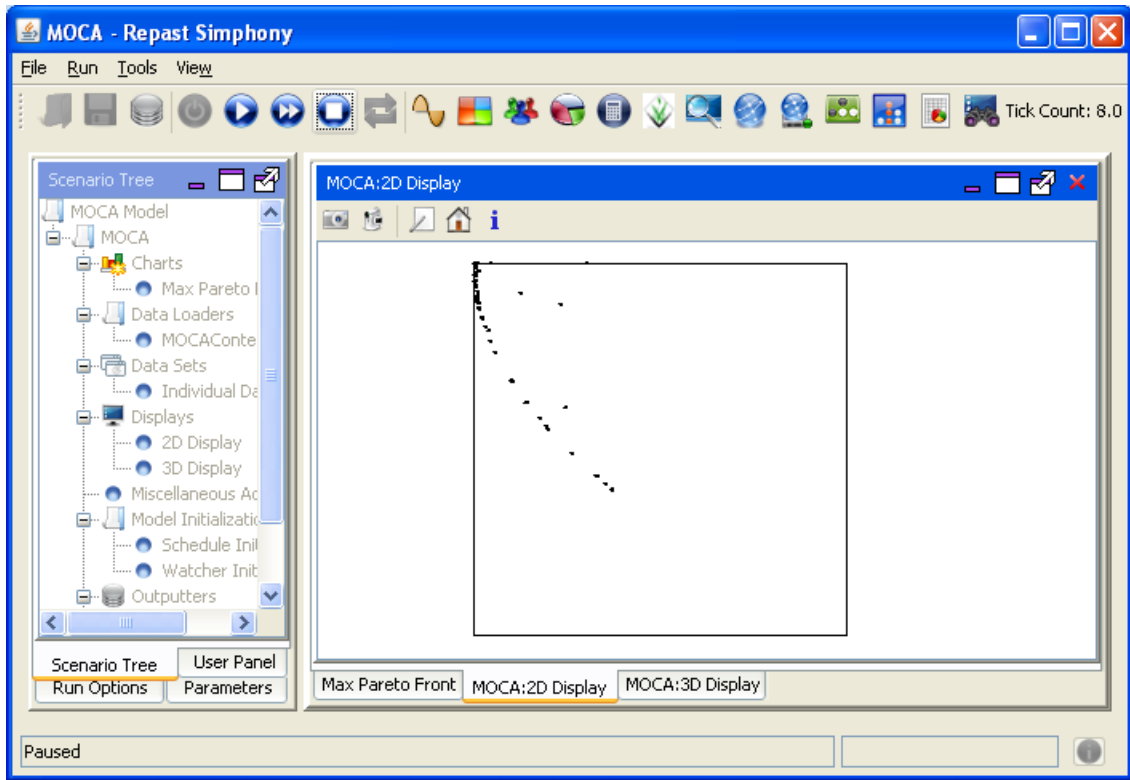
Figure 10.1 ZDT4 Pareto front

The Pareto front shown in Figure 10.1 for ZDT4 looks like the one of ZDT1, however, they are fundamentally different. While the Pareto front of ZDT1 is continuous, there the Pareto front of ZDT4 contains 2^{19} local Pareto fronts. This means that if an algorithm cannot jump out of local neighborhood it will be stuck in a small area. As such, it requires all of the knowledge needed to solve each of the first three problems in order for this one to be effectively solved.

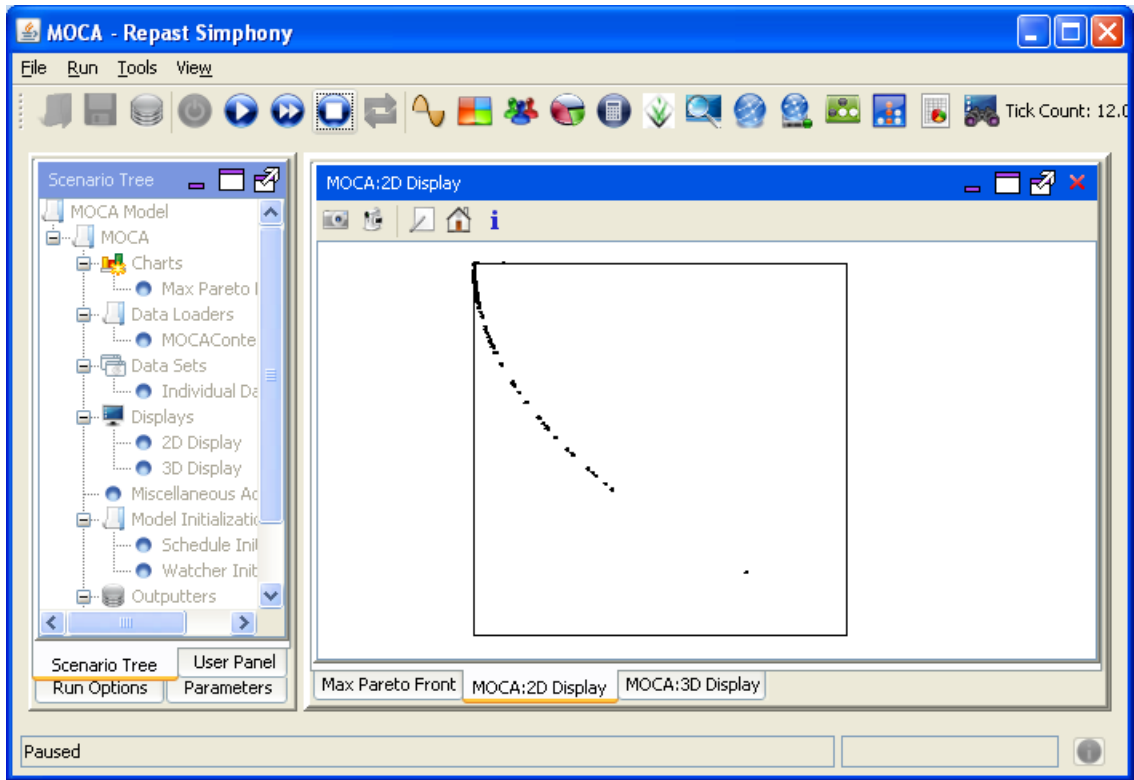
The runs were complete in an average of 14.38 seconds.

10.1.1 Evolving the Pareto Front

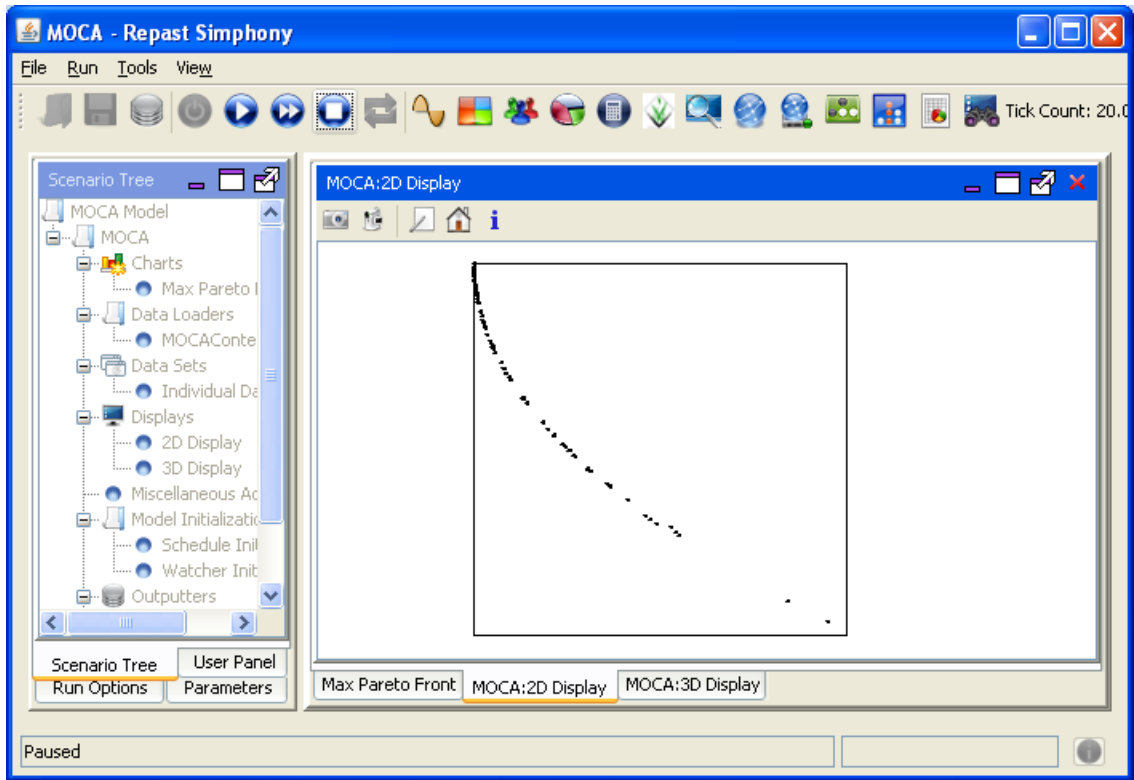
The following sequence of figures 10-a-e are taken from one run of the MOCAT system using the spread metric by which we can see that the evolution is similar to ZDT1 but a remarkable difference is that for ZDT4 at the beginning of evolution individuals do not spread to the far right bottom. Instead, they are attracted there during the evolution process.



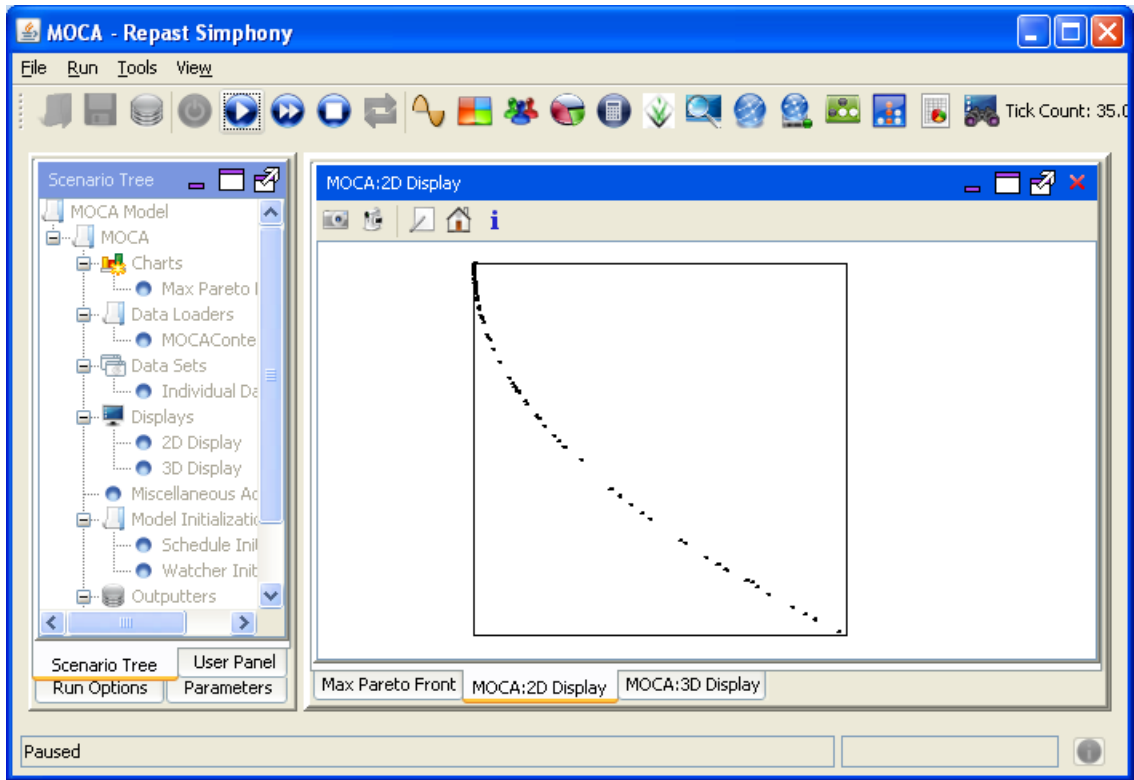
(a)



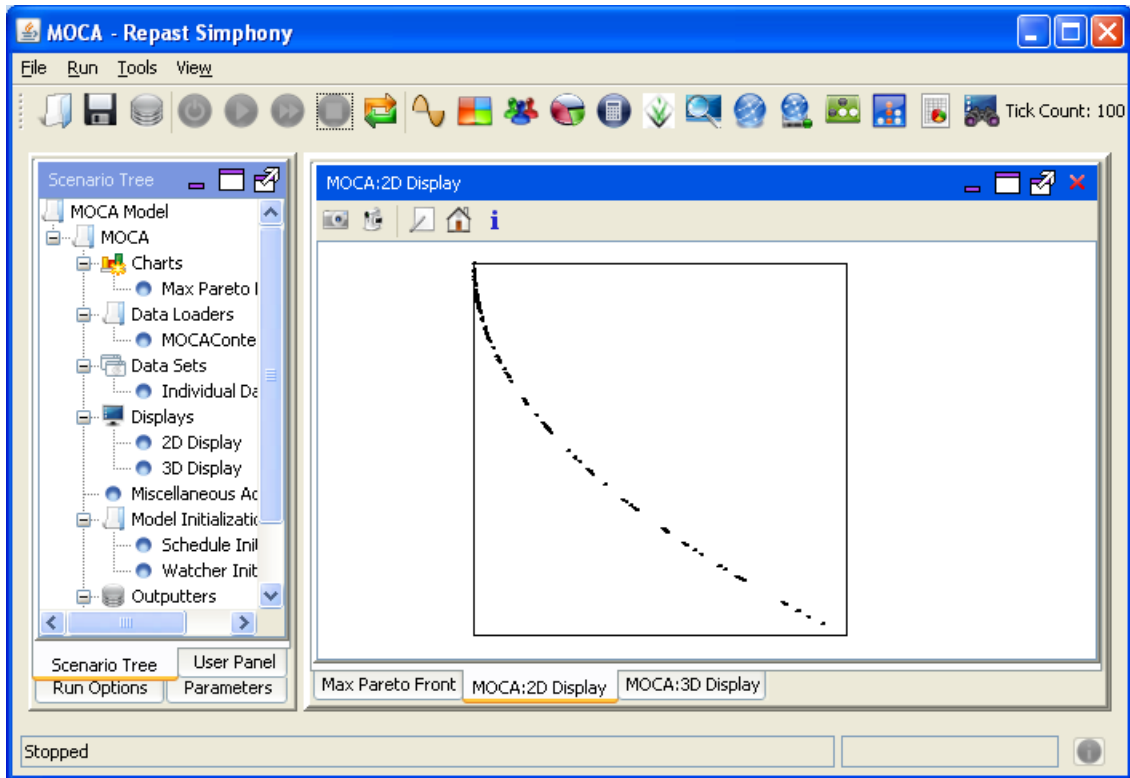
(b)



(c)



(d)



(e)

Figure 10.2 A series of screen copies along evolution (a-e)

10.1.2 Performance of MOCAT Using the Spread Metric

In Table 10-1 the performance statistics for MOCAT are given. The mean and standard deviation for the MOCAT system with the spread metric outperforms the benchmark results for all of the other systems. In fact, the mean error and the standard deviation are the least four all four problems that we have examined so far.

Table 10-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	1.79E-06	1.26E-05	2.28E-14	1.87E-04	2.47E-05

Run #2	3.71E-07	1.49E-06	-5.55E-17	1.31E-05	2.38E-06
Run #3	3.14E-07	5.08E-06	0	6.51E-05	1.08E-05
Run #4	3.07E-06	2.57E-05	-3.47E-17	2.92E-04	4.67E-05
Run #5	1.18E-06	1.20E-05	-2.78E-17	1.95E-04	3.02E-05
Run #6	1.63E-07	3.34E-06	0	1.03E-04	1.35E-05
Run #7	3.25E-08	1.80E-04	0	0.003007	4.74E-04
Run #8	4.28E-07	6.91E-06	-3.47E-17	8.24E-05	1.31E-05
Run #9	1.09E-08	3.07E-08	1.14E-13	1.55E-07	3.10E-08
Run #10	1.57E-07	2.64E-06	0	6.34E-05	8.10E-06
Run #11	3.83E-06	2.03E-05	-5.55E-17	2.69E-04	3.98E-05
Run #12	7.17E-07	3.20E-06	-5.55E-17	2.16E-05	4.31E-06
Run #13	1.70E-09	8.04E-09	-3.82E-17	6.71E-08	1.29E-08
Run #14	6.77E-09	3.48E-08	0	4.85E-07	6.54E-08
Run #15	4.69E-07	7.99E-06	3.01E-14	1.24E-04	2.11E-05
Run #16	1.02E-06	4.30E-05	1.01E-13	0.001302	1.85E-04
Run #17	4.05E-07	1.72E-06	0	1.22E-05	2.34E-06
Run #18	4.31E-06	4.86E-05	-2.78E-17	8.52E-04	1.10E-04
Run #19	1.17E-06	4.53E-06	0	2.94E-05	5.82E-06
Run #20	5.94E-07	2.31E-05	-2.78E-17	5.94E-04	8.23E-05
mean	1.00E-06	2.01E-05	1.34E-14	3.61E-04	5.37E-05
Stdev	1.25E-06	3.91E-05	3.24E-14	6.89E-04	1.06E-04

10.1.3 Statistics of Topologies using Spread metric

In terms of the usage of topologies over the runs Table 10-2 indicates that there is no significant difference in usage of the topologies with the spread metric.

Table 10-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	19	18	20	28	15	1.304667
Run #2	16	24	25	15	20	2.478954
Run #3	27	23	19	18	13	1.289089
Run #4	23	19	15	19	24	2.04255
Run #5	26	22	13	18	21	1.090334
Run #6	21	13	20	21	25	0.936189

Run #7	22	23	18	13	23	2.435693
Run #8	18	19	25	16	22	0.995736
Run #9	23	25	19	17	16	1.095636
Run #10	20	18	19	22	21	1.205241
Run #11	18	22	15	17	27	1.154544
Run #12	18	19	18	18	25	1.134337
Run #13	18	15	20	19	27	1.316264
Run #14	17	23	27	15	18	2.907291
Run #15	16	22	24	23	14	0.945172
Run #16	23	23	14	20	17	1.418885
Run #17	21	21	15	24	19	1.042119
Run #18	11	20	32	15	22	1.572977
Run #19	16	25	20	17	22	1.319642
Run #20	17	16	27	21	19	1.027526
mean	19.50	20.50	20.25	18.80	20.50	
stdev	3.72	3.25	4.89	3.52	4.02	
t-test	Lbest vs.	0.191596	0.298962	0.277362	0.215554	
	Square vs.		0.426909	0.064963	0.5	
	Octal vs.			0.150554	0.432106	
	Hex vs.				0.086712	

10.1.4 Behaviors of Knowledge sources using Spread Metrics

Table 10-3 gives the extent to which the different knowledge sources control individuals in the population. Notice that the MOCAT system uses the normative exploratory knowledge source more than for previous problems. This is also true for history knowledge. It is suggested that these two knowledge sources are necessary to make the leap between the different parts of the Pareto front

Table 10-3 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	293	1487	1420	1102	67
Run #2	261	1383	1434	1116	80
Run #3	278	1349	1502	975	60
Run #4	290	1395	1425	1148	68
Run #5	296	1465	1437	1102	58

Run #6	262	1317	1562	863	60
Run #7	296	1420	1464	898	86
Run #8	259	1379	1504	917	61
Run #9	303	1455	1445	1146	72
Run #10	283	1440	1439	1094	60
Run #11	290	1376	1493	875	75
Run #12	264	1407	1480	1039	61
Run #13	317	1393	1465	1171	67
Run #14	250	1439	1500	987	73
Run #15	270	1463	1433	1087	84
Run #16	256	1407	1518	1102	66
Run #17	283	1403	1412	1108	79
Run #18	284	1423	1494	976	77
Run #19	271	1404	1514	931	41
Run #20	308	1397	1421	1066	71
mean	280.70	1410.10	1468.10	1035.15	68.30
stdev	18.21	39.87	40.38	95.96	10.36
t-test	KS-N vs.	1.95E-37	1.08E-37	1.23E-19	2.88E-29
	KS-S vs.		3.58E-05	6.77E-15	8.05E-34
	KS-D vs.			2.15E-16	5.03E-34
	KS-H vs.				3.55E-21

10.1.5 Statistics of Topology-Knowledge tuple

While there was no significant difference in the usage of the topologies it is clear from table 10-4 that the Global topologies is clearly the best at generating non-dominated solutions over the 20 runs. The global domain knowledge tuple is the most successful in generating non-dominated solutions here. Topographic knowledge exhibits the most variability in the generation of the non-dominated solutions. The found Pareto front is shown in Figure in 10-6.

Table 10-4 Overall Statistics of Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	53.65	271.65	287.25	200.3	10.1
	stdev	13.04355	54.8455	58.0534	50.79484	6.835665
Square	mean	56.3	286.95	298.1	210.15	11.35

	stdev	16.86822	54.39086	52.3439	41.22375	6.037384
Octal	mean	50.8	292.65	294.65	206.7	10.55
	stdev	16.456	69.04634	80.26946	47.55728	6.492708
Hex	mean	55.45	260.7	274.55	201.05	13.35
	stdev	18.17886	51.99706	62.30103	46.43102	8.934116
Global	mean	64.5	298.15	313.55	216.95	22.95
	stdev	22.38773	59.87291	66.46366	46.09884	13.70929

Table 10-5 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.243123	0.201898	0.202101	0.253594	0.676799
Square	0.299613	0.189548	0.175592	0.196163	0.531928
Octal	0.323937	0.235935	0.272423	0.230079	0.615423
Hex	0.327842	0.199452	0.226921	0.230943	0.669222
Global	0.347097	0.200815	0.211971	0.212486	0.597355

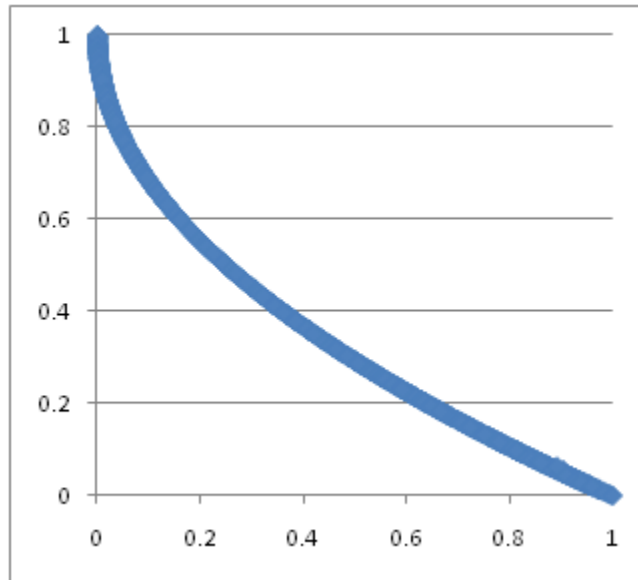


Figure 10.3 Overall found Pareto front

10.2 ZDT 4 10 with homogeneous Hyper-volume metric

This problem is similar to ZDT1 some similarities can be noticed. The spread metric tends to do well with problems of convex search. So it is not surprising that it outperforms the hyper-volume metric controlled MOCAT here.

The runs were complete in an average of 15.12 seconds.

10.2.1 Performance of MOCAT with the Hyper-Volume Metric

The hyper-volume controlled MOCAT outperformed all of the benchmarked function for this problem but still trailed the spread metric version in performance here. This confirms our previous understanding that the spread metric provides better control in convex search problems than does the hyper-volume metric. However, we will also find that as before, the hyper-volume metric excels at local search, thereby reducing the standard deviation of the error. This suggests that again their combined system will exhibit the best of both.

Table 10-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	Mean	min	max	Stdev
Run #1	4.79E-06	9.23E-06	0	7.06E-05	1.24E-05
Run #2	1.71E-06	1.80E-05	2.46E-14	3.17E-04	3.93E-05
Run #3	5.72E-10	4.21E-04	-5.55E-17	0.027293	0.002928
Run #4	7.95E-07	9.20E-06	-4.16E-17	3.00E-04	3.23E-05
Run #5	1.55E-05	6.62E-05	8.10E-12	6.18E-04	1.03E-04
Run #6	2.51E-06	1.79E-05	0	2.20E-04	3.06E-05
Run #7	1.45E-07	8.30E-07	0	9.97E-06	1.44E-06
Run #8	4.38E-08	2.26E-07	-5.20E-17	2.46E-06	3.62E-07
Run #9	1.81E-06	1.98E-05	-4.16E-17	1.80E-04	3.50E-05
Run #10	3.60E-08	1.32E-07	-5.55E-17	9.37E-07	1.80E-07
Run #11	1.55E-05	7.39E-05	0	4.95E-04	9.89E-05

Run #12	3.86E-06	1.24E-05	2.24E-11	7.43E-05	1.56E-05
Run #13	3.05E-08	6.90E-07	1.65E-14	2.87E-05	3.16E-06
Run #14	1.37E-08	2.93E-07	-2.78E-17	6.47E-06	8.02E-07
Run #15	2.77E-06	8.08E-05	5.63E-14	0.001074	1.71E-04
Run #16	1.18E-05	2.93E-05	0	1.48E-04	3.66E-05
Run #17	1.55E-08	2.54E-06	-5.07E-17	5.15E-05	8.67E-06
Run #18	4.87E-07	1.53E-06	-2.78E-17	1.10E-05	1.83E-06
Run #19	2.96E-10	1.59E-09	0	1.16E-08	2.06E-09
Run #20	1.50E-06	7.61E-06	0	6.93E-05	1.19E-05
mean	3.17E-06	3.86E-05	1.53E-12	1.55E-03	1.77E-04
stdev	4.91E-06	9.12E-05	5.11E-12	5.91E-03	6.33E-04

10.2.2 Statistics of Topologies using Hyper-volume metric

Table 10-7 Use Count of Topologies of each run

	Using Hyper-volume metric					Hyper-volume Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	34	25	14	19	8	2.59969
Run #2	7	35	34	21	3	2.816862
Run #3	22	23	13	12	30	1.153938
Run #4	25	6	6	33	30	2.16683
Run #5	16	9	23	16	36	2.807655
Run #6	9	41	1	25	24	1.311071
Run #7	26	21	17	27	9	2.625014
Run #8	13	23	27	25	12	1.103295
Run #9	36	4	19	13	28	2.732501
Run #10	35	31	10	20	4	4.489101
Run #11	19	2	18	14	47	2.723188
Run #12	10	19	48	14	9	1.870899
Run #13	16	8	23	19	34	2.631353
Run #14	16	22	8	27	27	2.565681
Run #15	5	18	30	35	12	14.05727
Run #16	30	13	19	18	20	3.435748
Run #17	20	12	24	15	29	1.944189
Run #18	38	10	23	22	7	3.322236
Run #19	7	35	1	22	35	7.289603
Run #20	30	22	6	19	23	4.763994

mean	20.70	18.95	18.20	20.80	21.35
stdev	10.34	10.73	11.33	6.21	12.28
t-test	Lbest vs.	0.305805	0.240853	0.485701	0.43044
	Square vs.		0.417557	0.260079	0.262534
	Octal vs.			0.193667	0.208148
	Hex vs.				0.431476

No statistical difference between any pair of topologies can be found. Once again, different topologies were selected with similar possibilities. The cause may be the high standard deviation: the highest and lowest means are corresponding to the two highest standard deviations.

10.2.3 Behaviors of Knowledge sources using Hyper-volume Metric

As with the spread metric , the system favors an increase in the use of exploratory knowledge sources, particularly the normative knowledge source. At the same time the history knowledge source is used extensively as a means to link the the parts of the global curve.

Table 10-8 Using hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	282	1380	1474	1057	62
Run #2	299	1406	1460	1317	88
Run #3	279	1407	1462	1083	69
Run #4	273	1418	1506	1031	76
Run #5	278	1475	1458	1017	80
Run #6	265	1437	1391	1042	69
Run #7	270	1447	1442	1169	84
Run #8	315	1436	1461	1115	72
Run #9	294	1418	1432	1142	73
Run #10	261	1445	1438	1105	64
Run #11	296	1410	1490	953	87
Run #12	299	1462	1447	1181	63
Run #13	259	1412	1428	1192	83
Run #14	330	1464	1373	953	58

Run #15	328	1350	1544	1083	64
Run #16	299	1371	1479	1150	55
Run #17	263	1378	1517	1105	57
Run #18	313	1405	1428	1191	81
Run #19	304	1410	1391	1153	88
Run #20	272	1402	1541	975	79
mean	288.95	1416.65	1458.10	1100.70	72.60
stdev	21.41	31.82	45.43	89.10	10.59
t-test	KS-N vs.	1.02E-46	9.4E-37	2E-21	3.02E-26
	KS-S vs.		0.001275	1.23E-13	6.11E-38
	KS-D vs.			1.06E-15	2.42E-32
	KS-H vs.				2.21E-22

10.2.4 Statistics of Topology-Knowledge tuple

Here in Table 10-9 the Hexagonal Social fabric is best at generating non-dominated solutions for the exploitative knowledge sources (situational, domain, and history) where the global topology does best at generating non-dominated solution for the exploratory knowledge sources. Again, this indicates the ability of the MOCAT system to conduct search at different levels of granularity, both the local and global level. Table 10-10 indicates that system is most variable in generating topographic solutions.

Table 10-9 Number of non-dominated solutions generated by Topology-Knowledge tuples

		N	S	D	H	T
Lbest	mean	57.55	288.9	304.85	225.1	15.75
	stdev	44.4942	142.3328	162.934	98.7852	16.16649
Square	mean	57.35	274.2	269.15	218.7	13.15
	stdev	40.49987	156.8075	156.4334	118.9118	13.6122
Octal	mean	50.15	272.6	271.4	212.4	13.25
	stdev	39.67005	162.8672	169.0803	126.9627	13.81409
Hex	mean	60.6	301.05	309.6	235.05	12.7
	stdev	36.0882	87.87699	105.1247	54.49334	9.841373
Global	mean	64	282.85	298.9	211.2	17.5
	stdev	54.67223	187.8609	204.828	123.1151	17.82813

Table 10-10 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.77314	0.492671	0.534473	0.43885	1.026444
Square	0.706188	0.571873	0.581213	0.543721	1.035149
Octal	0.791028	0.597459	0.622993	0.597753	1.042572
Hex	0.595515	0.291902	0.33955	0.231837	0.774911
Global	0.854254	0.664171	0.685273	0.582931	1.01875

10.2.5 Analysis of found Pareto front

The overall found pareto front does not contain any major gaps for this problem.

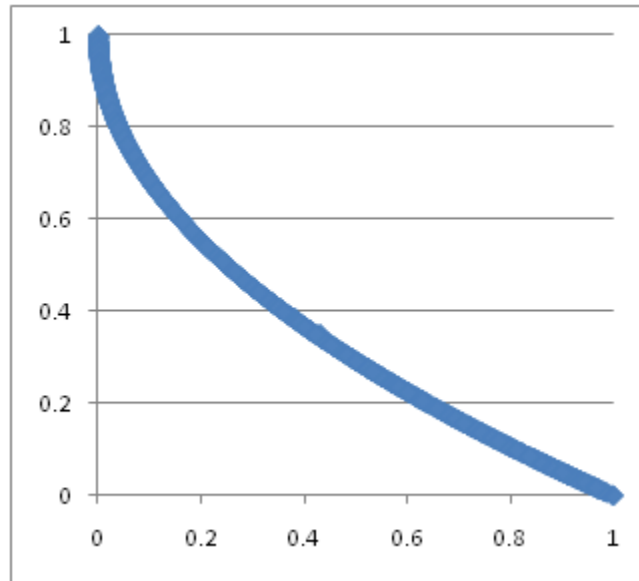


Figure 10.4 Overall found Pareto front

10.3 ZDT 4 10 with combined metrics

Although for the first convex function that we looked at the spread metric by itself performed well, here the performance of the combined system far exceeds the perform of either system. This is because as mentioned earlier it is able to integrate both local and global search and to exchange that information between control segments during the search process.

The runs were complete in an average of 16.12 seconds.

10.3.1 Performance of MOCAT with the Combined metric

As shown in Table 10-11 the performance of the combined system far exceeds that of either single metric guided system. In addition, it outperforms the benchmarked systems as well.

While we have seen the advantage of the combined metric approach for every problem so far, as the problems have increased in complexity from ZDT1 through ZDT4 the relative performance of the combined system have increased dramatically. Not only is it a better solution than the other systems, it is actually the most accurate of all solved fronts so far.

Table 10-11 Statistics for the fitness errors of ending solutions

Statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	1.57E-07	1.46E-06	0	2.12E-05	3.63E-06
Run #2	7.36E-07	6.41E-06	0	7.19E-05	1.27E-05
Run #3	3.45E-07	6.22E-06	-6.94E-18	1.54E-04	2.16E-05
Run #4	9.14E-07	4.64E-06	0	8.37E-05	9.45E-06
Run #5	1.37E-07	5.31E-07	0	4.80E-06	8.87E-07
Run #6	1.05E-07	3.21E-07	-2.78E-17	3.87E-06	6.15E-07
Run #7	1.59E-06	7.26E-06	-5.55E-17	4.54E-05	9.58E-06
Run #8	7.14E-07	2.54E-06	-5.55E-17	2.39E-05	3.52E-06
Run #9	2.37E-07	1.59E-06	-5.55E-17	2.97E-05	4.22E-06
Run #10	2.08E-06	1.05E-05	3.10E-13	1.03E-04	1.80E-05
Run #11	2.94E-06	8.58E-06	7.49E-13	5.01E-05	8.79E-06
Run #12	1.06E-07	2.59E-07	-5.55E-17	3.57E-06	5.34E-07
Run #13	8.22E-08	9.49E-07	8.73E-14	9.66E-06	1.95E-06
Run #14	8.78E-07	5.10E-06	-2.78E-17	5.01E-05	8.75E-06
Run #15	2.14E-08	4.70E-07	1.04E-13	9.22E-06	1.30E-06
Run #16	5.98E-07	2.94E-06	1.96E-11	2.99E-05	4.46E-06
Run #17	2.74E-08	1.36E-07	1.85E-13	1.07E-06	1.92E-07
Run #18	9.22E-07	2.40E-05	1.48E-14	3.35E-04	5.99E-05
Run #19	1.09E-08	5.90E-08	2.72E-14	4.55E-07	8.68E-08

Run #20	1.22E-06	5.43E-05	2.57E-10	0.001952	2.57E-04
mean	6.91E-07	6.92E-06	1.39E-11	1.49E-04	2.14E-05
stdev	7.60E-07	1.22E-05	5.60E-11	4.20E-04	5.57E-05

10.3.2 Statistics of Topologies for the Two Metrics

The usage of the topologies in each of the two phases of the combined metric system is given in Figures 10-12 and 10-13. For the spread metric, there is no significant different in the amount of usage among them. For the hyper-volume metric the usage of the square topology is significantly more frequent than that of LBEST or HEX. This suggests a more mid-grained knowledge distribution strategy, one that values between very local and global.

Table 10-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	9	9	13	5	14	1.004007
Run #2	3	12	11	10	14	7.554531
Run #3	11	10	11	5	13	1.207365
Run #4	13	10	9	11	7	1.181344
Run #5	12	3	7	8	20	1.116746
Run #6	2	7	14	11	16	0.916237
Run #7	13	3	8	18	8	1.154282
Run #8	1	8	25	12	4	5.225867
Run #9	5	10	15	11	9	4.892349
Run #10	12	15	7	6	10	1.097043
Run #11	10	9	17	6	8	1.050697
Run #12	8	12	3	8	19	7.583102
Run #13	14	14	11	6	5	1.066084
Run #14	9	3	9	18	11	1.245117
Run #15	14	15	4	14	3	1.062168
Run #16	14	10	6	7	13	7.774937
Run #17	15	15	8	6	6	1.072286
Run #18	9	10	10	8	13	1.857044
Run #19	15	2	13	8	12	2.155665

Run #20	8	13	14	7	8	0.996331
mean	9.85	9.50	10.75	9.25	10.65	
stdev	4.21	4.06	4.85	3.79	4.59	
t-test	Lbest vs.	0.397767	0.272431	0.323549	0.289345	
	Square vs.		0.197083	0.422738	0.209035	
	Octal vs.			0.14757	0.474127	
	Hex vs.				0.155932	

Table 10-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric					Hyper-volume metric value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	10	13	7	14	6	2.47357
Run #2	16	11	13	1	9	11.75539
Run #3	5	6	12	15	12	17.14695
Run #4	3	13	3	8	23	9.25986
Run #5	6	14	21	8	1	25.66597
Run #6	11	12	16	5	6	4.45085
Run #7	14	19	11	1	5	5.19906
Run #8	4	11	10	11	14	6.69113
Run #9	5	16	18	5	6	27.42015
Run #10	6	6	15	13	10	3.15135
Run #11	5	17	3	14	11	2.68119
Run #12	12	16	1	11	10	5.90485
Run #13	4	5	12	12	17	1.51482
Run #14	15	13	1	9	12	7.70107
Run #15	13	7	22	2	6	1.14883
Run #16	11	21	3	9	6	4.37885
Run #17	13	9	16	10	2	8.92897
Run #18	8	4	9	9	20	1.98441
Run #19	14	18	14	3	1	6.834429
Run #20	12	10	6	4	18	2.994
mean	9.35	12.05	10.65	8.20	9.75	
stdev	4.15	4.77	6.27	4.35	6.07	
t-test	Lbest vs.	0.03529	0.228248	0.204951	0.407042	
	Square vs.		0.221871	0.006643	0.101233	
	Octal vs.			0.085483	0.327862	
	Hex vs.				0.186116	

10.3.3 Behaviors of Knowledge sources for the combined metrics

The extent to which the knowledge sources control the population in each phase of the search process is given in tables 10-14 and 10-15. As was to be expected both spread metrics maintain significant differences in knowledge source control in their specific phases of the problem solving process. What is interesting is the complementarity that is displayed there. The spread metric phase makes more use of the exploratory knowledge sources than the hyper-volume phases. On the other hand, the hyper-volume phase makes more use of history knowledge. This again makes sense because it is the vehicle used to pass information from one phase of the process to the next.

Table 10-14 using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	159	728	730	448	33
Run #2	178	681	757	466	53
Run #3	177	696	731	514	48
Run #4	202	711	757	506	53
Run #5	144	726	768	480	40
Run #6	143	669	770	475	49
Run #7	149	650	753	486	45
Run #8	149	760	646	562	40
Run #9	187	714	695	503	42
Run #10	146	667	799	353	51
Run #11	149	687	783	386	38
Run #12	200	733	770	470	56
Run #13	194	692	700	462	59
Run #14	180	668	753	498	41
Run #15	180	678	737	491	42
Run #16	203	711	714	511	46
Run #17	176	642	776	509	65
Run #18	190	731	730	527	63
Run #19	202	670	759	454	60
Run #20	174	698	751	597	42

mean	174.10	695.60	743.95	484.90	48.30
stdev	20.92	30.08	34.44	52.01	8.73
t-test	KS-N vs.	8.66E-37	1.39E-34	3.76E-19	2.27E-19
	KS-S vs.		2.31E-05	3.99E-16	3.06E-30
	KS-D vs.			5.35E-19	6.2E-29
	KS-H vs.				4.96E-20

Table 10-15 Using Hyper-volume metric #Individuals influenced by KS

using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	105	730	667	593	19
Run #2	123	700	750	618	10
Run #3	97	737	734	580	32
Run #4	101	698	690	616	15
Run #5	104	680	681	579	20
Run #6	112	717	682	624	21
Run #7	97	753	728	552	26
Run #8	102	677	645	653	10
Run #9	112	746	705	607	31
Run #10	131	706	716	527	23
Run #11	120	724	743	577	11
Run #12	93	749	704	638	27
Run #13	116	714	710	595	16
Run #14	109	698	714	681	22
Run #15	89	750	703	619	31
Run #16	97	697	718	589	27
Run #17	114	726	674	595	37
Run #18	101	739	699	591	35
Run #19	107	718	691	614	25
Run #20	101	724	708	599	16
mean	106.55	719.15	703.10	602.35	22.70
stdev	10.36	22.25	25.24	32.99	7.96
t-test	KS-N vs.	2.23E-37	4.36E-34	3.2E-27	3.83E-26
	KS-S vs.		0.022256	1.04E-14	1.3E-35
	KS-D vs.			7.98E-13	5.41E-33
	KS-H vs.				1.95E-27

10.3.4 Statistics of Topology-Knowledge tuples

In tables 10-16 and 10-17 the generation of non-dominated solutions by the topology-knowledge sources tuples is presented. It is clear that the Octal topology is best at generating non-dominated solutions using the exploitation knowledge sources. It is also is the best at generating non-dominated solutions with topographic knowledge. In general, there is much more variability in the generation of non-dominated solutions across all combinations of topology knowledge source tuples. This is an indication of the difficulty of the problem and the need to maintain and exploit diversity within the system.

Table 10-16 generation of Non-Dominated Solutions using the Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	56.35	275.25	286.65	207.45	16.7
	stdev	27.08714	86.0348	95.26267	70.52097	11.98727
Square	mean	60.5	308.15	314.1	246.1	15.8
	stdev	18.16156	70.74214	70.40028	56.56938	9.666001
Octal	mean	58.65	301.8	303.45	227.65	16.1
	stdev	22.89628	124.052	106.937	93.62258	8.686953
Hex	mean	51.6	246.1	251.95	192.7	11.95
	stdev	19.04676	56.29986	54.06962	46.055	7.708335
Global	mean	53.15	284.05	292.75	207.6	12.75
	stdev	25.70536	93.50343	96.20585	77.2422	7.510519

Table 10-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.480695	0.31257	0.332331	0.339942	0.717801
Square	0.300191	0.22957	0.224133	0.229863	0.611772
Octal	0.390388	0.411041	0.352404	0.411257	0.539562
Hex	0.369123	0.228768	0.214605	0.238998	0.645049
Global	0.483638	0.329179	0.328628	0.372072	0.58906

10.3.5 Analysis of found Pareto front

The found Pareto front is given below for the combined system.

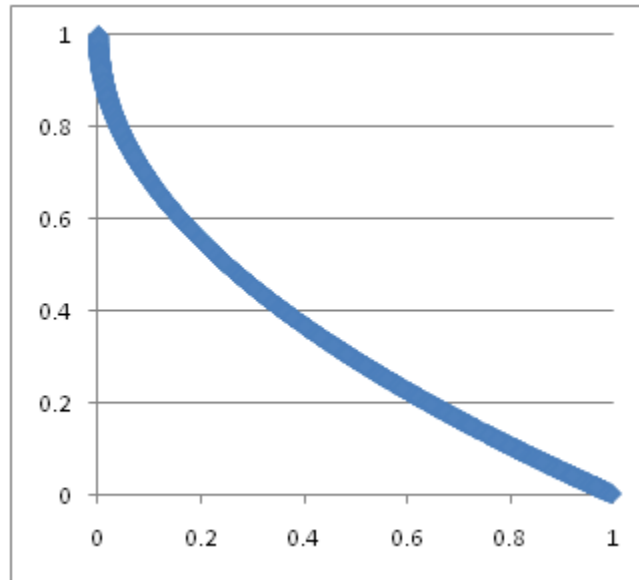


Figure 10.5 Overall found Pareto front

10.4 Summary of ZDT4

At the first glance, ZDT4 is the same as ZDT1; however they are fundamentally different. While for ZDT1 the real Pareto front is a continuous curve, for ZDT4 the real one contains 21^9 local Pareto fronts. This difference is not represented in figures due to the high density; but does bring changes of evolution internally.

In fact, as to experiment results, ZDT4 has many special exclusive features. MOCAT runs on ZDT4 as well as on the first two problems while some other evolutionary algorithms have much worse performance on ZDT4 than on ZDT1 and ZDT2. In fact, using MOCAT, evolution for ZDT4 is similar to it

on ZDT1 and ZDT2, with exception that for ZDT4 at the beginning of evolution individuals do not spread to the far right bottom. Instead, they are attracted there along with evolution.

What is particularly interesting is that while each single metric version has its own style of problem solving, when they are combined there are changes in the usage of the knowledge sources and topologies in each. This ability to adapt the nature and flow of information to the nature of the problem is very important here. Some specific highlights are:

1. Certain topologies begin to favor the generation of individuals for exploratory knowledge sources and other for exploitative knowledge sources.
2. The different metric takes different roles in guiding the system during their respective phases. The spread metric focuses on exploration while the hyper-volume metric concentrates on exploitation. This distinction was only slight in ZDT1 but now is very pronounced as the problem has become more complex.
3. History knowledge takes an active role in translating the results from the one phase to the next during the problem solving process.

CHAPTER 11 EXPERIMENTAL RESULTS FOR MOCAT ON DECPETIVE

PROBLEMS: ZDT5

This test function was described as “*a deceptive problem*” (Zitzler et al, 2000).The original meaning of the description referred to the fact that ZDT5 “*distinguishes itself from the other test functions in that represents a binary string*”. This is an important distinction since most MOEAs rely on hill climbing to solve problems. Such a representation is not at all conducive to that directly and was able to deceive several MOEAs.

While our belief space currently saves only numerical information, there is a need to map binary strings to numbers. In fact, in the definition of ZDT5, function u provides a mapping from strings to integers. In addition, the integers need to be mapped back to binary strings if we want to keep numerical encoding in our current implementation. Fortunately, this requirement can be easily satisfied because for any given integer that is legitimate for the dimension, we are able to create a string that produces such an integer value. Though this integer-to-string mapping is not injective, it does not impair the correctness of the encoding. Thereafter, in our implementation we directly evolve on u values instead of physical binary strings.

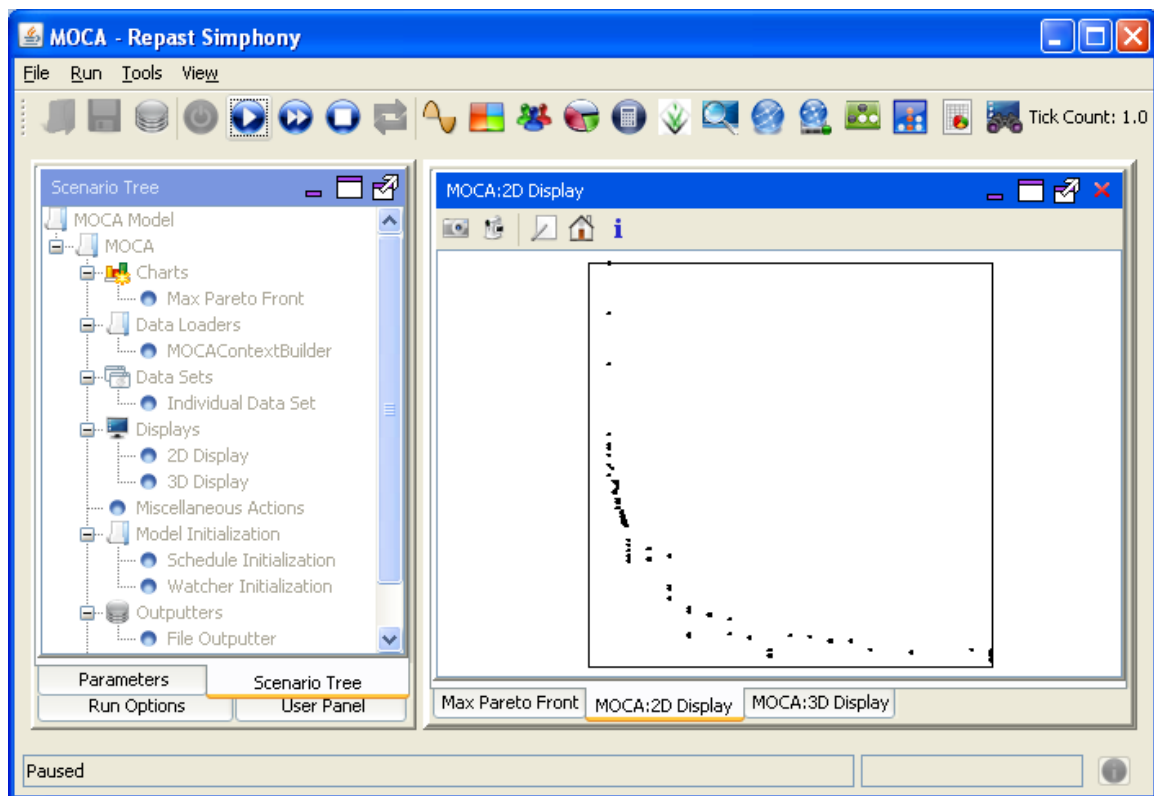
Again, we will use homogeneous spread metric, homogeneous Hyper-volume metric, and combined two metrics to conduct experiments and try to observe and summarize the correspondence among knowledge sources and topologies.

ZDT5 is special also in that it does not have the same range for all dimensions. According to its definition, the first dimension has a data range of $[0, 30]$ while all the rest have range $[1, 5]$. The standard ZDT5 has dimension of 11 (1 + 10) ZDT 5 11 with homogeneous Spread metrics.

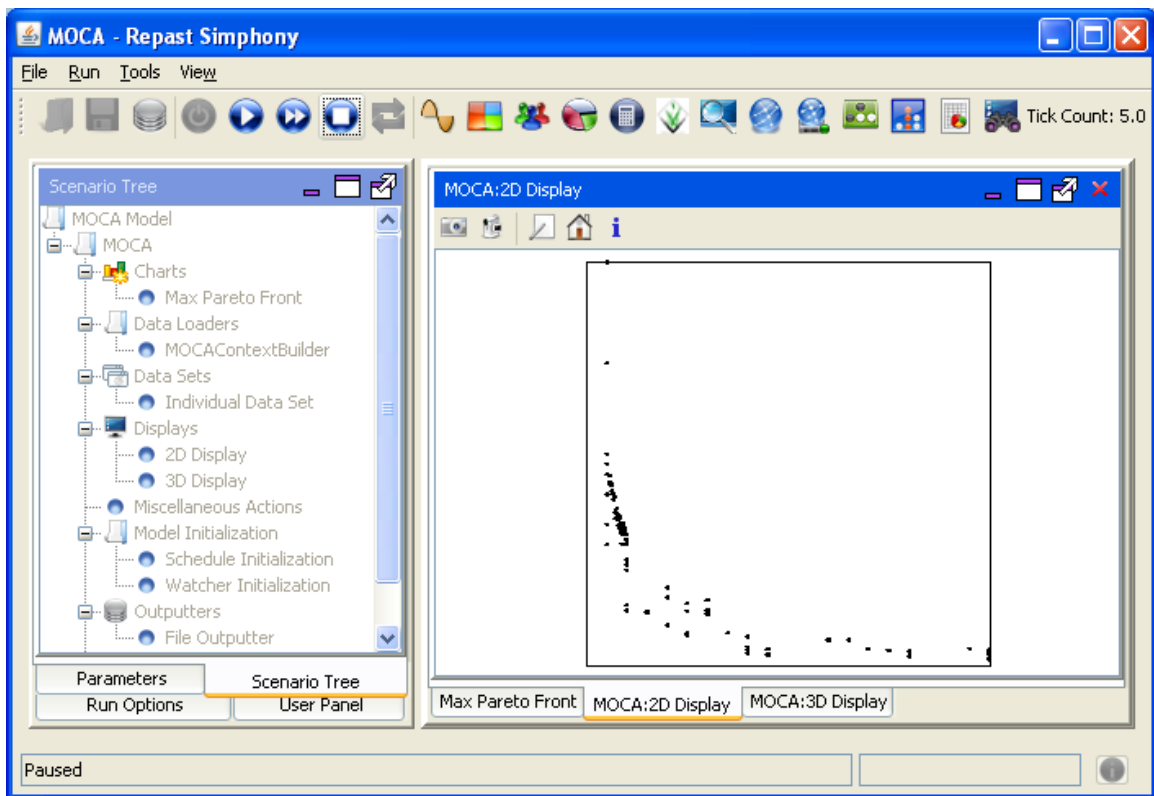
11.1 An Example of the Evolution of ZDT5 Optimal Curve

The runs were complete in an average of 11.78 seconds.

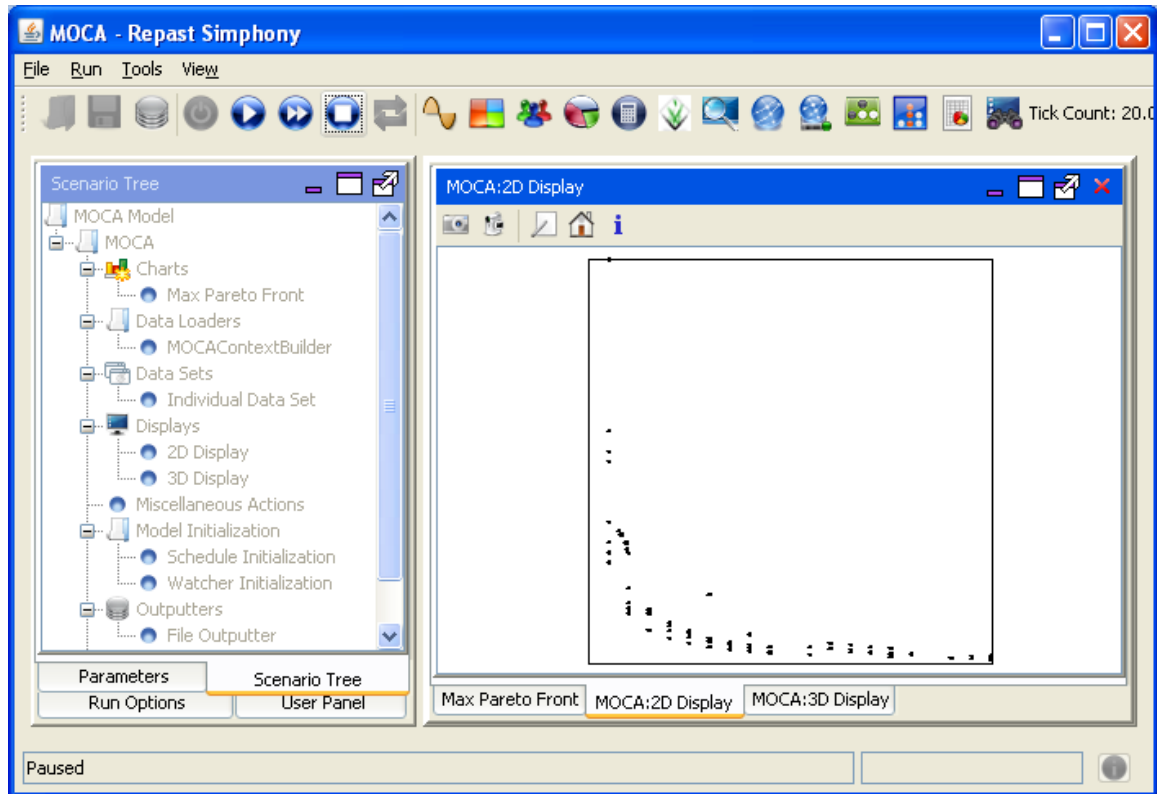
Figure 11-1 (a-f) illustrates how the MOCAT system as guided by the spread metric solves the ZDT5 problem. The problem, like ZDT1 and ZDT4, is essentially convex with a steep initial part and a relatively flat final portion. As with ZDT4 the system generated solutions to the left and then as the number of generations proceeds they begin to move to the portion with the least change.



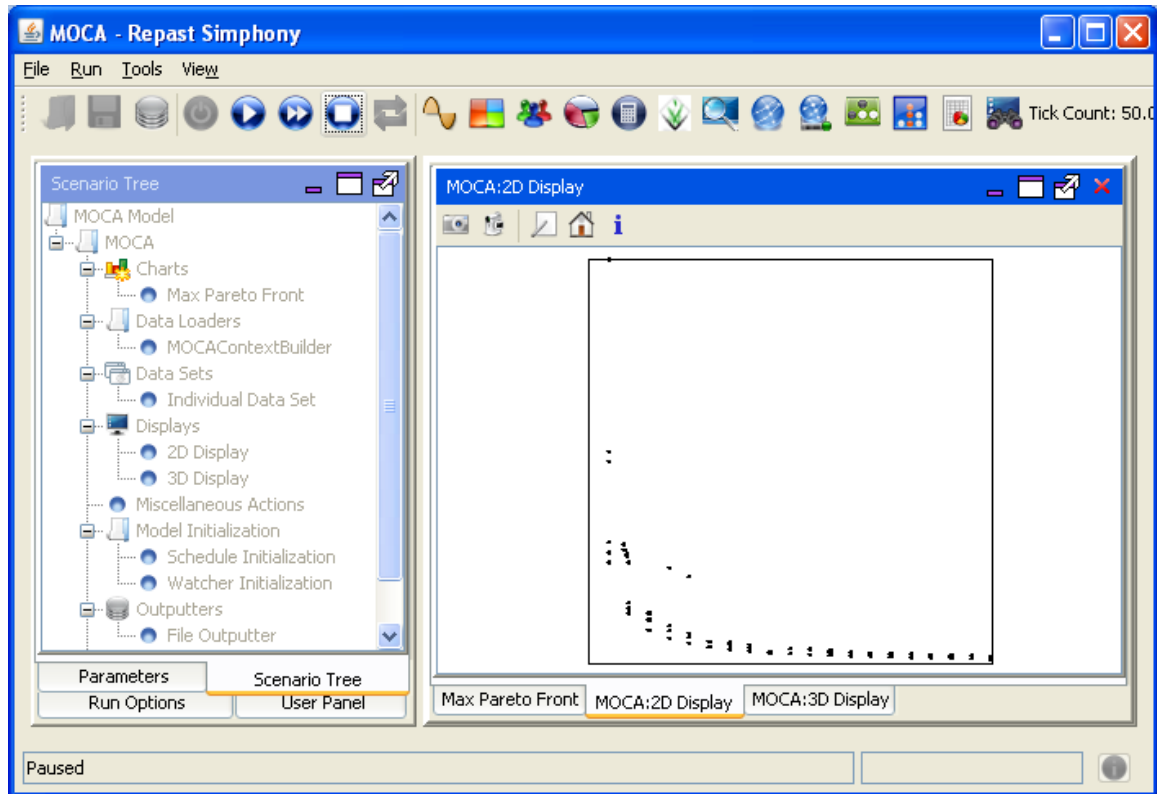
a. Generation 1



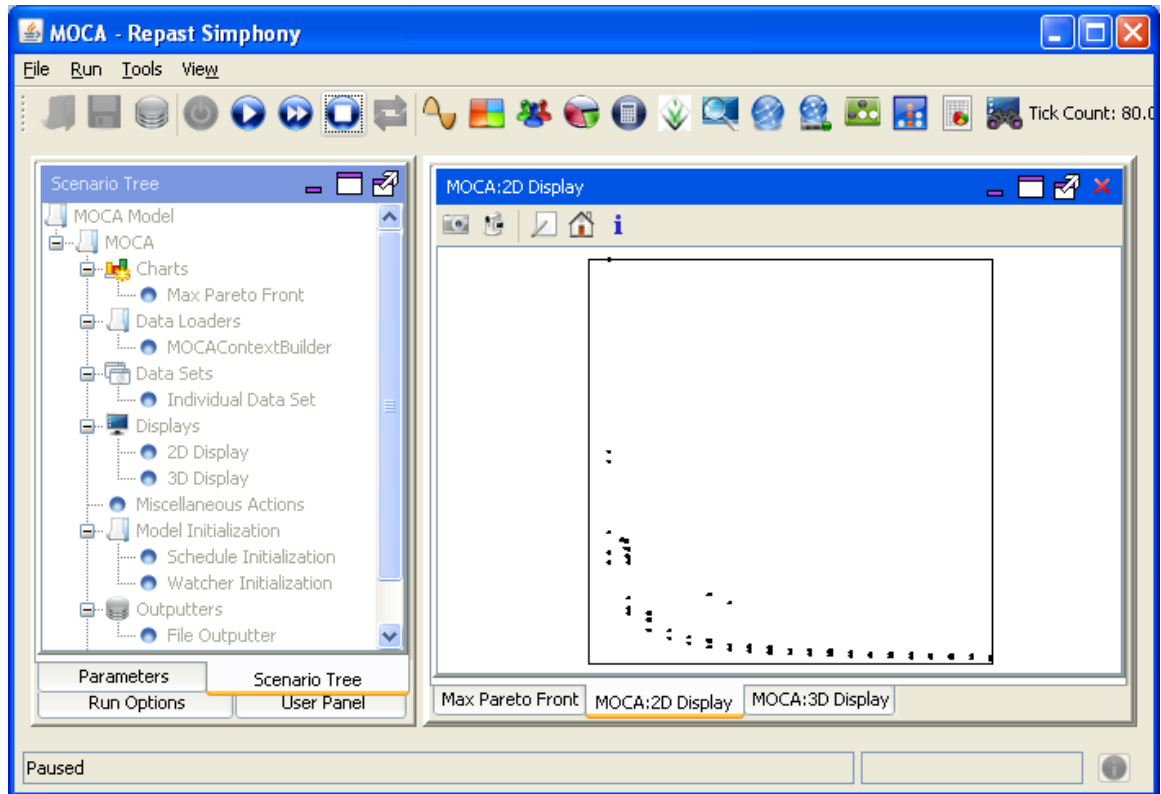
b. Generation 5



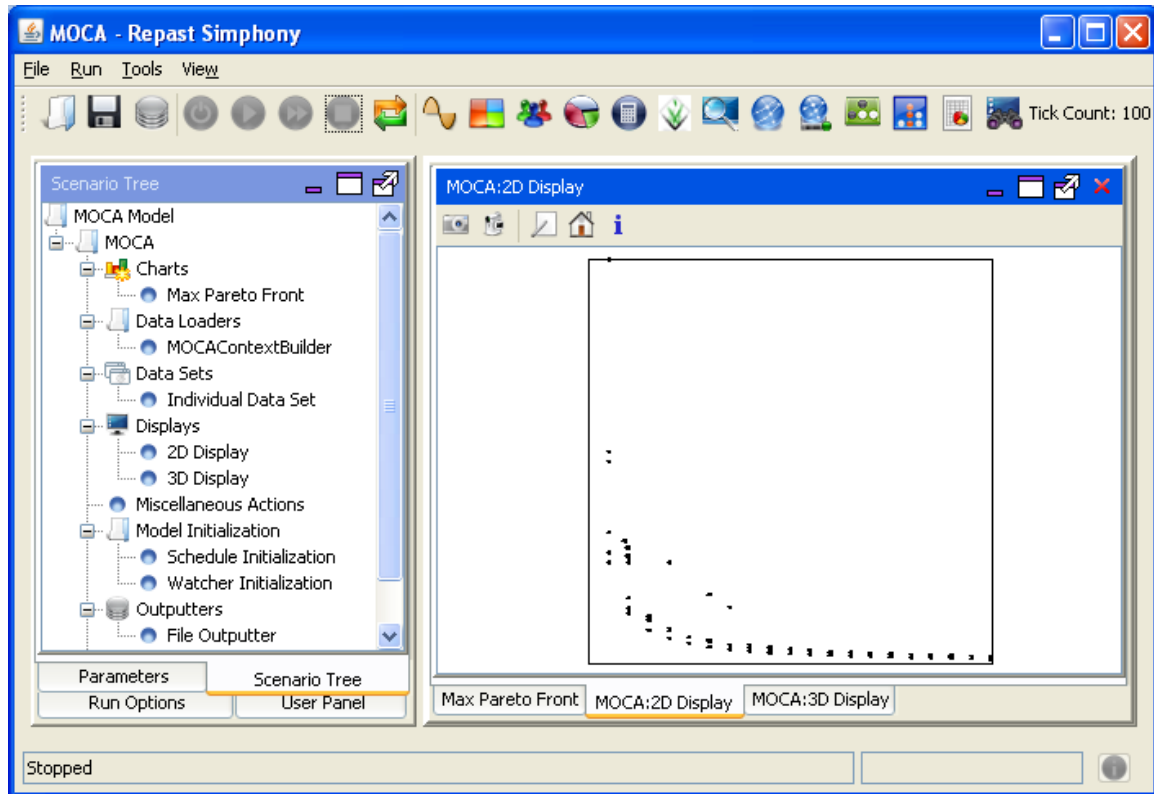
c. Generation 20



d. Generation 50



e. Generation 80



f. Generation 100

Figure 11.1 The sequence for Pareto front produced through cultural evolution.

11.1.1 Performance of MOCAT with the spread metric

For ZDT5 we lack public available existing experiment data to evaluate our performance. However, the performance shown in Table 11-1 below exhibits the least precision of all problems worked on so far.

Table 11-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	1.00E+00	3.34E+00	-2.22E-16	3.50E+01	6.11E+00
Run #2	1.28E+00	3.98E+00	-2.22E-16	3.60E+01	5.94E+00
Run #3	1.00E+00	3.67E+00	-2.22E-16	3.30E+01	5.70E+00

Run #4	1.00E+00	3.70E+00	-4.44E-16	3.60E+01	6.31E+00
Run #5	1.00E+00	3.68E+00	-2.22E-16	4.20E+01	7.16E+00
Run #6	1.00E+00	3.93E+00	-4.44E-16	3.40E+01	6.27E+00
Run #7	1.00E+00	3.04E+00	-4.44E-16	3.10E+01	5.43E+00
Run #8	1.00E+00	3.15E+00	-4.44E-16	2.90E+01	5.23E+00
Run #9	1.22E+00	2.84E+00	-2.22E-16	2.70E+01	4.12E+00
Run #10	1.00E+00	3.25E+00	-2.22E-16	3.60E+01	5.47E+00
Run #11	1.00E+00	3.42E+00	-2.22E-16	3.30E+01	5.51E+00
Run #12	1.00E+00	2.93E+00	-4.44E-16	3.10E+01	5.33E+00
Run #13	1.00E+00	3.83E+00	-2.22E-16	3.80E+01	7.38E+00
Run #14	1.00E+00	3.42E+00	-2.22E-16	4.00E+01	6.10E+00
Run #15	1.00E+00	3.11E+00	-4.44E-16	3.60E+01	5.72E+00
Run #16	1.00E+00	2.93E+00	-4.44E-16	3.50E+01	5.45E+00
Run #17	1.00E+00	2.98E+00	-2.22E-16	3.20E+01	5.10E+00
Run #18	1.23E+00	3.83E+00	-4.44E-16	4.40E+01	6.58E+00
Run #19	1.00E+00	2.97E+00	-2.22E-16	3.30E+01	5.60E+00
Run #20	1.00E+00	3.05E+00	-4.44E-16	4.20E+01	6.32E+00
mean	1.04E+00	3.35E+00	-3.22E-16	3.52E+01	5.84E+00
stdev	8.58E-02	3.69E-01	1.10E-16	4.30E+00	7.15E-01

11.1.2 Statistics of Topologies using Spread metric

In terms of the usage of topologies Table 11-2 indicates that there is no statistical difference in the usage here between the topologies. However, the spread metric typically uses the global topology to facilitate search.

Table 11-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	20	14	25	22	19	4.478448
Run #2	26	13	21	20	20	5.500802
Run #3	30	16	17	16	21	3.1204
Run #4	16	13	33	22	16	3.187408
Run #5	19	15	27	18	21	8.980074
Run #6	19	16	19	17	29	13.11974

Run #7	23	15	21	24	17	8.58566
Run #8	14	18	20	29	19	4.238895
Run #9	15	17	26	22	20	7.039125
Run #10	20	15	17	23	25	5.491045
Run #11	27	20	13	18	22	5.966925
Run #12	22	16	26	16	20	5.519773
Run #13	26	17	31	14	12	3.321442
Run #14	15	25	19	12	29	3.000329
Run #15	14	16	25	17	28	4.242641
Run #16	29	39	16	9	7	16.37494
Run #17	19	23	12	20	26	9.896835
Run #18	17	26	11	31	15	6.550872
Run #19	26	14	25	20	15	3.122606
Run #20	18	17	15	20	30	6.987281
mean	20.75	18.25	20.95	19.50	20.55	
stdev	4.97	5.96	6.02	5.06	5.90	
t-test	Lbest vs.	0.084234	0.455843	0.223628	0.455327	
	Square vs.		0.086377	0.245127	0.119666	
	Octal vs.			0.213456	0.418641	
	Hex vs.				0.279901	

11.1.3 Behaviors of Knowledge sources using Spread Metric

What is interesting here is that for the first time the MOCAT system has shifted to emphasizing exploration over exploitation. This suggests that due to the fact that hill climbing is less productive here there is more emphasis placed on systematic search. More individuals are controlled by exploratory knowledge sources than exploitative ones due to the reduced ability to do local hill climbing for this problem. All knowledge source usages are, as usual, statistically different from each other.

Table 11-3 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	1592	207	51	752	407
Run #2	952	223	15	832	388
Run #3	1325	229	67	1081	303

Run #4	1137	260	15	662	285
Run #5	1288	296	94	1022	320
Run #6	1555	277	74	822	460
Run #7	1577	277	51	648	406
Run #8	1465	259	61	865	516
Run #9	859	250	32	612	423
Run #10	1344	229	78	1085	305
Run #11	922	210	39	855	379
Run #12	1445	251	34	777	455
Run #13	1380	247	50	996	340
Run #14	441	161	8	459	291
Run #15	390	170	4	382	253
Run #16	1513	233	48	765	436
Run #17	1406	231	26	658	434
Run #18	1537	296	86	802	390
Run #19	422	186	3	443	275
Run #20	1089	221	22	543	341
mean	1181.95	235.65	42.90	753.05	370.35
stdev	386.32	36.56	27.02	199.85	70.89
t-test	KS-N vs.	8.06E-10	3.69E-11	9.12E-05	7.91E-09
	KS-S vs.		6.22E-20	2.28E-10	2.33E-08
	KS-D vs.			1.02E-12	2.45E-16
	KS-H vs.				2.31E-08

The table shows a phenomenon that is not seen in other problems. Individuals influenced by situational knowledge source had better chance to contribute to the evolution than those influenced by domain knowledge source. At first glance, this is contrary to being intuitive while situational knowledge source will choose a random location in near neighborhood but domain knowledge source will probe the adjacent positions on each dimension. Nonetheless, a deep analysis finds that situational knowledge may influence an individual to jump to a new position that is much farther than the distance that can be got under the influence of domain knowledge source. Using the situational knowledge source, on each dimension a random offset is moved so that sum of such offsets can be big upon accumulation. Using domain knowledge source, the new individual is chosen from a pool of candidates whose locations are

different from the parent on only one dimension (Che, 2008) and the difference is only one since ZDT5 has discrete domain space. It looks like walking around inside a small neighborhood does not bring the evolution any progress for ZDT5. If this conjecture is true, ZDT5 will impose unconquerable challenges to the evolutionary algorithms that heavily depend on local exploration.

11.1.4 Statistics of Topology-Knowledge tuple

Here it is clear the exploratory knowledge source tuples are doing the best at generating non-dominated solutions. Of the exploitative knowledge sources history is best at exploring areas around existing solutions. All of the tuples exhibit variability in the generation process.

Table 11-4 The Generation of non-dominated solutions by Topology-Knowledge tuples

		N	S	D	H	T
Lbest	mean	186.15	1.15	5	161.7	67.3
	stdev	96.3187	1.531253	6.488857	65.54637	30.87343
Square	mean	190.85	0.55	6.1	164.15	66.3
	stdev	91.41823	0.759155	6.896986	80.09815	27.8683
Octal	mean	167.05	0.65	4.85	148.25	54.7
	stdev	66.09919	0.933302	4.319783	71.83891	22.66948
Hex	mean	167.15	0.9	6	148.9	59.8
	stdev	77.0183	1.372665	6.844129	65.72823	20.45946
Global	mean	173.4	0.8	4.25	154.5	65.8
	stdev	65.34475	1.361114	3.739899	54.58697	22.83949

Table 11-5 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.517425	1.331525	1.297771	0.405358	0.458743
Square	0.479006	1.380281	1.130654	0.487957	0.420336
Octal	0.395685	1.435849	0.890677	0.484579	0.414433
Hex	0.460774	1.525184	1.140688	0.441425	0.342131
Global	0.376844	1.701393	0.879976	0.353314	0.347105

11.1.5 Analysis of found Pareto front

Finally, the addition of found Pareto front of each run is represented.

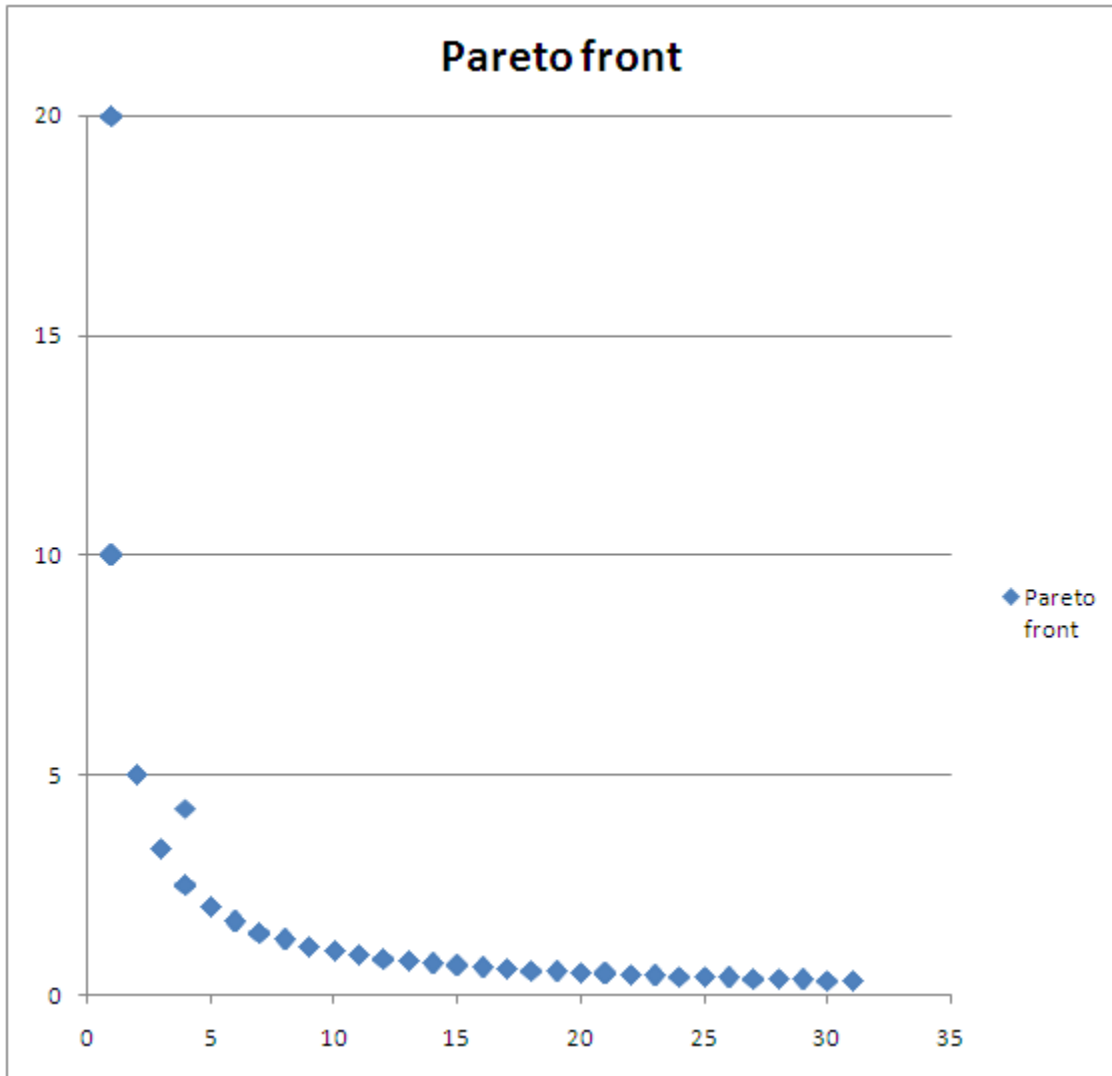


Figure 11.2 Overall found Pareto front

For ZDT5, the explored Pareto front consists of a series of discrete locations. For $f_1 = 1$, f_2 should be 10, however, in our final result set 20 is also presented. There is one location (4, 4) which is far from the real Pareto front. This indicates that in one run we have missed locations (3, 3.333) and (4, 2.5). And

we can see some overlap at locations (6, 1.667), (8, 1.25), etc. Nonetheless, in total, the Pareto front that MOCAT has identified is a good representation of the ideal.

11.2 ZDT 511 with homogeneous Hyper-volume metrics

The runs were complete in an average of 12.12 seconds.

11.2.1 Performance of MOCAT with Hyper-volume

The performance of the MOCAT system with hyper-volume is slightly better than with the spread metric in terms of both mean error and standard deviation of the error. This is consistent with ZDT2, ZDT3, and ZDT4.

Table 11-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	Mean	min	max	stdev
Run #1	1.00E+00	3.34E+00	-4.44E-16	3.00E+01	5.72E+00
Run #2	8.00E-01	2.94E+00	-4.44E-16	3.20E+01	5.41E+00
Run #3	1.32E+00	4.34E+00	-2.22E-16	3.60E+01	6.79E+00
Run #4	1.35E+00	3.79E+00	-4.44E-16	3.20E+01	5.46E+00
Run #5	1.00E+00	3.25E+00	-2.22E-16	3.40E+01	5.13E+00
Run #6	9.52E-01	3.01E+00	-4.44E-16	3.30E+01	5.47E+00
Run #7	1.00E+00	3.13E+00	-2.22E-16	3.50E+01	5.29E+00
Run #8	1.00E+00	3.48E+00	-2.22E-16	3.60E+01	6.69E+00
Run #9	8.57E-01	2.96E+00	-2.22E-16	3.60E+01	5.53E+00
Run #10	1.00E+00	3.40E+00	-2.22E-16	3.90E+01	5.64E+00
Run #11	1.00E+00	3.06E+00	-4.44E-16	3.20E+01	4.83E+00
Run #12	1.00E+00	3.18E+00	-4.44E-16	3.10E+01	5.32E+00
Run #13	1.00E+00	3.37E+00	-2.22E-16	3.70E+01	6.25E+00
Run #14	1.00E+00	3.24E+00	-2.22E-16	3.30E+01	5.55E+00
Run #15	1.13E+00	3.75E+00	-2.22E-16	3.60E+01	5.96E+00
Run #16	6.63E-01	3.30E+00	-2.22E-16	4.20E+01	7.12E+00
Run #17	1.00E+00	3.95E+00	-4.44E-16	4.10E+01	6.73E+00

Run #18	1.00E+00	3.37E+00	-2.22E-16	2.70E+01	4.83E+00
Run #19	1.18E+00	3.50E+00	-4.44E-16	3.80E+01	5.96E+00
Run #20	6.90E-01	2.34E+00	-4.44E-16	3.20E+01	4.55E+00
mean	9.97E-01	3.34E+00	-3.22E-16	3.46E+01	5.71E+00
stdev	1.64E-01	4.11E-01	1.10E-16	3.64E+00	6.86E-01

11.2.2 Statistics of Topologies using Hyper-volume metric

Unlike the spread metric MOCAT system, favored certain topologies during the runs. The hyper-volume version used the square topology significantly more often than LBEST, HEX, and Global as shown in Table 11-7.

Table 11-7 Use Count of Topologies of each run

	Using Hyper-volume metric					Hyper-volume Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	19	22	18	20	21	100.8287
Run #2	23	17	15	24	21	101.062
Run #3	23	17	26	19	15	96.44014
Run #4	20	26	23	15	16	120.8224
Run #5	16	24	18	24	18	113.3882
Run #6	21	21	18	21	19	137.8363
Run #7	19	19	21	16	25	154.6373
Run #8	22	15	25	18	20	113.098
Run #9	24	18	23	12	23	104.3497
Run #10	22	29	15	16	18	105.062
Run #11	21	20	20	20	19	105.8229
Run #12	16	21	19	26	18	137.3104
Run #13	18	18	18	28	18	126.496
Run #14	18	23	22	14	23	118.1137
Run #15	21	19	23	20	17	129.9446
Run #16	14	27	17	21	21	119.7099
Run #17	20	24	24	14	18	112.0656
Run #18	16	21	31	10	22	119.8202
Run #19	18	24	19	21	18	113.6576
Run #20	20	26	13	25	16	96.44519
mean	19.55	21.55	20.40	19.20	19.30	

stdev	2.64	3.68	4.20	4.72	2.57
t-test	Lbest vs.	0.031218	0.230202	0.389835	0.384438
	Square vs.		0.187573	0.047782	0.017963
	Octal vs.			0.206433	0.16882
	Hex vs.				0.467947

11.2.3 Behaviors of Knowledge sources using Hyper-volume Metric

Table 11-8 describes the extent of control over the population by each of the knowledge sources. Given the local hill climbing is effectively disabled search control has moved over to the exploratory knowledge sources as we saw with the spread metric version earlier. While both exploratory knowledge sources are used frequently, the hyper-volume metric increases its use of situational history knowledge to replace the hill climbing activities of the domain knowledge. This is reasonable since the hyper-volume metric is more supportive of exploitative search than the spread metric, so it tries to identify other ways to improve it.

Table 11-8 Using Hyper-volume metric #Individuals influenced by KS

usingHyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	1685	283	69	753	582
Run #2	1702	315	99	887	510
Run #3	1414	263	25	968	497
Run #4	1330	245	34	1243	428
Run #5	1225	292	36	772	358
Run #6	1714	280	36	732	504
Run #7	513	261	4	354	244
Run #8	1421	257	36	806	303
Run #9	1445	252	58	785	385
Run #10	1185	266	51	836	317
Run #11	1499	277	67	712	370
Run #12	1605	291	63	903	542
Run #13	1659	265	59	877	322
Run #14	639	157	15	557	314

Run #15	1293	248	94	1165	448
Run #16	1441	270	45	884	370
Run #17	1549	235	39	1080	321
Run #18	1727	260	34	775	374
Run #19	1267	262	15	664	327
Run #20	1323	248	37	472	250
mean	1381.80	261.35	45.80	811.25	388.30
stdev	316.19	30.16	24.04	208.12	94.50
t-test	KS-N vs.	1.32E-12	5.97E-14	9.29E-08	2.75E-12
	KS-S vs.		2.38E-24	1.89E-10	5.77E-06
	KS-D vs.			6.1E-13	2.52E-13
	KS-H vs.				6.57E-09

11.2.4 Statistics of Topology-Knowledge tuple

As shown in Table 11-9, the only exploitative knowledge source that is produce with the hyper-volume measure is History. However, with the hyper-volume performance metric it is much more productive then with the spread metric. Both situational and domain knowledge are ineffective in generating non-dominated solution for this problem because of the reliance on local improvements. Another interesting note, is that the variability of domain knowledge is high which means that if it did find something in a region it would do some limited hill climbing.

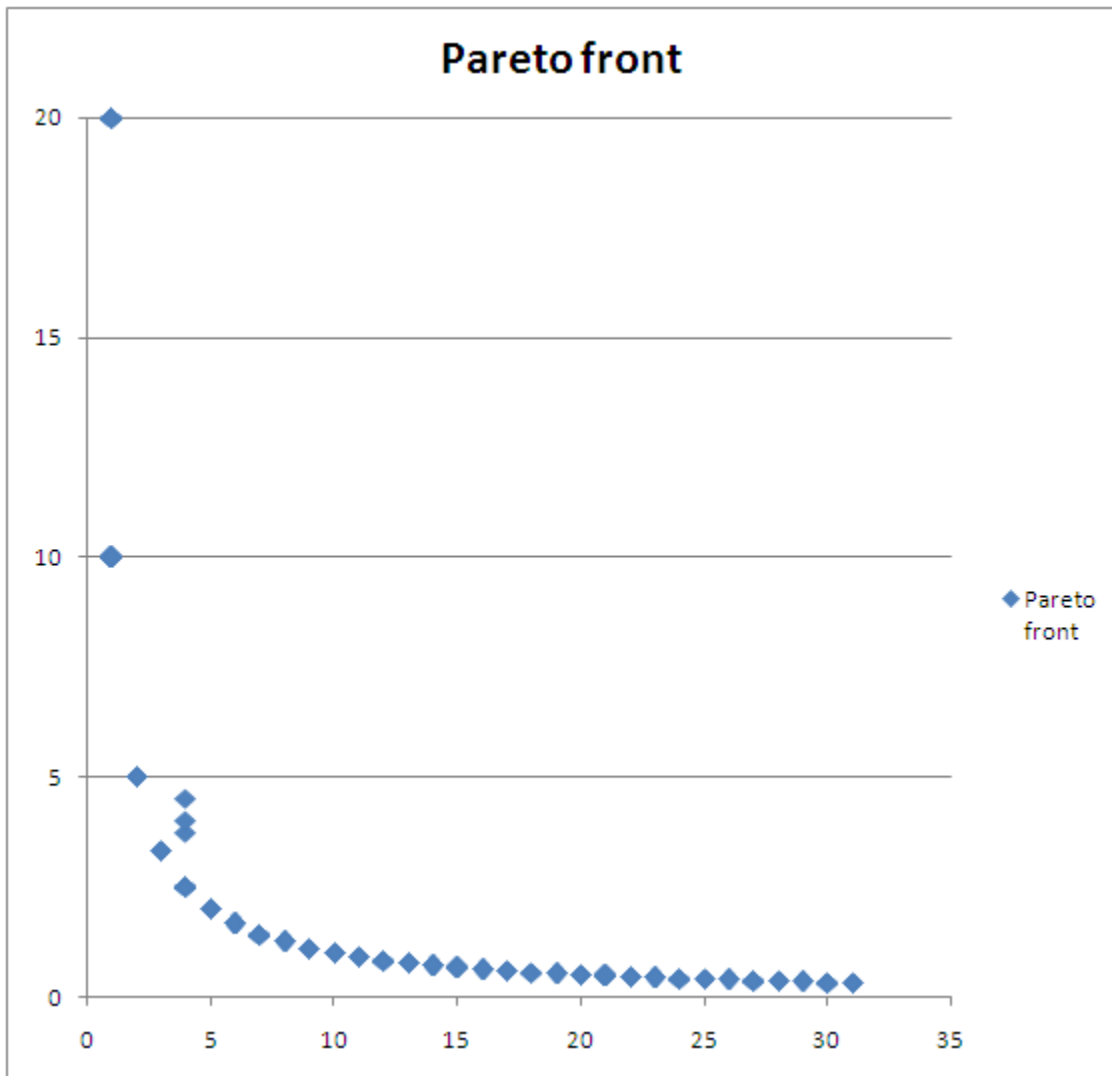
Table 11-9 Production of non-dominated solution by Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	190.9	0.6	6.3	170.4	71.3
	stdev	59.45356	0.680557	4.117996	54.24545	23.53519
Square	mean	169.25	0.55	6.25	148.7	58.3
	stdev	78.04646	0.944513	5.552382	52.68087	18.53617
Octal	mean	183.95	0.5	6.65	181.3	66.7
	stdev	42.44188	0.888523	5.314083	53.9845	24.07904
Hex	mean	214	0.55	6.4	186.85	69.95
	stdev	83.90408	1.145931	5.255573	74.07768	26.77877
Global	mean	204.55	0.6	6.8	176.7	70.5
	stdev	63.82829	0.88258	5.084548	41.05978	18.53446

Table 11-10 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.311438	1.134262	0.65365	0.318342	0.330087
Square	0.461131	1.717297	0.888381	0.354276	0.317945
Octal	0.230725	1.777047	0.79911	0.297763	0.361005
Hex	0.392075	2.083511	0.821183	0.396455	0.382827
Global	0.312042	1.470967	0.747728	0.23237	0.2629

11.2.5 Analysis of found Pareto front

**Figure 11.3 Overall found Pareto front**

The system produced a good approximation. Notice that the only flaw is in the transitional point between the steep part of the curve and the flatter part. It will be interesting to see how the combined metric approach deals with this situation.

11.3 ZDT 511 with combined metrics

In this section MOCAT system that uses the combined metric is examined. The key problem with the deceptive function is the removal of hill climbing as a search option. Thus, a critical issue is to how to replace the search power lost.

The runs were complete in an average of 12.89 seconds.

11.3.1 Performance of MOCAT with the Combined Metrics

The performance of MOCA with the combined metrics is slightly worse than that of the hyper-volume system on its own. The system synergy is best when both exploration and exploitation can take place together. In ZDT1 the emphasis was on exploitation and here the emphasis was on exploration. In both situations a single performance metric approach dominated rather than a combination. As will be demonstrated the two subsystems perform in very similar ways for the problem, so there is little opportunity for synergy. Both focus on the extreme topologies (lbest or square, and global) and on the exploratory knowledge sources along with a single exploitative knowledge source, history.

Table 11-11 Statistics for the fitness errors of ending solutions

Statistics for the fitness errors of ending generation					
	median	mean	Min	max	stdev
Run #1	7.50E-01	2.76E+00	-4.44E-16	2.70E+01	4.14E+00
Run #2	1.13E+00	4.13E+00	-4.44E-16	3.70E+01	7.01E+00
Run #3	1.00E+00	3.71E+00	-4.44E-16	3.70E+01	6.61E+00

Run #4	6.90E-01	3.04E+00	-2.22E-16	3.80E+01	6.60E+00
Run #5	2.00E+00	4.10E+00	-2.22E-16	3.60E+01	6.18E+00
Run #6	1.00E+00	3.18E+00	-2.22E-16	3.70E+01	6.12E+00
Run #7	1.00E+00	2.76E+00	-2.22E-16	2.10E+01	3.91E+00
Run #8	1.00E+00	3.13E+00	-2.22E-16	3.70E+01	5.59E+00
Run #9	8.97E-01	2.79E+00	-2.22E-16	3.60E+01	5.18E+00
Run #10	1.00E+00	3.55E+00	-2.22E-16	3.60E+01	6.04E+00
Run #11	1.00E+00	3.50E+00	-2.22E-16	3.70E+01	6.95E+00
Run #12	1.00E+00	3.20E+00	-4.44E-16	2.10E+01	4.02E+00
Run #13	9.17E-01	3.00E+00	-2.22E-16	3.30E+01	5.37E+00
Run #14	1.10E+00	3.13E+00	-2.22E-16	3.60E+01	4.67E+00
Run #15	9.41E-01	3.26E+00	-4.44E-16	3.60E+01	5.51E+00
Run #16	1.00E+00	3.76E+00	-2.22E-16	3.60E+01	6.64E+00
Run #17	1.42E+00	3.98E+00	-2.22E-16	3.70E+01	6.31E+00
Run #18	1.00E+00	3.40E+00	-2.22E-16	3.60E+01	5.76E+00
Run #19	1.35E+00	3.66E+00	-4.44E-16	3.10E+01	5.58E+00
Run #20	9.69E-01	3.35E+00	-2.22E-16	3.70E+01	6.25E+00
mean	1.06E+00	3.37E+00	-2.89E-16	3.41E+01	5.72E+00
stdev	2.68E-01	4.13E-01	1.02E-16	5.02E+00	9.20E-01

11.3.2 Statistics of Topologies for the combined Metrics

There is no statistically significant difference in usage between the topologies for the spread metric in the combined system similar to how it performed on its own. It tends to favor use of the simplest and most complex topologies as shown in Table 11-12. On the other hand, on its own the hyper-volume favored the square topology whereas in the combined system it favored the LBEST topology over the others.

Table 11-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	6	14	11	8	11	6.468369
Run #2	12	10	16	8	4	4.636591
Run #3	7	11	13	8	11	4.385989

Run #4	8	6	10	11	15	11.40611
Run #5	9	10	12	7	12	6.889167
Run #6	10	9	7	6	18	9.806533
Run #7	10	4	10	8	18	8.092147
Run #8	13	20	1	2	14	3.141398
Run #9	6	11	7	16	10	12.70618
Run #10	12	14	13	4	7	8.430844
Run #11	7	5	16	9	13	13.95457
Run #12	11	3	6	14	16	5.968624
Run #13	8	10	9	11	12	3.754187
Run #14	15	12	7	9	7	9.377765
Run #15	14	15	8	7	6	3.800018
Run #16	13	5	10	10	12	5.581133
Run #17	12	14	8	12	4	3.642426
Run #18	9	4	16	11	10	8.796178
Run #19	8	10	3	16	13	11.01232
Run #20	10	17	1	12	10	17.39656
mean	10.00	10.20	9.20	9.45	11.15	
stdev	2.61	4.56	4.37	3.50	3.98	
t-test	Lbest vs.	0.434607	0.248988	0.293133	0.149841	
	Square vs.		0.246962	0.286454	0.248914	
	Octal vs.			0.423335	0.079196	
	Hex vs.				0.0851	

Table 11-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric				Hyper-volume metric value	
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	12	10	14	7	7	96.96355
Run #2	13	8	11	7	11	134.6014
Run #3	9	7	17	6	11	119.1882
Run #4	11	10	8	9	12	105.3409
Run #5	14	9	6	11	10	100.9886
Run #6	16	5	14	9	6	117.065
Run #7	8	7	15	11	9	110.8687
Run #8	10	12	9	10	9	106.9506
Run #9	9	13	6	11	11	117.0063
Run #10	9	6	11	13	11	107.2813
Run #11	10	12	11	7	10	123.9455
Run #12	14	10	12	9	5	102.2482

Run #13	9	5	12	12	12	131.5261
Run #14	11	10	6	10	13	136.8117
Run #15	11	9	9	11	10	93.67658
Run #16	15	12	9	8	6	105.1895
Run #17	14	6	8	15	7	120.1524
Run #18	10	4	13	14	9	103.1443
Run #19	13	7	6	14	10	101.5949
Run #20	14	13	7	7	9	90.30836
mean	11.60	8.75	10.20	10.05	9.40	
stdev	2.31	2.74	3.23	2.58	2.15	
t-test	Lbest vs.	0.000673	0.066938	0.02925	0.002165	
	Square vs.		0.072102	0.070028	0.210614	
	Octal vs.			0.437636	0.187999	
	Hex vs.				0.202247	

11.3.3 Behaviors of Knowledge sources in the combined system

The different in influence of the knowledge sources over individuals in the population is given in Figures 11-14 and 11-15. Both exhibit statistically different influence values for all knowledge sources. Both emphasize the two exploratory knowledge sources and the exploitative history knowledge source.

Table 11-14 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	681	156	9	333	163
Run #2	492	137	13	578	156
Run #3	594	143	19	432	242
Run #4	551	121	23	356	176
Run #5	516	135	32	527	232
Run #6	555	156	35	413	355
Run #7	631	140	3	337	216
Run #8	619	128	18	446	253
Run #9	609	155	23	528	172
Run #10	567	144	18	403	247
Run #11	606	129	18	323	179
Run #12	225	129	4	380	135

Run #13	625	131	33	393	277
Run #14	583	155	7	527	182
Run #15	609	172	18	325	150
Run #16	517	117	10	468	160
Run #17	441	142	38	701	230
Run #18	629	133	15	341	200
Run #19	505	125	13	518	165
Run #20	563	144	14	357	218
mean	555.90	139.60	18.15	434.30	205.40
stdev	94.45	13.59	9.80	99.24	51.79
t-test	KS-N vs.	1.76E-14	1.99E-16	0.000208	5.26E-15
	KS-S vs.		2.18E-27	2.54E-11	1.18E-05
	KS-D vs.			6.12E-14	4.93E-13
	KS-H vs.				4.7E-10

Table 11-15 Using Hyper-volume metric #Individuals influenced by KS

usingHyper-volume metric #Individuals influenced by KS					
	778	132	14	275	107
Run #1	807	141	27	602	167
Run #2	787	132	28	496	248
Run #3	807	86	41	295	112
Run #4	747	113	65	528	227
Run #5	840	134	51	357	263
Run #6	969	139	5	386	163
Run #7	916	102	38	435	197
Run #8	849	126	30	506	199
Run #9	800	113	35	438	196
Run #10	951	111	41	361	169
Run #11	500	118	6	277	100
Run #12	939	119	40	371	238
Run #13	949	147	20	477	187
Run #14	737	131	22	306	125
Run #15	870	119	39	492	201
Run #16	605	114	49	721	158
Run #17	984	109	20	363	199
Run #18	784	118	14	531	172
Run #19	703	108	21	297	132
Run #20	816.10	120.60	30.30	425.70	178.00

mean	120.24	14.38	15.18	115.91	45.59
stdev	778	132	14	275	107
t-test	KS-N vs.	1.28E-16	1.17E-17	1.03E-12	1.04E-17
	KS-S vs.		3.97E-21	2.19E-10	1.35E-05
	KS-D vs.			2.19E-12	1.07E-12
	KS-H vs.				2.88E-09

11.3.4 Production of non-dominated solutions by Topology-Knowledge tuples

As with the previous statistics there is little difference in the behavior of the spread metric controlled phase of the combined system with that of the hyper-volume controlled phase. In each case the three most productive knowledge sources are normative, topographic, and history. The octal metric is the most productive topology with the octal normative pair being the most productive overall. In the standalone systems the most productive topologies tended towards the extremes. Here, the combination has reduced the complexity of the social fabric.

Table 11-16 Generation of non-dominated solutions by Topology-Knowledge tuples

		N	S	D	H	T
Lbest	mean	197.75	0.55	7.95	180	67.05
	stdev	53.9199	1.099043	4.978639	51.33686	21.57112
Square	mean	194.5	0.5	8.25	178	70.45
	stdev	50.5595	0.688247	4.982865	59.35797	26.64676
Octal	mean	223.5	0.6	8.25	196.35	81.1
	stdev	67.4689	0.940325	3.537394	51.40784	26.27166
Hex	mean	209.55	0.7	8.8	191.05	78.3
	stdev	63.54814	1.080935	5.818301	55.67242	29.53161
Global	mean	179.5	0.65	6	168.3	67.1
	stdev	71.35346	1.268028	4.768316	48.934	22.82865

Once again, variations for S are always bigger than one.

Table 11-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.272667	1.998259	0.626244	0.285205	0.321717
Square	0.259946	1.376494	0.603984	0.333472	0.378237

Octal	0.301874	1.567208	0.428775	0.261817	0.323942
Hex	0.30326	1.544193	0.661171	0.291402	0.37716
Global	0.397512	1.950812	0.794719	0.290755	0.340218

11.3.5 Analysis of found Pareto front

While the system performs slightly less well than the hyper-volume system on its own, it has done a slightly better job at the interface between the steep curve and the gradual one. That was the improvement that we had hoped for in the combined metric.

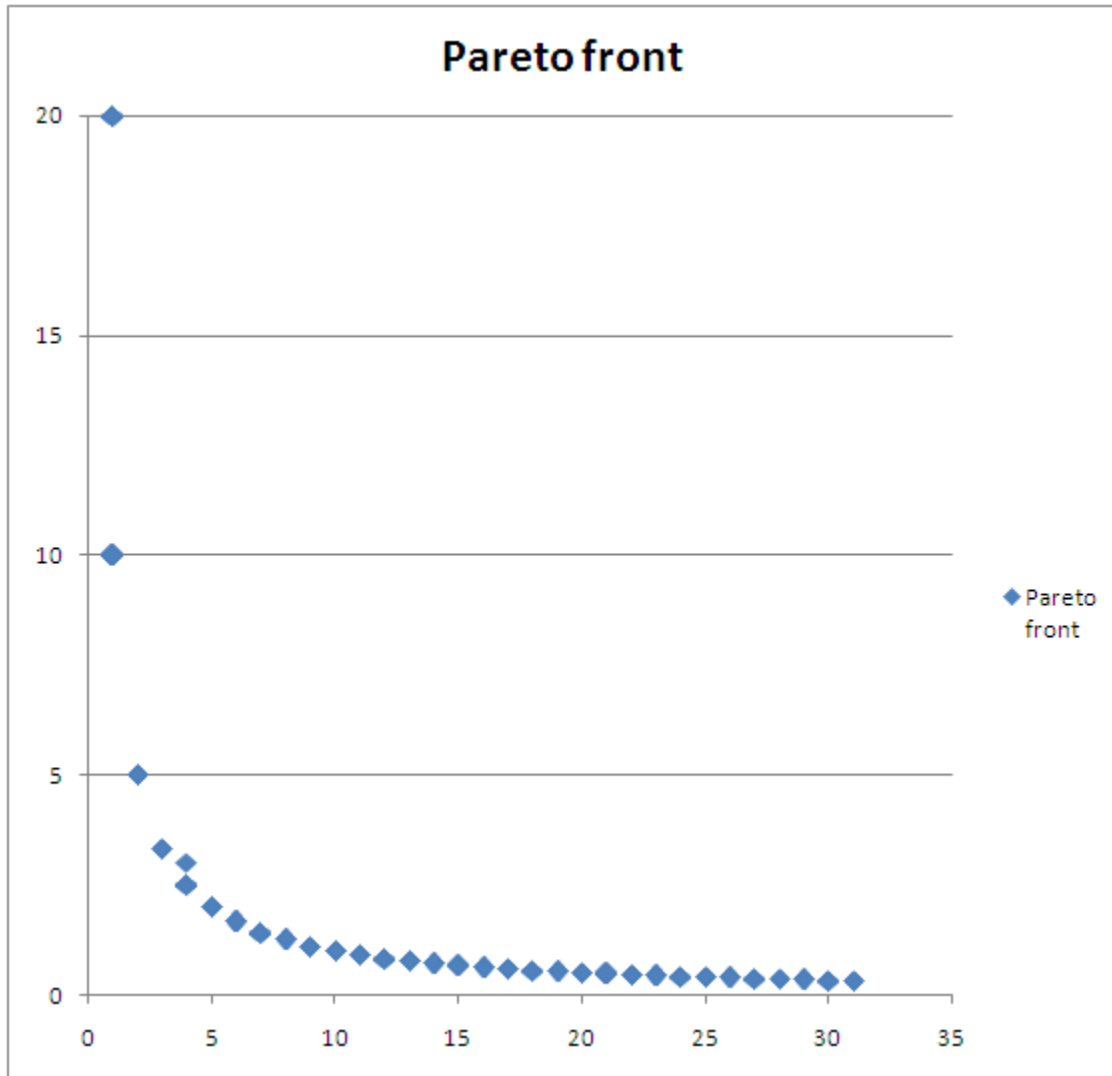


Figure 11.4 Overall found Pareto front

11.4 Summary of the MOCAT performance for ZDT5

The deceptive nature of the problem means that the knowledge sources that utilize hill climbing as a vehicle for exploitation in problem solving will be less effective. In fact, it was the first problem where the exploratory knowledge sources dominated the exploitative ones in usage. As a result the options for each were limited to the remaining three knowledge sources. Both the spread and hyper-

volume versions were forced to exploit the same knowledge sources and as a result there were only minor differences between the two stand-alone metric systems and the combined system although the hyper-volume system was slightly better.

The two phased combined system is less effective when the phases are inherently the same. There were some adjustments in the knowledge and topology usage in the combined system but nothing that made a strong impact on overall performance. However, since history knowledge was still the primary vehicle for the exchange of information between phases there was some opportunity for interaction. The result of that interaction showed up in the transitional portion of the convex curve between the steep and the shallow portions.

In summary, the combined system can serve to blend the results of both phases. When there is sufficiently different activity in both phases then the combined system is worth more likely to perform better than the individual ones. In both ZDT1 and ZDT5, there was less need for both exploratory and exploitation activities so that there was less benefit from the interaction. Yet, even in such cases we did notice some smoothing of the curves and overall reduction in standard error deviation.

CHAPTER 12 EXPERIMENT RESULTS OF ZDT6

ZDT6 looks similar to ZDT2 but its structure is the most complicated of the set for two reasons. First the Pareto optimal solutions are not uniformly distributed along the concave front and they are more dense near certain regions where $f(x) = 1$. Also, there are fewer solutions near the optimal curve than away from it. So there are features from ZDT2, ZDT3, and ZDT4 combined in the problem.

We will use homogeneous spread metric, homogeneous hyper-volume metric, and combined two metrics to conduct experiments and try to observe and summarize the correspondence among knowledge sources and topologies.

12.1 ZDT 6

This problem combines features from problems ZDT2 through ZDT4. As such it is likely to be best solved using the combined system.

$$f_1 = x_1$$

$$g(x_2, \dots, x_m) = 1 + 9 * \left[\sum_{i=2}^m x_i / (m - 1) \right]^{0.25}$$

$$h(f_1, g) = 1 - \left(\frac{x_1}{g} \right)^2$$

$$f_2 = g * h$$

where $m=30$ and $0 \leq x_1 \leq 1$ and $x_i \in (0,1]$ for $i = 2, \dots, m$.

The Pareto front is:

$$x_2 = 1 - x_1^2 \text{ while } x_1 \in [0.2807753191, 1].$$

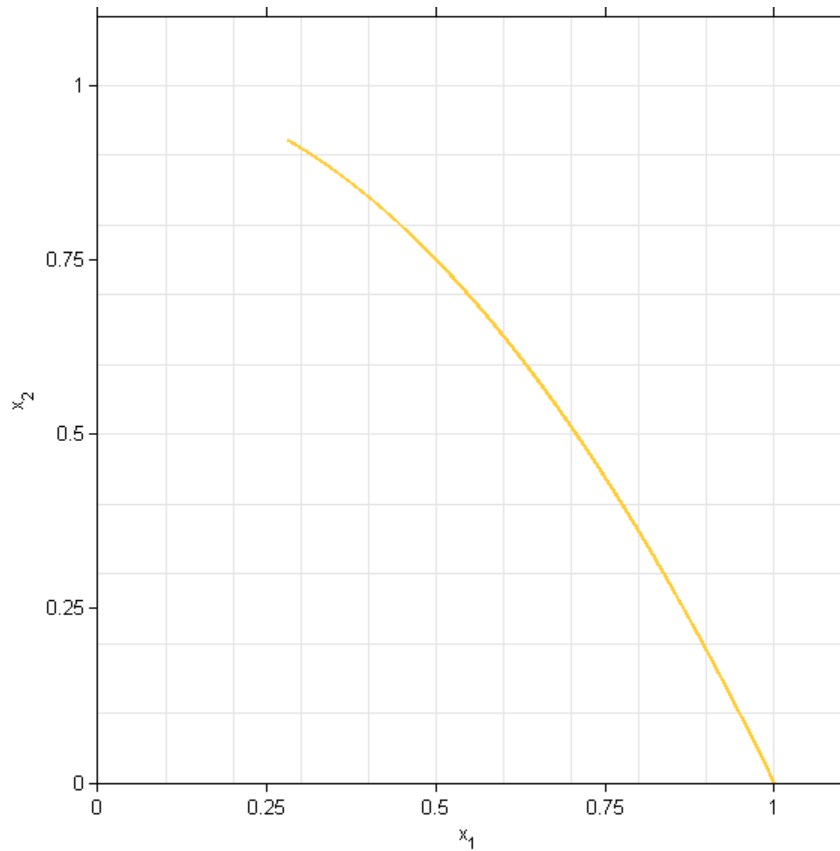


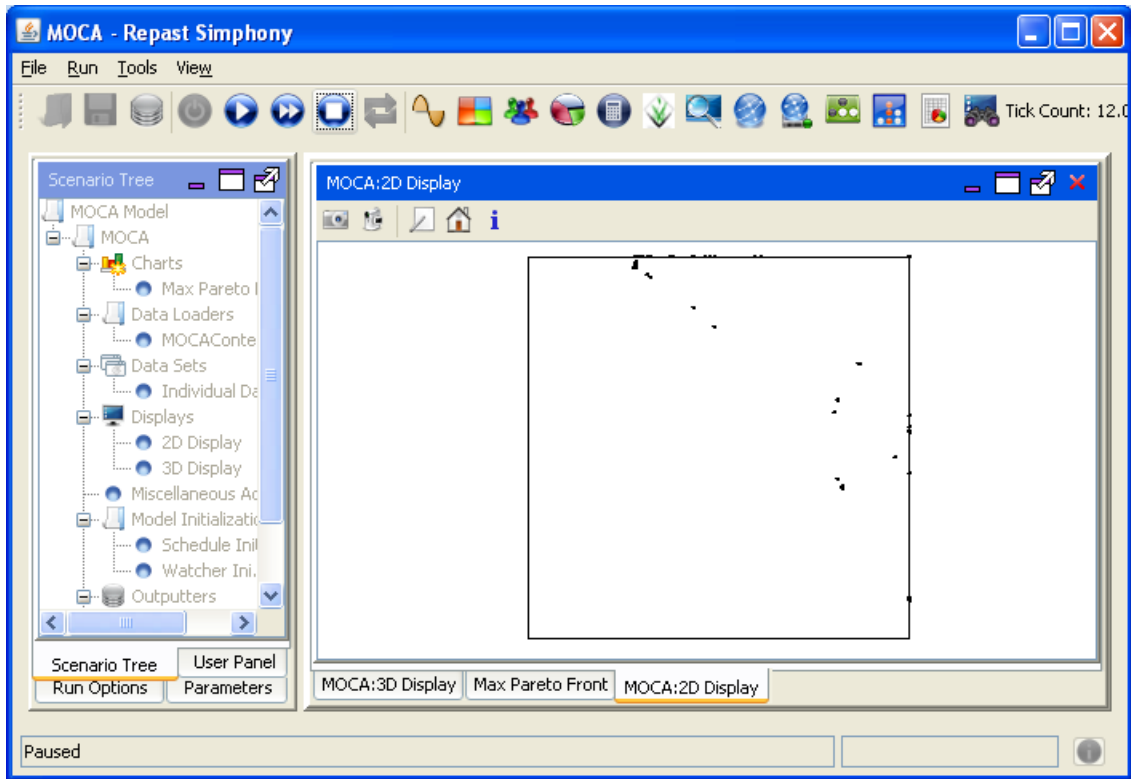
Figure 12.1 ZDT6 Pareto front

For selected test problems, we will run MOCAT 20 times with population of 100, generation of 100, and may use Spread metric, Hyper-volume metric, and both of them alternatively after 25 generations. All results will be presented and summarized with statistical analyses.

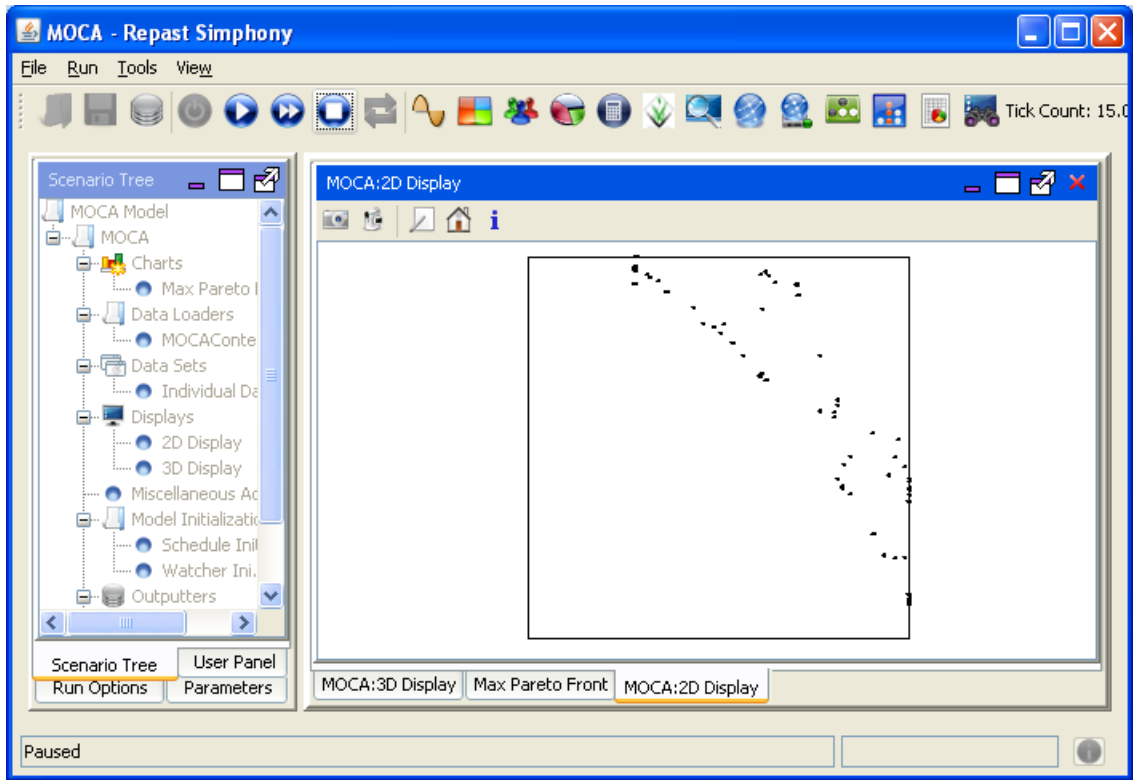
12.1.1 Analysis of found Pareto front

From the following sequence of found Pareto front along with evolution, we can see that MOCAT approximates the real Pareto front slower than it does in ZDT1, 2, and 4. For ZDT6, the individuals on the first front still could not compose a smooth curve at generation 30 which is done at

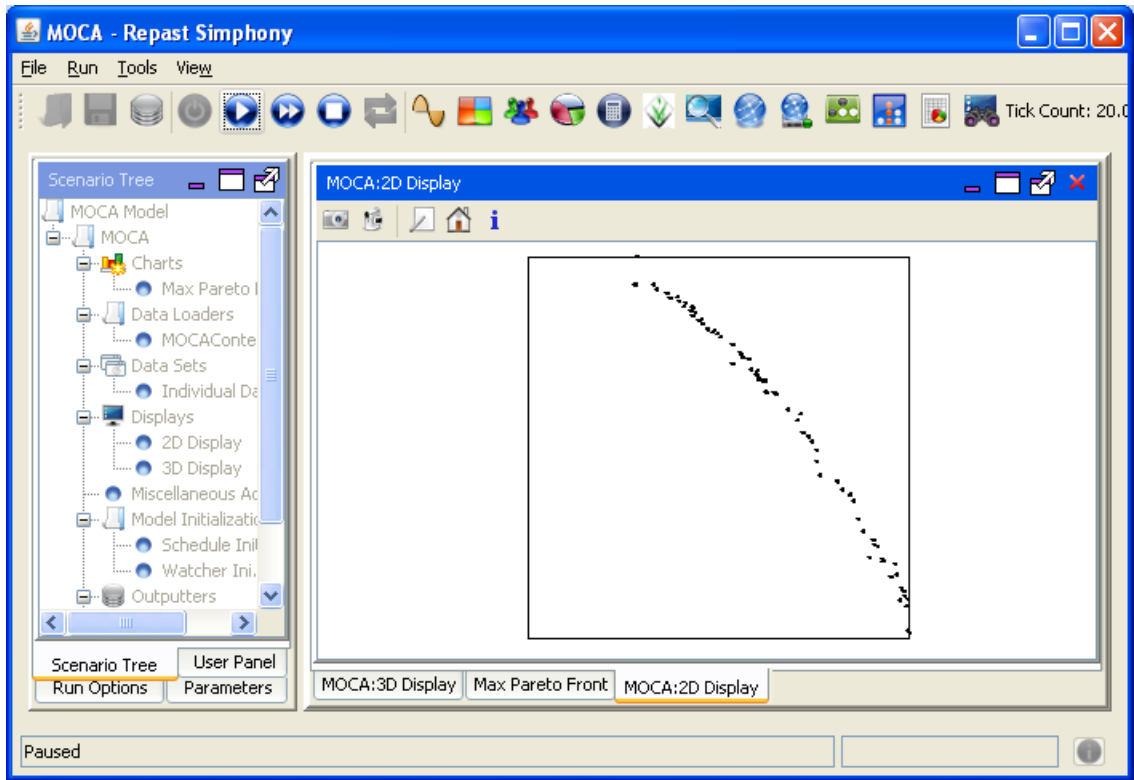
generation 40, while in other similar problems the curve has become smooth at generation 20; thereafter the evolution is twice slower. In fact, since ZDT4 has only 10 dimensions, while ZDT1 and 2 have 30, the curve is seen at generation 12 but does not reach far right corner yet.



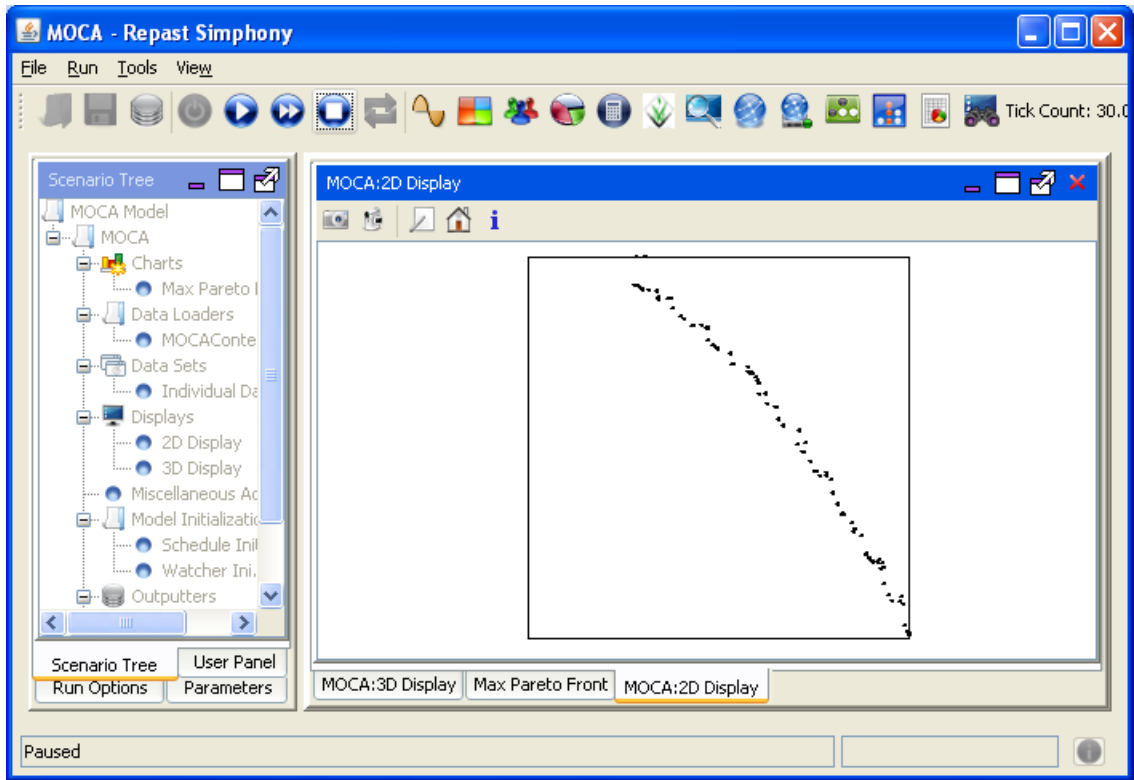
(a)



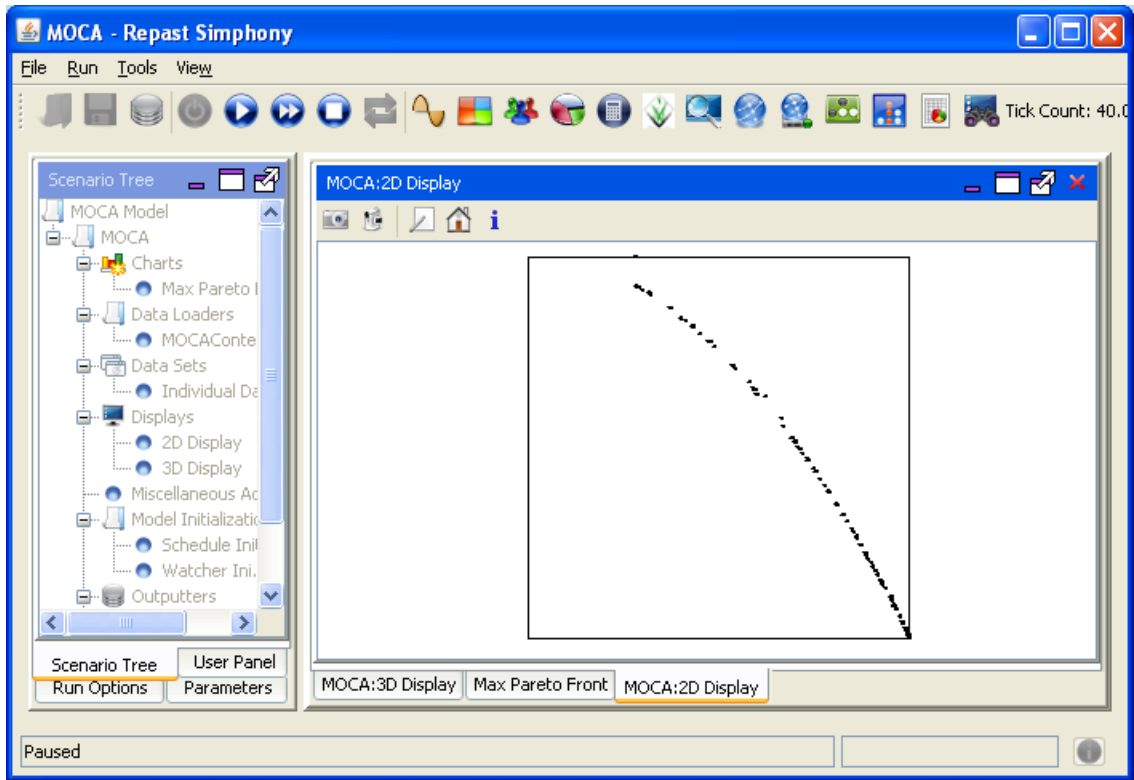
(b)



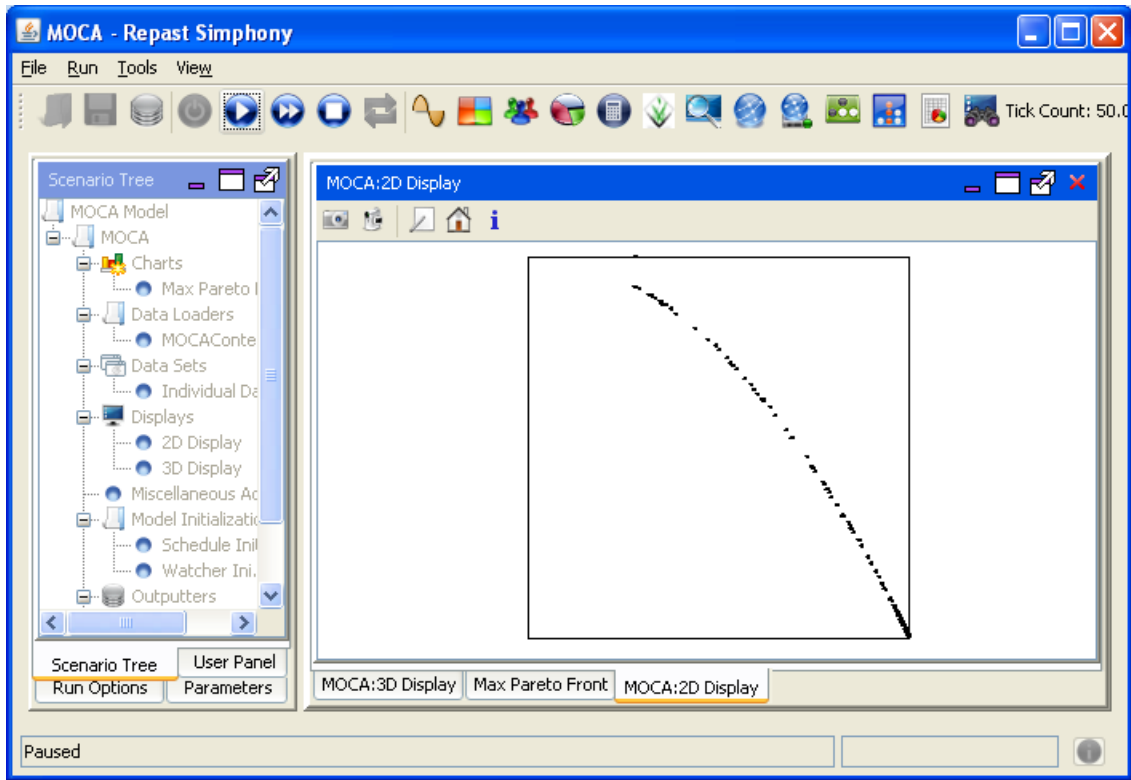
(c)



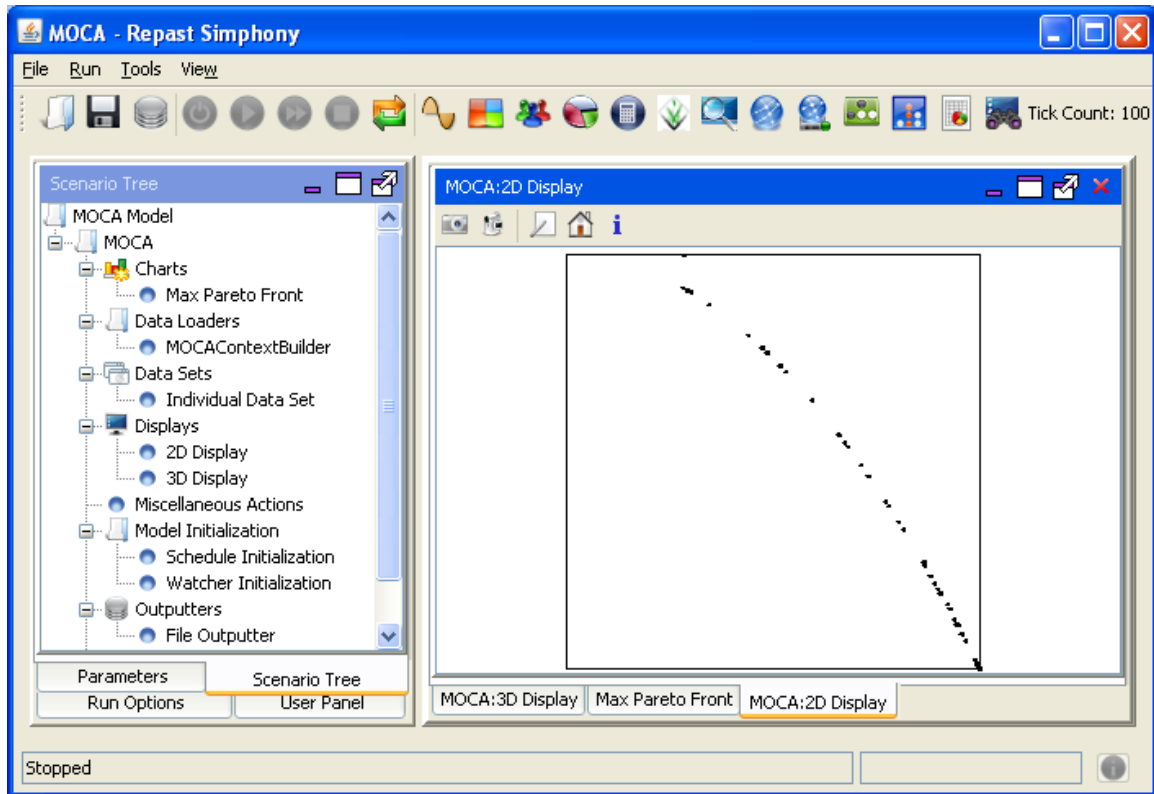
(d)



(e)



(f)



(g)

Figure 12.2 A series of screen copies along evolution (a-g)

Finally, the addition of found Pareto front of each run is represented. It is a close match to the real Pareto front.

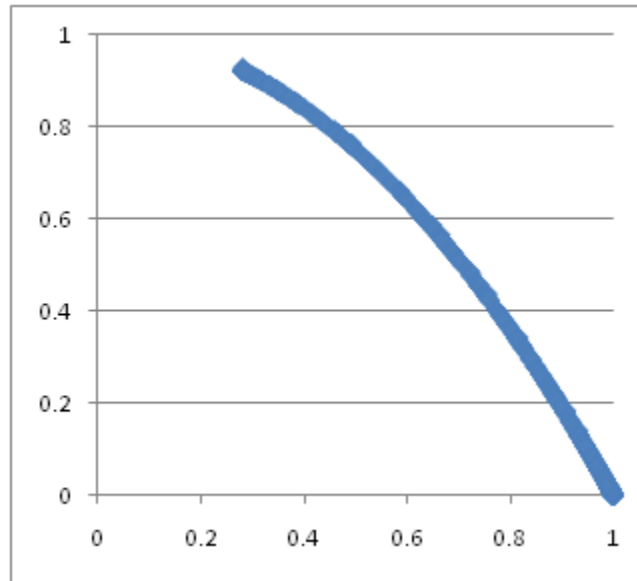


Figure 12.3 Overall found Pareto front

12.2 ZDT 610 with homogeneous Spread metrics

The runs were complete in an average of 16.31 seconds.

12.2.1 Performance of MOCAT

In table 12-1 the performance results for the 20 runs are given. The results in terms of mean error and standard deviation of the error are much better than any of the other benchmarked function performances. In addition, some individuals lay on top of the real Pareto front within the tolerance of floating precision. Once again, this may be due to the low dimension.

Table 12-1 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	7.18E-05	1.30E-04	0	7.92E-04	1.79E-04
Run #2	4.57E-06	3.32E-05	0	5.19E-04	9.48E-05

Run #3	5.17E-04	4.86E-04	-5.55E-17	0.001019	1.86E-04
Run #4	5.17E-04	0.001485	-2.78E-17	0.011747	0.001672
Run #5	1.07E-04	2.52E-04	3.54E-06	7.87E-04	2.12E-04
Run #6	1.02E-07	6.31E-06	3.77E-09	4.90E-04	5.10E-05
Run #7	7.04E-04	0.001079	0	0.00888	0.001709
Run #8	3.49E-07	1.47E-06	1.66E-08	1.10E-05	2.62E-06
Run #9	2.88E-04	3.10E-04	-5.55E-17	8.05E-04	1.66E-04
Run #10	2.43E-05	6.89E-05	-2.78E-17	7.49E-04	1.43E-04
Run #11	3.22E-05	2.95E-04	0	0.002017	4.57E-04
Run #12	3.68E-04	3.60E-04	-5.55E-17	8.82E-04	2.30E-04
Run #13	1.89E-05	2.66E-05	-5.55E-17	1.11E-04	2.47E-05
Run #14	3.27E-06	5.01E-06	0	1.38E-04	1.44E-05
Run #15	3.17E-06	1.78E-04	0	0.008146	0.001113
Run #16	6.25E-06	6.35E-05	-5.55E-17	3.23E-04	9.30E-05
Run #17	4.10E-07	2.87E-06	0	3.58E-05	6.56E-06
Run #18	7.99E-07	1.16E-05	0	1.19E-04	2.45E-05
Run #19	6.49E-05	1.38E-04	0	9.02E-04	9.33E-05
Run #20	1.99E-08	4.36E-05	0	9.80E-04	1.84E-04
mean	1.37E-04	2.49E-04	1.78E-07	1.97E-03	3.33E-04
stdev	2.12E-04	3.76E-04	7.72E-07	3.29E-03	5.11E-04

12.2.2 Statistics of Topologies using Spread metric

For the system there is no real difference in usage of the topologies based upon the 20 runs displayed in Table 12-2.

Table 12-2 Use Count of Topologies of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	21	20	23	15	21	13.83631
Run #2	20	14	23	30	13	13.45325
Run #3	17	18	29	17	19	12.8008
Run #4	20	19	24	11	26	13.39179
Run #5	31	22	18	11	18	12.68232
Run #6	20	25	17	23	15	13.9696
Run #7	18	23	13	24	22	17.74052

Run #8	15	14	25	26	20	13.94492
Run #9	24	23	12	17	24	6.847155
Run #10	26	20	22	20	12	5.605137
Run #11	27	20	24	15	14	6.30135
Run #12	21	14	22	23	20	11.98244
Run #13	18	15	26	21	20	12.4234
Run #14	17	25	18	21	19	14.96249
Run #15	27	15	17	15	26	14.27985
Run #16	19	20	24	19	18	12.91152
Run #17	24	19	13	20	24	15.33948
Run #18	22	12	21	27	18	10.4571
Run #19	13	29	19	18	21	8.703409
Run #20	12	17	25	22	24	6.017679
mean	20.60	19.20	20.75	19.75	19.70	
stdev	4.75	4.34	4.58	4.92	3.95	
t-test	Lbest vs.	0.174476	0.460789	0.295489	0.264633	
	Square vs.		0.145626	0.358436	0.356278	
	Octal vs.			0.260282	0.227041	
	Hex vs.				0.486315	

12.2.3 Behaviors of Knowledge Sources using Spread Metrics

While in ZDT5 the exploratory knowledge sources dominate search, here the exploitative knowledge sources again control the most individual agents over the course of the runs. However, the normative exploratory knowledge sources are still an important contributor. The relative counts of use are all significantly different from each other. However, we notice that the contribution of H is much worse than those in ZDT1, ZDT2, and ZDT4. This indicates that historical information does not bring as much contribution as it does in other three problems.

Table 12-3 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	253	1037	1726	541	88
Run #2	195	1118	1631	541	80

Run #3	180	1125	1625	609	75
Run #4	195	1010	1815	280	64
Run #5	296	1115	1682	606	77
Run #6	316	1024	1761	414	69
Run #7	229	995	1737	437	72
Run #8	257	1058	1737	499	83
Run #9	222	1040	1679	489	74
Run #10	187	960	1700	464	76
Run #11	224	999	1610	463	69
Run #12	206	1113	1602	558	69
Run #13	241	1075	1630	587	70
Run #14	249	990	1625	473	84
Run #15	265	942	1704	440	79
Run #16	201	1019	1633	483	54
Run #17	263	952	1712	474	70
Run #18	269	1022	1669	479	67
Run #19	169	999	1804	413	72
Run #20	190	1084	1266	552	84
mean	230.35	1033.85	1667.40	490.10	73.80
stdev	39.35	55.03	110.29	75.50	7.79
t-test	KS-N vs.	1.58E-34	1.41E-26	4.48E-14	7.37E-14
	KS-S vs.		1.11E-19	2.62E-24	4.45E-26
	KS-D vs.			1.21E-29	5.49E-24
	KS-H vs.				3.66E-16

12.2.4 Statistics of Non-Dominated solution production

In 12-4 it is clear that the LBEST topology is the best generator of non-dominated solutions across all knowledge sources. The LBEST domain knowledge tuple is the most effective combination here. Topographic knowledge varies most dramatically in effectiveness from run to run.

Table 12-4 Number of non-dominated solutions produced by Topology-Knowledge tuples

		N	S	D	H	T
Lbest	mean	47.2	212.8	340.45	106.55	17.85
	stdev	24.14343	48.49482	90.61746	33.59429	9.702279
Square	mean	48.55	195.8	325	92.1	12.75

	stdev	18.15569	46.99115	80.26207	24.7001	7.032818
Octal	mean	45.4	218.8	340	102.65	15.15
	stdev	16.35269	58.13741	78.31515	32.84297	7.617673
Hex	mean	46.9	201.85	330.45	95.8	14.5
	stdev	19.19128	55.82612	82.2272	30.67246	7.823716
Global	mean	42.3	204.6	331.5	93	13.55
	stdev	16.67996	44.71359	74.50998	22.31356	8.738752

Table 12-5 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.511513	0.227889	0.26617	0.315291	0.543545
Square	0.373959	0.239996	0.24696	0.268188	0.551594
Octal	0.360191	0.26571	0.230339	0.319951	0.502817
Hex	0.409196	0.276572	0.248834	0.320172	0.539567
Global	0.394325	0.218542	0.224766	0.239931	0.644926

12.3 ZDT 610 with homogeneous Hyper-volume metrics

The runs were complete in an average of 15.32 seconds.

12.3.1 Performance of MOCAT

Table 12-6 gives the performance statistic for the hyper-volume MOCAT system. The results are an improvement over the spread metric system. In addition, they outperform all of the benchmarked MOEAs given in chapter 6 on both the mean and standard deviation of the error.

Table 12-6 Statistics for the fitness errors of ending solutions

statistics for the fitness errors of ending generation					
	median	Mean	min	max	stdev
Run #1	5.58E-07	2.59E-05	0	4.22E-04	8.03E-05
Run #2	8.29E-07	5.56E-06	0	9.41E-05	1.22E-05
Run #3	1.58E-05	3.27E-05	-2.78E-17	6.89E-05	1.75E-05
Run #4	5.52E-07	4.91E-06	-5.55E-17	3.89E-04	4.02E-05
Run #5	2.30E-06	1.00E-05	9.73E-08	1.75E-04	2.92E-05
Run #6	1.10E-06	3.57E-06	0	5.95E-05	7.72E-06

Run #7	1.13E-04	1.83E-04	0	5.70E-04	1.90E-04
Run #8	3.58E-07	3.76E-06	-4.16E-17	7.73E-05	9.28E-06
Run #9	3.35E-06	2.65E-05	-2.78E-17	1.04E-03	1.52E-04
Run #10	1.87E-06	6.34E-06	-2.78E-17	5.89E-05	1.37E-05
Run #11	3.92E-05	7.70E-05	0	1.54E-04	3.89E-05
Run #12	8.12E-06	8.12E-05	-4.16E-17	0.002913	3.12E-04
Run #13	7.93E-06	1.46E-05	-5.55E-17	3.36E-05	9.27E-06
Run #14	4.75E-04	9.59E-04	-5.55E-17	0.003248	8.37E-04
Run #15	7.02E-06	2.75E-05	-5.55E-17	2.41E-04	4.77E-05
Run #16	3.85E-04	7.38E-04	0	1.69E-03	3.57E-04
Run #17	3.10E-07	5.26E-06	-5.55E-17	1.33E-04	2.47E-05
Run #18	5.93E-04	1.29E-03	0	1.02E-02	9.93E-04
Run #19	5.10E-06	3.19E-05	6.09E-07	4.48E-04	7.22E-05
Run #20	1.64E-05	5.99E-05	-5.55E-17	9.07E-04	1.49E-04
mean	8.38E-05	1.79E-04	3.53E-08	1.15E-03	1.70E-04
stdev	1.73E-04	3.57E-04	1.33E-07	2.28E-03	2.68E-04

12.3.2 Statistics of Topologies using Hyper-volume metric

While the spread metric system did not exhibit any statistical differences in topology usage, the hyper-volume system does as shown in Table 12-7. The Octal topology is used significantly more often than both the square and LBEST topologies.

Table 12-7 Use Count of Topologies of each run

	Using Hyper-volume metric					Hyper-volume Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	13	28	31	11	17	2.829497
Run #2	15	16	24	22	23	4.037976
Run #3	19	24	19	15	23	3.842902
Run #4	20	21	24	22	13	5.142704
Run #5	18	21	18	15	28	4.617802
Run #6	22	23	19	19	17	12.46701
Run #7	21	21	23	15	20	4.124842
Run #8	17	16	28	22	17	10.75945
Run #9	22	13	18	27	20	11.05558

Run #10	17	16	18	22	27	4.095092
Run #11	25	18	20	20	17	3.616337
Run #12	17	16	13	33	21	4.194218
Run #13	19	12	28	18	23	4.135767
Run #14	29	19	25	11	16	4.509909
Run #15	16	12	22	27	23	9.179675
Run #16	20	12	32	19	17	2.066416
Run #17	25	15	19	19	22	4.888691
Run #18	13	29	21	21	16	5.238227
Run #19	19	21	22	11	27	4.694531
Run #20	20	19	16	20	25	4.038914
mean	19.35	18.60	22.00	19.45	20.60	
stdev	3.90	4.85	4.84	5.49	4.16	
t-test	Lbest vs.	0.301368	0.035599	0.474391	0.172879	
	Square vs.		0.018443	0.30803	0.090468	
	Octal vs.			0.06859	0.172604	
	Hex vs.				0.235875	

12.3.3 Behaviors of Knowledge sources using Hyper-volume Metric

As with the spread metric system all influence counts are significantly different from each other. Also, the emphasis is on exploitation knowledge source even though the use of the history knowledge source is down while the normative exploratory knowledge source usage is up.

Table 12-8 Using Hyper-volume metric #Individuals influenced by KS

Using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	249	1052	1588	483	110
Run #2	244	1054	1676	452	88
Run #3	200	1067	1595	543	62
Run #4	224	1021	1678	471	78
Run #5	293	1103	1634	499	74
Run #6	282	1007	1709	399	75
Run #7	214	990	1695	475	64
Run #8	229	1105	1638	581	68
Run #9	253	990	1669	491	76

Run #10	230	1147	1602	580	93
Run #11	194	1054	1654	457	86
Run #12	240	1016	1565	483	74
Run #13	215	1066	1615	461	75
Run #14	194	1042	1694	466	85
Run #15	206	1136	1563	478	63
Run #16	267	1073	1624	637	65
Run #17	197	1095	1259	555	88
Run #18	203	1143	1739	448	74
Run #19	241	1033	1684	564	89
Run #20	195	985	1681	383	68
mean	228.50	1058.95	1628.10	495.30	77.75
stdev	29.10	49.22	97.18	61.76	11.83
t-test	KS-N vs.	1.62E-34	1.51E-26	2.77E-16	1.04E-17
	KS-S vs.		5.64E-20	5.12E-28	1.39E-28
	KS-D vs.			2.79E-30	3.87E-25
	KS-H vs.				2.44E-18

12.3.4 Statistics of Topology-Knowledge tuple

In the spread metric version of MOCAT the most productive topology was the LBEST. Here the most productive topology is that of the OCTAL. The octal domain tuple is the best performer as shown in table 12-9. Also, in table 12-10 it is clear that topographic knowledge varies the most between runs as with the spread metric. However, the other exploratory knowledge source, normative, exhibits more variability here than for the spread system.

Table 12-9 Generation of non-dominated individuals by Topology-Knowledge tuples

		N	S	D	H	T
Lbest	mean	45.15	205.25	316.6	89.85	15.05
	stdev	19.43755	45.06122	59.73661	26.19014	9.659874
Square	mean	36.05	197.8	303.95	89.85	12.6
	stdev	16.9441	56.07382	87.70613	27.62584	9.005262
Octal	mean	58	232.15	357.5	110	19.55
	stdev	24.06789	54.09278	79.006	39.4915	10.51553
Hex	mean	48.15	208	311.35	99.1	16.35

	stdev	21.73591	64.12898	87.61776	33.33388	10.36327
Global	mean	41.15	215.75	338.7	106.5	14.2
	stdev	17.97447	51.9087	73.04368	30.42938	7.487533

Still, MOCAT is really dynamic because the data have high variation.

Table 12-10 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.430511	0.219543	0.188682	0.291487	0.641852
Square	0.470017	0.283487	0.288554	0.307466	0.714703
Octal	0.414964	0.233008	0.220996	0.359014	0.537879
Hex	0.451421	0.308312	0.281412	0.336366	0.633839
Global	0.436804	0.240597	0.215659	0.285722	0.527291

12.3.5 Analysis of found Pareto front

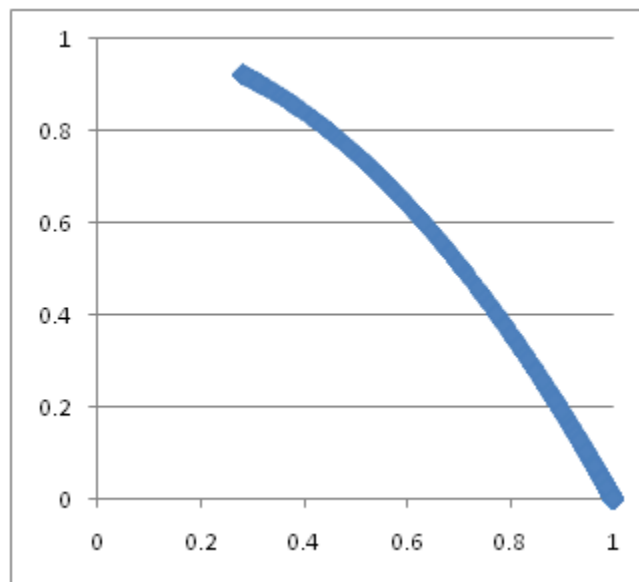


Figure 12.4 Overall found Pareto front

The result is close to the one that in ZDT6 10 using pure spread metric.

12.4 ZDT 610 with combined metrics

The runs were complete in an average of 15.42 seconds.

12.4.1 Performance of MOCAT using the combined metrics.

The performance of the combined system exceeds that of either single metric system. In fact the mean error is one full magnitude better than the hyper-volume case. Also, it clearly outperforms all other benchmarked functions for this problem.

Table 12-11 Statistics for the fitness errors of ending solutions

Statistics for the fitness errors of ending generation					
	median	mean	min	max	stdev
Run #1	3.44E-05	5.53E-05	-4.16E-17	3.59E-04	6.82E-05
Run #2	4.56E-08	1.50E-04	-5.55E-17	0.001742	3.61E-04
Run #3	1.82E-08	7.84E-07	3.67E-10	9.74E-06	2.25E-06
Run #4	5.53E-06	7.15E-06	-4.16E-17	3.80E-05	7.61E-06
Run #5	9.04E-08	1.63E-05	3.40E-09	3.73E-04	6.12E-05
Run #6	4.92E-06	7.06E-05	-2.78E-17	8.18E-04	1.18E-04
Run #7	2.19E-05	1.04E-04	-4.16E-17	0.001001	1.65E-04
Run #8	4.23E-07	1.67E-06	1.33E-08	9.90E-06	2.23E-06
Run #9	6.56E-05	6.12E-05	-2.78E-17	1.10E-04	2.60E-05
Run #10	1.82E-08	4.03E-05	-2.78E-17	0.00165	2.39E-04
Run #11	4.69E-08	3.70E-07	1.86E-09	2.87E-06	5.74E-07
Run #12	7.71E-06	2.28E-05	-2.78E-17	2.75E-04	3.95E-05
Run #13	4.56E-08	7.73E-05	-4.16E-17	0.001024	2.49E-04
Run #14	4.54E-06	4.94E-05	-5.55E-17	8.76E-04	1.48E-04
Run #15	1.91E-05	1.77E-04	-5.55E-17	0.002931	4.83E-04
Run #16	8.14E-06	3.30E-05	-5.55E-17	5.91E-04	9.77E-05
Run #17	4.56E-08	1.81E-07	7.68E-09	1.09E-05	1.12E-06
Run #18	1.82E-08	2.28E-06	-2.78E-17	1.48E-04	1.66E-05
Run #19	3.57E-07	2.32E-05	-5.55E-17	5.01E-04	9.75E-05
Run #20	2.37E-05	8.97E-05	-4.16E-17	0.001395	2.17E-04
mean	9.83E-06	4.91E-05	1.33E-09	6.93E-04	1.20E-04
stdev	1.61E-05	4.93E-05	3.28E-09	7.47E-04	1.29E-04

12.4.2 Statistics of Topologies using Spread Metrics

From table 12-12 the square topology is used significantly more than the LBEST topology in the spread metric phase of the solution process. When the spread metric was use alone there was no difference in the usage of topologies. Likewise, the Sqare metric significantly outperforms LBEST in the hyper-volume phase..

Table 12-12 Use Count of Topologies using spread metric of each run

	Using Spread metric					Spread Metric Value
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	6	9	5	11	19	15.08273
Run #2	5	8	19	6	12	6.585798
Run #3	8	13	11	12	6	15.04582
Run #4	11	14	7	7	11	10.67421
Run #5	10	12	9	11	8	13.0653
Run #6	10	8	16	10	6	14.5444
Run #7	10	12	9	12	7	1.660286
Run #8	14	14	6	9	7	14.66218
Run #9	7	12	4	12	15	10.88384
Run #10	12	14	5	8	11	7.123939
Run #11	8	12	12	12	6	2.022871
Run #12	7	6	14	12	11	6.633822
Run #13	10	11	11	11	7	25.05995
Run #14	10	9	13	12	6	1.874203
Run #15	11	10	9	8	12	12.82533
Run #16	8	7	11	11	13	14.72305
Run #17	10	11	11	7	11	14.75448
Run #18	6	13	10	7	14	24.43156
Run #19	8	9	11	11	11	7.037685
Run #20	10	10	11	5	14	12.00201
mean	9.05	10.70	10.20	9.70	10.35	
stdev	2.18	2.35	3.64	2.28	3.55	
t-test	Lbest vs.	0.015328	0.123232	0.18746	0.091829	
	Square vs.		0.309161	0.095513	0.361231	
	Octal vs.			0.307776	0.449213	

Hex vs.	0.253537
----------------	----------

Table 12-13 Use Count of Topologies using hyper-volume metric of each run

	Hyper-volume metric				Hyper-volume metric value	
	T-LBEST	T-SQUARE	T-OCTAL	T-HEX	T-GLOBAL	
Run #1	4	8	11	8	19	4.070232
Run #2	13	5	9	14	9	3.166685
Run #3	10	9	11	14	6	5.157879
Run #4	9	16	7	11	7	5.310816
Run #5	9	13	11	9	8	9.433638
Run #6	9	13	7	7	14	5.412966
Run #7	11	10	7	12	10	2.594558
Run #8	9	11	11	8	11	9.331143
Run #9	6	16	19	1	8	3.465811
Run #10	9	14	10	8	9	3.911616
Run #11	7	16	8	10	9	4.277458
Run #12	11	12	9	7	11	3.189111
Run #13	11	10	9	9	11	5.876553
Run #14	8	14	8	12	8	3.576724
Run #15	13	9	9	11	8	3.234325
Run #16	5	9	11	14	11	3.613724
Run #17	14	7	17	6	6	5.67747
Run #18	7	13	8	12	10	3.845922
Run #19	6	11	14	16	3	4.918367
Run #20	9	3	8	15	15	3.75549
mean	9.00	10.95	10.20	10.20	9.65	
stdev	2.63	3.50	3.14	3.54	3.41	
t-test	Lbest vs.	0.030051	0.104677	0.121854	0.257306	
	Square vs.		0.245575	0.257772	0.126704	
	Octal vs.			0.5	0.304022	
	Hex vs.				0.314367	

12.4.3 Behaviors of Knowledge sources with combined metrics

Tables 12-14 and 12-15 give the number of individuals controlled by each of the knowledge sources during the problem solving process. The usage of all knowledge sources in both phases is

statistically significant. Notice that the exploratory knowledge sources are used more often with the spread metric phase than with the hyper-volume metric as we expected.

Table 12-14 Using Spread metric #Individuals influenced by KS

using Spread metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T
Run #1	187	566	814	318	44
Run #2	127	533	833	250	69
Run #3	160	545	802	277	55
Run #4	193	505	846	230	43
Run #5	141	553	828	258	52
Run #6	149	527	816	274	43
Run #7	152	548	778	319	48
Run #8	190	532	820	227	61
Run #9	120	511	764	245	53
Run #10	127	526	793	237	60
Run #11	189	529	845	244	57
Run #12	129	578	812	274	50
Run #13	174	513	860	253	58
Run #14	129	462	829	267	55
Run #15	175	510	836	198	50
Run #16	141	522	794	303	45
Run #17	180	555	799	271	49
Run #18	135	585	682	256	56
Run #19	174	476	822	309	53
Run #20	168	533	800	303	63
mean	157.00	530.45	808.65	265.65	53.20
stdev	24.18	29.71	37.10	31.76	6.82
t-test	KS-N vs.	5.52E-33	3E-36	3.29E-14	5.85E-15
	KS-S vs.		4.6E-25	2.32E-26	1.92E-26
	KS-D vs.			1.82E-35	6.56E-28
	KS-H vs.				2E-18

Table 12-15 Using Hyper-volume metric #Individuals influenced by KS

Using Hyper-volume metric #Individuals influenced by KS					
	KS-N	KS-S	KS-D	KS-H	KS-T

Run #1	28	493	765	316	23
Run #2	57	529	808	157	30
Run #3	110	496	894	271	28
Run #4	77	531	832	215	31
Run #5	103	473	880	220	27
Run #6	89	495	890	276	35
Run #7	66	503	874	225	31
Run #8	71	440	884	297	26
Run #9	35	561	864	260	29
Run #10	62	497	684	235	33
Run #11	111	456	866	222	24
Run #12	95	503	835	245	19
Run #13	66	482	708	238	31
Run #14	82	456	836	217	33
Run #15	62	492	847	195	18
Run #16	19	458	796	312	35
Run #17	100	508	868	283	38
Run #18	79	561	492	307	49
Run #19	63	470	857	240	32
Run #20	85	530	831	275	27
mean	73.00	496.70	815.55	250.30	29.95
stdev	25.04	32.68	92.78	40.85	6.67
t-test	KS-N vs.	3.2E-33	1.46E-20	4.42E-17	1.62E-07
	KS-S vs.		2.54E-13	7.47E-22	4.9E-25
	KS-D vs.			9.39E-20	1.42E-19
	KS-H vs.				3.07E-16

With no exception, all topologies are listed at 5 levels: D, S, H, N, and T.

12.4.4 Statistics of Topology-Knowledge tuple non-domination production

For the combined system the most productive topology is Square. This is interesting since the most productive one for the hyper-volume system was Octal, and that for the spread metric system was LBest. Square represents a middle ground between the two. The combined system produces less variability in the generation of non-dominated solution from run to run which attests for its stability.

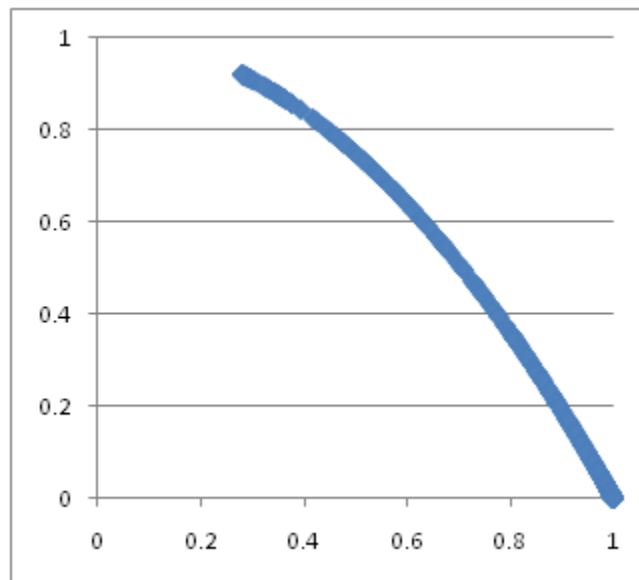
Table 12-16 Non-dominated solutions produced by Topology-Knowledge tuple

		N	S	D	H	T
Lbest	mean	40.4	185.15	300.75	96.95	15.05
	stdev	11.59583	39.35302	80.77185	20.96984	7.937088
Square	mean	53.9	221.8	342.4	112	20.6
	stdev	20.98596	51.04343	89.26035	25.42481	8.5557
Octal	mean	44.7	211.9	329.8	105.95	16.7
	stdev	14.6327	47.61236	70.83755	29.9868	8.118206
Hex	mean	47	198.45	325.1	102.45	16.35
	stdev	17.82665	38.62025	79.25236	26.79645	7.321741
Global	mean	44	209.85	326.15	98.6	14.45
	stdev	16.52749	67.59265	93.45208	37.57995	6.047836

Table 12-17 Randomness of Topology-Knowledge tuple

stdev/mean	N	S	D	H	T
Lbest	0.287025	0.212547	0.268568	0.216295	0.527381
Square	0.38935	0.230133	0.26069	0.227007	0.415325
Octal	0.327353	0.224693	0.214789	0.283028	0.48612
Hex	0.37929	0.194609	0.243778	0.261556	0.447813
Global	0.375625	0.3221	0.286531	0.381135	0.418535

12.4.5 Analysis of found Pareto front

**Figure 12.5 Overall found Pareto front**

Still, 20 runs closely cover the real Pareto front.

12.5 Summary of ZDT6

In term of the Pareto front, ZDT6 is similar to ZDT1, 2, and 4. The combined system outperforms each of the individual ones. The problem combines most of the problem features together and is challenging for MOEAs. In the detailed example it is clear that our system takes more time to form the approximation and then attract solutions to the area via the knowledge sources. There is now sufficient complexity in the problem to generate the synergy between the two phases of the problem solving process. The combined system clearly outperforms all of the other MOEAs as a result.

CHAPTER 13 CONCLUSION AND FUTURE WORK

The basic premise of the thesis is that Cultural Systems deal with multi-objective problems on a daily basis. So there should be some inner mechanisms that allow them to effectively solve such problems over time. Our goal is to extend the basic Cultural Algorithm framework to incorporate multi-objective problem solving into it. Then, the system was applied to a variety of benchmark problems that posed problems for various other multi-objective evolutionary optimizers. The goal was to identify the computational features that make cultural systems so effective in this regard.

The key aspect to a successful solution for any multi-objective problem is the maintenance of diversity within the populations so as to provide a search that was broad enough to identify the Optimal Pareto front for a given problem. There were two different performance metrics in the literature (Bastos-Filho and Miranda 2011) that were used to assess the spread of solutions along the curve, the spread metric and the hyper-cube metric.

In our Cultural systems, MOCAT, knowledge sources determined the location of points in the search space. The knowledge source performances were assessed in terms of the quality of spread that they produced. If a knowledge source was successful in influencing the positioning of an agent in the space, then agents that are connected to it within the social network, the social fabric, were more likely to use it as well. That way influential knowledge sources can spread through the social network and influence the direction of the search process.

The action of one set of knowledge sources can set the stage for the action of others. From this perspective there are two categories of knowledge sources that were previously identified relative to optimization problem solving, exploratory and exploitative knowledge sources. It was conjectured that

the different spread metrics might be more likely to selectively reward different knowledge sources and social networks that connected the problem solving agents together.

In the MOCAT system, the goal was to allow it to learn the frequency with which to apply the various knowledge source along with the social fabric through which the knowledge sources directed their influences. The goal was to identify certain problem features that posed problems for multi-objective systems and to observe how the MOCAT system configured itself relative to those problems. The results were then compared with well-known benchmark performance data on these problems.

While in a traditional MOEA the spread metrics provided the main source of information in the search process, the situation was different for the MOCAT system. Here the metrics were used to evaluate the ability of the knowledge sources to generate and distribute solutions within the problem space. The key is that the knowledge sources collected information from the search activity that was done independently of the spread metric assessment.

The system was run against a benchmark set of problems that represented categories of hard problems. These categories include convex and concave optimal Pareto Fronts, multi-modality, discretization of the curve, uneven distribution of points along the front, and uneven distribution between the optimal front and sub-optimal ones.

Some basic results that were produced are as follows:

1. The MOCAT system was very effective in the generation of an appropriate configuration for solving problems with different combinations of these features. Even for a given problem, as information was added to the knowledge sources, adjustments in the topologies could be made effectively.

2. As the complexity of the problems increased in terms of the number of problem features, the MOCAT system's relative performance increased.
3. A problem with just a single problem feature, such as ZDT1 and ZDT5, was often effectively solved by just using one metric guide the solution process. However, if there were multiple problems, combining the two metrics together produced a synergy that outperformed each single metric based system.
4. This synergy resulted from the fact that they rewarded spread production in different ways. The spread metric focused on global distribution while the hyper-volume tended to support local optimization.
5. The configuration of the top performing MOCAT system varied markedly from one problem to the next.

The results suggest three possible directions for future work:

1. The structure of a social system may reflect the nature of the problems that are presented to it. This can be a useful tool in comparing and contrasting cultural systems in terms of the properties of the multi-objective decision-problems that they are likely to face.
2. Also, by characterizing real-world problem in terms of these problem properties we may be able to prescribe a particular Cultural Algorithm system that is best suited to solve the problem.
3. Also, we can observe how slight changes in problem constraints can impact the social fabric and knowledge sources used to solve the problem.
4. The population may have an adaptive size according to how many Pareto fronts have been found so that the population increases while just a few Pareto fronts are found but decreases while too many Pareto fronts are found.

5. MOCAT should be tested on real-world problems to test its potentials.

APPENDIX—SYSTEM IMPLEMENTATION

In this chapter we begin by describing the Repast Symphony software that will represent the population space and upon which the rest of the MOCAT system is built. Next we provide a description of the MOCAT interface. Finally, we will demonstrate how MOCAT displays the results of the problem solving process.

As mentioned above, multiple agent systems (MAS) are often used in solving MOPs. A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents who have only local views but is unaware of full global view of the system. While in the CA knowledge is stored in the belief space, the population space of CA can be implemented as a Multi-Agent System. In all of our previous implementations, all individuals are autonomous and there is no hierarchical structure among them here although there could be.

In this work we took advantage of a mature MA framework, Repast Symphony (Argonne National Laboratory 2010), in our latest CAT implementation. Repast Symphony, as an integrated environment for building special purpose simulation tools for modeling organizational systems, provides various services that enable the developer to easily control different behaviors as simulating behaviors, graphical representation, statistics, and animation. Such facilitates provide the building blocks that we need to create simulation models for different problem solving domains.

For Repast, there were several concrete implementations of its conceptual specification. While all of them share the same core services, such as Repast.NET which is used with the Microsoft.Net framework and Repast Py which uses Python Scripting. Repast J is the concrete specification used here because it was the reference implementation that defines the core services used in MOCA and the

earlier CAT systems. In this text Repast J and Repast are used interchangeably unless Repast Symphony specifically refers to a latest version that is used in our newest CAT implementation. In terms of implementation, both CA and Repast use a timer to evolve the populations; with the consideration of improving computing efficiency by combining autonomous agents and high-level guidance of CA.

One of the advantages of Repast is its ability to support real-time dynamic visualization of the population space and the belief space, to query for details of specific individuals, and to produce a real-time display of knowledge source performance.

Repast Symphony, while containing all the core services that Repast J has supplied, provides various services that enable the developer to easily control different behaviors relating to simulation, graphical representation, statistics, and animation. In addition, by taking advantage of the 3D interaction ability of Repast Symphony, it is very straightforward to use the mouse to drag and rotate the graphical representation in MOCAT. Such facilities provide the building blocks that we need to create simulation models for different problem solving domains.

The following series of screen shots shows the 3D pictures obtained when the objective coordinate system is rotated so that it is easier to observe the flat surface in the objective space. Figure 0.1 shows two coordinates in domain space and one value in objective space for generation 36 when MOCAT is evolving DTLZ1. Certainly, we are able to represent any combinations of coordinates in both domain space and objective space for individuals. In this figure, we are able to visually check the solution surface, i.e., the current Pareto front and we can easily tell that the solutions are good in terms of individual's performance.

In the figure, colors are used to display the Pareto fronts discovered during the problem solving process. The magenta points indicate that the representing individual is on Pareto front 1; the color blue indicates Pareto front 2, etc. Please note that in this implementation we represent all individuals in the cube whose edges are shown in white lines and outliers are dragged back into the cube and therefore stick to boundary edges. In addition, in this figure the three coordinates are normalized for convenience of observation.

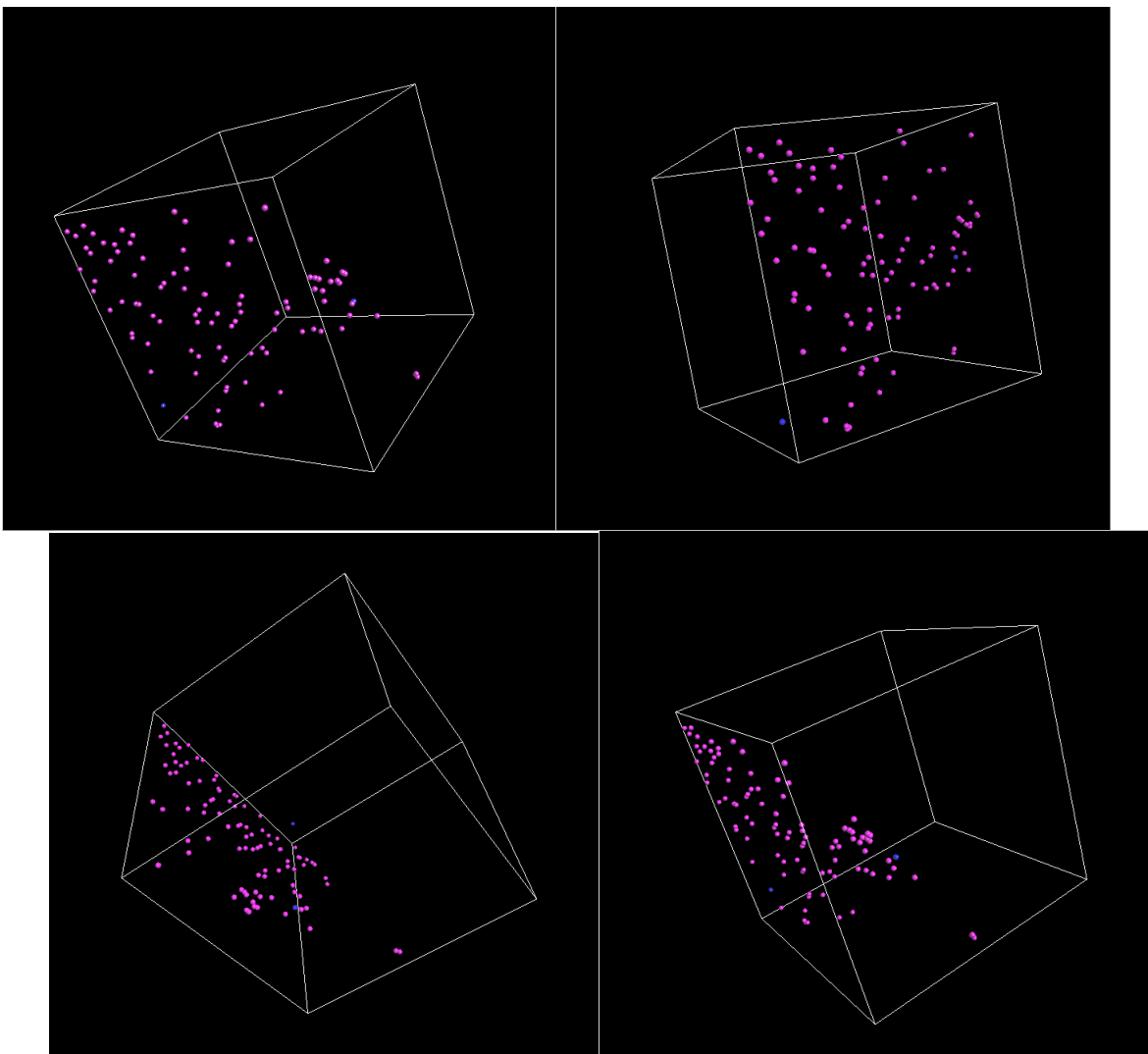


Figure 0.1 Screen copies of MOCAT's 3D visualization

To take advantage of the facilitates that Repast Symphony provides in addition to normal Repast functionalities, some specific programming techniques have to be implemented. For example, in the new system, individuals, i.e., agents, provide a consistent programming interface for the Symphony system to query its internal status.

Moreover, in the concrete system design, system performance was focused since evolution algorithms are calculation-intensive and consume much more computation power. For example, to select an individual with a good performance for single-objective evolution, a linear array sort is sufficient. However, in order to arrange individuals onto a Pareto front, comparisons have to be done on each dimension so that sort needs more time.

Therefore, in this section, the concrete implementation details are explained with the intention to be a self-contained programming guide and user manual. First, specific programming concepts of Repast Symphony are introduced. Second, its concrete programming environment is explained. Third, particular implementation details of new MOCAT are explained. Finally, a simple how-to-use guide is supplied for MOCAT since Repast Symphony has a crowded interface which, though efficient, may pose a challenge to first-time users.

13.1 Introduction to Repast Symphony

Repast Symphony is an open source, agent-based modeling toolkit. It has the following features that facilitate our new implementation.

- Ready-to-use dynamic and interactive visualization
- Point-and-click to show properties of any agent

- Charts showing data series

Additionally, there are potentials that make our future work easier.

- Logging of data series so that we can remove messy hard coding in the source code to record data in specific format;
- Automated connections to a variety of optional external tools including Matlab;
- A range of data storage "freeze dryers" for model check pointing and restoration including XML file storage, text file storage, and database storage; so that we are able to pause experiment at one time and resume it later;
- A fully concurrent multi-threaded discrete event scheduler;
- Libraries for genetic algorithms, neural networks, regression, random number generation, and specialized mathematics that makes the comparison of MOCAT and other evolutionary algorithms practical;
- An automated Monte Carlo simulation framework which supports multiple modes of model results. However, we have already implemented a high-quality random number generator which will be used in its place.
- In the model execution environment a mouse click on an agent will reveal all of its properties to the public. No coding is needed.

There are two options to install Repast Symphony: as a plug-in of Eclipse or as a stand-alone application which actually is a subset of Eclipse. While the former needs fine and detailed work though the download volume is smaller, the latter installation approach is suggested for its convenience. The installer at the website is for the newest version 1.2.0; an installation guide can be found online (Anon. 2008b).

The core data structure in Repast Symphony is called a Context. The Context is a simple container based on set semantics that means that it provides the basic infrastructure to define a population and the interactions of that population. In reality, Context does not exert such interaction constraints over its content. In other words, any population can be added into a Context.

The particular structure of populations, i.e., how agents are connected to and interact with each other, is defined by a Projection. Projections are data structures designed to define and enforce relationships between agents within a given Context. Context and Projection are the two most important concepts in Repast Symphony.

Each Context can have an arbitrary number of Projections associated with it which means that within each Context, the agents can create an arbitrary number of types of relationships with each other. This ability is intended to release the constraint of writing agents that are designed to work with grids or networks specifically.

Projection is designed to work with arbitrary objects and switching between projections does not require any changes of objects so that it is easy to be done. In terms of coding, this means that no code changes are required to allow a projection to work with a particular agent.

In reality, in Repast Symphony, a context can be complex. For example, a context can contain sub-Contexts. As an example, a farming village consists of multiple families; in this case each family is a sub-Context of the larger village. Membership in a Context is inherited by definition. This hierarchical structure allows for the model designer to consciously define the granularity of the model. In CA, there is no hierarchy in population space, (nor in belief space,) therefore we can practically skip this note. Membership in a Context is designed to be fluid. As a result, agents that are designed to engage in a

behavior on the basis of their environment can switch behaviors very easily when they migrate into another context.

An illustration of the relationships among Context, Projection, and agents are shown in Figure 0.2, taken from Repast Symphony's online document (Anon. 2008a).

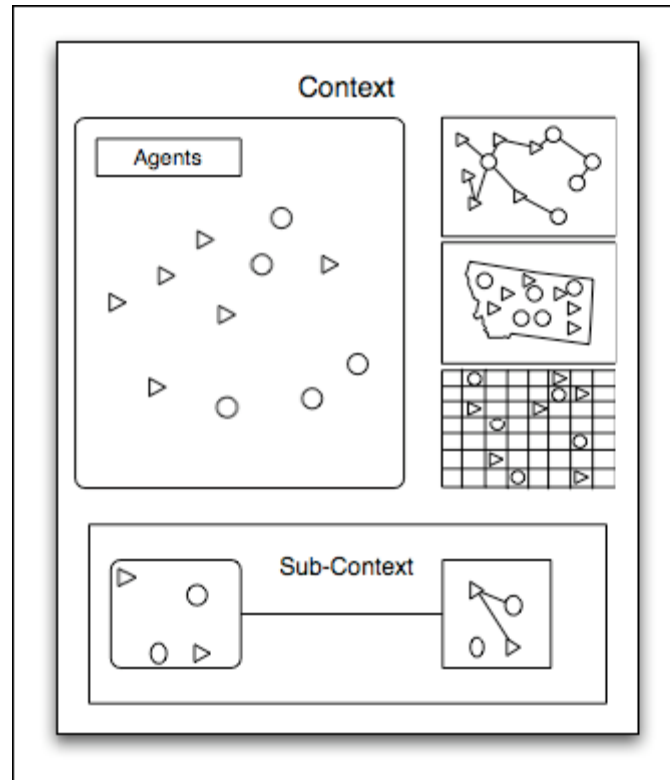


Figure 0.2 Context, Projection, and agents in Repast Symphony

A Context will be initialized by Repast Symphony automatically at the very beginning of startup and will stay there for the whole execution session. During the initialization, all setup is expected to be done, including creating and setup of Projections, which is not indispensable for a Repast Symphony application to run. In order to visualize agents, either a grid or a continuous space is needed to be created and all agents have to be added into them to be processed.

In addition to charts, we can consider using other multi-dimensional visualization. There are two kinds of visualization interfaces, 2D and 3D. Creating 2D visualization, which is called space here, is generally easy to do. However, creating 3D spaces requires an explicit mapping layer between the concrete data and objects that are used by the spaces. For example, 3D spaces require a literal definition of three dimensions when an agent is added into it; otherwise, in runtime Java exceptions complaining about non-affine operations will be thrown out when the matrix calculations are in fact affine. The development environment created by the installer does not support runtime debugging well in addition to the fact that JNI is used in 3D visualization which further prevents step debug; thereafter the mechanism causing the exceptions is unclear.

13.2 Introduction to Repast IDE

There are two ways to set up the Repast Symphony Integrated Development Environment (IDE). One is to download the plug-in from the update site to an existing Eclipse IDE; another one, which is easier for beginners, is to download Repast Symphony IDE which embeds an Eclipse IDE inside. The IDE plugin from the update site provides the same functionality of the stand-alone installers, while permitting the flexibility to add Repast Symphony to an existing Eclipse installation. This is useful for advanced programmers who already have additional Eclipse components and do not want to use the Eclipse framework that is included with the installers.

In spite of the same core functionality of two configurations, the plug-in configuration does not require the development environment components or source code from the repository, whereas the installer configuration needs the development environment components and comes with the source code from the repository. Therefore, the installer configuration is recommended for our purpose.

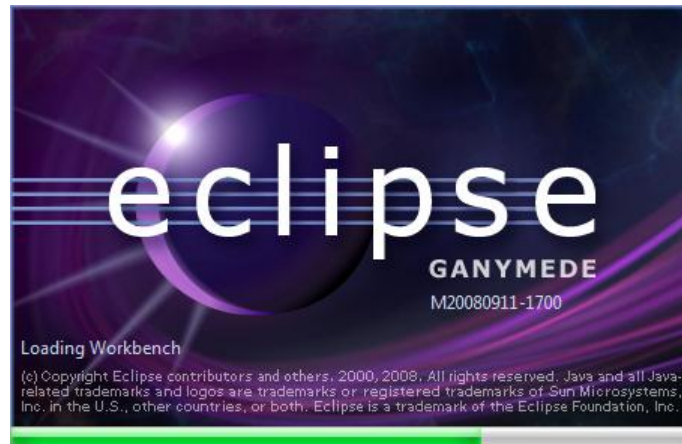


Figure 0.3 Splash window of the Repast Simphony

The integrated development environment (IDE) is a classic Eclipse.

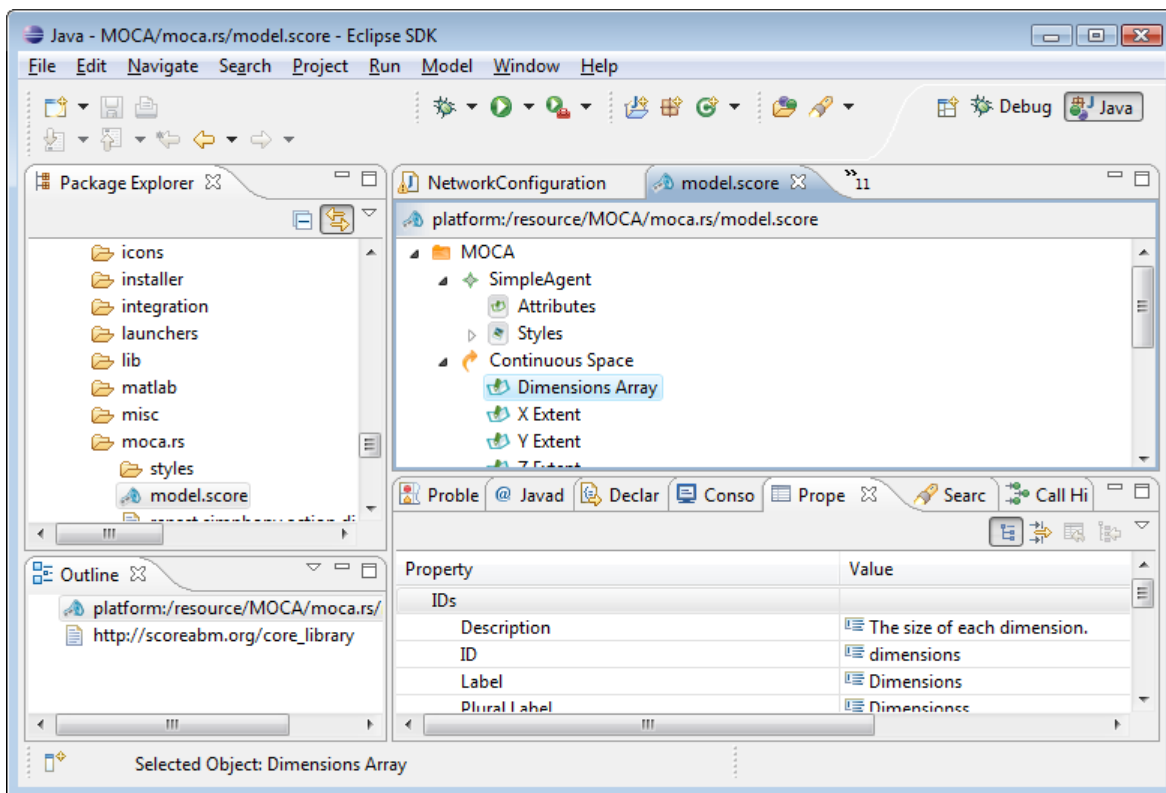


Figure 0.4 IDE of Repast Simphony

From Figure 0.3 and Figure 0.4 we can verify that Repast Symphony is basically an extension of Eclipse, though much auxiliary functionality are added. To get familiar with the IDE, please refer to Eclipse's official website for detailed and updated documents. With the consideration that Eclipse is a popular development tool that has been extensively used in Java, C/C++, web development; further introduction of how to use the IDE is eliminated here.

One point worth once more mentioning is that Repast J and Repast S work in totally different ways. For Repast J, there is no dedicated IDE. Repast Symphony IDE cannot support Repast J projects or provide more help on normal Java projects. To make Repast J code fit into Repast Symphony, a lot of code, especially which related to interface stuff, should be modified.

13.3 Building MOCAT in Repast Symphony

In this section, at first how to create a Repast Symphony for MOCAT is explained. At the end, the calculation of spread metric that is specific to MOCAT is introduced.

In a nutshell, the procedures below been followed in migrating MOCA into Repast Symphony:

1. Create a blank Repast Symphony project with the name MOCAT and copy the source code of previous CA implementation into the project folder; an alternative is to import MOCA into the IDE and add Repast Nature to it. There are no technical differences of the results. In either way, a few files and folders that are critical to Repast Symphony are automatically created.
2. Groovy code can be deleted since pure Java is used in this implementation. Existence of it does not harm the integrity or hinder the execution of the project.

3. Remove all Repast Java code from the project. Such code is used to step evolution and now should be consigned to Repast Symphony.
4. Some files are essential to Repast S, especially `model.score`, which records the system's parameters. Work on it and fill all necessary parts. Details will be given underneath right after this brief introduction.
5. After coding to satisfy Symphony specifications, other code units were added that created system properties as global parameters, such as maximum number of generations. Repast Symphony code uses a lot of Java Annotation which is similar to compiler instructions, and a feature that came with Java 1.5. (As of this thesis writing, the newest Java version is 1.6.)
6. When coding is done, click the little green triangle at the toolbar and locate and click "*Run Repast MOCAT*", which will bring us a primitive Repast Symphony runtime interface that is bound to MOCAT and wait to be further setup.
7. In the loaded execution program, setup for correct visualization, including display, loggers, and charts. In other words, we have to configure the score, write the code, and setup the execution program correspondingly and sync them manually.
8. Additionally, to spread and deploy MOCAT on a machine on which an execution environment is not ready, click the little green triangle and select Build MOCAT Installer. A single-file installer which contains all necessary components to run will be created.

If a copy of Repast Symphony project is ready, then import source file into the IDE and the later will automatically create a Run Configuration "Repast Project MOCAT". Click it and a Repast Symphony MOCAT will run.

Figure 0.5 shows the essential model.score in bulletin format, in which the system attribute (parameter) number of population is highlighted. This file has to be located under the folder moca.rs, in which rs is an abbreviation of resources. In Eclipse, in the menu popped out after right click on top of the score file, choosing Open With | Text Editor will bring us the XML file describing the content. The score files configures the system's runtime parameters.

In order to guarantee visibility of details of the score, the tab Properties has to be brought up by selecting the menu item Show Properties in the popup menu trigger by right click in any place in the score window. Double click on a visible item won't make the Properties tab appear.

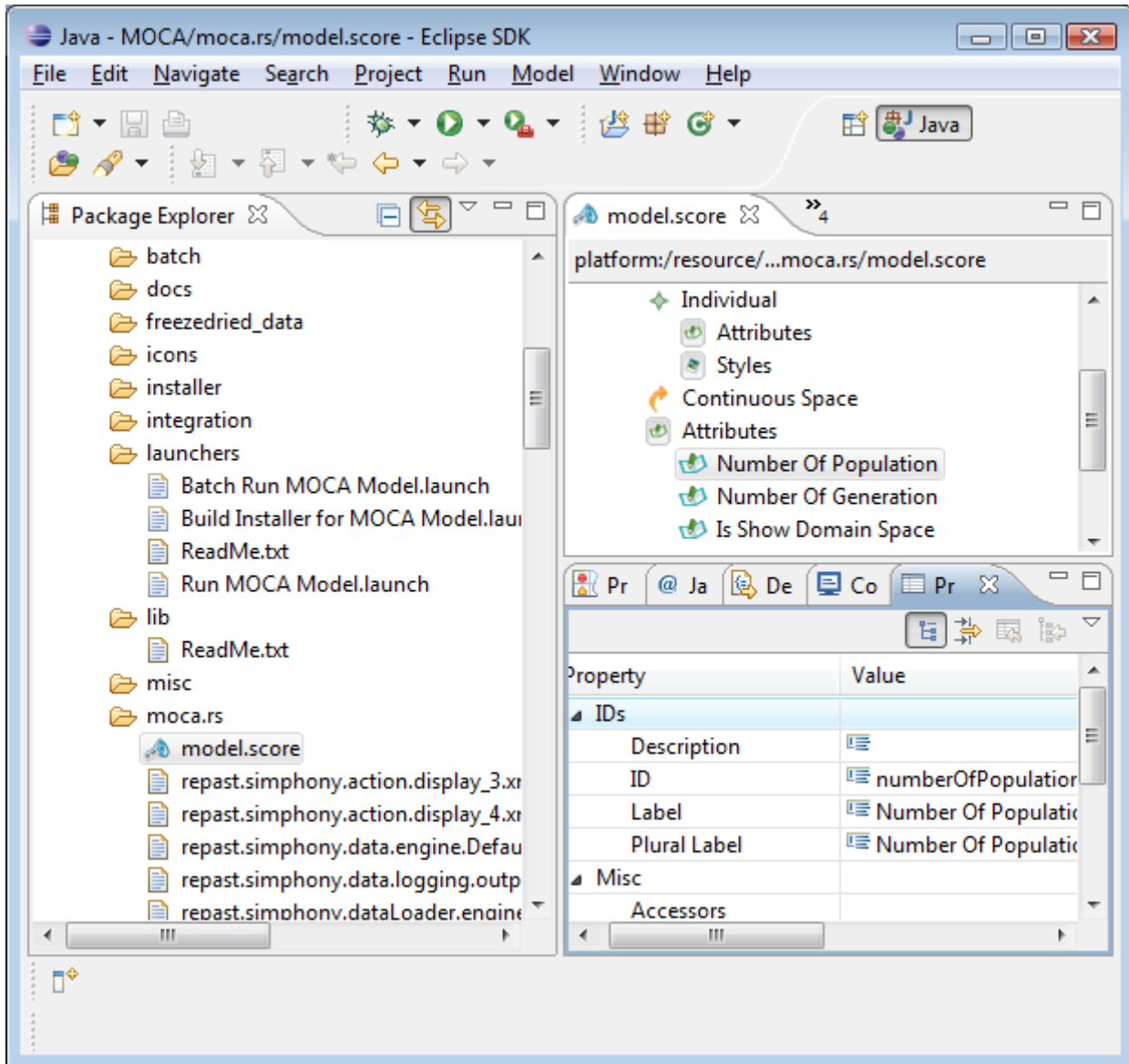


Figure 0.5 Model score

There are a few things that can be inserted into the project: agent, Context, and a variety of Projections. Agent must be registered here so that the runtime system can recognize it. In registering available agent, source code and compiled class have to be specified. In reality, it was found that deep package levels will prevent the runtime system from locating the agent Java class. However, there is an open document related to this requirement. In fact, shallow package levels are suggested for all classes

that are critical to Repast Symphony implementation. Context is not set here but configured in source code. Projections, here named as Spaces, have to be set both here and in source code. There are a few Space types; continuous space, grid, network, geography, and scalar, in which the first two are of our concern.

In a score, Attributes indicate system global parameters, whose type can be integer, float, string, Boolean, file etc. Parameters set here will be automatically shown by the runtime program for users to check and set values to; new parameter values can be read by source code easily through runtime access object.

In the current implementation of MOCAT, evolution is controlled in Context as a global event, while all individuals (agents) are in charge of their visualization. In this way, minimum code change is requested without hurting the system execution. After each step of evolution, the display needs to be updated. All agents have to be added into the projection to be able to get shown. We specify a set of eternal individuals in favor of display, which is named Display Set, because in CA populations are born and die all the time, as what happens in nature. Thereafter, after each step of evolution, information of the current population will be copied into the display set; the concrete update of graphical drawing is handled by Repast Symphony.

An adder—which is used by Repast Symphony to *add* an agent into the display—was created to explicitly match the dimensionality of individuals in display set and the displays should they are 3D; otherwise, a non-affine exception is thrown before the display is really about to show even though emulation calculations can be executed normally. With the consideration that Java 3D is actually delegated to JNI and finally reaches Java3D.dll, we suspect that strict declaration check is exerted and

causes this exception; however, we were unable to verify the conjecture and have not found any related documents.

In new MOCAT, another big modification is the addition of spread metric and its impact on evolution. Most of the code aggregates in Java class `NetworkConfigurationRouletteWheel`. As its name suggests, similar to the selection of knowledge sources, a roulette wheel is dedicated to selecting network configurations with the concern of randomness while we mainly focus on the spur of expanding spread. In this version of MOCAT, all available network configurations will be test run on top of the initial population to gain primitive evaluations of their performance in terms of spread; such primitive evaluations serve as the initial values for the quote in the roulette wheel. To promise fairness of comparison, the population space and belief space are all rolled back to the original status after test runs. During our test it was found that the initial spread metrics collected this way, though obviously bigger than the peers in later generations, are comparable to the later values. Thereafter, there is no need to enforce a minimum chance for any network configuration to be selected, unlike what had been done in the roulette wheel for knowledge sources where in each generation a minimum of five new individuals are guaranteed to be propagated underneath.

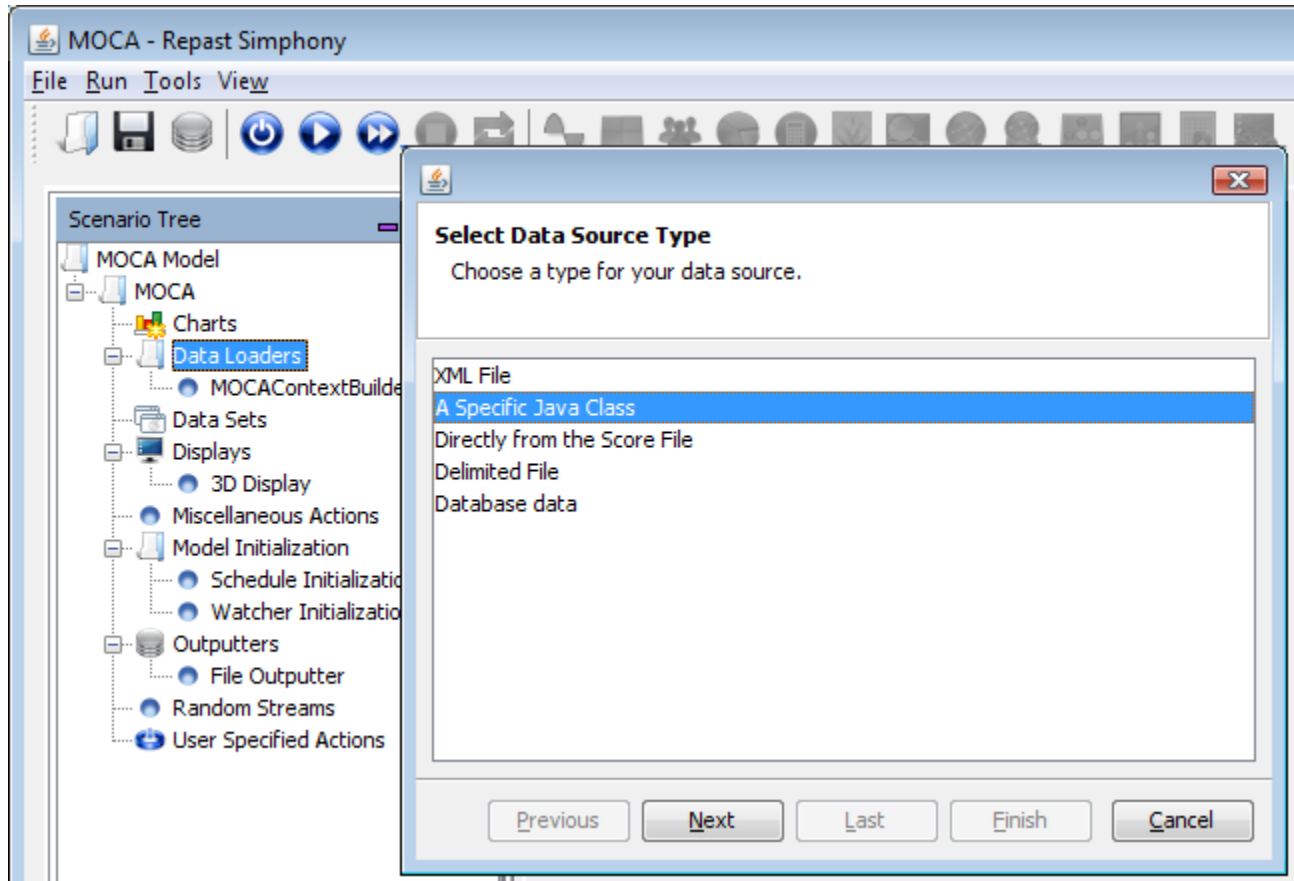
Concrete calculations of the spread metric happen in the method `processOneGeneraion` of class `MOCulturalAlgorihtm`, in which the spread value is calculated upon the current generation and the network configuration which has been selected for it is thereafter updated accordingly. In other words, in the network configuration roulette wheel, a moving average is maintained for network configurations for a past period. In current implementation, the math average of the stored spread value and the new spread value is saved.

13.4 How to Run MOCAT

To enable a Repast Symphony project to run, three peers have to be guaranteed to match each other, the Java source code, the score file, and the runtime configurations. In this section, all necessary setup in the runtime environment will be explained step by step. Without these correct setups, CA can evolve its emulation but no data will be collected or shown visually.

When first starting up MOCAT, after the model is loaded into the execution environment, we have to specify the data loader; even though it has been setup in the score file.

To create a start point of the evolution, in the left pane, right click “Data Loader” and then click “Set Data Loader” in the popup menu, choose “A Specific Java Class” in the next dialog and then select the correct context builder class. Please note here the context builder has to be located at the root of the project; otherwise it cannot be correctly located by the execution environment.



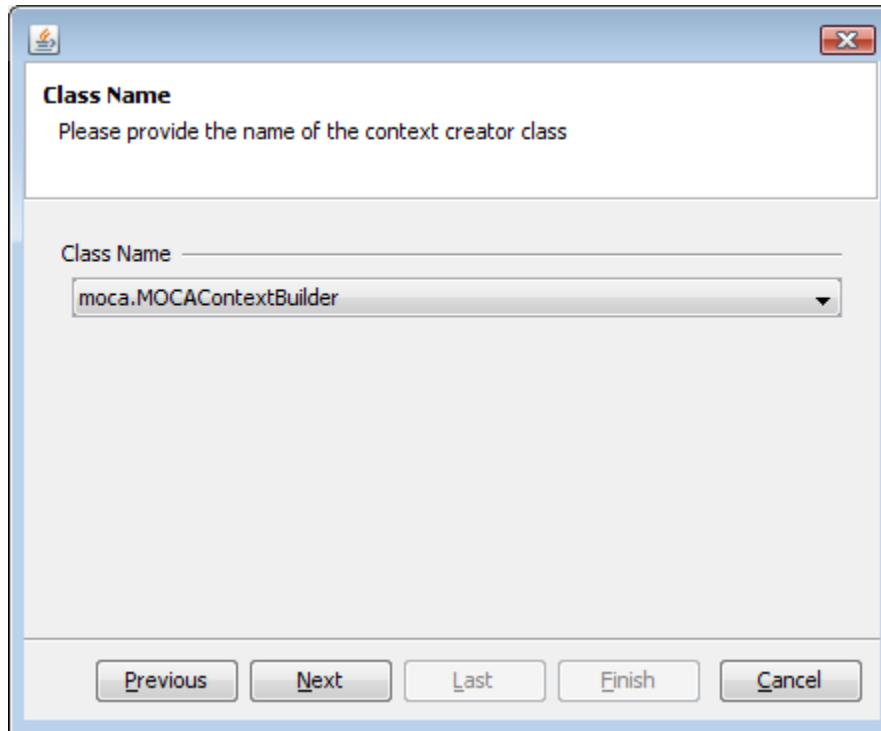


Figure 0.6 Setup Context loader in the runtime environment

After setting of the context builder, data sets can be set too though it is not indispensable for the core implementation of MOCAT. A data set is a series of fields that are extracted from a specific agent class and can be used to create logs.

Right click "Data Sets" and then click "Add Data Set", in the next dialog set up as follows, in which name and id can be arbitrary. Methods `getXPosition()` and `getYPosition()` belong to class `Individual` and can be automatically recognized by Repast Symphony.

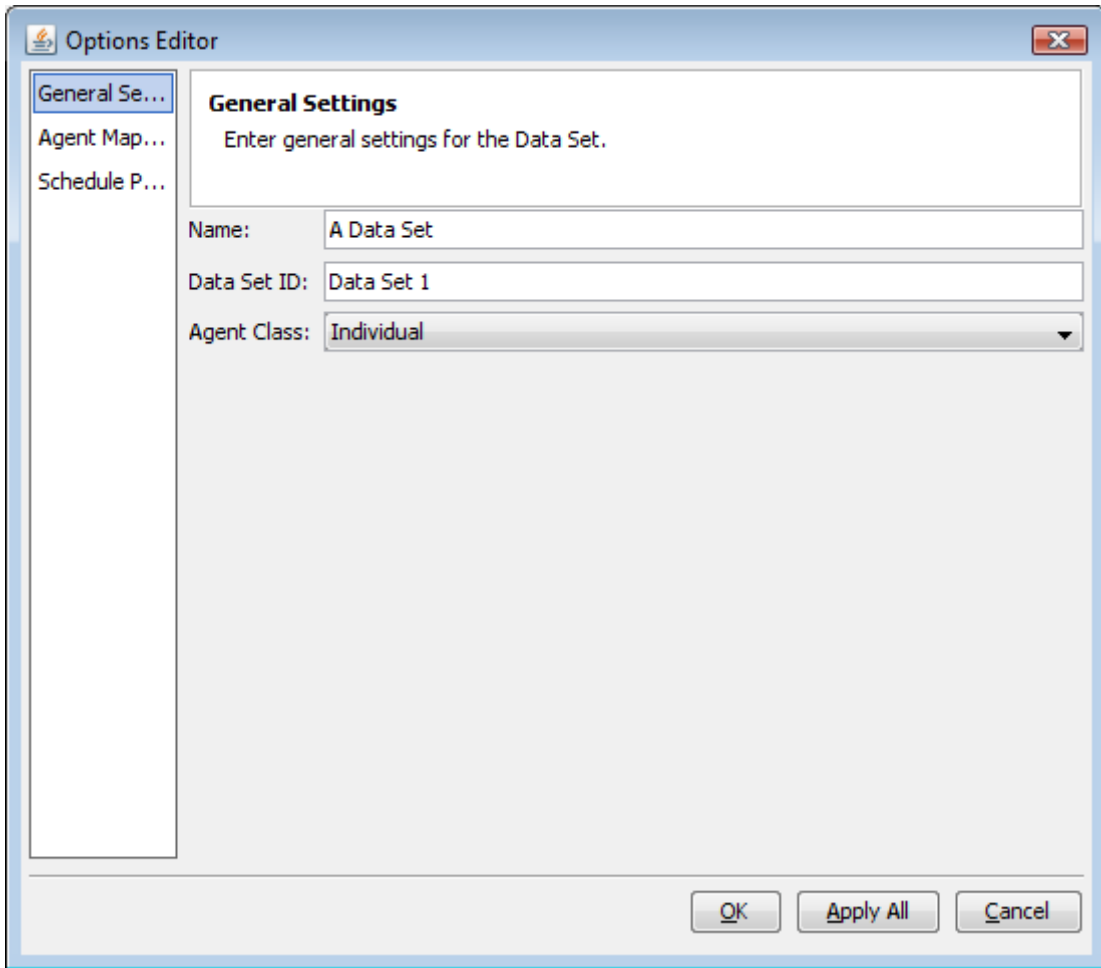
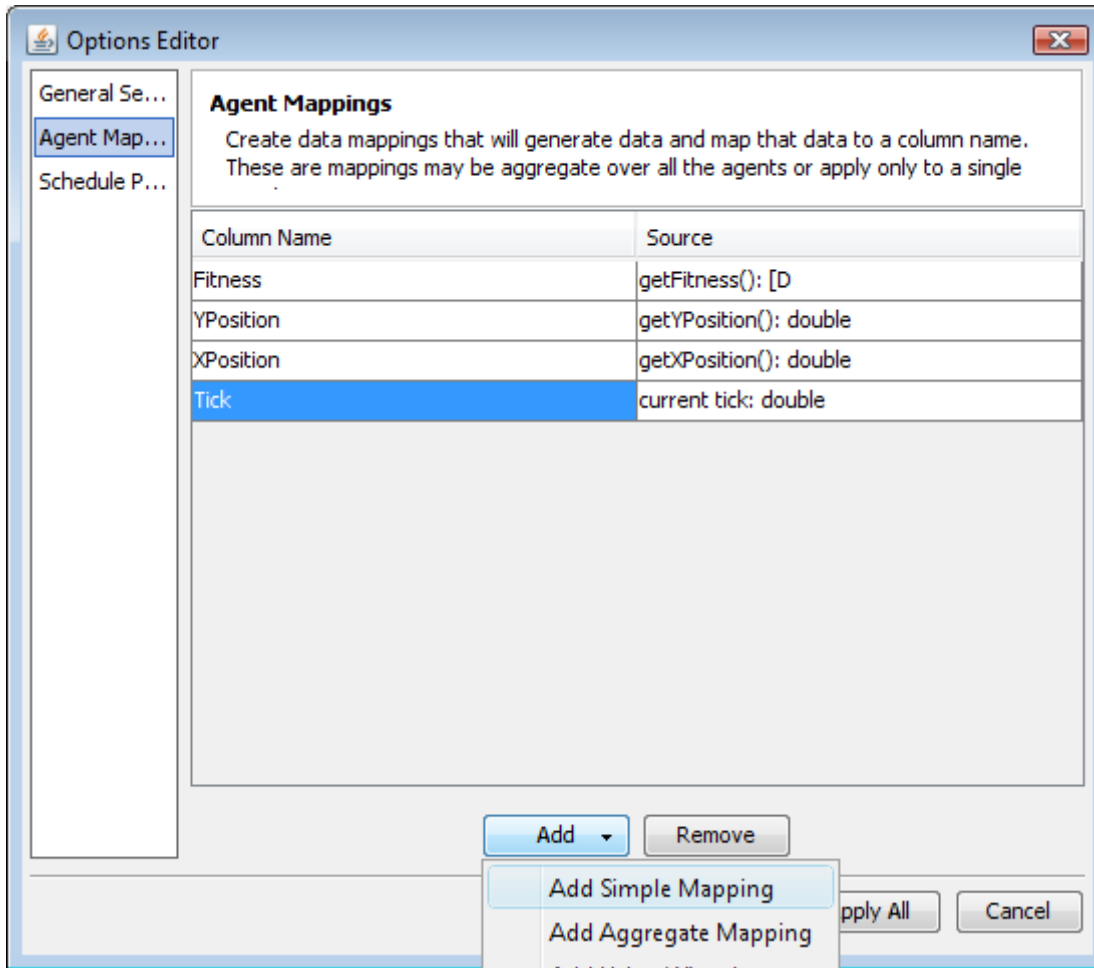
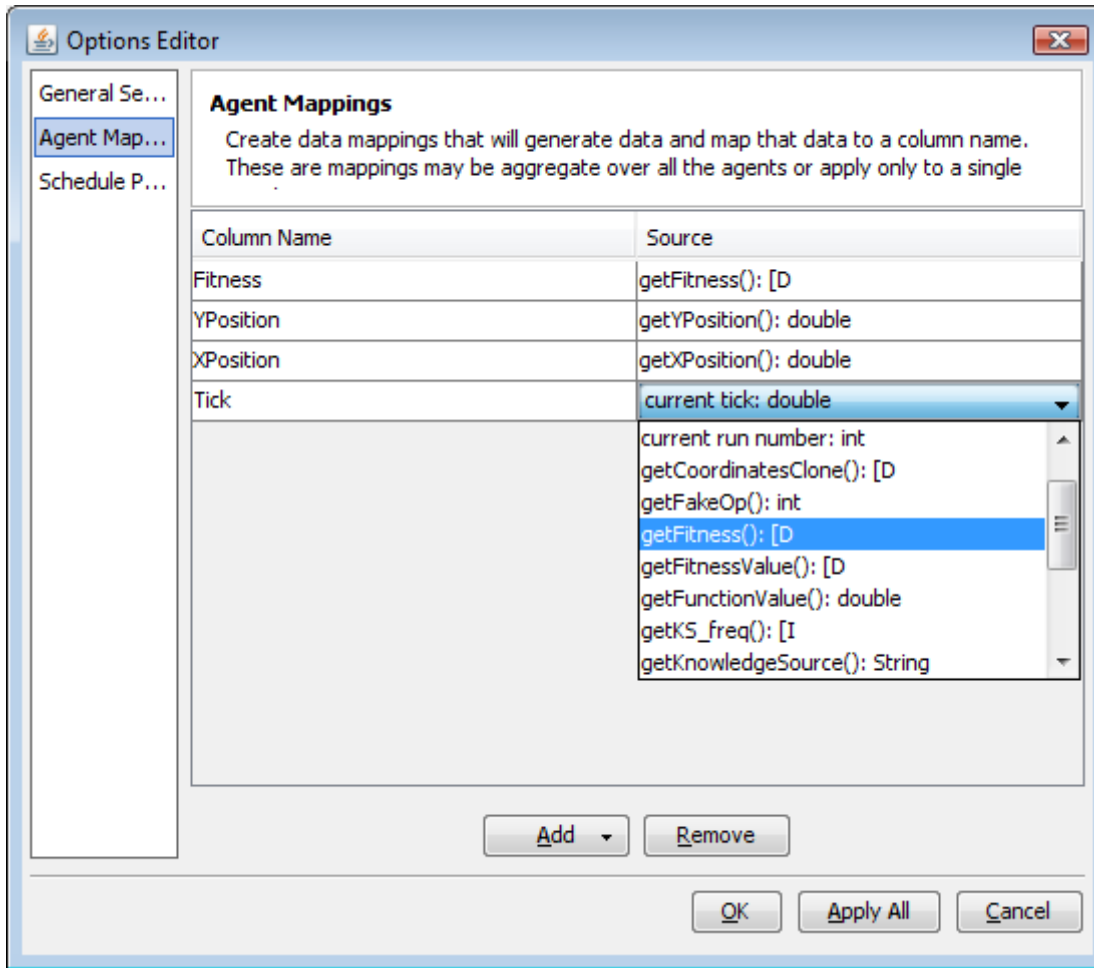


Figure 0.7 Creating a Data set by extracting fields from agent class

Click the button "Add" or select "Add Simple Mapping" precisely; a default item named "Tick" will show in the mapping table.



Click the right column and select a wanted method; and the left column will change to the name according to Java Bean's naming convention. Double click on the left column make it possible to modify the name.



Outputters can be built upon data sets. Right click it and select “Add File Outputter” and give an arbitrary name and select an available data set, which in this case is “Date Set 1”. Then add any field of interest from the left list into the right one by pressing the green right arrow in the middle.

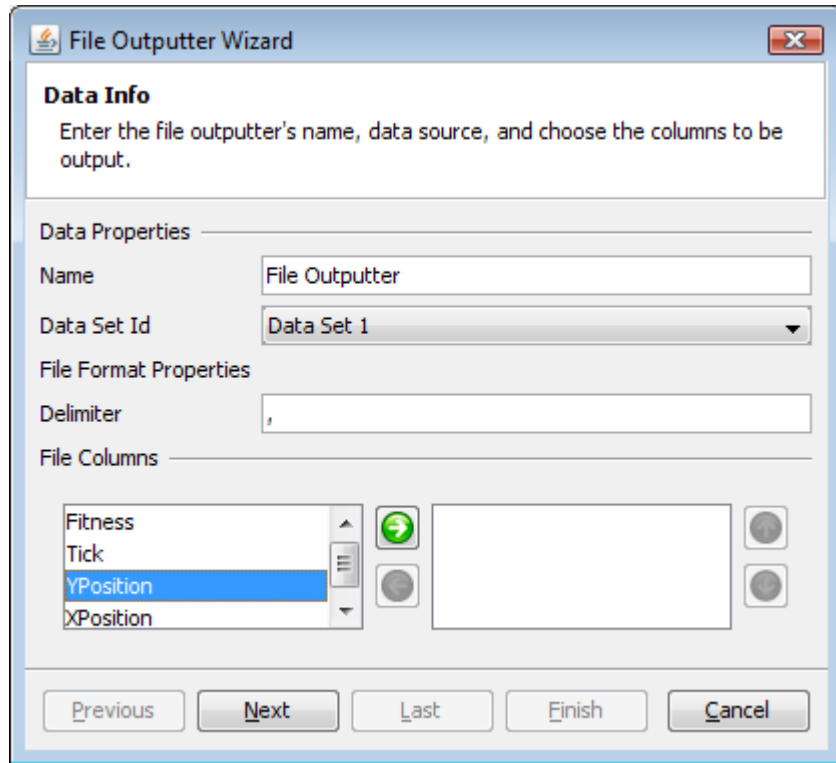
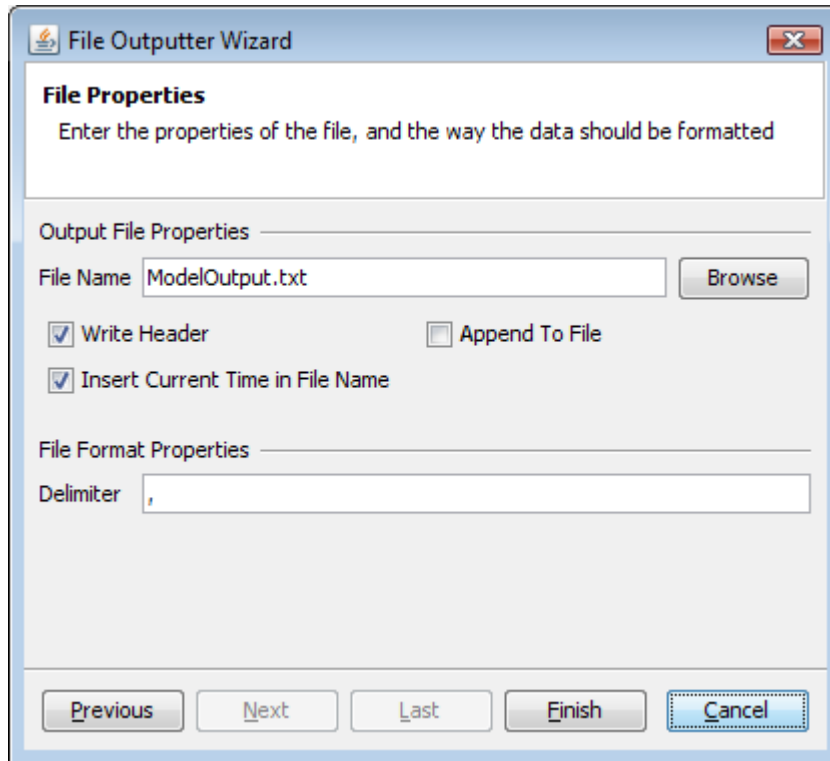
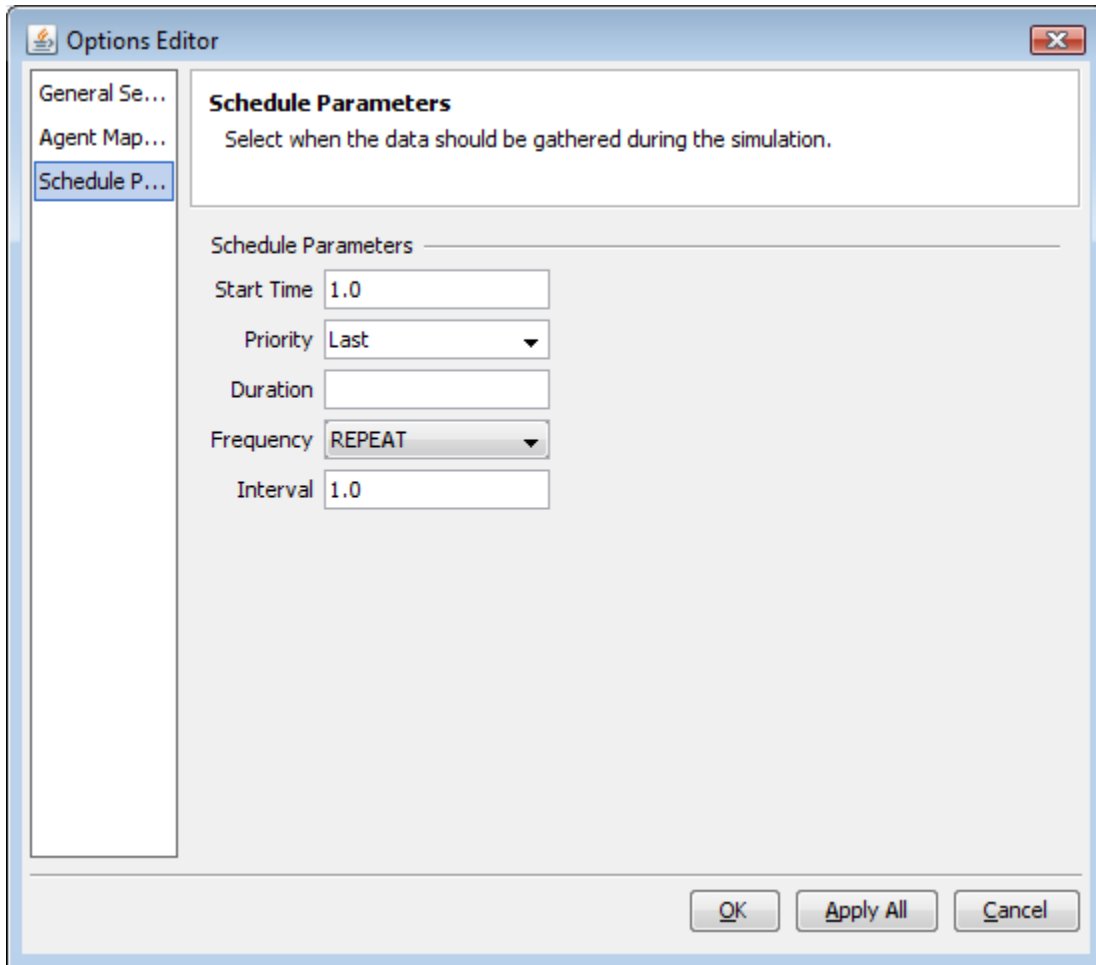


Figure 0.8 Creating Outputter upon Data Set

The next dialog is self-explanatory, in which the log file is given a file and format is specified. In this way, we don't have to manually set up and maintain any log files in the source code; instead, we have the freedom to create them in runtime and change the data series to record.



The last step is to setup schedule parameters, where no particular attention is required, except that frequency has to be repeat instead of the other choice: one_time. In this way, those data will be updated automatically along evolution.



Additionally, displays have to be set too. We may think that displays we created in source and in the score file are possible choices and in the runtime environment they are nominated to show.

Right click “Display” and select “Add Display” in the popup menu, give an arbitrary name, here we choose “3D display”, in the following dialog and add “Continuous Space” into the right list by clicking the green right arrow in the middle which is disabled since there are no further possible choices. Recall that “Continuous Space” has been added in the score file so that it can be recognized and listed by the runtime environment. In the source code the continuous space has been created and configured as well;

however, the matching between the source code and the setup here will not be exerted until the execution of the evolution.

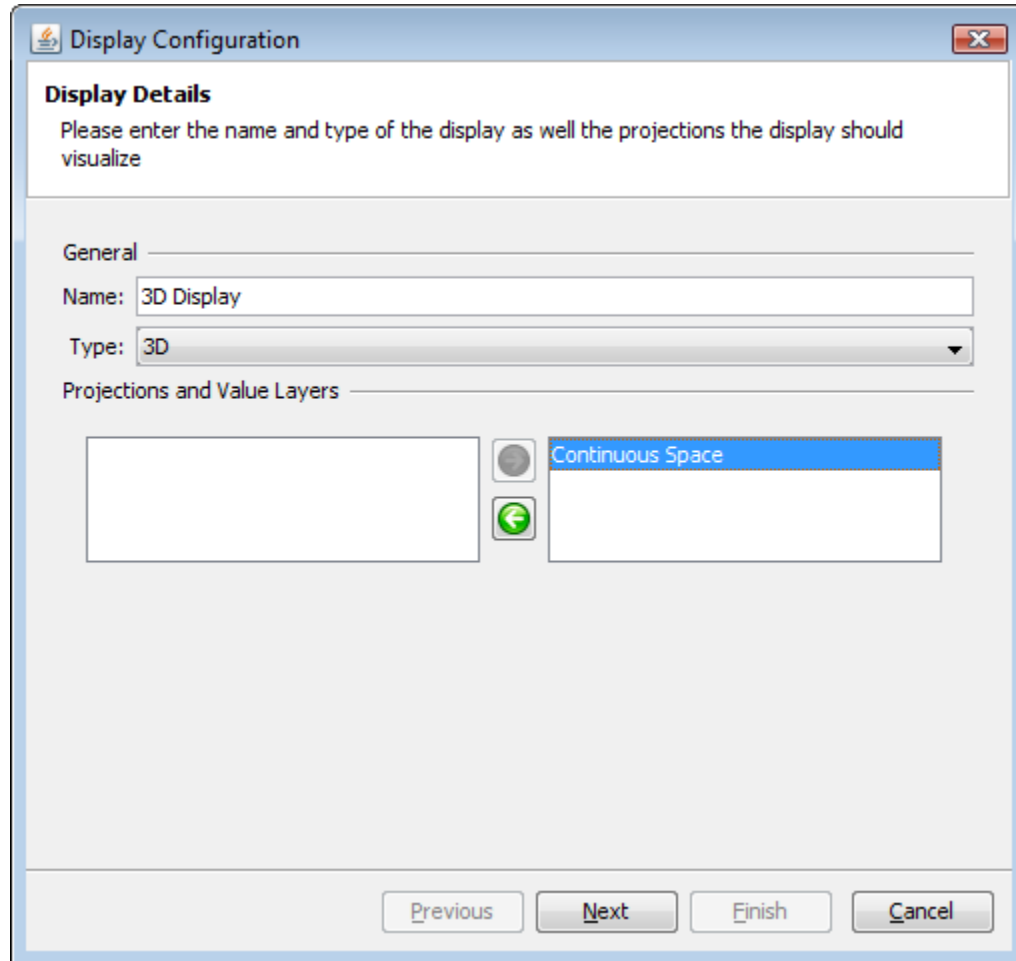
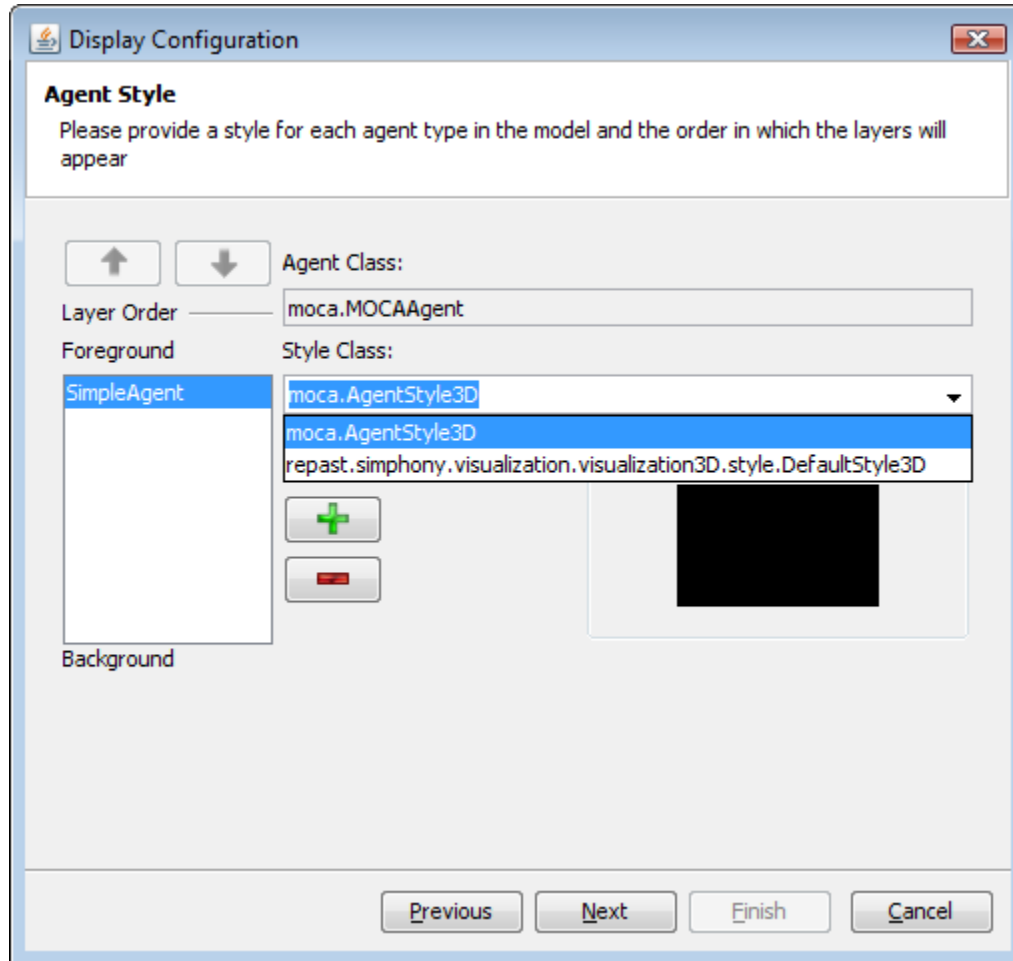


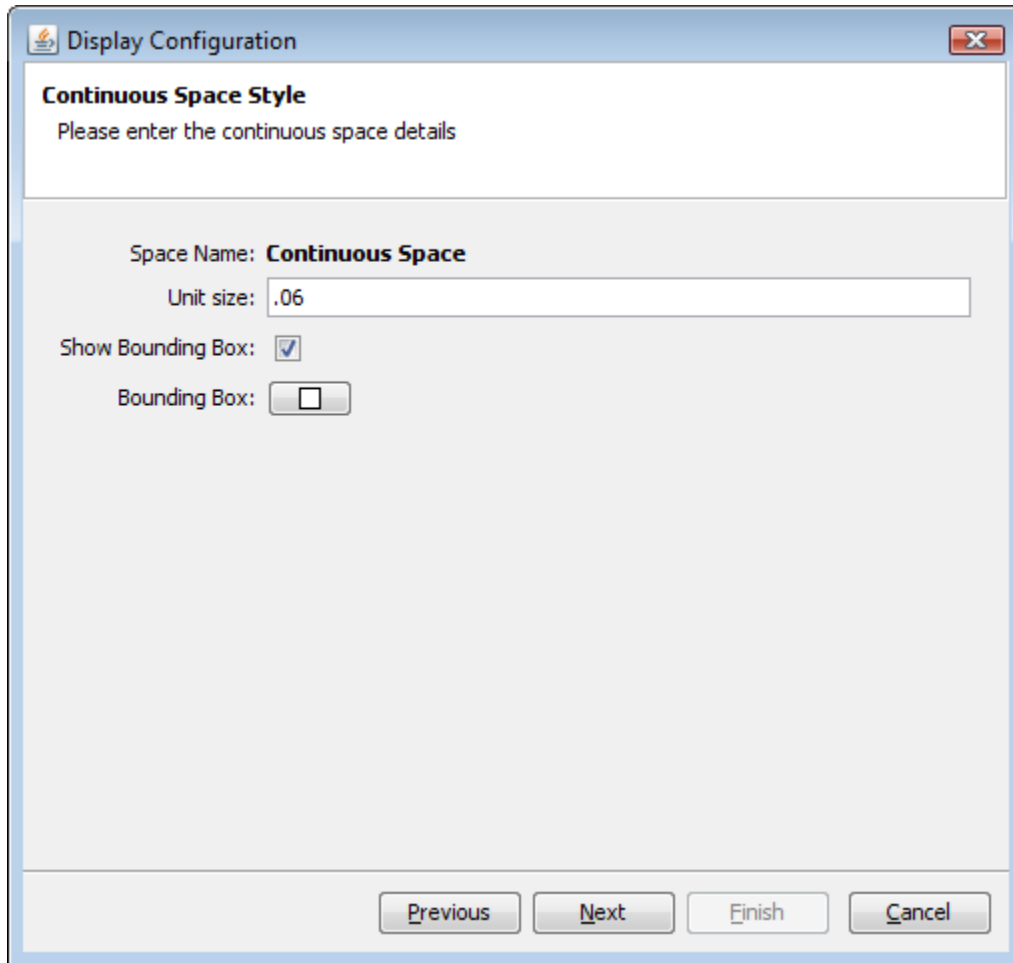
Figure 0.9 Creating a 3D display (Projection) for agents

A display has to be filled with agents in this setup though in source code agents have already added themselves into Projections. In the next dialog, SimpleAgent is the display name that is created in the source code by Java annotation for the class MOCAgent that is used in the display set. Be sure to select an appropriate style class that inherits a suitable Style class of Repast Symphony is in charge of select visual elements for agents to show in the display. In our implementation, colors are used to mark

the order of the pareto front at which the agent is located. Otherwise, a pure white ball is shown for an agent thereafter other information is lost.



The last two screens for creating a display is simple, as shown in the following figures.



Display Configuration [Close]

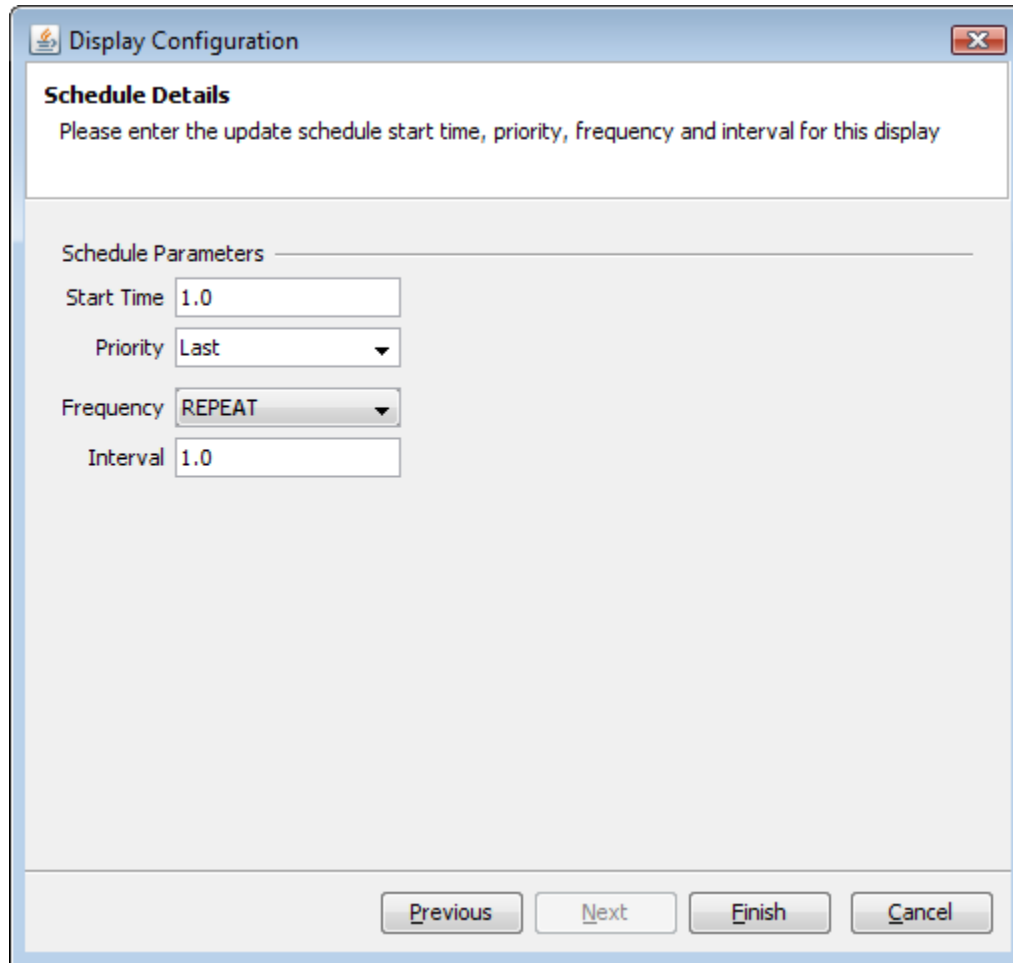
Continuous Space Style
Please enter the continuous space details

Space Name: **Continuous Space**

Unit size:

Show Bounding Box:

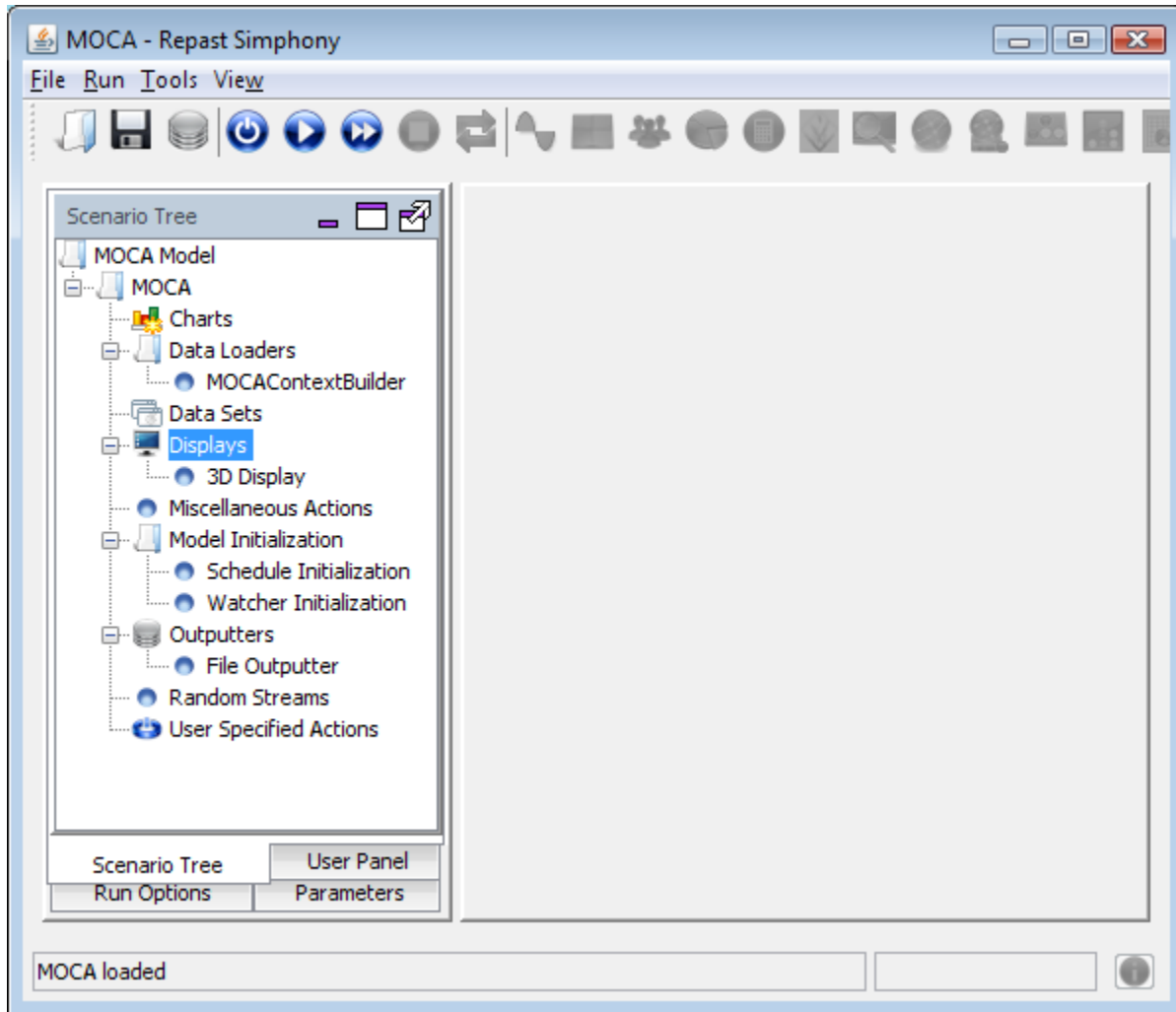
Bounding Box:



The image shows a software dialog box titled "Display Configuration" with a close button in the top right corner. The main section is titled "Schedule Details" and contains the instruction: "Please enter the update schedule start time, priority, frequency and interval for this display". Below this, there is a section labeled "Schedule Parameters" with a horizontal line. It contains four input fields: "Start Time" with the value "1.0", "Priority" with a dropdown menu showing "Last", "Frequency" with a dropdown menu showing "REPEAT", and "Interval" with the value "1.0". At the bottom of the dialog, there are four buttons: "Previous", "Next", "Finish", and "Cancel".

Similarly, be sure to selection repeat for the frequency option so that the display is updated along each step of evolution.

Finally, we get a ready-to-go MOCAT execution environment as shown underneath. Press the floppy disk icon to remember the current configuration and reuse it later. From now on, any subsequent use of MOCAT will return this interface.



The power button will call source code to create the Context and initialize, date set, outputter, and display that have been created during the previous setup. The Start button will notify CA to run until in the source code calls *repast.simphony.essentials.RepastEssentials.EndSimulationRun()* to stop. The Step Run button will make evolution step forward and the pause.

All such buttons have hints that will show when mouse is hovering over; and the functionalities are easy to get understood.

In the ready interface, after an evolution is paused, double click an agent will make its properties to be shown in a dedicated panel, as illustrated in the bottom left tab of the underneath screen copy. Here we can not only observe its current information but are able to see its history. In this screen copy, we can see that the agent had moved to the second Pareto front three times.

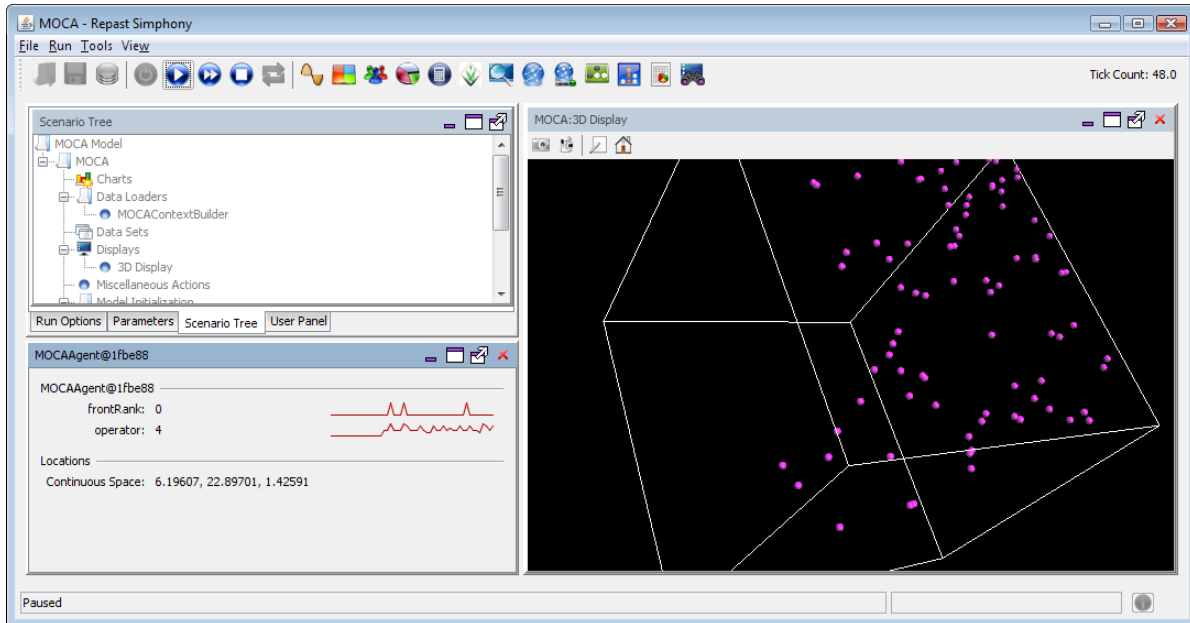


Figure 0.10 Double click an agent and show its properties

In addition to the necessary visualization, other straightforward representation can be added to MOCAT. For example, we can thereafter observe the number of Pareto fronts by viewing the maximum Pareto front rank. To do this, right click the category Charts and selected Add Chart in the popup menu.

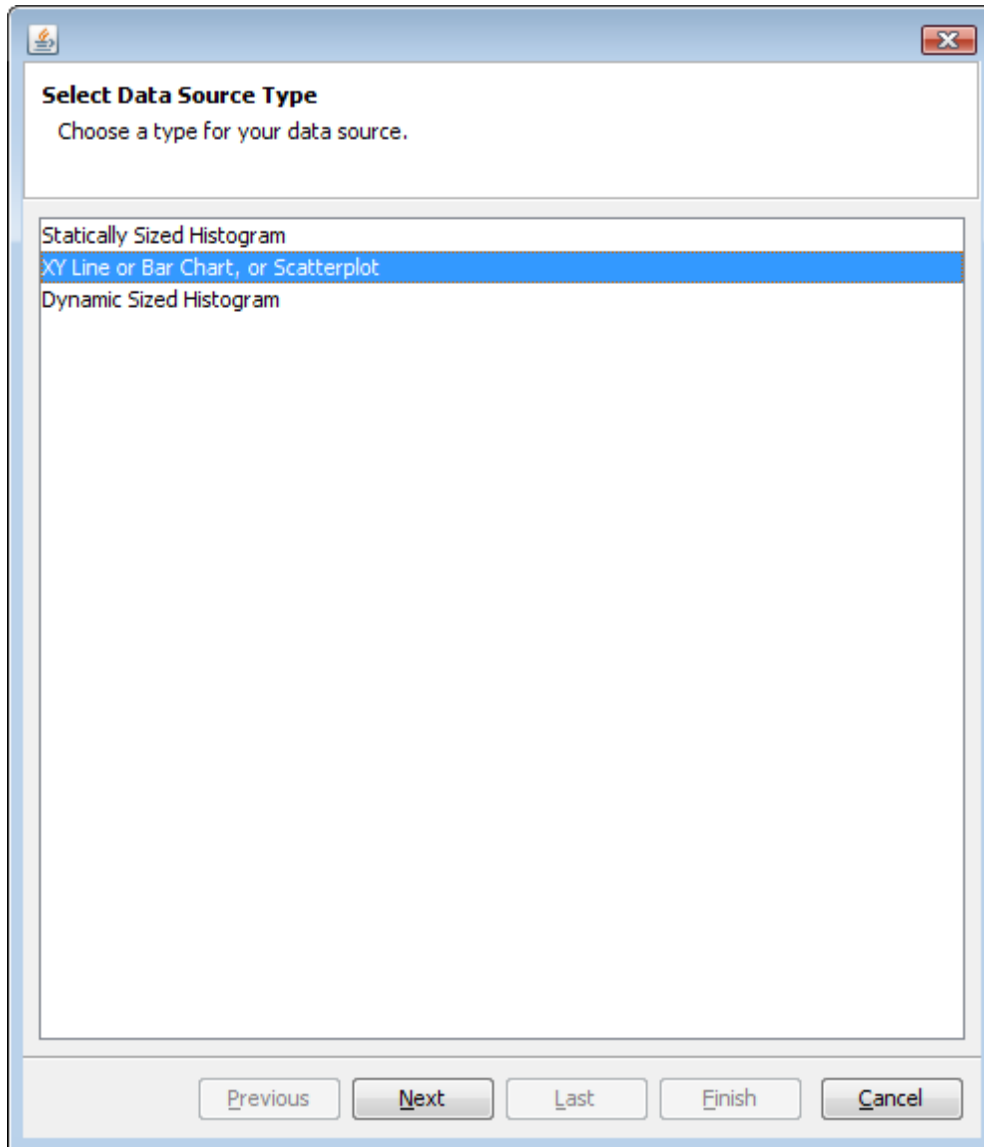


Figure 0.11 Choose Chart type

Select the chart type in need. In this example, please select the second one, i.e., XY Line.

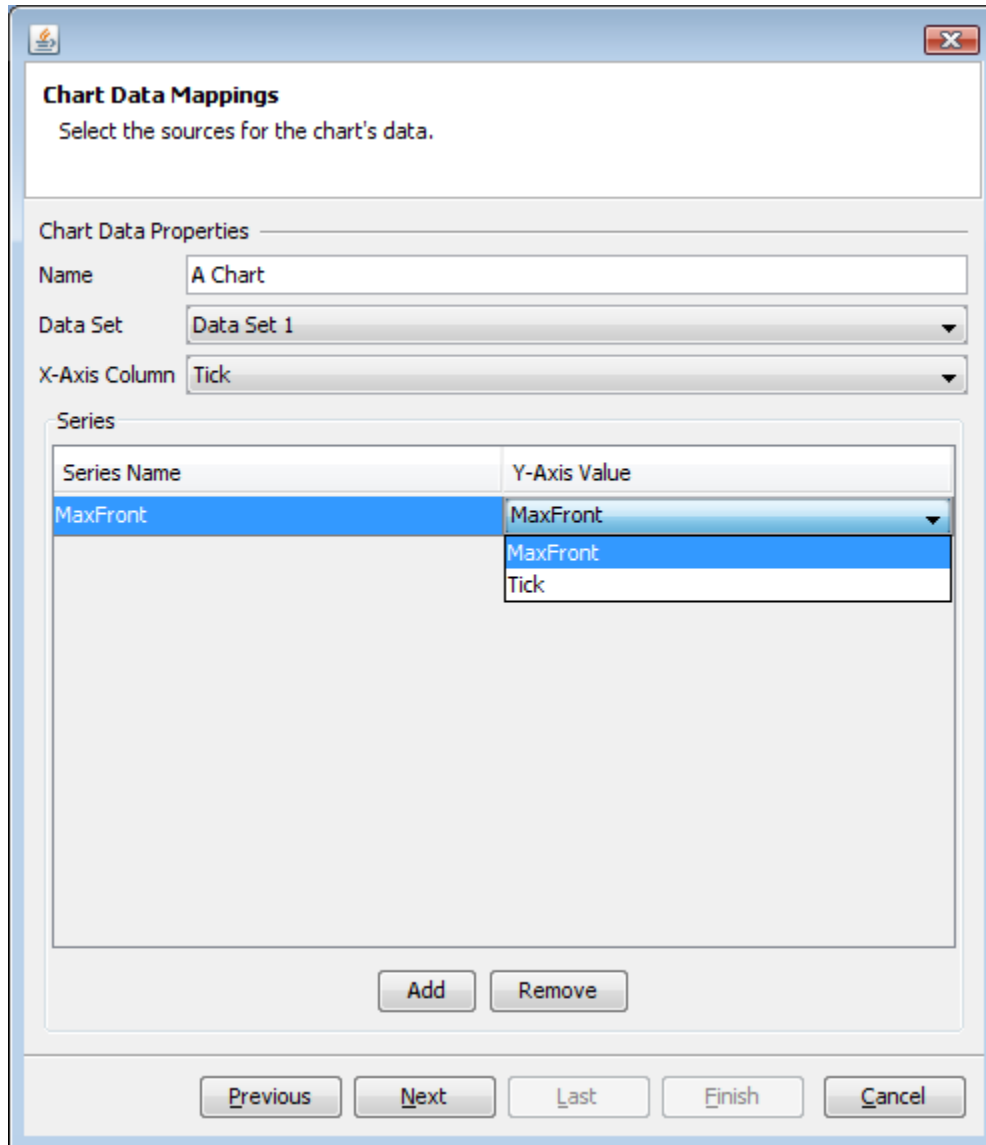


Figure 0.12 Select Data Serial for X and Y Axes

XY Line chart is not complex; here we select Tick, i.e., number of generation. For Y axis, we use the aggregate Max value of front ranks which is come predefined data set 1.

The rest is the detailed setup of the appearance of the chart.

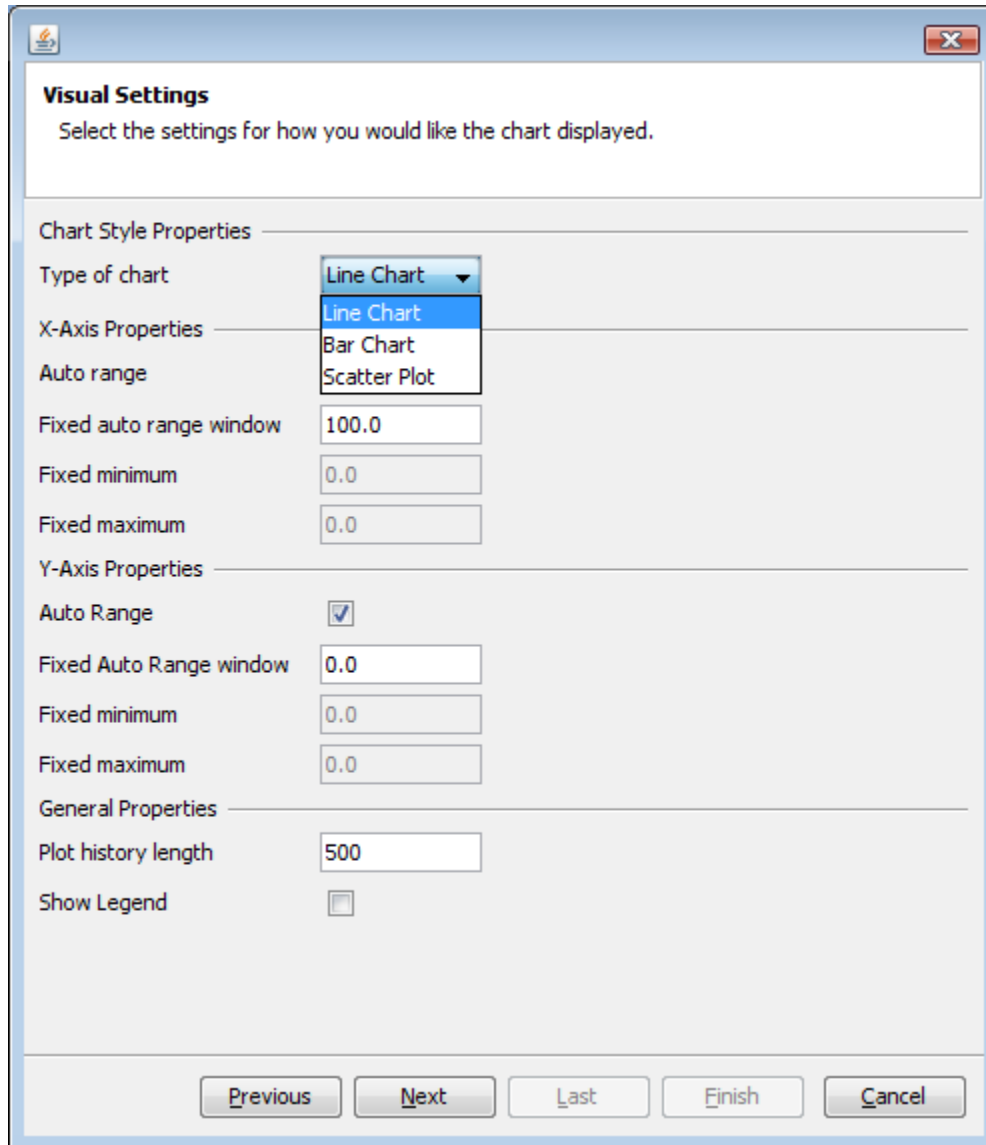


Figure 0.13 Configuration of the Chart

The last screen is a notification of the completion of the creation of the chart.

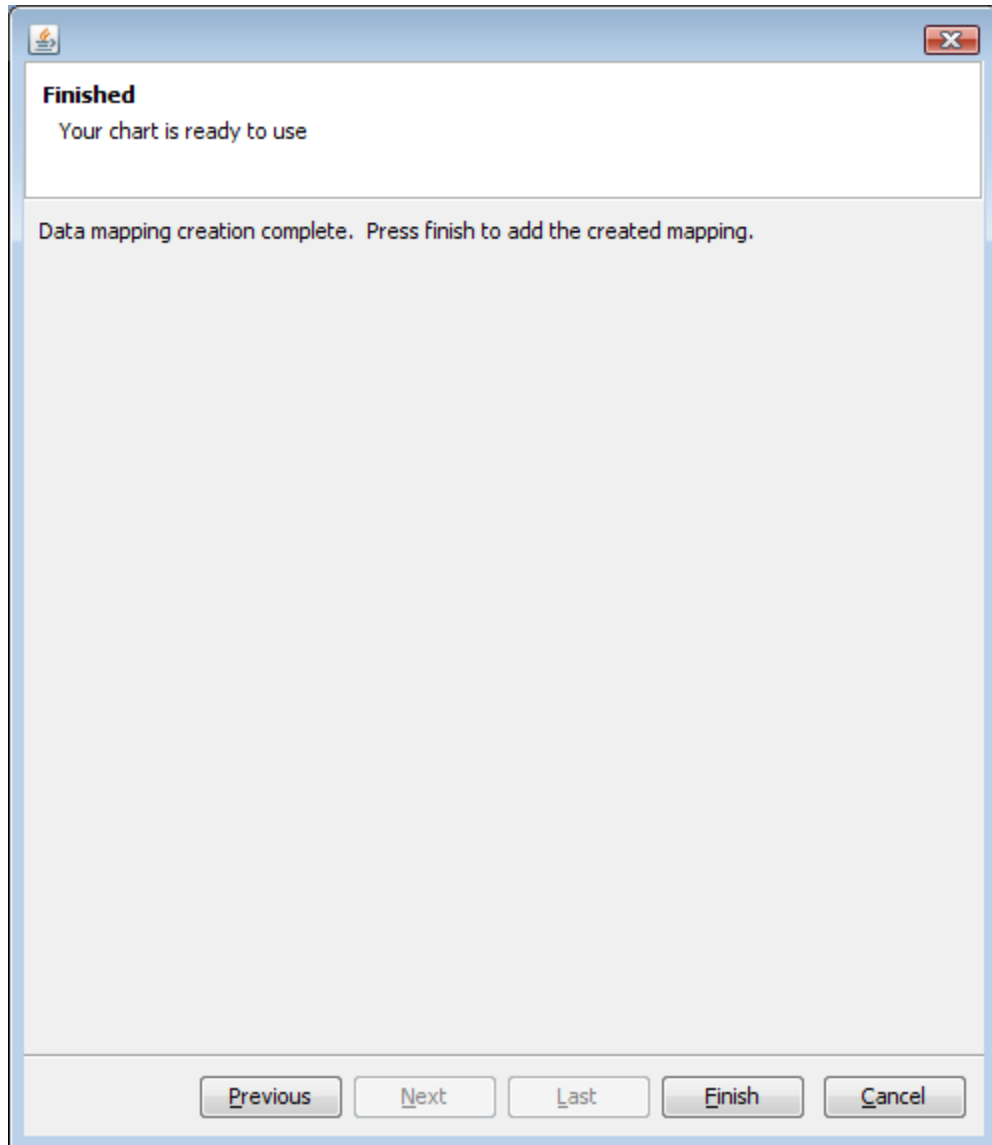


Figure 0.14 Chart is created completely

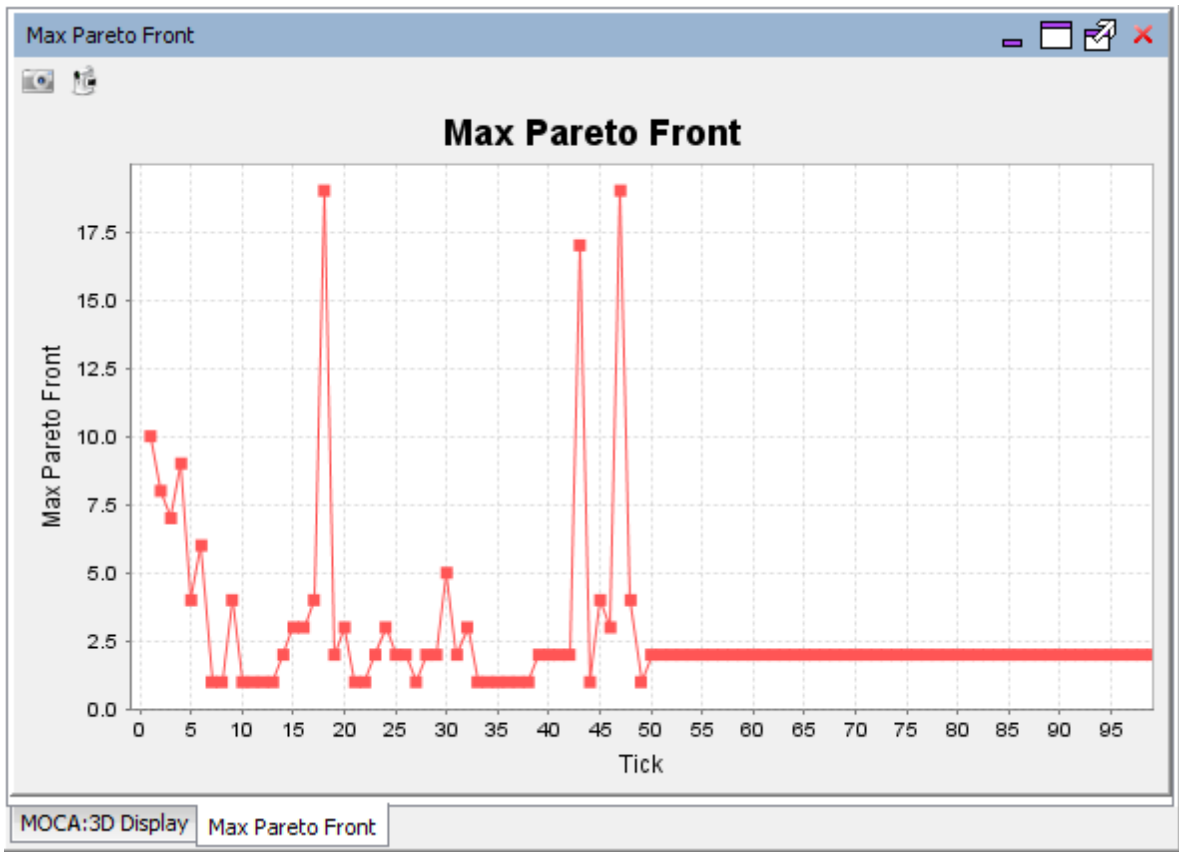


Figure 0.15 Real-time display of the Chart

After execution, a chart will be automatically created and adjacent to our main display. The content will be updated along execution so that it is convenient for us to observe the fluctuation of the generations. From this sample screen copy, we can see that evolution stay stable after generation 50. In the first 50 generations, there are three ones in which individuals aggregated into crowded areas so that the maximum pareto front ranks were large, either 17 or 18. With the reference to this figure, we may get a hint of when to stop evolution.

REFERENCES

- Ali, M., Using cultural algorithms to solve optimization problems with a social fabric approach, Ph.D. thesis. Wayne State University. Detroit, MI. 2008.
- Argonne National Laboratory. Repast Agent Simulation Toolkit. Argonne National Laboratory, retrieved June 10, 2010. <http://repast.sourceforge.net/index.html>.
- Barkow, H. Jerome, Cosmides, Leda , and Tooby, John . The Adapted Mind: Evolutionary Psychology and the Generation of Culture, Oxford University Press, Huntington Beach, 1995.
- Bastos-Filho, Carmelo, J. A. and Miranda, B.C. Pericles. Multi-Objective Particle Swarm Optimization using Speciation, in Proceedings of 2011 IEEE Symposium on Swarm Intelligence, Paris, France, April 11-15, 2011.
- Beckers, R., Deneubourg, J. L , and Goss, S. Trails and u-turns in the selection of a path by the ant *Lasius Niger*, JOURNAL OF THEORETICAL BIOLOGY , 1992. pp 397-415.
- Beheshti, R. and Rahmani, A. T. A Multi-Objective Genetic Algorithm Method to Support Multi-Agent Negotiations. In the proceedings of Second International Conference on Future Information Technology and Management Engineering, 2009.
- Bellman, E. Richard. Dynamic Programming. Princeton University Press, Princeton, 1957.
- Best, Christopher. Multi-objective Cultural Algorithms. Master thesis, Wayne State University. Detroit, MI. 2009.
- Best, Christophe, Che, Xiandong, Reynolds, G. Robert, and Liu, Dapeng. Multi-objective Cultural

- Algorithms, in Proceedings of 2010 IEEE Congress on Evolutionary Computation, Barcelona, 2010. pp 1-9.
- Bonabeau, Eric, Theraulaz, Guy, and Dorigo, Marco. Swarm intelligence: from Natural to Artificial Systems, Oxford University Press, Huntington Beach , 1999.
- Casti, L. John, Would-be Worlds: How Simulation is Changing the World of Science, Wiley, New York, 1998.
- Chamaani, S. and Mirtaheri, S. A. and Teshnehlab, M. and Shoorehdeli, M. A. and Seydi, V. Modified Multi-objective Particle Swarm Optimization for electromagnetic absorber design. in Proceedings of Asia-Pacific Conference on Applied Electromagnetics, 2007 (APACE'07)., Melaka, Malaysia, 4-6 Dec, 2007. pp 1-5.
- Charnov, E. "Optimal Foraging: the Marginal Value Theorem", Theoretical Population Biology, 9, no. 2 (April), 1976. pp. 129-136.
- Che, Xiangdong, Weaving the Social Fabric: Optimization Problem Solving in Cultural Algorithms Using Cultural Engine, Ph.D. thesis, Wayne State University, Detroit, 2009.
- Cheng , Li-Te, Patterson, John, Rohall, L. Steven, Hupfer, Susanne, and Ross, Steven, Weaving a Social Fabric into Existing Software, in Proceedings of the 4th international conference on Aspect-oriented software development (AOSD'05), Chicago, USA, March 14-18, 2005. pp 147-158,
- Chung, Chan-Jin and Reynolds, G. Robert. CAEP: An Evolution-based Tool for Real-Valued function Optimization Using Cultural Algorithms, International Journal on Artificial Intelligence Tools, Vol. 7, No. 3, 1998. pp 239-291.

- Clayton, S. Nicola, Griffiths, P. Daniel, and Dickinson, Anthony, Declarative and Episodic-like Memory in Animals: Personal Musings of a Scrub Jay, in *The Evolution of Cognition*, the MIT Press, Cambridge, 2000. pp 273-288.
- Coello-Coello, A. Carlos and Ricardo L. Bécerra. *Evolutionary Multi-objective Optimization Using a Cultural Algorithm*, IEEE Swarm Intelligence Symposium, Piscataway, 2003. pp 6-13.
- Coello-Coello, A. Carlos, David A. Van Veldhuizen, and Lamont, B. Gary, *Evolutionary Algorithms for Solving Multi-Objective Problems*. 1st ed. Springer, June 30, 2002.
- Coello-Coello, A. Carlos and Maximino Salazar Lechuga, *MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization*, proceedings of 2002 Congress on Evolutionary Computation. Vol. 2, Honolulu, 2002. pp 1051–1056.
- Deb, Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, New York, 2001.
- Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2000. pp 182-197.
- Deb, Kalyanmoy, Pratap, A., Agarwal, S., Meyarivan, T. A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2) 2002. pp 182–197.
- Deb, Kalyanmoy, Thiele, Lothar, Laumanns, Marco, and Zitzler, Eckart. Scalable Multi-Objective Optimization Test Problems. In *Proceedings of the 2002 Congress on Evolutionary*

- Computation, Honolulu, HI, USA, May 12-17, 2002. pp 825–830.
- Doerner, K., Hartl, R.F., and Reimann, M. CompetAnts for problem solving—the case of full truckload transportation. *Central European Journal for Operations Research and Economics*. vol. 11, no. 2, 2003. pp. 115–141.
- Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C., and Stummer, C. Pareto ant colony optimization in multi-objective project portfolio selection with ILP preprocessing. *European Journal of Operational Research*, vol. 171, no. 3, 2006. pp. 830–841.
- Dorigo, M., Birattari, M., and Stutzle, T. Ant colony optimization, *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4. 2006. pp 28-39.
- Dorigo, Marco. *Learning and Natural Algorithms*, Ph.D. Thesis. Italy: Politecnico di Milano, 1992.
- Dorigo, Marco and Gambardella, Maria Luca. Ant Colony System: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation* 1997.
- Dorigo, Marco , Maniezzo, Vittorio , and Colorni, Alberto. Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, Vol. 26, No. 1, 1996. pp 29-41.
- Dowty, David. Thematic proto-roles and argument selection, *Language*, Vol. 67, No. 3, 1991. pp 547-619.
- Drezewski, Rafał, and Siwik, Leszek. The Application of Agent-Based Co-Evolutionary System with Predator-Prey Interactions to Solving Multi-Objective Optimization. in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision*

- Making (MCDM 2007). Honolulu, HI, USA. 1-5 April 2007. pp 294-301.
- Durham, William. *Coevolution: Genes, Culture, and Human Diversity*, Stanford University Press, Palo Alto, 1992.
- Ebel, Holger, Davidsen, Jörn, and Bornholdt, Stefan. Dynamics of social networks. *Complexity*, vol 8, no. 2 (11), 2002. pp 24-27
- Eberhart, Russell C., Shi, Yuhui, and Kennedy, James. *Swarm Intelligence*, Morgan Kaufmann Press, 2001.
- Fonseca, Carlos M, and Fleming, J. Peter. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation* vol 3, 1995. pp 1--16.
- Gambardella, L. M., Taillard, E.D., and Agazzi, G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. in *New Ideas in Optimization*. McGraw Hill, London, UK, 1999. pp. 63–76.
- Goldberg, E. David. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, Boston, 1989.
- Gong, Maoguo, Chao Liu, Licheng Jiao, and Gang Cheng. Hybrid immune algorithm with Lamarckian local search for multi-objective optimization, *Memetic Computing*, Vol. 2, No. 1, 2010. pp 47-67.
- Hajela, P., and Lin, C. Y. Genetic search strategies in multicriterion optimal design. *Structural and Multidisciplinary Optimization*, vol 4, no. 2, June 1, 1992. pp 99-107.
- Häckel, Sascha, Fischer, Marco, Teich, Tobias, and Zechel, David. A Multi-objective Ant Colony Approach for Pareto-optimization Using Dynamic Programming, in *Proceedings of the*

- 10th Annual Conference on Genetic and Evolutionary Computation. Atlanta, 2008. pp 33-40.
- Heyes, Cecilia and Huber, Ludwig. The evolution of cognition, The MIT Press, Cambridge, 2000.
- Holldobler, Bert and Wilson, O. Edward. The Ants, 1st ed. Belknap Press of Harvard University Press, 1990.
- Horn, Jeffrey, Nafpliotis, Nicholas , and Goldberg, E. David. A Niche Pareto Genetic Algorithm for Multiobjective Optimization, in Proceedings of the first IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, 1994. pp 82-87.
- Ibanez, M.L., Paquete, L., and Stutzle, T. On the design of ACO for the biobjective quadratic assignment problem. in Proceedings of 6th International Symposium on Algorithmic Number Theory, (ANTS'2004), Springer-Verlag, vol. 3172, 2004. pp. 214–225.
- Iredi, Steffen, Merkle, Daniel, and Middendorf, Martin. Bi-criterion optimization with multi colony ant algorithms, in Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization, Zurich, 2001. pp 359-372.
- Jennings, R. Nicholas and Michael Wooldridge. On Agent-based Software Engineering, Artificial Intelligence, vol. 117, no. 1, 2000. pp 277-296.
- Jin, Emily, Girvan, Michelle, and Newman, M. Structure of growing social networks. Physical Review E vol 4, no. 9, 2001.
- Jin, Xidong and Reynolds, G. Robert. Using Knowledge-based Evolutionary Computation to Solve Nonlinear constraint Optimization Problems: a Cultural Algorithm Approach, in

- Proceedings of the 1999 Congress on Evolutionary Computation, Washington D.C., 1999. pp 1672-1678.
- Johnson, Allen and Earle, Timothy. *The Evolution of Human Societies: From Foraging Group to Agrarian State*, Stanford University Press, Palo Alto, 2000.
- Kennedy, James and Eberhart, Russell, Particle Swarm Optimization, in Proceedings of IEEE International Conference on Neural Networks. Perth, 1995. pp 1942-1948.
- Kennedy, James. The particle swarm: social adaptation of knowledge, in IEEE International Conference on Evolutionary Computation, 1998. pp 303-308.
- Knowles, Joshua and Corne, David, Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, *Evolutionary Computation*, vol. 8, no. 2, 2000. Pp 149-172.
- Kursawe, Frank. 1991. A Variant of Evolution Strategies for Vector Optimization. *Parallel Problem Solving From Nature*, vol 496, 1991. pp 193--197.
- Langton, Christopher. Life at the edge of chaos. *Artificial Life*, vol 10, 1992. pp 41-91.
- Laumanns, M., Rudolph, G., and Schwefel, H.-P. A spatial predatorprey approach to multi-objective optimization: A preliminary study, in *Parallel Problem Solving from Nature*, vol. 1498. Springer-Verlag, 1998.
- Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation*, vol 10 no. 3, 2002. pp 263–282.
- Li, Yinghai, Zhou, Jianzhong, Qin, Hui, Lu, Youlin, Yang, Junjie, Adaptive Niche Multi-objective Particle Swarm Optimization Algorithm, in Proceedings of International Conference on

- Natural Computation, Jinan, China, August 25-27, 2008. pp 418-422.
- Macal, C.M. ; North, M.J. Tutorial on agent-based modeling and simulation part 2: how to model with agents. Association for Computing Machinery, Winter Simulation Conference, Monterey, CA, 2006. pp 73 – 83.
- Mariano, C. E. and Morales, E. MOAQ an ant-Q algorithm for multiple objective optimization problems. In Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, 13-17 July 1999. pp 894.
- Maniezzo, Vittorio and Carbonaro, Antonella. Ant Colony Optimization: An Overview, Essays and Surveys in Metaheuristics, 1999. pp 21--44.
- Melin, Patricia, Castillo, Oscar, Ramírez, G. Eduardo, Kacprzyk, Janusz, and Pedrycz, Witold. Analysis and Design of Intelligent Systems Using Soft Computing Techniques. Springer, 1st edition. July, 2007
- Miraglia, E., Law, R., and Collins, P. What is culture?
<http://www.wsu.edu:8001/vcwsu/commons/topics/culture/culture-index.html>,
retrieved Nov 16, 2010.
- Multiobject optimization, http://en.wikipedia.org/wiki/Multiobjective_optimization retrieved
Nov 16, 2010.
- North, J. M. and T. R. Howe. The Repast Symphony Development Environment, Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, Chicago, 2005. pp 159-166.
- Osyczka, A., and S. Kundu. A New Method to Solve Generalized Multicriteria Optimization

- Problems Using the Simple Genetic Algorithm. *Structural Optimization* vol 10, no. 2, 1995. pp 94-99.
- Pattison, Philippa, and Robins, Garry. Neighborhood-Based Models For Social Networks. *Sociological Methodology* vol 32, no. 1, 2002. pp 301-337.
- Peng, Bin. Cultural algorithms: Knowledge learning in dynamic environments. Ph.D. thesis. Wayne State University, Detroit, 2004.
- Peng, Bin. Knowledge Swarms in Cultural Algorithms for Dynamic Environment presented at the University Microfilms International, Ann Arbor, MI. 2005.
- Pulido, Gregorio Toscano and Coello Coello, A. Carlos, Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer, *Proceedings of the 2004 Genetic and Evolutionary Computation Conference, Part I, Seattle, 2004.* pp 225-237.
- Pulido, Gregorio Toscano, Luis Vicente Santana-quintero, and Carlos A. Coello Coello. EMOPSO: a Multi-objective Particle Swarm Optimizer with Emphasis on Efficiency, in *Proceedings of the 4th international conference on Evolutionary Multi-criterion Optimization, Matsushima, Japan, 2007.* pp 272-285.
- Pulido, Toscano Gregorio. Coello-Coello, A. C. A constraint-handling mechanism for particle swarm optimization. *IEEE Congress on Evolutionary Computation, 2004 (CEC2004).* vol.2, 19-23 June 2004. pp 1396 – 1403.
- Repast Symphony a. Introduction to Repast Symphony, <http://repast.sourceforge.net/docs/tutorial/SIM>, Extracted on March 23, 2011. 2008a.
- Repast Symphony b. Repast Symphony Installation Guide. July 24.

<http://repast.sourceforge.net/docs/tutorial/SIM/1%20Installation.html>. 2008b.

Reyes-sierra, Margarita and Coello-Coello, A. Carlos. Multi-Objective Particle Swarm Optimizers:

A Survey of the State-of-the-art, *International Journal of Computational Intelligence Research*, Vol. 2, No. 1, 2006. pp 287-308.

Reynolds, G. Robert and Dapeng Liu. Multi-objective Cultural Algorithms, *Proceedings of 2011*

IEEE Congress on Evolutionary Computation, to appear.

Reynolds, G. Robert. An Adaptive Computer Model of the Evolution of Agriculture in the Valley

of Oaxaca, Mexico. Ph.D. thesis, University of Michigan, Ann Arbor, 1979.

Reynolds, G. Robert. An introduction to cultural algorithms, *Proceedings of the Third Annual*

Conference on Evolutionary Programming, San Diego, 1994. pp 131-139.

Reynolds, G. Robert. An Adaptive Computer Model for the Evolution of Plant Collecting and

Early Agriculture in the Eastern Valley of Oaxaca, *Guila Naquitz: Archaic Foraging and Early Agriculture in Oaxaca, Mexico*, 1986. pp 439-500.

Reynolds, G. Robert and Saleem, Saleh. The impact of environmental dynamics on cultural

emergence, *Perspectives on Adaptation in Natural and Artificial Systems*, 2001. pp 253–280.

Reynolds, G. Robert and Saleem, Saleh. The impact of environmental dynamics on cultural

emergence, *Perspectives on Adaptation in Natural and Artificial Systems*. 2005. pp 253–280.

Reynolds, G. Robert and Ali, M. Computing with the social fabric: The evolution of

social intelligence within a cultural framework. *IEEE Computational Intelligence*

Magazine. 2008.

Reynolds, G. Robert and Ayman, Nazzal. Using Cultural Algorithms with Evolutionary Computing to Extract Site Location Decisions from Spatio-Temporal Databases, Proceedings of the 6th International Conference on Evolutionary Programming, 1997. pp 443-456.

Reynolds, G. Robert, Whallon, R., Ali, M. , and Zadegan, B. Agent Based Modeling of Early Cultural Evolution, in Proceedings of the IEEE World Congress on Computational Intelligence, Vancouver, B.C., July 16-21, 2006.

Richerson, Peter J., and Boyd, Robert. Not By Genes Alone: How Culture Transformed Human Evolution. 1st ed. University Of Chicago Press, December 31, 2004.

Saleem, Saleh. Knowledge-based Solution to Dynamic Optimization Problems Using Cultural Algorithms, University Microfilms International, P. O. Box 1764, Ann Arbor, MI,48106, USA. 2001.

Schaffer, David James. Some experiments in machine learning using vector evaluated genetic algorithms. Ph.D. thesis. Vanderbilt University, Nashville, 1984.

Schott, J. R. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Ph.D. thesis, Department of Aeronautics and Astronautics, MIT, Cambridge, Massachusetts, 1995.

Schwimmer, B. Anthropology on the internet: a review and evaluation of networked resources. Current anthropology. 1996. pp 561-568.

Sharaf, A.M. El-Gammal, A. A novel discrete multi-objective Particle Swarm Optimization

- (MOPSO) of optimal shunt power filter, Proceeding of Power Systems Conference and Exposition, 2009 (PSCE '09). Seattle, WA, USA. 15-18 March 2009. pp 1-7.
- Shi, Y and Eberhart, R. A Modified Particle Swarm Optimizer, in Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence. Anchorage, AK, USA, 1998. pp 69-73.
- Srinivas, N. and Kalyanmoy Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, Vol. 2, No. 1, 1994. pp 221-248.
- Stützle, Thomas and Holger H. Hoos. MAX-MIN Ant system, Future Gener. Comput. Syst. 16, no. 9, 2000. pp 889-914.
- T'kindt, V., Monmarche, N., Tercinet, F., and Laugt, D. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. European Journal of Operational Research, vol. 142, no. 2, 2002. pp. 250–257.
- Tylor, Sir Edward Burnett. Primitive culture: researches into the development of mythology, philosophy, religion, language, art, and custom. Gordon Press. 1920.
- Valenzuela-Rendon, Manuel, Eduardo Uresti-charre, and Itesm Monterrey. A Non-Generational Genetic Algorithm for Multiobjective Optimization, Proceedings of the Seventh International Conference on Genetic Algorithms, 1997, pp 658--665.
- Van Veldhuizen, David A, and Lamont , B. Gary. Multiobjective Evolutionary Algorithm Research: A History and Analysis. 1998.
- Van Veldhuizen, David A, and G. B Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance, Congress on Evolutionary Computation, vol 1, 2000. pp 204--

211.

Wu, Yong-hai, Fan, Qin-man, and Wang Feng. Application of Multi-objective Particle Swarm Optimization in Automobile Transmission Design. in Proceedings of Third International Conference on Information and Computing 2010 (ICIC'10). Vol 1, Wuxi, Jiang Su, China. 4-6 June 2010. pp 215-218.

Wynne, C. D., Animal Cognition - The Mental Lives of Animals. Palgrave Macmillan, Great Britain. 2001.

Zhang, Qingfu. n.d. Special Session on Performance Assessment of Multiobjective Optimization Algorithms/CEC 09 MOEA Competition. 2009.

Zitzler, E. Evolutionary algorithms for multiobjective optimization: methods and applications, Ph.D. dissertation, Swiss Federal Institute of Technology, Zurich, 1999.

Zitzler, Eckart, Deb, Kalyanmoy, Thiele, Lothar. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results, Evolutionary Computation, Vol. 8, No. 1, 2000. pp 173-195.

Zitzler, Eckart, and Thiele, Lothar. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 1999. pp 257-271.

ABSTRACT**Multi-Objective Cultural Algorithms**

by

DAPENG LIU**August 2011****Advisor:** Dr. Robert G. Reynolds**Major:** Computer Science**Degree:** Doctor of Philosophy

Evolutionary algorithms, including the Cultural Algorithms and other bio-inspired approaches are frequently used to solve problems that are not tractable for traditional approaches. Previously, research in the field of evolutionary optimization has focused on single-objective problems. On the contrary, most real-world problems involve more than one objective where these objectives may conflict with each other.

The newest implementation of the Cultural Algorithms to solve multi-objective optimization is named MOCAT. It is not the first time that the Cultural Algorithms have been used to solve multi-objective problems. Nonetheless, it is the first time that the Cultural Algorithms systematically merge techniques that have been popular in other evolutionary algorithms, such as non-domination sorting and spacing metrics, among other features. The goal of the thesis is to test whether MOCAT can

efficiently handle multi-objective optimization. In addition to that, we want to observe how the knowledge sources and agent topologies within a Cultural Algorithm interact with each other during the problem solving process.

The MOCA system was evaluated against the ZDT test set proposed by Zitzler (2000). Some basic results that were produced are as follows:

1. The MOCAT system was very effective in the generation of an appropriate configuration for solving problems with different combinations of these features. Even for a given problem, as information was added to the knowledge sources, adjustments in the topologies could be made effectively.
2. As the complexity of the problems increased in terms of the number of problem features, the MOCAT system's relative performance increased.
3. A problem with just a single problem feature, such as ZDT1 and ZDT5, was often effectively solved by just using one metric guide the solution process. However, if there were multiple problems, combining the two metrics together produced a synergy that outperformed each single metric based system.
4. This synergy resulted from the fact that they rewarded spread production in different ways. The spread metric focused on global distribution while the hyper-volume tended to support local optimization.
5. The configuration of the top performing MOCAT system varied markedly from one problem to the next.

Our experiments proved the potential of applying the Cultural Algorithms on multi-objective problems and open a gate to observing internal behaviors of various knowledge sources and social fabrics.

AUTOBIOGRAPHICAL STATEMENT

DAPENG LIU

Dapeng Liu entered the Department of Computer Science in 2003. He has worked in the database, software engineering, and Artificial Intelligence laboratories. At present, he is a software developer in “TheBrain” Technologies Corporation located in Los Angeles California. There he employs techniques he has acquired from his research experiences at Wayne State. Dapeng is in charge of addressing bugs in the Personal versions of the “Brain” software product. In his spare time he enjoys reading and looking deep inside software code.