3-28-2008

Multi-Objective Optimization of Mixed-Variable, Stochastic Systems Using Single-Objective Formulations

Todd J. Paciencia

MULTI-OBJECTIVE OPTIMIZATION OF MIXED VARIABLE, STOCHASTIC SYSTEMS USING SINGLE-OBJECTIVE FORMULATIONS

THESIS

Todd J. Paciencia, Captain, USAF

AFIT/GOR/ENS/08-17

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/08-17

MULTI-OBJECTIVE OPTIMIZATION OF MIXED VARIABLE, STOCHASTIC SYSTEMS USING SINGLE-OBJECTIVE FORMULATIONS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Todd J. Paciencia, BA

Captain, USAF

March 2008

AFIT/GOR/ENS/08-17

MULTI-OBJECTIVE OPTIMIZATION OF MIXED VARIABLE, STOCHASTIC SYSTEMS USING SINGLE-OBJECTIVE FORMULATIONS

Todd J. Paciencia, BA
Captain, USAF

Approved:

[illegible]
Dr. James W. Chrissis, Ph.D. (Chairman)

28 Mar 08
date

[illegible]
LtCol Mark A. Abramson, Ph.D. (Member)

26 MARCH 2008
date

AFIT/GOR/ENS/08-17

Abstract

Many problems exist where one desires to optimize systems with multiple, often competing, objectives. Further, these problems may not have a closed form representation, and may also have stochastic responses. Recently, a method expanded mixed variable generalized pattern search/ranking and selection (MVPS-RS) and Mesh Adaptive Direct Search (MADS) developed for single-objective, stochastic problems to the multi-objective case by using aspiration and reservation levels. However, the success of this method in approximating the true Pareto solution set can be dependent upon several factors. These factors include the experimental design and ranges of the aspiration and reservation levels, and the approximation quality of the nadir point. Additionally, a termination criterion for this method does not yet exist. In this thesis, these aspects are explored. Furthermore, there may be alternatives or additions to this method that can save both computational time and function evaluations. These include the use of surrogates as approximating functions and the expansion of proven single-objective formulations. In this thesis, two new approaches are developed that make use of all of these previous existing methods in combination.

AFIT/GOR/ENS/08-17

Acknowledgements

I would like to thank Dr. James Chrissis for his continual support and assistance throughout the process, and for allowing me to complete this in my own unique fashion. I would also like to thank Maj Jennifer Walston for her assistance both during and after her days here at AFIT. Without her quick answers and assistance, I may never have cleared some disconnects between my thought processes and her work. I must also thank LtCol Mark Abramson for his help in all areas, to include coding and theory/application. Having such a vast wealth of knowledge available was certainly an advantage in this process. Thank you to Dr. J.O. Miller for allowing me to hog computers to finish my computer runs. Thank you to Capt Dave Bethea for his collaboration in the surrogate realm and GPS/MADS. Thank you to Capt Ryan Kappedal for assisting in the brainstorming of possible crossover methods of discrete variables in GAs. Thank you to Capt Bryan Sparkman for helping when my home computer decided it had had enough. I must also thank family and friends, for attempting to keep me sane amidst the countless hours spend upon this undertaking. And finally, thank you to my thesis, for being done.

Table of Contents

List of Figures

List of Tables

MULTI-OBJECTIVE OPTIMIZATION OF MIXED VARIABLE, STOCHASTIC SYSTEMS USING SINGLE-OBJECTIVE FORMULATIONS

I. Introduction

1.1. Problem Setting

Optimization over multiple objectives is not as simple or straightforward as optimization of a single objective. There is typically no single optimal solution, as a solution may be better in one objective but worse in another. This causes a competition among the objectives, and so a true optimum is viewed in terms of a set versus a single point. This set, called the *Pareto set* or *Pareto front*, consists of those solutions that are not dominated, or those that are not worse for every objective than another solution in the set. Further complicating matters is that the decision variables may also be discrete or categorical, and that there may be some uncertainty in the objective function(s) or constraint(s). These problem settings are referred to as mixed variable and stochastic optimization, respectively.

The classical optimization problem for a stochastic system can be formulated as follows,

$$\min \; Z(w) = F(x, w) \tag{1.1a}$$

subject to

$$g_i(x, w) \leq 0 \,, \; i \in \{1, ..., M\} \,, \tag{1.1b}$$

$$x \in \mathbb{R}^{n_1} \,, \tag{1.1c}$$

$$w \in \mathbb{R}^{n_2} \,, \tag{1.1d}$$

where *x* represents the controllable design variables and *w* represents the random environment-determining variables. Therefore, the goal is to minimize in some manner over all feasible x and all possible values of w. For stochastic systems, the notions of feasibility and optimality are highly dependent on the problem, and must be precisely defined [70].

All constraints are assumed to be deterministic, and the system under study is assumed to have an objective function that cannot be explicitly evaluated and must be estimated through some sort of simulation (in which input or control variables produce a response). For simulation-based optimization, the general form of the stochastic objective function is typically replaced with its mathematical expectation. Under the assumption that the observed response is an unbiased approximation of the true system response, the observed response can be represented by $F(x,w) = f(x) + \varepsilon_w(x)$ where f is the deterministic, "true" objective function value and $\varepsilon_w(x)$ is the random error function associated with the simulation, where $E[\varepsilon_w(x)] = 0$.

In this research the mixed variables are included as follows. The decision space is partitioned into continuous and discrete variables, Ω^c and Ω^d respectively, as categorical variables may be mapped to discrete values. By further mapping the discrete values to the integers, the discrete part of the decision space can be represented as a subset of the integers, *i.e.* $\Omega^d \in \mathbb{Z}^{n^d}$, where n^d is the dimension of the discrete space. A solution $x \in \Omega$ is denoted as $x = (x^c, x^d) \in \left(\mathbb{R}^{n_c} \times \mathbb{Z}^{n_d}\right)$ where $x^c \in \mathbb{R}^{n^c}$, $x^d \in \mathbb{Z}^{n^d}$, and $n = n^c + n^d$ is the dimension of the decision space. With the inclusion of stochastic and multi-objective elements to the classic formulation, the problem can be formulated as:

$$\min E[F(x)] = E[f(x) + \varepsilon_w(x)] \tag{1.2a}$$

subject to

$$g_i(x) \le 0, \quad i \in \{1,...,M\}, \tag{1.2b}$$

where there are J objectives and $F(x):\left(\mathbb{R}^{n^c}\times\mathbb{Z}^{n^d}\right)\rightarrow\mathbb{R}^J$. That is, $F=\left(F_1,F_2,\ldots,F_J\right)$ and a solution x^* optimizes this set of objectives such that no other feasible point yields a better function value in all objectives.

There has been much work done using genetic algorithms and other methods to solve deterministic, multi-objective problems. However, these solutions can be random in their success and can vary in their completeness. Recently, a provably convergent algorithm, known as Stochastic Multi-Objective Mesh Adaptive Direct Search (SMOMADS), was developed by Walston to solve the stochastic, multi-objective class of problems [70]. The algorithm combines mixed-variable generalized pattern search/ranking and selection (MVPS-RS) and Mesh Adaptive Direct Search (MADS) developed for single-objective stochastic problems, with three multi-objective methods: interactive techniques for the specification of aspiration/reservation levels, scalarization functions, and multi-objective ranking and selection. Originally, the purpose of this thesis was to further develop SMOMADS; however, the research quickly evolved beyond that scope.

1.2. Purpose of the Research

SMOMADS samples *aspiration* and *reservation levels* to find points of intersection between the line, or plane, formed by a single set or design of aspiration and reservation levels and the Pareto front. The aspiration and reservation levels represent levels at which a solution is either ideal (aspiration) or unacceptable (reservation). The intersection is found using an achievement scalarization function of the objectives input into the pattern search method. The achievement scalarization function uses the *utopia point*, *nadir point*, aspiration level, and reservation level to form a single objective formulation. Although SMOMADS is convergent to Pareto solutions, the experimental design used may generate a front with considerable gaps in the objective space, thus excluding desirable solutions. Additionally, the reservation levels are likely dependent upon the estimate of the nadir point, which is the worst possible solution in the objective space. This point is often overestimated by using the worst value for each objective, as its value is typically hard to determine in practice. As mentioned, the achievement scalarization function uses the nadir point in its weighting of the objective functions. Therefore, using an incorrect nadir point may have some negative impact on SMOMADS. The same may be true for the utopia point; however, the utopia point is typically easier to find as its components are the best value in each objective irrespective of the other objectives. Additionally, little research has been conducted on the sensitivity of SMOMADS to the level of noise in the objective functions.

Once a design has been run using SMOMADS, it is important to be able to quantify the quality of the Pareto front approximation, as with several objectives the quality cannot be visually determined. Although the points found are Pareto optimal, there may exist large gaps or clusters, and desirable portions of the front may be missing. Quantification is not easily done, as fronts are not necessarily continuous, and for new problems the front is unknown. There are a few methods for comparing approximations quantitatively, but they generally cannot be used to determine the completeness of an approximation (*i.e.* are any portions missing).

Finally, because SMOMADS can be time-consuming, it may be more useful to use surrogates (models that approximate the true objectives) to help better determine the Pareto front after an initial set of design points have been used. Furthermore, no true methodology exists for using SMOMADS in a manner that guarantees a "full" Pareto front approximation upon completion. That is, no method exists to identify gaps or determine a point of termination. Methods used to fill the gaps may be used in conjunction with, or perhaps even in lieu of, SMOMADS.

1.3. Problem Statement

A main focus of this research is to determine the best experimental design to explore the Pareto objective space within SMOMADS. More specifically, the focus is to look at various performance measures of the approximation to see which design performs best based upon desired attributes (spread, lack of clusters, etc.). Further, this research examines the impact of the quality of the nadir point and the use of surrogates to help generate the Pareto front, so as to make SMOMADS as efficient as possible. Additionally, the sensitivity of SMOMADS to various levels of noise is evaluated, and an adaptive methodology is developed to use SMOMADS to find a representative Pareto front for any problem. Finally, existing methods other than SMOMADS are also evaluated. In particular, a bi-objective algorithm, BiMADS, is expanded to work for any number of objectives.

1.4. Overview

This thesis is organized as follows. Chapter II reviews SMOMADS and the methods and techniques it uses. In addition, existing multi-objective optimization methods, experimental designs, surrogate methods, nadir point approximations, and Pareto front quality metrics are reviewed. Chapter III presents the specific implementations used and investigated, as well as the methodology used. Chapter IV presents the data collection and analysis procedures, and the resulting analysis and computational results. Algorithms developed in this research are also presented. Chapter V presents the final conclusions and recommendations for future research.

II. Literature Review

This chapter begins with an overview of the SMOMADS algorithm, and the methods that it uses. The remaining sections cover experimental design concepts, surrogate methods, nadir point approximations, and Pareto front metrics as they apply to the SMOMADS algorithm.

2.1. SMOMADS

SMOMADS uses an achievement scalarization function to combine the multiple objectives into a single objective. In this form, the problem can then be solved using single objective optimization methods. Specifically, in the case of stochastic, linearly-constrained problems, Generalized Pattern Search with Ranking and Selection (GPS-RS) can be used. An extended version, Mesh Adaptive Direct Search with Ranking and Selection (MADS-RS) can be used when the problem is nonlinearly constrained. Both methods can also be applied to mixed variable cases (MVPS-RS, MVMADS-RS). A brief description of these methods follows, beginning with ranking and selection, followed by GPS-RS and MADS-RS.

2.1.1. Ranking and Selection. Problems with stochastic responses require a method to select a "best" point to account for variation, while also providing statistical assurance of correct selection. Ranking and selection (R&S) considers multiple candidates simultaneously at a reasonable cost. To do so, R&S detects a relative order of the candidates rather than generating precise estimates.

Let X_k denote the kth element of a sequence of random vectors and x_k denote a realization of X_k. For a finite set of candidate points $C = \left\{Y_1, Y_2, ..., Y_{n_c}\right\}$ with $n_c \geq 2$, let $f_q = f\left(Y_q\right) = E\left[F\left(Y_q, \cdot\right)\right]$ denote the true mean of the response function F at Y_q for each $q = 1, 2, ..., n_c$. These means can be ordered (minimum to maximum) as $f_{[1]}, f_{[2]}, ..., f_{[n_c]}$.

Denote by $Y_{[q]} \in C$ the candidate from C with the qth lowest true objective function value.

Given some $\delta > 0$, called the *indifference zone parameter*, no distinction is made between two candidate points whose true means satisfy $f_{[2]} - f_{[1]} < \delta$. In such a case, the method is *indifferent* in choosing either candidate as best. The probability of correct selection (*CS*) is defined as

$$P[CS] = P\left[\text{select } Y_{[1]} \mid f_{[q]} - f_{[1]} \geq \delta; q = 1,2,\ldots,n_c\right] \geq 1-\alpha\,, \tag{2.1}$$

where $\alpha \in (0,1)$ is the statistical significance level. Because random sampling guarantees $P[CS] = \frac{1}{n_c}$, the significance level must satisfy $0 < \alpha < 1 - \frac{1}{n_c}$.

Because the true objective function values are unavailable, it is necessary to work with the sample means of *F*. For each $q = 1,2,\ldots,n_c$, let s_q be the total number of replications at Y_q, and let $\left\{F_{qs}\right\}_{s=1}^{s_q} = \left\{F\left(Y_{qs}, W_{qs}\right)\right\}_{s=1}^{s_q}$ be the set of simulated responses, where $\{Y_{qs}\}_{s=1}^{s_q}$ are the replications at candidate point Y_q, and W_{qs} are realizations of the random noise. For each $q = 1,2,\ldots,n_c$, the sample mean $\bar{F}_q$ is given by

$$\bar{F}_q = \frac{1}{s_q}\sum_{s=1}^{s_q} F_{qs}\,. \tag{2.2}$$

The sample means can be ordered and indexed , letting $\hat{Y}_{[q]} \in C$ denote the candidate with the qth lowest estimated objective function value as determined by the R&S procedure. The candidate corresponding to the minimum mean response $\hat{Y}_{[1]} = \arg(\bar{F}_{[1]})$ is chosen as the best point. A generic R&S procedure is shown in Figure 2.1.1.

2.1.2. GPS-RS. Pattern search algorithms are defined through a finite set of directions used at each iteration. The direction set and a step length parameter are used to generate a discrete set of points, or mesh, around the current iterate. The mesh at iteration *k* is defined to be

$$M_k = \bigcup_{x \in O_k} \left\{x + \Delta_k^m D z \ : \ z \in \mathbb{N}^{n_D}\right\} \tag{2.3}$$

where O_k is the set of points for which the objective function f has been evaluated by the start of iteration k, Δ_k^m is called the *mesh size parameter*, and D is a positive set of directions that span $\mathbb{R}^n$. An additional restriction on D is that each direction $d \in D$, $j = 1, 2..., n_D$, must be the product of some fixed nonsingular generating matrix $G \in \mathbb{R}^{n \times n}$ by an integer vector $z_j \in \mathbb{Z}^n$ [67]. For bound and linearly constrained problems, the directions in D must be sufficiently rich to ensure that polling directions can be chosen that conform to the geometry of the constraint boundaries, and that these directions be used infinitely many times. A finite set of trial points, called the *poll set*, is then chosen from the mesh, evaluated, and compared to the incumbent solution. If improvement is found, the incumbent is replaced and the mesh is retained or coarsened by increasing the mesh size parameter Δ_k^m. If not, the mesh is refined and a new set of trial points is selected.

```
Procedure $RS(C, \alpha, \delta)$
**Inputs**: $C = \{Y_1, Y_2, ..., Y_{n_c}\}$, $\alpha \in (0,1)$, $\delta > 0$.
**Step 1**: For each $Y_q \in C$, use an appropriate statistical technique to determine the number
        of samples $s_q$ required to meet the probability of correct selection guarantee in
        (2.1), as a function of $\alpha$, $\delta$ and response variation of $Y_q$.
**Step 2**: For each $q = 1, 2, ..., n_c$, obtain replicated responses $F_{qs}$, $s = 1, 2, .., s_q$, and
        compute the sample mean $\bar{F}_q$, according to (2.2).
**Return**: $\hat{Y}_{[1]} = \arg(\bar{F}_{[1]})$
```

Figure 2.1.1: A Generic R&S Procedure [70]

At each iteration, an optional search may be conducted that although does not contribute to the convergence theory, does improve efficiency and performance. The search evaluates a finite number of mesh points that may be generated using a variety of methods; in this research a Latin Hypercube. If the search fails, the poll step is used.

In 1997, Torczon [67] defined and analyzed the derivative-free class of pattern search algorithms for unconstrained problems with continuously differentiable objective functions. In this work, it was shown that a subsequence of pattern search iterates $\{x_k\} \in \mathbb{R}^n$ converges to a first order stationary point x^*. The connection between pattern search and the positive basis theory of Davis [26] was introduced by Lewis and Torczon [40]. Pattern search was subsequently extended by Lewis and Torczon to problems with bound constraints [41] and a finite number of linear constraints [42]. Audet and Dennis [14] introduced a slightly generalized version called generalized pattern search (GPS), adding a hierarchy of convergence results for unconstrained and linearly constrained problems, including a new thory based on the nonsmooth calculus of Clarke [22]. Abramson [6] studied second-order behavior of GPS and showed that, under certain algorithmic choices, strict local maximizers and an entire class of saddle points can be eliminated from convergence consideration.

Audet and Dennis [15] extended their approach to handle nonlinear constraints by adding a filter method [31] for GPS that accepts new iterates if improvement in the objective function or an aggregate constraint violation function is found. Alternatively, Lewis and Torczon [43] handled nonlinear constraints by solving a sequence of bound constrained augmented Lagrangian subproblems [23].

Audet and Dennis [11] extended GPS to mixed variable problems, mixed variable pattern search (MVPS), with bound constraints by including user-specified discrete neighborhoods in the definition of the mesh, where the objective function f is assumed to be continuously differentiable for fixed discrete variable values. Abramson *et.al.* extended the results of [11] to linear [5] and non-linear constraints [1], again making use of the Clarke calculus [22], and the latter being augmented with a filter [15] to handle the nonlinear constraints.

The GPS framework, in conjunction with ranking and selection, was used by Sriver to address the random response case with mixed variables [64]. In this case, the poll set at each iteration is given by $P_k(x_k)\bigcup N(x_k)$ where $N(x_k)$ is a user-defined set of discrete neighbors around x_k and

$$P_k = \{x_k + \Delta_k(d,0) :\ d \in D_k^i\} \tag{2.4}$$

where $(d,0)$ denotes that continuous variables have been partitioned and that the discrete variables remain unchanged. The set of discrete neighbors is defined by a set-valued function $N\ :\ \Omega \to 2^{\Omega}$, where 2^{Ω} denotes the power set of Ω. By convention, $x \in N(x)$ for each $x \in \Omega$, and it is assumed that $N(x)$ is finite. A generic indifference-zone ranking and selection procedure $RS(P_k, \alpha, \delta)$ with indifference-zone parameter δ and significance level α is used to select among points in the poll set for improved solutions; *i.e.*, δ-near-best mean. Given a fixed rational number $\tau > 1$ and two integers $m^- \le -1$ and $m^+ \ge 0$, the mesh size parameter Δ_k^m is updated according to

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m \tag{2.5}$$

where

$$w_k \in \begin{cases} \{0,1\ldots,m^+\}, & \text{if an improved mesh point is found} \\ \{m^-, m^- + 1, \ldots, -1\}, & \text{otherwise.} \end{cases} \tag{2.6}$$

If no improvement is found, an extended poll step is conducted to search about any discrete neighbor $y \in N(x_k)$ that satisfies $f(x_k) \le f(y) < f(x_k) + \xi_k$, where ξ_k is called the *extended poll trigger*. Each neighbor satisfying this criteria, in turn, becomes the poll center, and the extended poll continues until either a better point than the current iterate is found, or else they are all worse than the extended poll center. Sriver showed that this algorithm has an iteration subsequence with almost sure convergence to a stationary point "appropriately defined" in the mixed-variable domain [63]. The mixed-variable GPS-RS Algorithm is shown in Figure 2.1.2.

A General MVPS-RS Algorithm

- INITIALIZATION: Let $X_0 \in \Omega$, $\Delta_0 > 0$, $\xi > 0$, $\alpha_0 \in (0,1)$, and $\delta_0 > 0$. Set the iteration and R&S counters $k = 0$ and $r = 0$ respectively.

- SEARCH STEP (OPTIONAL): Employ a finite strategy to select a subset of candidate solutions, $S_k \subset M_k(X_k)$ defined in (2.3) for evaluation. Use R&S procedure $RS(S_k \cup \{X_k\}, \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]} \in S_k \cup \{X_k\}$, update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r+1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$, see (2.5)-(2.6), and $k = k+1$, and repeat SEARCH STEP. Otherwise, proceed to POLL STEP.

- POLL STEP: Set extended poll trigger $\xi_k \geq \xi$. Use R&S procedure $RS(P_k(X_k) \cup N(X_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r+1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$, see (2.5)-(2.6), and $k = k+1$, and return to SEARCH STEP. Otherwise, proceed to EXTENDED POLL STEP.

- EXTENDED POLL STEP: For each discrete neighbor $Y \in N(X_k)$ that satisfies the extended poll trigger condition $\bar{F}(Y) < \bar{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$, and do the following.
  - Use R&S procedure $RS(P_k(Y_k^j), \alpha_r, \delta_r)$, to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r+1$. If $\hat{Y}_{[1]} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}_{[1]}$ and $j = j+1$, and repeat this step. Otherwise, set $Z_k = Y_k^j$ and go to the next step.
  - Use R&S procedure $RS(X_k \cup Z_k, \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r+1$. If $\hat{Y}_{[1]} = Z_k$, the step is successful, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$, see (2.5)-(2.6), and $k = k+1$, and return to the SEARCH STEP. Otherwise, repeat the EXTENDED POLL STEP for another discrete neighbor that satisfies the extended poll trigger condition. If no such discrete neighbors remain in $N(X_k)$, set $X_{k+1} = X_k$, $\Delta_{k+1} < \Delta_k$, and $k = k+1$, and return to the SEARCH STEP.

Figure 2.1.2: The Mixed-variable GPS-RS Algorithm [63]

2.1.3. Mesh Adaptive Direct Search. Mesh Adaptive Direct Search (MADS) is a class of algorithms developed by Audet and Dennis for minimization of nonsmooth functions of the type $f \ : \mathbb{R}^n \leftarrow \mathbb{R} \bigcup \{+\infty\}$ under general constraints $x \in \Omega \subseteq \mathbb{R}^n$ where $\Omega \neq \varnothing$. The feasible region Ω may be defined by blackbox constraints [12]. Thus, this class of algorithms is applicable also to nonlinearly constrained problems.

MADS is similar to GPS in the generation of the mesh as well as in the rules for updating the mesh. However, the key difference is that in MADS a separate poll size parameter Δ_p^k is introduced which controls the magnitude of the distance between the incumbent solution and trial points generated for the poll step, and that satisfies $\Delta_k^m \leq \Delta_k^p$ for all *k* such that $\lim_{k \in K} \Delta_k^m = 0 \Leftrightarrow \lim_{k \in K} \Delta_k^p = 0$ for any infinite subset of indices in *K*. In GPS, only one value $\Delta_k = \Delta_k^p = \Delta_k^m$ is used, and a set of positive spanning directions $D_k \subset D$ is chosen at each iteration.

In the poll step of MADS, neither restriction generally holds and the MADS frame (analogous to the poll set in GPS) is defined to be

$$P_k = \left\{ x_k + \Delta_k^m d \ : \ d \in D_k \right\} \subset M_k \, , \tag{2.7}$$

where D_k is a positive spanning set such that $0 \notin D_k$ and for each $d \in D_k$ the following conditions must be met [12]:

- *d* can be written as a nonnegative integer combination of the directions in *D*: $d = Du$ for some vector $u \in \mathbb{N}^{n_{D_k}}$ that may depend on the iteration number *k*,
- the distance from the frame center x_k to a frame point $x_k + \Delta_k^m d \in P_k$ is bounded above by a constant times the poll size parameter:
$$\Delta_k^m \|d\| \leq \Delta_k^p \max\{\|d'\| \ : \ d' \in D\} \, ,$$
- limits of the normalized sets $D_k = \left\{ \dfrac{d}{\|d\|} : d \in D_k \right\}$ are positive spanning sets.

The mesh size parameter typically decreases to zero at a faster rate than the poll size parameter, which allows the set of directions in D_k used to define the MADS frame to be chosen from increasingly larger sets as a limit point is approached. Audet and Dennis [12] showed that if this set is dense in the limit, convergence to a stationary point in the nonsmooth case can be ensured. They also provided an implementable instance in which directions are chosen randomly and a dense set of directions is acheieved with probability one [12].

The general MADS algorithm is shown in Figure 2.1.3. The extended algorithm for stochastic and mixed variable problems, the mixed variable MADS with ranking and selection (MVMADS-RS), is shown in Figure 2.1.4.

A General MADS Algorithm

- INITIALIZATION: Let $x_0 \in \Omega$, $\Delta_0^m \le \Delta_0^p$, D, G, τ, w^-, and w^+ satisfy the requirements of a MADS frame set given in (2.7). Set the iteration counter *k*=0.
- SEARCH AND POLL STEP: Perform the SEARCH and possibly the POLL steps (or part of them) until an improved mesh point x_{k+1} is found on the mesh M_k, where M_k is defined as for GPS in (2.3).
    - OPTIONAL SEARCH: Evaluate f_Ω on a finite subset of trial points on the mesh M_k.
    - LOCAL POLL: Evaluate f_Ω on the frame P_k, where P_k is as given in (2.7).
- PARAMETER UPDATE: Update Δ_{k+1}^m and Δ_{k+1}^p. Set *k*=*k*+1 and go back to the SEARCH AND POLL step.

Figure 2.1.3: A General MADS Algorithm [12]

A General MVMADS-RS Algorithm

- INITIALIZATION: Let $X_0 \in \Omega$, $\Delta_k^p \geq \Delta_k^m > 0$, $\xi > 0$, $\alpha_0 \in (0,1)$, and $\delta_0 > 0$. Set the iteration and R&S counters $k = 0$ and $r = 0$ respectively.

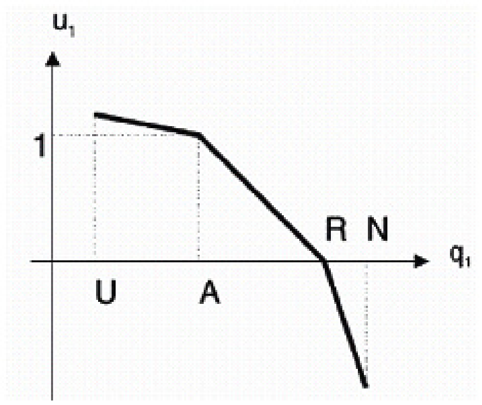
- SEARCH STEP (OPTIONAL): Employ a finite strategy to select a subset of candidate solutions, $S_k \subset M_k(X_k)$ defined in (2.3) for evaluation. Use R&S procedure $RS(S_k \bigcup \{X_k\}, \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]} \in S_k \bigcup \{X_k\}$, update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r+1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1}^p \geq \Delta_k^p$, $\Delta_{k+1}^m \geq \Delta_k^m$, and $k = k+1$, and repeat SEARCH STEP. Otherwise, proceed to POLL STEP.

- POLL STEP: Set extended poll trigger $\xi_k \geq \xi$. Use R&S procedure $RS(P_k(X_k) \bigcup N(X_k), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta$, and $r = r+1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1}^p \geq \Delta_k^p$, $\Delta_{k+1}^m \geq \Delta_k^m$ and $k = k+1$, and return to POLL STEP. Otherwise, proceed to EXTENDED POLL STEP.

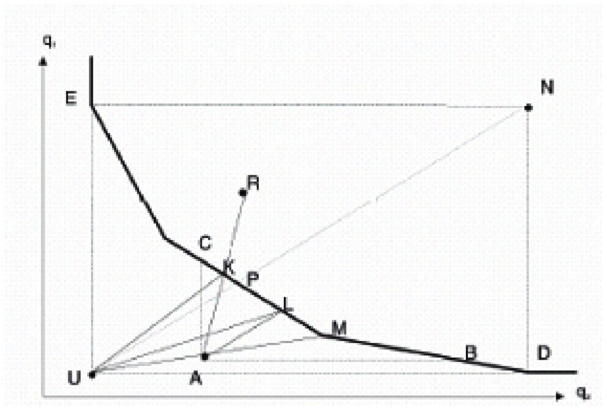
- EXTENDED POLL STEP: For each discrete neighbor $Y \in N(X_k)$ that satisfies the extended poll trigger condition $\bar{F}(Y) < \bar{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$, and do the following.
  - Use R&S procedure $RS(P_k(Y_k^j), \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta$, and $r = r+1$. If $\hat{Y}_{[1]} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}_{[1]}$ and $j = j+1$, and repeat this step. Otherwise, set $Z_k = Y_k^j$ and go to the next step.
  - Use R&S procedure $RS(X_k \bigcup Z_k, \alpha_r, \delta_r)$ to return the estimated best solution $\hat{Y}_{[1]}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta$, and $r = r+1$. If $\hat{Y}_{[1]} = Z_k$, the step is successful, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1}^p \geq \Delta_k^p$, $\Delta_{k+1}^m \geq \Delta_k^m$ and $k = k+1$, and return to the SEARCH STEP. Otherwise, repeat the EXTENDED POLL STEP for another discrete neighbor that satisfies the extended poll trigger condition. If no such neighbors remain in $N(X_k)$, set $X_{k+1} = X_k$, $\Delta_{k+1}^p < \Delta_k^p$, $\Delta_{k+1}^m < \Delta_k^m$, and $k = k+1$, and return to the SEARCH STEP.

Figure 2.1.4: MVMADS-RS

2.1.4. Aspiration/Reservation Levels and Scalarization Functions. Now considering the case of multiple objectives, points on the Pareto front can be found by varying the relative importance, *i.e.* trade-off coefficients or weights, of the distance to a given point, as shown in Figure 2.1.5. Using the utopia point **U**, any point between points **D** and **E** can be found. By using aspiration point **A** and varying the weights or slope of the ray emanating from it, points between **B** and **C** can be found. There are multiple methods for determining which ray to use [70]. The particular method implemented by SMOMADS uses the reservation point **R** as the second point in determining the direction of the ray [70]. This assumes that the decision maker has an idea of what is desired for each objective, as well as what minimum, or maximum, values are acceptable. These values are referred to as the aspiration and reservation levels, respectively points **A** and **R** from Figure 2.1.5.



(a) Component Achievement Functions for Minimized Criteria (Figure 4 in [47])



(b) Pareto solutions corresponding to different component achievement functions (Figure 3 in [47])

Figure 2.1.5: Component Achievement Functions for Pareto Optimal Solutions

The aspiration and reservation levels for each objective, a_i and r_i, respectively, where $i = 1,...,M$ and M is the number of objectives, are then used inside of an achievement scalarization function of the form

$$fx = -(\min(u) + \varepsilon \cdot \sum_{i=1}^{m} u_i). \tag{2.8}$$

The function u_i, where

$$u_i = \begin{cases} \alpha_i \cdot w_i \cdot (a_i - f_i) + 1, & f_i < a_i \\ w_i \cdot (a_i - f_i) + 1, & a_i \le f_i \le r_i \\ \beta_i \cdot w_i \cdot (r_i - f_i), & r_i < f_i \end{cases} \tag{2.9}$$

is of the type called *component achievement functions*; *i.e.*, strictly monotone functions of the objective vector components f_i (these functions are shown in Figure 2.1.5 (a)). The minimization of (2.8) provides proper Pareto optimal solutions nearest the aspiration level (point **K** in Figure 2.1.5 (b)). The notation used here was simplified from [70] to become more intuitive.

Walston used $w_i = \dfrac{1}{r_i - a_i}$ and $\varepsilon = 5$ [70]. Defining the *nadir point* as the component-wise supremum of all Pareto points (f_i^b), and the *utopia point* as the component-wise minimum of all feasible points (f_i^g), $\alpha_i = (0.1)\left(\dfrac{r_i - a_i}{a_i - f_i^g}\right)$ if $a_i \ne f_i^g$. Otherwise, $\alpha_i = (0.1)\left(\dfrac{r_i - a_i}{10^{-7}}\right)$. Similarly, $\beta_i = (-10)\left(\dfrac{r_i - a_i}{a_i - f_i^b}\right)$ if $\alpha_i \ne f_i^b$, and $\beta_i = (-10)\left(\dfrac{r_i - a_i}{10^{-7}}\right)$ otherwise. Walston used these specifics in her implementation of SMOMADS [70]. The nadir and utopia points are defined in more detail in Section 2.2.

2.1.5. SMOMADS Results. Walston proved that the sequence of iterates generated by each subproblem of SMOMADS contains a limit point that meets the first-order necessary conditions for Pareto optimality, almost surely. In addition, Walston proved if the sequence of iterates generated by a subproblem of SMOMADS converged to $\hat{x} \in \Omega$, then $\hat{x}$ meets the first-order necessary conditions for optimality almost surely [70].

Solving the set of subproblems, *i.e.* using a set of different aspiration and reservation levels, results in a set of Pareto optimal solutions. However, in general, if the frontier is non-convex or discontinuous, the resulting approximation to the Pareto front may be missing points of potential interest (note it will always be missing points due to the infinite nature of the front) [32]. To account for this, Walston proposed as future research a second stage to SMOMADS, replacing the single-objective ranking and selection routine of MVPS-RS with the Multi-Objective Computing Budget Allocation algorithm (MOCBA). However, there was indication that extending SMOMADS in some way may eliminate the need for the MOCBA phase. For the purposes of this research, SMOMADS is considered a one-stage algorithm and methods are evaluated to find the best Pareto front and eliminate gaps (or missing portions of the front). Another limitation of the SMOMADS algorithm is that there is no way to ensure the solutions found are as spread as possible along the Pareto front. Specifically, extreme points are not identified so that a user can know if they are finding points along the whole front or only a small portion. The SMOMADS method is summarized in Figure 2.1.6.

SMOMADS Algorithm

- Generate a set of Apriration/Reservation levels.
- For each choice of Aspiration/Reservation levels, combine the objective functions into an achievement scalarization function, and solve using MVMADS-RS or MVPS-RS.
- In the case of stochastic problems, because the solution converges to an efficient point with probability one in infinite iterations, check to ensure that a point is non-dominated before adding to the efficient set by comparing to solutions found thus far.

Figure 2.1.6: SMOMADS Algorithm [70]

2.2. Nadir and Utopia Point Approximation

Assuming at least one Pareto optimal solution exists, the nadir point $y^N \in \mathbb{R}^m$ is characterized by the component-wise supremum of all efficient points [29]:

$$y_m^N := \sup_{x \in \text{Pareto}} f^m(x); \quad m = 1, \ldots, M . \tag{2.10}$$

This point is not to be confused with an objective-wise maximum. The utopia point is the objective-wise minimum over the feasible set, or component-wise infimum of the Pareto set. That is, the utopia point is found by minimizing each objective and the ith component of the utopia point is the ith objective's minimum.

As previously mentioned, SMOMADS uses both the utopia point and nadir point when creating the achievement scalarization function. Furthermore, the Pareto front quality metrics that are dicussed in Section 3.2 also require use of these points, and it is likely the user will use the nadir point as a basis for choosing reservation levels and the utopia point as a basis for aspiration levels. Therefore, it is important to have accurate estimations of these points. The determination of the utopia point for any number of objectives involves only the solution of *M* single-objective problems over the whole feasible set Ω [29]. However, trying to estimate the nadir point using *M* single-objective problems could possibly lead to an overestimation of the true nadir point as shown in Figure 2.2.1.

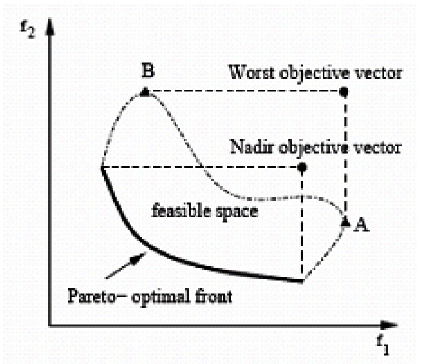


Figure 2.2.1: Nadir and Worst Objective Vectors [27]

Pay-off tables, or single objective optimal solutions evaluated for all objectives, are sometimes used to estimate the nadir point, but this can result in either underestimation or overestimation. Additionally, this optimization may be computationally expensive. In the case of two objectives, lexicographic optimization can be used to find the true nadir point. In the case of three objectives the PARETO^{Q-1} algorithm, which solves bi-objective subproblems, or faces of the original feasible set, can be used [29]. However, in the case of nonlinear problems, or more than three objectives, lexicographic optimization and PARETO^{Q-1} fail.

Another method for approximating the nadir point is to use a genetic or evolutionary algorithm. Such algorithms are often used to do the multi-objective optimization itself, but by emphasizing extreme Pareto-optimal solutions, an estimate of the nadir point can be achieved quickly without doing the full optimization. For this research, a slight modification of the Non-dominated Sorting Genetic Algorithm (NSGA-II) with elitist extremized crowding was used. For both the utopia and nadir points, weighted objective functions can be solved, using MVMADS-RS or MVPS-RS. This is presented in Section 3.1.

2.3. Pareto Set Quality Metrics

For the purpose of this research, the definition of a Pareto optimal solution is taken from [70], given as follows:

Definition 2.3.1. *A solution to a multi-objective optimization problem of the form* $\min_{x\in\Theta} F(x,w), F : \Theta \to \mathbb{R}^J$ *is said to be Pareto optimal at the point* $\hat{x}$ *if there is no* $x \in \Theta$ *such that* $F_k(x) \le F_k(\hat{x})$ *for* $k = 1,\dots,J$ *and* $F_i(x) < F_i(\hat{x})$ *for some* $i \in \{1,\dots,J\}$.

A solution is said to be *dominated* if it is not Pareto optimal with respect to the current Pareto approximation.

Given a set of points output from the SMOMADS algorithm, it is desirable to have some metric to determine the quality of the approximation to the true Pareto front, either to use as a termination criteria or as a means of comparison between fronts. Clearly, such points will be part of the true front, but large gaps may exist. One aspect that makes such a metric difficult is that the true Pareto front may not be known. Therefore, the metric needs to allow for the fact that fronts are not necessarily known *a priori*. Furthermore, metrics must also account for discontinuous or poorly-shaped fronts.

Few papers in the literature deal with instances where the true front is unknown *a priori*. However, Wu and Azarm developed five quality metrics that do not make the *a priori* assumption [72]. These metrics use the utopia point, nadir point, and regions in the objective space to, at a minimum, be able to compare two approximated fronts. Farhang-Mehr and Azarm furthered these concepts by developing an information-theoretic entropy metric that, in the best case, not only can be used to compare fronts, but may also be able to assess the quality of a single front, without the *a priori* assumption [30]. These metrics are presented further in Section 3.2.

2.4. Experimental Designs

Sampling the infinite space of all possible aspiration and reservation levels for a given range to produce Pareto optimal solutions during SMOMADS is certainly intractable. Therefore, it is important to sample in an intelligent manner, using experimental design methods. Such methods allow the user to incorporate several considerations into each design investigated, and typically, to also fit a response surface. Furthermore, it is desired to sample smartly and quickly, so as to achieve the best Pareto front approximation as fast as possible.

Traditional designs are factorial-based and allow estimation of linear and quadratic terms in least squares models. Some designs may also be fractionated, or reduced in size, by aliasing effects (*i.e.*, assuming some effects are not significant; $A = A + BC$). This can greatly reduce the number of runs if not all effects are significant. Typically, only some main effects and two-factor interactions are significant, and being able to estimate these effects un-aliased can be important. These traditional designs are presented further in Section 3.3.1.

If the dimension of the sample space is large, the number of samples required for a factorial based design may grow rapidly. It may also be the case that fitting a model is less important than sampling the space. Therefore, designs that uniformly sample the design space with fewer points are desired. The trade-off is that designs with fewer points may generate gaps in the sample space where no samples are taken. Such designs include Latin hypercubes, orthogonal arrays, and quasi-Monte Carlo sampling [35]. These are presented in detail in Section 3.3.2.

2.5. Surrogates

A surrogate is used to approximate a function that may be expensive to evaluate. Several surrogate approaches exist, with accompanying benefits and limitations. Interpolating surrogates, such as Kriging and radial basis functions, use an underlying weighted sum of basis functions to fit the data. Least-squares regression may provide a good fit, but may only be useful to identify significant terms in a model. Multi-adaptive regression splines (MARS) use a least-squares approach but fit the data piecewise (but with overlapping partitions).

Mulitvariate interpolation is not as well developed as univariate. Hermite interpolation has been expanded to the multivariate case, called *Hermite-Birkhoff interpolation*, where certain derivative information is known [28]. Quasi-interpolants use the sum of decaying functions centered at a point in the sample set to create approximating functions. For this research, interpolation methods are restricted to variations of Kriging and radial basis functions. These are presented further in Section 3.4.

Other methods also exist, such as Artificial Neural Networks (ANN). These models train and validate on sets of data and attempt to learn characteristics, so that a formed model can then be used to correctly predict a response from new data. The limitation of such models is that a model may train differently on the same set of data due to random weights. Nonetheless, these are also evaluated.

2.6. Existing Multi-Objective Methods and Their Limitations

It is perhaps important to explain why this research is even important. There are many existing multi-objective methods, but most are limited in some fashion. Much of the following comes from an excellent summary by Audet, Savard, and Zghal [13].

Genetic algorithms are obviously random in their solutions and are thus limited in the confidence they can generate in the resulting solutions. These algorithms also experience trouble in the mixed-variable case and with random elements present.

The linear weighting method converts a multi-objective problem into a single-objective problem by minimizing a convex combination of objectives,

$$\min_{x \in X} \sum_{i=1}^{p} w_i f_i(x), \tag{2.11}$$

where the weights w_i for $i = 1, 2, .., p$ are positive and sum to one. However, this method is unable to generate points in any nonconvex part of the Pareto front.

Another method uses approximations to reference points, formulated as

$$\min_{x \in X} \left\| F(x) - r \right\|_q = \left(\sum_{i=1}^{p} \left| f_i(x) - r_i \right|^q \right)^{1/q}, \tag{2.12}$$

where these formulations try to find feasible solutions close to the reference points *r*. This method may generate non-efficient points.

The weighted geometric mean approach uses a single-objective formulation to maximize the weighted geometric mean of differences between the components of the nadir point u and the objective functions

$$\max_{x} \prod_{i=1}^{p} (u_i - f_i(x))^{\lambda_i}, \tag{2.13}$$

where $f_i(x) \leq u_i$, $x \in X$, and $\lambda_i > 0$. This approach adds general constraints and requires the objective functions to be convex for a solution to be Pareto optimal.

The Normal Boundary Intersection approach by Das and Dennis [25] solves a series of single-objective optimization problems, with an additional equality constraint. This constraint maps the objective function value to a point on the normal emanating from a point in the Convex Hull of Individual Minima (CHIM), or the set of points in $\mathbb{R}^n$ that are convex combinations of $F(x_i^*)$ where x_i^* is a global minimizer for $i = 1,...,n$, and the boundary of the set of attainable objective vectors. This approach can be impractical in the blackbox optimization context [13]. Furthermore, NBI can have trouble finding extreme solutions in more than two objectives because there may be Pareto optimal points not in the CHIM, and NBI may find local solutions when the boundary is "folded" [25].

Audet, Savard, and Zghal recently devised a method with the intention of avoiding all of the previously mentioned shortcomings. This method is discussed in more detail in Section 3.6, but, as reported, is only applicable for two objectives.

Walston's work on SMOMADS applies to the stochastic case, but was more of a proof-of-concept rather than an optimal algorithm. One goal of this research investigates using Walston's work [70] in a more efficient manner and also implementing Audet, Savard, and Zghal's work [13] for any number of objectives.

Chapter III presents the specific algorithms and methods analyzed in this thesis to further both the use of SMOMADS and BiMADS. Concepts are introduced in an appropriate amount of detail, so that the reader may understand how they were implemented, but also so as to be concise. Also included in Chapter III are new algorithms and methods that resulted from this research, and any changes made to those from previous research.

III. Approach to the Problem

The following sections detail the various pre-existing methodologies that were evaluated during the course of this research to better implement SMOMADS and to create an alternative algorithm to SMOMADS. In addition, any modifications made to these methodologies are given here, as are a few new concepts and algorithms to be used in conjunction with SMOMADS and the alternative algorithm.

3.1. Nadir and Utopia Point Approximation

3.1.1. Genetic Algorithm Approach. To find the nadir point, two methods are evaluated in this research as alternatives to doing a maximization for each objective (and thus over-estimating the true nadir point). First, an elitist extremized crowding NSGA-II algorithm is used to approximate the nadir point. The concept for this algorithm came from Deb, Chaudhuri, and Miettinen [27]. Doing NSGA-II alone would perform the multi-objective optimization (or really the approximation thereof). However, by using extremized crowding, only those solutions that may assist in developing the nadir point are emphasized (recall Figure 2.2.1).

In general, genetic algorithms begin with an initial population of feasible points. Members of the population are then chosen for crossover and mutation operations, according to some fitness function, and then, depending upon the algorithm, either the resulting solutions or a best percentage of the two populations carry on to the next generation. In an elitist scheme, a best number of solutions from one generation carry on to the next generation regardless. Code written for NSGA-II by Aravind Seshadri [58] was used as a starting reference for implementing actual code for the elitist extremized crowding NSGA-II.

The initial population is constructed by taking the lower bound (based off the simple linear bounds) of a given decision variable and adding the range (*i.e.*, difference between upper bound and lower bound) multiplied by a 0-1 random number. For discrete variables, the initial value is randomly selected (uniformly) from the possible values for that variable. Chromosomes consist of the decision variable and objective function values and are checked for feasibility. If a chromosome has variable values that are infeasible, the chromosome is re-generated until feasible. In each generation, a non-dominated sort is then conducted which adds a ranking, or Pareto front number, to the chromosome according to the algorithm show in Figure 3.1.1, as taken from [58]. All completely non-dominated solutions are given a ranking of 1, that is, they most likely belong to the true Pareto front with respect to the current population.

Once the non-dominated sort is complete, a *crowding distance* is added to each chromosome. Solutions on a particular front are sorted from maximum to minimum based on each objective. The extreme solutions, minimum and maximum, for each objective get a rank equal to N', where N' is the number of solutions on the front. The solutions next to these extreme solutions get a rank of ($N'-1$) and so on. After a rank is assigned to a solution for each objective, the maximum value of the ranks is declared as the crowding distance for that solution. This helps to emphasize the solutions closer to the extreme solutions and therefore find the extreme points quicker. In addition, this maintains a good diversity of solutions and reduces the chance of having non-Pareto optimal solutions remain in the first non-dominated front [27]. The solutions are then sorted, based upon their rank and crowding distance.

Additionally, a uniqueness check may be conducted so that, if there are redundant solutions in the population, they are replaced by random solutions, similar to how the initial population was created. This is done to help prevent stagnation. As convergence of the population is desirable, this feature may not be entirely advantageous.

```
Non-Dominated Sort
  • Initialize the front counter to one, i=1, $F_1 = \{\ \}$.
  • For each individual p in main population P do the following:
      - Initialize $S_p = \varnothing$. This set will contain all the individuals
        dominated by p. Initialize $n_p = 0$. This will be the number of
        individuals that dominate p.
      - For each individual q in P,
          • If p dominates q, then add q to $S_p$, i.e. $S_p = S_p \bigcup \{q\}$,
          • Else if q dominates p, then increment the domination counter for
            p, i.e. $n_p = n_p + 1$.
      - If $n_p = 0$ (i.e., no individuals dominate p), then p belongs to the first
        front. Set rank of individual p to one, $p_{rank} = 1$. Update the first
        front set by adding p to $F_1$, $F_1 = F_1 \bigcup \{p\}$.
  • While the ith front is non-empty, $F_i \neq \varnothing$,
      - $Q \neq \varnothing$. This is the set for storing individuals on the (i+1)th front.
      - For each individual p in front $F_i$,
          • For each individual q in $S_p$ (those individuals dominated by p),
              o $n_q = n_q - 1$, decrement the domination count for
                individual q.
              o If $n_q = 0$, then none of the individuals in the subsequent
                fronts dominate q. Set $q_{rank} = i + 1$. Update Q with
                individual q, $Q = Q \bigcup q$.
      - Set $i = i + 1, F_i = Q$ (the next front).
```

Figure 3.1.1: Non-dominated Sort [58]

Once the non-dominated sort and crowding distances, as well as the final sort, are complete, a binary tournament selection is conducted. A mating pool with a size of approximately half the population is filled by repeatedly selecting two solutions from the population and choosing the one with lower rank, or in the case of an equal rank, the one with higher crowding distance. The selection of chromosomes for the tournament takes place by further ranking the population based on Pareto front rank and crowding distance.

These ranks are then used to build a cumulative probability distribution with which to compare random number draws. This process takes the place of a typical fitness function. Once the mating pool is filled, chromosomes are chosen at random, and perhaps more than once, for crossover or mutation.

The crossover operator used is Simulated Binary Crossover (SBX) and the mutation operator used is polynomial mutation [58]. SBX simulates the binary crossover observed in nature and is given as

$$\begin{aligned} c_{1,k} &= 0.5[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}] \\ c_{2,k} &= 0.5[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}], \end{aligned} \tag{3.1}$$

where $c_{i,k}$ is the ith child with kth component, $p_{i,k}$ is the selected parent and $\beta_k \geq 0$ is a sample from a random number. That random number is generated using the density

$$p(\beta) = \begin{cases} 0.5(\eta_c+1)\beta^{\eta_c}, & \text{if } 0 \leq \beta \leq 1 \\ 0.5(\eta_c+1)\dfrac{1}{\beta^{\eta_c+2}}, & \text{if } \beta > 1. \end{cases} \tag{3.2}$$

This distribution may be obtained from a random number u uniformly sampled between (0,1) according to

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{\eta+1}}, & \text{if } u \leq 0.5 \\ \dfrac{1}{\left[2(1-u)\right]^{\frac{1}{\eta+1}}}, & \text{if } u > 0.5. \end{cases} \tag{3.3}$$

η_c in (3.2) is the distribution index for crossover. Deb, Chaudhuri, and Miettinen used a distribution index of 20 [27].

The mutation operator uses polynomial mutation,

$$c_k = p_k + (p_k^u - p_k^l)\delta_k \tag{3.4}$$

where c_k is the resulting child and p_k is the parent with p_k^u as the upper bound on the parent component, p_k^l the lower bound, and δ_k a small variation calculated from a polynomial distribution of the form

$$\delta_k = \begin{cases} (2r_k)^{1/\eta_m+1} - 1, & \text{if } r_k < 0.5 \\ 1 - [2(1-r_k)]^{1/\eta_m+1}, & \text{if } r_k \geq 0.5, \end{cases} \tag{3.5}$$

where r_k is an uniformly sampled random number between (0,1), and η_m is the mutation distribution index. Deb, Chaudhuri, and Miettinen also used a mutation distribution index of 20 [27].

Discrete variables present a problem with regard to mutation and crossover because SBX and polynomial mutation are for continuous variables and resulting values will likely not be a part of the discrete set. An analysis of various ways to account for this, and their effectiveness, is presented in Section 4.3.

In the event of mixed variables or constraints other than simple bounds, the children are checked for feasibility, and if not feasible, the crossover or mutation is run again. In the case of crossover, a maximum of 100 attempts are made at feasibility, with completion when two feasible children are obtained. If 100 attempts complete without two feasible children, the single feasible child and one of the parents (randomly selected), or in the event of no feasible children, both parents, become the children. Similarly for mutation, 100 attempts are made, and if a feasible child does not occur, the parent becomes the child. The number of attempts is limited to 100, so as to limit the run-time of the algorithm. Again, this process limits the speed of evolution when constraints or discrete variables are included, but increased generations should account for the effect.

Once the crossovers and mutations have completed in a generation, the starting population and the children are pooled into one population, where the non-dominated sort is again conducted and the extremized crowding distances are again calculated. Here, the solution with maximum objective function value for each objective is made elite. Additionally, the remaining survivors are selected based on low rank, nearing the Pareto front, and high crowding distance. The entire process is repeated for a number of generations.

It is important to note here that although the aforementioned algorithm in its entirety is based on NSGA-II and the literature, it was developed specifically for this research. Again, performance of this algorithm on a suite of test problems and analysis on parameters are included in Section 4.3.

3.1.2. GPS/MADS Approach. Obviously, a user may not have prior knowledge of the utopia or nadir points whatsoever. Additionally, genetic algorithms, no matter how robust, are nonetheless heuristics. Therefore, without requiring the speed of a heuristic, it would be advantageous to have a more "mathematically sound" method of determining the nadir point (and utopia point), that could still be efficient. In addition, as much of this research is dependent upon use of MADS and GPS, it would be advantageous to also use MADS and GPS for this method.

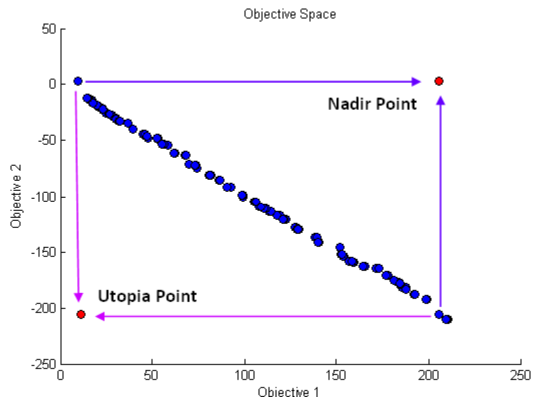


Figure 3.1.2: Utopia and Nadir Points

As mentioned in Section 2.2, the utopia point is found by performing an optimization for each objective separately. Let x_i^* be the global minimizer of objective i and $F(x_i^*)$ be the vector of all objective function values for x_i^*. Then $F_i(x_i^*)$ is the *i*th component of the utopia point; but also, some $F_j(x_i^*)$, where $j \neq i$, is the *j*th component of the nadir point. This is true because the utopia point components must occur at the extremes of the Pareto front to be non-dominated, and it is from the extremes of the

Pareto front that the nadir point is formulated, as shown in Figure 3.1.2. Therefore the nadir point can be determined once the minimizers corresponding to the utopia point components are known. Alternatively, finding the nadir point directly and accurately can be harder, as it constitutes a weighted objective method (minimizing all but one objective at a time). GPS/MADS can be used to perform the single-objective optimizations to find the utopia.

3.2. Approximated Pareto Front Quality Metrics

A true Pareto front is infinite in nature, and therefore, any set of solutions output from SMOMADS is only an approximation to the true front. In addition, not all Pareto fronts are continuous or well-shaped (*e.g.*, a curve). Therefore, just because a set of numerical solutions appears to be equally distributed over a region and well-shaped, it does not mean the complete front has been found. It is important to be able to determine when a representative Pareto front has been found, under any circumstances, and under the assumption that the actual Pareto front is unknown.

3.2.2. Quality Metrics. Wu and Azarm first attempted to solve this problem using a set of five quality metrics [72]. Using the utopia point or its estimate (where *g* denotes "good"), $p_g = (f_1^g, ..., f_m^g)$ and the nadir point or its estimate (where *b* denotes "bad"), $p_b = (f_1^b, ..., f_m^b)$, objective values are scaled, denoted by $\overline{f}_i(x_k)$ for some point $x_k \in X$. The number of Pareto solutions found is denoted n_p.

To fully develop the concepts behind the metrics, the definitions of the *inferior*, *non-inferior*, and *dominant* regions with respect to a point in the scaled objective space are needed.

Definition 3.2.1. An inferior region of a point p_j is defined as a hyper-rectangle, $S_{in}(p_j)$, such that for all $p_k \in S_{in}(p_j)$, $\overline{f}_i(x_k) > \overline{f}_i(x_j)$ and $\overline{f}_i(x_k) < 1$ for all $i = 1, ..., m$ where $p_k = (\overline{f}_1(x_k), ..., \overline{f}_m(x_k))$ and $p_j = (\overline{f}_1(x_j), ..., \overline{f}_m(x_j))$.

Definition 3.2.2. A non-inferior region of a point p_j, $S_{nin}(p_j)$, is the complementary region of $S_{in}(p_j)$; that is, $space(S_{nin}(p_j)) = 1 - space(S_{in}(p_j))$ where *space* denotes some portion of the scaled objective space hyper-rectangle between 0 and 1.

Definition 3.2.3. A dominant region of a point p_j, is the hyper-rectangle, $S_{do}(p_j)$, such that for all $p_k \in S_{do}(p_j)$, $\overline{f}_i(x_k) < \overline{f}_i(x_j)$ and $\overline{f}_i(x_k) > 0$ for all $i = 1,...,m$.

Therefore, for an observed Pareto solution set in the scaled objective space: $P = \left(p_1,...,p_{n_p}\right)$, the inferior, non-inferior, and dominant regions can be expressed as follows:

$$S_{in}(p) = \bigcup_{j=1}^{n_p} S_{in}(p_j) \tag{3.6}$$

$$space(S_{nin}(P)) = 1 - space(S_{in}(P)) \tag{3.7}$$

$$S_{do}(P) = \bigcup_{j=1}^{n_p} S_{do}(p_j). \tag{3.8}$$

1. **Hyperarea Difference (HD)**

This metric quantitatively evaluates the difference between the size of the objective space dominated by an observed Pareto solution set and that of the space dominated by the true Pareto solution set, or rather the space difference between the inferior regions of the two sets. The true set dominates the entire objective space; however, it is assumed to be unknown. Therefore, the utopia point is used as an estimate of the true Pareto solution set, giving a space of the inferior region equal to 1. Additionally, because the true set is unknown, it then becomes only possible to identify whether or not an observed Pareto solution set is worse than the true set when compared to another Pareto set. Therefore, an observed set with a lower *HD* is considered to be better than an observed set with a higher *HD*. Mathematically, *HD* is defined as:

$$HD(P) = 1 - space(S_{in}(P))$$
$$= 1 - \left\{ \sum_{r=1}^{n_p} \left\{ (-1)^{r+1} \times \left[\sum_{k_1=1}^{n_p - r+1} \cdots \sum_{k_l = k_{l-1}+1}^{n_p-(r-l+1)+1} \cdots \times \sum_{k_r = k_{r-1}+1}^{n_p} \prod_{i=1}^{m} \left[1 - \max_{j=1}^{r} (\bar{f}_i(x_{k_j})) \right] \right] \right\} \right\}. \tag{3.9}$$

Clearly, calculation of this metric becomes computationally expensive as the number of points becomes large, due to its recursive nature. In the test runs for this research, approximately 24 points seemed to be the point at which the computation became expensive for problems with only two or three objectives.

2. Pareto Spread

The Pareto Spread metric is in fact a set of metrics. The first metric is Overall Pareto Spread (*OS*), which quantifies how widely the observed Pareto solution set spreads over the objective space when the design objective functions are considered altogether. The volume ratio of two hyper-rectangles, that of one defined by the utopia and nadir points, and one defined by extreme points of an observed Pareto solution set, is *OS(P)*. This metric is given by

$$OS(P) = \prod_{i=1}^{m} \left| \max_{k=1}^{n_p} \left[\bar{f}_i(x_k) \right] - \min_{k=1}^{n_p} [\bar{f}_i(x_k)] \right|. \tag{3.10}$$

The second metric quantitatively depicts the solution range with respect to each individual design objective. For a particular objective *k*, it is given by

$$OS_k(P) = \left| \max_{k=1}^{n_p} \left[\bar{f}_k(x_i) \right] - \min_{k=1}^{n_p} [\bar{f}_k(x_i)] \right|. \tag{3.11}$$

These metrics also can only be used to compare two observed sets, as the true Pareto solution set may not spread across the entire objective space. A set with a higher spread is preferred to one with a lower spread.

3. Accuracy of the Observed Pareto Frontier (AC)

Pareto solutions not belonging to the current observed set must be non-inferior with respect to the current observed set and thus do not belong to either the observed set's inferior or dominant region. For an observed Pareto solution set or frontier approximation *P*, the quantity *AP(P)* denotes the region wherein an observed Pareto frontier falls. As the approximation becomes more accurate, *AP(P)* will go to zero. It is given by

$$AP(P) = 1 - space(S_{in}(P)) - space(S_{do}(P)), \tag{3.12}$$

where

$$space(S_{in}(P)) = \sum_{r=1}^{n_p} \left\{ (-1)^{r+1} \times \left[\sum_{k_1=1}^{n_p-r+1} \cdots \sum_{k_l=k_{l-1}+1}^{n_p-(r-l+1)+1} \cdots \times \sum_{k_r=k_{r-1}+1}^{n_p} \prod_{i=1}^{m} \left[1 - \max_{j=1}^{r} (\overline{f}_i(x_{k_j})) \right] \right] \right\},$$

$$space(S_{do}(P)) = \sum_{r=1}^{n_p} \left\{ (-1)^{r+1} \times \left[\sum_{k_1=1}^{n_p-r+1} \cdots \sum_{k_l=k_{l-1}+1}^{n_p-(r-l+1)+1} \cdots \times \sum_{k_r=k_{r-1}+1}^{n_p} \prod_{i=1}^{m} \left[\min_{j=1}^{r} (\overline{f}_i(x_{k_j})) \right] \right] \right\}.$$

Important to note is that in [72], the dominant space equation used $1 - \min_{j=1}^{r}(\overline{f}_i(x_{k_j}))$. As the dominant region is between the minimums and the origin, the original equation has the ability to double-count the inferior region in its calculation and result in a negative numerical measure, which is not valid. Again, it is clear the evaluation of these formulas becomes computationally expensive as the number of points becomes large.

The quantitative accuracy of the observed Pareto frontier *AC(P)*, is then $1/AP(P)$. An observed set with a higher *AC(P)* is preferred. Again, this can only be used to compare two sets, as the true frontier may be discontinuous, and therefore, a pre-defined criteria may be misleading; *i.e.*, the observed set is missing some region of solutions. However, a value of 1 is achieved when the observed set is empty.

4. Number of Distinct Choices (NDC_μ)

Let the quantity $\mu \in (0,1)$ be such that the *m*-dimensional objective space is divided into $1/\mu^m$ small grids or hyper-cubes (assume $1/\mu$ is integer for simplicity). This number should be chosen such that the decision-maker considers as similar any two solution points within a hypercube; *i.e.*, an indifference region $T_\mu(q)$, where *q* is an intersection of *m* grid lines in the objective space.

The number of distinct choices is defined by

$$NDC_\mu(P) = \sum_{l_m=0}^{v-1} \cdots \sum_{l_2=0}^{v-1} \sum_{l_1=0}^{v-1} NT_\mu(q,P), \tag{3.13}$$

where $q = (q_1, q_2, ..., q_m)$ with $q_i = l_i / v$ and $v = 1/\mu$, and where

$$NT_\mu(q,P) = \begin{cases} 1 \text{ if } \exists\, p_k \in P,\ p_k \in T_\mu(q) \\ 0 \text{ if } \forall p_k \in P,\ p_k \notin T_\mu(q). \end{cases} \tag{3.14}$$

This metric can be used to compare two solution sets, with a higher value being preferred. However, again, this metric cannot be used to determine the quality of the set in relation to the true set unless there is some prior knowledge of the true frontier.

5. Cluster (CL_μ)

The cluster metric accounts for the fact that sets of different sizes may give an equivalent number of distinct choices, and that in such a case, the smaller cardinality set is likely preferred. The cluster metric is defined by

$$CL_\mu(P) = \frac{N(P)}{NDC_\mu(P)}, \tag{3.15}$$

where *N(P)* is the number of observed Pareto solutions. If every solution is distinct, a value of 1 is achieved. Therefore, a lower value, or closer to 1, is preferred. A lower value implies the method being used to find the Pareto front is not finding redundant solutions. In any case, this metric can only be used to compare solutions.

As explained, these metrics can only be used to compare observed, non-empty Pareto solution sets. Furthermore, they can be conflicting, forcing tradeoffs among quality aspects. Therefore, because different aspects may be more valuable to different decision-makers, for the purposes of this research, the metrics are not combined into a single metric and are left for interpretation. This is additionally justified because some of the metrics do not possess a problem-specific range, and therefore, an equal consideration of more than one metric may be impossible in any aggregation. Of course, in the event one solution set is better in every metric, the decision is trivial.

3.2.2. Entropy Metric. Farhang-Mehr and Azarm [30] sought to create a metric that could not only be used for comparison of sets, but also assessing the quality of a single set. They created an information-theoretic entropy metric that quantifies the quality of a set in terms of distribution quality, or diversity, over the Pareto frontier. This *entropy* encapsulates into a single scalar different aspects of the Pareto approximation such as uniformity of distribution, coverage, number of solution points, and clustering.

The basic concept is to use influence functions that provide information about the neighborhood in the feasible space of each solution point, and to create a density function that aggregates the influence functions for each hypercube on a grid. Specifically, considering the *m*-dimensional objective space $F^m \subseteq \mathbb{R}^m$, the influence function of the ith solution point, $\Omega_i : F^m \to \mathbb{R}$ (here Ω is no longer denoting the feasible set, but rather a function), is a decreasing function of the distance to the ith solution point. For example, Farhang-Mehr and Azarm [30] recommended the Gaussian influence function:

$$\Omega(r) = \frac{1}{\sigma\sqrt{2\pi}} e^{-r^2/2\sigma^2} \tag{3.16}$$

where $r_{i\to y}$ is a scalar that represents the Euclidean distance of the point y and the ith solution point. A large value of σ yields a level influence function with no significant peaks, while a small value yields a sharp influence function with significant peaks.

The density function at any point y in the feasible objective space is defined as the sum of the influence functions from all solution points. That is,

$$D(y) = \sum_{i=1}^{N} \Omega_i(r_{i\to y}) \tag{3.17}$$

where $\Omega_i(.)$ is the influence function for the ith solution point.

The end result is that the generated density hyper-surface consists of peaks and valleys that can be easily identified. The peaks correspond to those areas with many nearby points, and the valleys correspond to those areas with few nearby points. The entropy metric measures how level the surface is. Again using the concept of indifference regions, a grid is constructed in the feasible domain, where the density $D = D(y)$ of each cell is computed using the center of each cell, y. The density is then normalized as:

$$\rho = \frac{D}{\sum_{k_1=1}^{a_1} \sum_{k_2=1}^{a_2} \dots \sum_{k_m=1}^{a_m} D_{k_1 k_2 \dots k_m}} \tag{3.18}$$

where a_i is the number of indifference regions in Objective *i*.

The normalized densities then sum to 1, and the entropy is

$$H = -\sum_{k_1=1}^{a_1} \sum_{k_2=1}^{a_2} \dots \sum_{k_m=1}^{a_m} \rho_{k_1 k_2 \dots k_m} \ln(\rho_{k_1 k_2 \dots k_m}) \tag{3.19}$$

where $H_{\max} = \ln(n)$ is the maximum possible value of H and n is the total number of grid centers. Therefore, a set with higher entropy is more evenly spread throughout the feasible region in the objective space and provides a better coverage of the space. For this research, the entropy metric is scaled such that $H / H_{\max} = H_{\text{Scaled}} \in [0,1]$.

However, the Pareto frontier obviously is not the entirety of the *m*-dimensional objective space. Therefore, using the normalized objective space, the observed Pareto set is projected into a $m-1$ dimensional objective space that gives a more representative density hyper-surface for the Pareto frontier. The vectors $\vec{u}_1, \dots, \vec{u}_m$ are the Cartesian unit vectors along each normalized objective, respectively. The projection direction $\vec{v}_1$ is the unit vector along $p^g p^b$ (utopia and nadir points) and the projection hyperplane is $m-1$ dimensional, passing through p^g, and is normal to the projection direction. The remaining projection vectors $\vec{v}_j$ are generated using Gram-Schmidt orthogonalization:

$$\vec{v}_j = \frac{\left[\vec{u}_j - \left(\vec{u}_j \cdot \vec{v}_1\right)\vec{v}_1 - \left(\vec{u}_j \cdot \vec{v}_2\right)\vec{v}_2 \dots - \left(\vec{u}_j \cdot \vec{v}_{j-1}\right)\vec{v}_{j-1}\right]}{\left\|\vec{u}_j - \left(\vec{u}_j \cdot \vec{v}_1\right)\vec{v}_1 - \left(\vec{u}_j \cdot \vec{v}_2\right)\vec{v}_2 \dots - \left(\vec{u}_j \cdot \vec{v}_{j-1}\right)\vec{v}_{j-1}\right\|}, \; j = 2,3,\dots,m. \tag{3.20}$$

The solution points, $\bar{f} = (\bar{f}_1, ..., \bar{f}_m)$, are then projected, using these projection vectors. This process is depicted in Figure 3.2.1.

It is in this projected space that the influence functions and density functions are calculated, and thus the entropy. In addition, the entropy should be comparable between two sets, and a given set may not contain the entire front. Therefore, the hypercube between the utopia point and nadir point is also projected to represent the feasible area and to be able to construct the indifference grid. For this research, the center points of the grid are projected, so as to maintain the decision-maker's true indifference regions. Constraints could be projected instead; however, this is sometimes difficult and computationally expensive in practice [30].

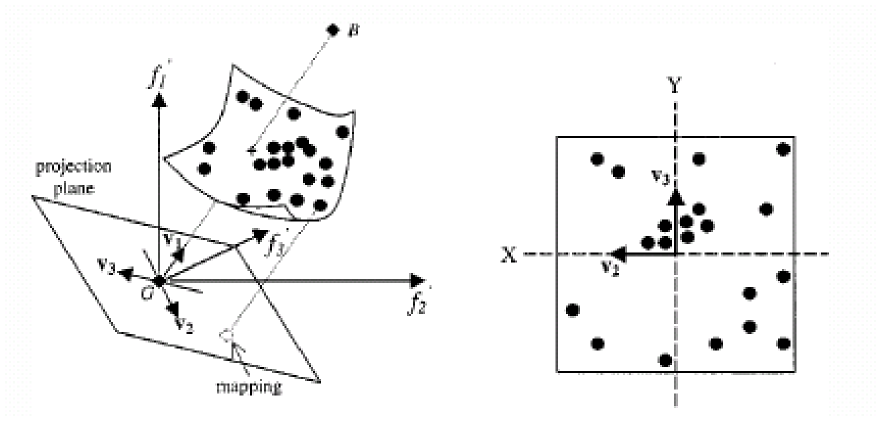


Figure 3.2.1: Projection of Solution Points [30]

The Pareto front may be discontinuous, and therefore, entropy cannot be used to quantify the quality of a single observed Pareto set unless separate projections are performed on known sub-regions (peaks will occur even in the case of a good approximation). Additionally, the value of σ impacts the value of entropy differently based upon the observed Pareto points and grid size, further complicating any interpretation of the metric beyond a comparison of two sets. Finally, a boundary effect may occur; *i.e.*, points near any boundary will likely have a smaller density simply because there is not as large a neighborhood around them for other feasible points to exist. However, this mainly impacts the visual density surface only, with minimal impact on the actual entropy [30].

Only unique projected density centers should be used to calculate entropy, as hyperdiagonals project to the same grid point, and the center may be falsely inflated. Figure 3.2.2 depicts a Pareto set both before and after projection. Looking at the corresponding density surfaces in Figure 3.2.3, with σ too large, sensitivity is lost. Conversely, with σ too small, the sensitivity may become too great. In general, in evaluating sample data for two and three objectives, $\sigma = 1/12$ seemed to provide the most reasonable, yet smooth, density surfaces and provided what appeared to be appropriate entropy values. Therefore 1/12 is used in this research.

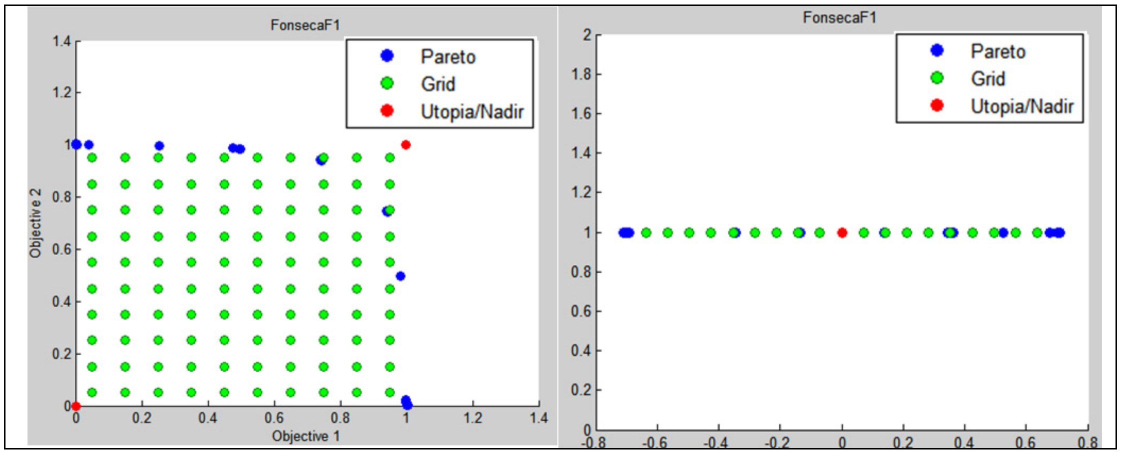


Figure 3.2.2: Example Pareto Set

A three objective example is shown in Figure 3.2.4 for further clarity, where the first plot depicts a Pareto approximation and the second plot depicts the density surface. Here indifference values of 0.2 in each objective and $\sigma = 1/12$ were used.

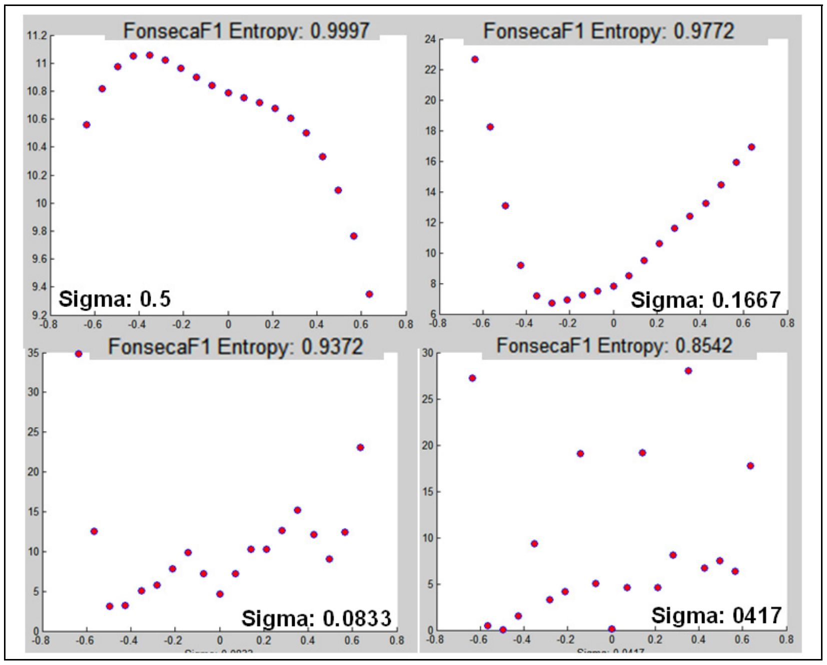


Figure 3.2.3: Example Density Surfaces

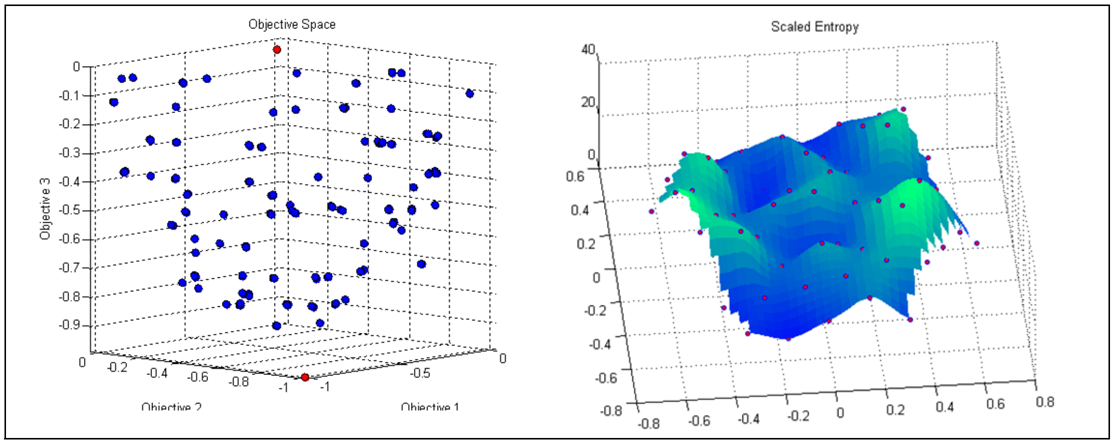


Figure 3.2.4: 3 Objective Example

3.2.3. Further Considerations. In the case of NDC_{μ}, μ is an important parameter. There are two alternatives in deciding a value for this parameter: either specifying a value for every dimension, or using a single value, as presented in [72]. For this research, and to provide robustness, a value for every dimension is used. The decision-maker is allowed to enter an indifference value, ω_i, for each original objective i. The parameter μ_i that determines the number of cells in each objective space is then calculated using:

$$\mu_i = \frac{1}{\left\lceil \frac{\left|p_i^b - p_i^g\right|}{\omega_i} \right\rceil} , \tag{3.21}$$

where the utopia and nadir points are not yet scaled. Similarly, different indifference values are allowed for the entropy metric, which, as mentioned previously, are used to construct the grid before projection. This allows the decision-maker's preferences to be incorporated, as each dimension in the projected space no longer corresponds to a single objective.

It is extremely important to mention something about the computational expense of these metrics in more than three objectives. An entropy metric in four objectives requires 10000 indifference hypercubes when dividing each objective into only 10 bins. This further complicates the use of these metrics as termination criteria.

3.3. Experimental Design

Typically, experimental designs are used to screen factors or to fit models to data. For the purposes of this research, the interest is more in sampling such that the most representative set of Pareto points for the entire front, in as few runs as possible, is achieved. In addition, if the resulting points are not representative enough, being able to fit models that yield the remaining points can be important. These two objectives may be conflicting, in that designs that yield the best front may, in fact, yield a bad predictive model.

3.3.1. Factorial and Composite Designs.

1. Full Factorial Designs

Full factorial designs are produced combinatorially, using every possible combination of levels of factors (things being sampled to determine their relationship to the response), or in the case of this research, the aspiration and reservation levels. The designer can choose a number of levels for each factor. These designs grow rapidly in size according to the number of factors and levels. Fractional factorials use a subset of these runs using alias structures based on significance assumptions.

2. Central Composite Design (CCD)

The CCD is considered most useful for sequential experimentation and is an efficient method to fit a second-order model. The CCD typically consists of a 2^k factorial (k factors, 2 levels), or fractional factorial of Resolution V (no main effects or 2-factor interactions aliased with each other, that is, no single column of the design matrix is the same as two columns of the design matrix multiplied by each other element by element) with n_F runs, $2k$ axial runs, and a number of center runs n_C. For clarification, in design of experiments a specific design level or sample is often referred to as a run. Using a distance $\alpha = (n_F)^{1/4}$ for the axial runs yields a rotatable design; that is, the variance of the fitted values remains unchanged when the design is rotated about the center [48]. For the implementation of this research, the best fraction (fewest runs) for up to five objectives, yielding a Resolution V design, is used for the factorial portion. Beyond five objectives, a half fraction is used.

Additionally, two variations of the standard or circumscribed CCD are investigated. The face-centered CCD uses $\alpha = 1$ to place axial points on the faces of the hypercube. The second variation, an inscribed CCD, effectively scales down the circumscribed CCD to the design space hypercube.

3. Box-Behnken

The Box-Behnken design requires three levels and is formed by combining 2^k factorials with incomplete block designs. These are usually very efficient in terms of runs and are rotatable or nearly rotatable [48].

4. Small Composite Design

Other designs also exist that are based off of factorial or composite designs, with the aim of reducing the numbers of runs as much as possible, or increasing the efficiency.

These designs are typically saturated or near-saturated and are used when the cost prohibits the use of a standard design. First, Plackett-Burman designs must be defined. These designs are two-level fractional factorial designs for studying $k = N - 1$ factors in *N* runs where *N* is a multiple of 4 [48]. As there is always an even number of factors in this research, these designs are not directly of interest, but are useful when constructing the small composite design.

The Small Composite Design (SCD) is a design such that the factorial portion is Resolution III* (defining relation does not contain any four-letter words, *i.e.*, *ABCD*), augmented with center and axial runs. This design aliases some main effects with 2-factor interactions (2FI) and, therefore, main effects and 2FI are highly correlated [54]. This could present a problem in the accuracy of the regression coefficients, although all coefficients are estimable in the second-order model [51]. For the initial sampling design (four and six factors), Draper-Lin SCDs are used. These are constructed as described in [34] by:

i) Calculating the minimum number of points for the cube-portion, $m = p - 2k$, where $p = (k+1)(k+2)/2$ and k is the number of factors

ii) Starting from a two-level Plackett-Burman design with a number of experiments equal to or higher than m

iii) Selecting k columns of the original Plackett-Burman design and removing the rest

iv) In the case of duplicate rows, removing one row for each duplication

v) Establishing the cube-portion with the rest of the rows

vi) Adding the selected axial and center points

For this research, appropriate SCDs are considered for two and three objectives. In the case of two objectives (four factors), columns 1,2,3, and 6 from the seven-factor Plackett-Burman are used with $\alpha = 1.41$ (which is $4^{1/4}$) and three center points, for a total of 19 runs. In the case of three objectives, columns 1-6 are used from the 19-factor Plackett-Burman with $\alpha = 1.57$ (which is $6^{1/4}$) and three center points, for a total of 35 runs [68]. For any additional sampling that may be required with different than four or six factors (due to factor-screening), SCDs from [51] are used, with $\alpha = 1.41$ for two and three factors, and $\alpha = 1.57$ for five factors.

5. Hybrid Designs

Hybrid designs were developed to achieve the same degree of orthogonality as a CCD, to be near-minimum-point in size, and to be near-rotatable. These designs use a k-1 factor CCD, with the kth factor set according to some optimality criteria [54]. Unfortunately, these designs thus far are only for k =3,4,6,7, but are evaluated for two and three objectives in this research despite the lack of generality [54,57]. For this research, 416A (this designator entails the number of factors, number of runs, and variant) with an added center point, and 628A are used. The design matrices can be found in [57]. These specific designs were chosen because, of the available hybrids, they provide the best efficiency, smallest maximum regional variance, rotatability, and information at the center of the design space [57]. In the case of three factors in further sampling, 311B is used due to its efficiency.

6. Minumum-Run Resolution V Designs

Minimum-run Resolution V designs are equireplicated two-level irregular fractions of Resolution V that can be used standalone, or as the factorial portion of a CCD. These designs are typically beneficial for greater than five factors [44], allowing a savings of runs beyond a typical fractional design. This design is only used in the present work for the three-objective case (22 runs), and axial and center points are added. The specific matrix was obtained from [19]. Methods do exist to create these designs, but they are not implemented in this research.

7. Koshal Designs

Koshal designs are saturated designs for modeling response surfaces of order greater than zero. In the case of a first-order model, the Koshal design is really just the one-factor-at-a-time design. The first-order plus interaction and second-order Koshal designs are also evaluated in this research, with actual design matrices being available in [51]. For strict DOE purposes, this design is likely not a good choice; however, the goal is really a balance between forming a model and exploring the design space.

8. Alphabetic Optimality Criteria

Designs can be generated for any number of runs according to some alphabetic optimality criteria. Examples are D-Optimality, where $\left|(X^T X)^{-1}\right|$ is minimized; A-Optimality, where the sum of the variances of the regression coefficients is minimized; G-Optimality, where the maximum scaled prediction variance over the design region is minimized; and V-Optimality, where the average prediction variance is minimized. These computer-generated designs are generally inferior to either small composite or hybrid designs with respect to reducing the number of runs [48]. Furthermore, many of the previously mentioned designs were developed specifically for alphabetic optimality criteria. In fact, using design optimality with a single criterion is the antithesis of design robustness [50].

Regardless, a D-Optimal algorithm from MATLAB® is included for investigation as part of this research due to its immediate availability. Here, the design is formed in order to fit the best quadratic model, and five center points, as well as axial points, are added to the design. An exchange algorithm is used to optimize the design.

It is interesting to note that alphabetic optimality criteria for multiple responses do not yield the same design as in the single-response case. A method for finding such designs is available in the univariate case [36]. However, this research typically has more than one regressor variable, and so the D-Optimal design will only be best with respect to a single response.

9. Other Design Criteria

Other methods have been developed to create designs that require fewer runs while optimizing some measure. Specifically, low cost response surface measures (LCRSM) seek to minimize expected integrated mean squared error (EIMSE) for some number of runs and factors, while finding a best model among some set of candidates [8]. EIMSE is used so as to not ignore bias errors in a fitted model. Although the resulting designs are useful, on a Pentium 450 MHz machine they can, for example, require an entire day to generate while only looking at 10 candidate models [8]. Therefore, they are not included in this research.

3.3.2. Other Sampling Methods. Factorial and composite-based designs can grow in size rapidly. Therefore, it may be desirable to sample as uniformly as possible with a restriction on the number of runs. Furthermore, uniformity may be desirable in general. The following sampling methods provide alternatives that allow a designer to perform such sampling. Unfortunately, these designs may also be far less desirable when forming a model, in this case, using aspiration and reservation levels.

1. Latin Hypercube Sampling

With Latin Hypercube Sampling (LHS), for a number of samples *k*, each variable is divided into *k* bins of equal probability. Uniform probability distributions are assumed for the variables in this research. Then, *k* samples are randomly taken with the following restrictions: 1) each sample is randomly placed inside a bin, and 2) for all one-dimensional projections of the *k* samples and bins, there will be one and only one sample in each bin. There exists more than one arrangement of bins and samples; thus, the performance of such a sampling may vary. However, methods exist to reduce correlation among samples, and in this research, this criterion is also included. Lattice sampling is also investigated, where a sample is placed at the center of its respective bin.

2. Orthogonal Array Sampling

LHS is a special case of orthogonal array sampling (OA). An OA produces a set of samples that yield uniform sampling in any *t*-dimensional projection of an *n*-dimensional design space, where $t < n$ and *t* is called the strength. In LHS, $t = 1$.

Additional parameters include *p*, the number of bins in each variable, and λ, the number of samples in each bin following the projection. The OA, denoted by OA(*k,n,p,t*), is such that, for any *t* columns of the array, each ordered *t*-tuple appears exactly λ times, and $k = \lambda p^t$ [35]. For this research, OAs were constructed for the two and three-objective cases using OA(8,4,2,3), OA(80,6,2,4), OA(9,4,3,2), and OA(64,6,4,3) [10]. In addition OAs were constructed for a number of factors other than four or six using OA(4,3,2,2), OA(8,5,2,2), and OA(16,5,4,2). OAs are non-trivial to construct and are therefore often taken from publications. However, Owen [65] has made available an archive of C-code that builds OAs. A four sample, $t = 2$ OA is shown in Figure 3.3.1 for three dimensions.

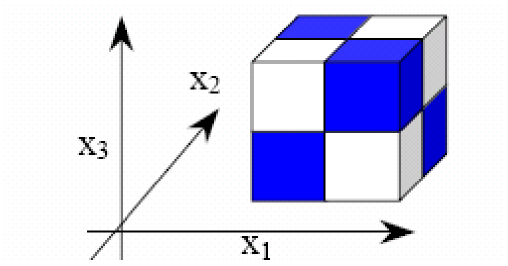


Figure 3.3.1: Example OA [35]

3. Hammersley Sampling

Hammersley sampling is a quasi-Monte Carlo sampling method that uniformly disperses sample sites throughout the design space for any number of samples *N*. First, note that the radix-R notation of an integer *p* is defined as:

$$p = p_0 + p_1 R + ... + p_m R^m , \tag{3.22}$$

where $m = [\ln(p)/\ln(R)]$, and the brackets denote the integer portion. The inverse number radix function for p is:

$$\phi_R(p) = p_0 R^{-1} + p_1 R^{-2} + ... + p_m R^{-m-1} . \tag{3.23}$$

The Hammersley sequence of n-dimensional points is generated as:

$$x_n(p) = \left(p/N, \phi_{R_1}(p), \phi_{R_2}(p), ..., \phi_{R_{n-1}}(p)\right), \tag{3.24}$$

where $p = 0,...,N-1$ and the values for $R_1,...,R_{n-1}$ are the first $n-1$ prime numbers; *i.e.*, (2,3,5,…) [35]. This algorithm is considered a modern design of experiment.

4. Nearly Uniform Designs

A uniform design is a space-filling design, the forming of which is an NP-Hard problem [46]. Therefore, various methods are used to form uniform, or nearly uniform, designs by minimizing a given *discrepancy* (measure of non-uniformity). These designs are often used in quasi-Monte Carlo methods. A uniform design (UD) for *n* runs and *s* factors is a $n \times s$ matrix, where each column is a permutation of $\left[1, 2, ..., n\right]$. Let $U = \left[u_{kj}\right]$ be a uniform design and $x_k = (x_{k1}, ..., x_{ks})$, where

$$x_{kj} = \frac{2u_{kj} - 1}{2n} , \tag{3.25}$$

for $k = 1,...,n$; $j = 1,...,s$. $P = \{x_1,...,x_n\}$ is called the *induced* design of *U*, *i.e.*, the corresponding 0-1 range design to *U*. The discrepancy value (quantitative measure of discrepancy) is denoted as $D(U) = D(P)$. A *U*-type design that minimizes the discrepancy (D-value) for a given *n* and *s* is the UD $U_n(n^s)$. A design with a near-minimal D-value is a nearly uniform design (*NUD*).

Many measures of uniformity have been defined, to include star discrepancy (really just the Kolmogorov-Smirnov statistic), symmetrical discrepancy, and centered L$_2$-discrepancy. For the purposes of this research, the centered L$_2$-discrepancy is used, denoted by *CD(P),* where

$$CD(P)^2 = \left(\frac{13}{12}\right)^s - \frac{2}{n}\sum_{k=1}^{n}\prod_{j=1}^{s}\left(1+0.5\left|x_{kj}-0.5\right|-0.5\left|x_{kj}-0.5\right|^2\right)$$
$$+\frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\prod_{i=1}^{s}\left(1+0.5\left|x_{ki}-0.5\right|+0.5\left|x_{ji}-0.5\right|-0.5\left|x_{ki}-x_{ji}\right|\right), \tag{3.26}$$

and P is as defined in [46]. Centered L_2-discrepancy is invariant under coordinate rotation.

The method used in this research to form the *NUD* improves the design generated from the good lattice point (glp) method. The cutting method of Ma and Fang [46] first uses the glp method to generate a *NUD* $U_p(p^s)$ using the following steps:

1. Find the candidate set of positive integers

$$A_{p,s} = \{a \,:\, a < p, \quad \gcd(a^j, p) = 1, \quad j = 1,...,s\}$$

where gcd is the greatest common divisor.

2. For each $a \in A_{p,s}$, construct $U^a = \left[u_{kj}^a\right]$, where

$$u_{kj}^a = ka^{j-1}(\text{mod}\, p) + 1,\ k = 1,...,p\text{, and } j = 1,...,s\,.$$

3. Find $a_* \in A_{p,s}$ such that $D(U^{a_*}) = \min_{a \in A_{p,s}} D(U^a)$. Then U^{a_*} is a *NUD* $U_p(p^s)$.

With these steps alone, the cardinality of $A_{p,s}$ may be smaller than desired. For the cutting method, *NUD* $U_p(p^s)$ is found such that $p >> n$ and p or p+1 is prime. In this research $p = 79$ is used for $n < 50$; otherwise, the closest prime number to $1.5n$ is used. This worked fairly well in getting similar results to [46]. The resulting induced design is denoted as $P = \{c_1,...,c_p\} = C$. The following steps are conducted to complete the cutting method:

1. For $l = 1,...,s$, the rows of C are reordered by sorting column l of C. Each resulting matrix is denoted as $C^{(l)} = \left[c_{kj}^{(l)}\right]$.
2. For $m = 1,...,p$, let $C^{(l,m)} = \left[c_{kj}^{(l,m)}\right]$ where

$$c_{kj}^{(l,m)} = \begin{cases} c_{k+m-n-1,j}^{(l)}, & m > n,\ k = 1,...,n \\ c_{k,j}^{(l)}, & m \le n,\ k = 1,...,m-1,\ j = 1,...,s \\ c_{k+p-n,j}^{(l)}, & m \le n,\ k = m,...,n. \end{cases}$$

3. The elements of each column of $C^{(l,m)}$ are relabeled by 1,2,…,*n*, according to their magnitude. The resulting matrix is $U^{(l,m)}$.
4. The matrices $U^{(l,m)}$ are compared, and the one with the smallest *CD(P)* is chosen as the *NUD* $U_n(n^s)$.

The cutting method takes cuts, Figure 3.3.2(b) and Figure 3.3.2(c), of the original design, Figure 3.3.2(a), and for a coordinate wraps the cuts such that points are either near 0 or 1, Figure 3.3.2(d). These points are uniformly scattered over the wrapped space and are linearly transformed, such that they are uniformly scattered over the unit space. Further details and examples may be found in [46].

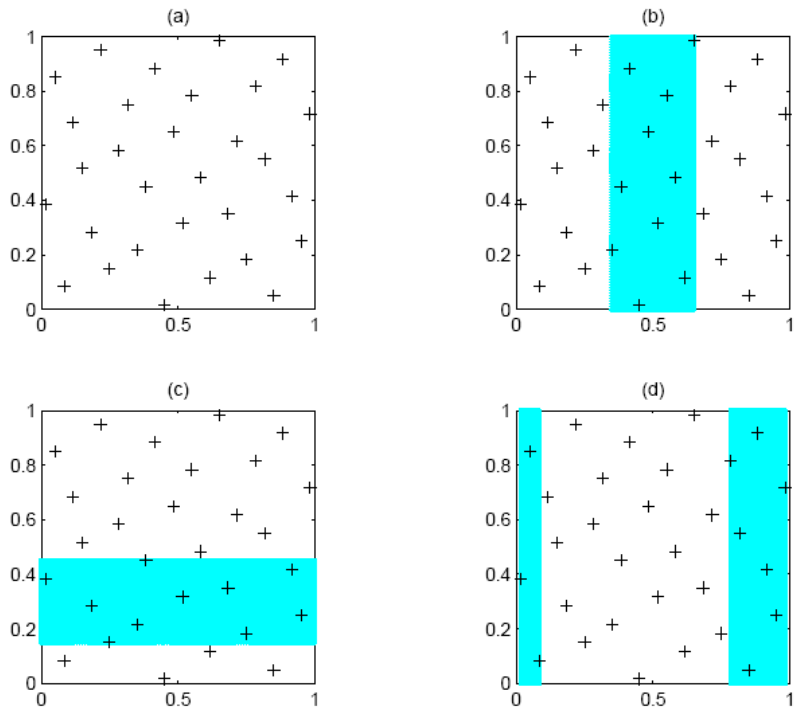


Figure 3.3.2: Cutting Method [46]

The cutting method is advantageous over just using the glp method, regardless of the discrepancy used. As the number of runs and factors increases, the cutting method can become time-consuming, comparative to glp, but it is nonetheless fairly efficient and produces a more uniform design.

3.4. Surrogates

Surrogates have several uses in this research. First, function evaluations may be expensive, so it may be beneficial to form surrogates that can be used inexpensively. Secondly, instead of continuing to use GPS or MADS, it may be possible to simply use the surrogates in some manner to complete the Pareto front.

Many surrogates have some underlying global polynomial. Typically up to quadratic terms are included, although cubic terms may be used as well. Terms of higher order often lose meaning or relevance and instabilities may arise [60]. Nonetheless, cubic terms are evaluated in this research, as appropriate. Descriptions of some surrogates not entirely based upon a least squares approach follow.

3.4.1. Kriging. Kriging is an approximation scheme that interpolates data, relying on two component models, expressed as

$$y(x) = f(x) + Z(x) \tag{3.27}$$

where $f(x)$ represents a global model, and $Z(x)$ is the realization of a stationary Gaussian random function with zero mean and non-zero covariance that gives a localized deviation from the global model [71]. The function *Z(x)* typically produces localized deviations to interpolate the data, although a non-interpolating model is possible. For this research Design and Analysis of Computer Experiments (DACE) is used to form Kriging surrogates [45]. Kriging assumes deterministic functions, that is, repeated runs for the same inputs give the same reponse. Since the functions in this research are stochastic, the mean response is used for the surrogate. In some cases, the noise affects the function enough so that very different responses can be achieved. As it may be difficult to label some response more likely or more important than another, the mean is still considered the best method for aggregating multiple responses into one value, with the hope that the mean approaches the true response in the limit.

Specifically, in DACE data is normalized and the global model is built using a regression model of p functions, f_i, $i = 1,...,p$, such that

$$\widetilde{F}(\beta_{:,l}, x) = \beta_{1,l} f_1(x) + ... + \beta_{p,l} f_p(x). \tag{3.28}$$

The function *Z(x)* has a covariance between realizations $z(w)$ and $z(x)$ of

$$E[z_l(w) z_l(x)] = \sigma_l^2 \widetilde{R}(\theta, w, x), \tag{3.29}$$

where σ_l^2 is the process variance for the lth component of the response, x and w are design sites in $\mathbb{R}^n$, and $\widetilde{R}(\theta, w, x)$ is the correlation model with parameters $\theta \in \mathbb{R}^n$.

The set S of m design sites has the expanded design matrix,

$$F = \left[f(s_1),...,f(s_m)\right]^T. \tag{3.30}$$

The predictor at a point x is defined as

$$\hat{y}(x) = f(x)^T \beta^* + r(x)^T \gamma^*, \tag{3.31}$$

where $\beta^* = (F^T R^{-1} F)^{-1} F^T R^{-1} Y$ from generalized least squares, Y is the matrix of responses, γ^* is computed via the residuals $R\gamma^* = Y - F\beta^*$, and

$$R_{ij} = \widetilde{R}(\theta, s_i, s_j),\ i, j = 1,...,m, \tag{3.32}$$

$$r(x) = \left[\widetilde{R}(\theta, s_1, x),...,\widetilde{R}(\theta, s_m, x)\right]^T. \tag{3.33}$$

DACE also restricts the correlations to the form:

$$\widetilde{R}(\theta, w, x) = \prod_{j=1}^{n} \widetilde{R}_j(\theta, w_j - x_j). \tag{3.34}$$

Specifically, $\widetilde{R}_j(\theta, d_j)$, where $d_j = w_j - x_j$, can be defined in one of seven ways [45]:

1. Exponential: $\exp(-\theta_j |d_j|)$
2. General Exponential: $\exp(-\theta_j |d_j|^{\theta_{n+1}})$, where $0 < \theta^{n+1} \le 2$
3. Gaussian: $\exp(-\theta_j d_j^2)$
4. Linear: $\max\left\{0, 1 - \theta_j |d_j|\right\}$
5. Spherical: $1 - 1.5\xi_j + 0.5\xi_j^3$, $\xi_j = \min\left\{1, \theta_j |d_j|\right\}$
6. Cubic: $1 - 3\xi_j^2 + 2\xi_j^3$, $\xi_j = \min\left\{1, \theta_j |d_j|\right\}$

7. Spline:

$$\varsigma(\xi_j) = \begin{cases} 1 - 15\xi_j^2 + 30\xi_j^3 , & 0 \le \xi_j \le 0.2 \\ 1.25(1-\xi_j)^3, & 0.2 < \xi_j < 1 \\ 0, & \xi_j \ge 1. \end{cases} , \ \xi_j = \theta_j \left| d_j \right|$$

The choice of regression polynomial, correlation function, and θ parameters can significantly affect the quality of a surrogate in its prediction. The parameter θ can take any positive value, with a smaller value corresponding to a flatter approximation. It is optimized within DACE, given that more restrictive upper and lower bounds are provided by the user. Without knowledge of appropriate bounds, θ can be estimated by maximizing a likelihood function as an unconstrained nonlinear optimization problem, or by an alternative method proposed by Mardia and Marshall [37]. However, both of these methods could require exhaustive evaluation of values for θ.

In this research, the lower and upper bounds on the θ vector are determined by the NOMADm software [2]. In this approach, the conditioning of the correlation matrix from GLS is used to iteratively halve the lower bound from initial θ values, and the correlation matrix itself is used to iteratively double the upper bound until a criterion is met.

The discussion on bounds for θ is important because lower values for the θ vector are generally desirable, although not always. Low values are often achieved by using more sample sites [21], and a higher θ causes the correlation to deteriorate more quickly [55]. An example is given in Figure 3.4.1, depicting a reduced quadratic polynomial on Dias Γ2 Objective 1 data (the objective is the z-axis) in plots (a) and (b). The effect of θ is clear as the small bumps, or mounds, in the surface disappear with the smaller θ vector. Of course, with a constant polynomial the θ vector can cause significant bumps in the surface, shown in Figure 3.4.1(c).

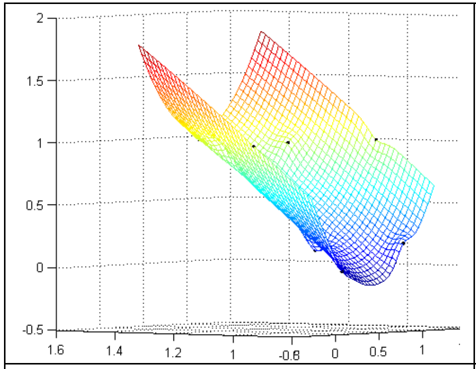
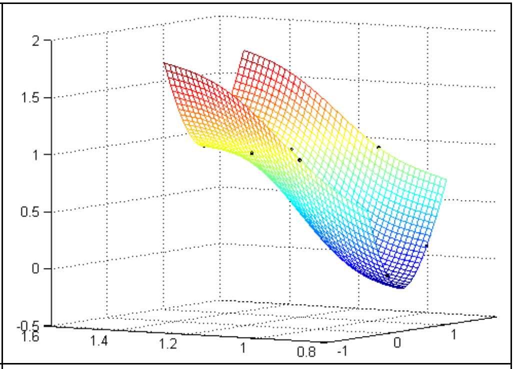
	
(a) Thetas: 20, 30	(b) Thetas: 0.5, 0.5

Figure 3.4.1: Example Reduced Quadratic Kriging Surrogate

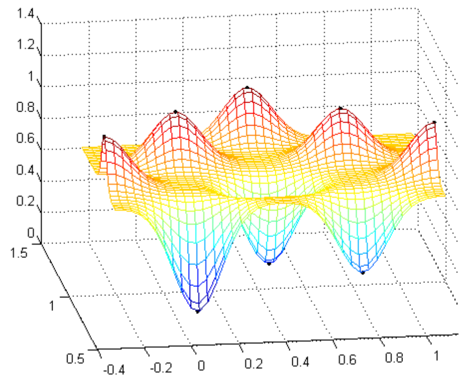


Figure 3.4.2: Constant Kriging Surrogate (Thetas: 10,10)

In the case of high noise, a nugget parameter can be introduced into the correlation function to smooth the data for a non-interpolating Kriging model. However, in this research the noise level is generally not assumed to be large, so this should not be required. To quantify the quality of the Kriging surrogate, a cross-validation approach can be employed. In general, the surrogate can be formed using subsets of samples, with the remaining samples used to estimate mean square error [62,59]. This is necessary because, although interpolation has zero error, it does not mean the surrogate is an effective predictor. The cross-validation method developed in Section 4.11 is based on analysis from this research.

It should be mentioned that a method called cokriging exists. Cokriging is similar to Kriging, but uses secondary performance functions highly correlated to the primary function, such as gradient information, in the event the primary function is expensive to evaluate. Unfortunately, the inference from auxiliary data becomes extremely demanding as dimensionality increases, because correlation and cross-correlation between variables and their partial derivatives is required [71].

3.4.2. Radial Basis Functions. Radial Basis Functions (RBFs) also have the ability to approximate non-linear functions and to interpolate data. A RBF ϕ has a symmetric output around a center μ, a sample site. That is $\phi(x) = \phi\left(\|x-\mu\|_p\right)$, where $\|\cdot\|_p$ is a vector *p-norm*, normally with $1 \le p \le 2$ (in this research, we assume the Euclidean-norm, $p = 2$). A set of RBFs serves as a basis for representing multiple functions expressible as linear combinations of chosen RBFs and a polynomial function *p(x)*:

$$y(x) = p(x) + \sum_{j=1}^{m} w_j \phi\left(\|x - x_j\|\right). \tag{3.35}$$

(3.35) can be expensive with a large number of RBFs, and so a k-means clustering algorithm is sometimes used to reduce the number of RBFs employed [71]. In this research however, there should be no need to limit the number of RBFs.

Several RBFs exist with various advantages to each. The bi-harmonic, $\phi(r) = r$ with a linear polynomial, and the tri-harmonic or cubic spline, $\phi(r) = r^3$ with a quadratic polynomial, are popular for fitting functions of three variables. The multi-quadric, $\phi(r) = \sqrt{r^2 + c^2}$, is useful for fitting topographical data. The thin-plate spline, $\phi(r) = r^2 \log(r)$, is popular for fitting smooth functions of two variables. Other RBFs include the inverse quadric, $\phi(r) = (r^2 + c^2)^{-1/2}$, and the Gaussian, $\phi(r) = \exp(-cr^2)$ [9]. Recall that *r* is the norm from (3.35), and $c \in \mathbb{R}$ is a positive, fixed constant. In this research, all of the mentioned RBFs are evaluated, as well as constant, linear, quadratic, and cubic polynomials (with and without interaction terms). The RBF estimator code developed by Abramson [3] was used as a starting point and expanded for this research.

It should be mentioned that RBFs are much more general than presented here. The following are the general definitions of classes of RBFs [56]:

1. Surface splines are any RBF such that $\phi(r) = r^k$, $k \in \mathbb{N}$ and odd, or $\phi(r) = r^k \log(r)$, $k \in \mathbb{N}$ and even.
2. Multiquadrics are any RBF such that $\phi(r) = \sqrt{r^2 + c^2}^k$, $k > 0$ and $k \notin \mathbb{N}$.
3. Inverse multiquadrics are any RBF such that $\phi(r) = \sqrt{r^2 + c^2}^k$ and $k < 0$.
4. Gaussians are any RBF such that $\phi(r) = \exp(-cr^2)$.

To solve for the weights *w* and polynomial coefficients c' (note this c' vector is different than the scalar in the RBF), the weights are such that for interpolation values $f = (f_1, ..., f_m)$, $y(x) = f(x)$, where $y(x)$ is the true response. Additionally, because there are more parameters than data,

$$\sum_{j=1}^{m} w_j p(x_j) = 0. \tag{3.36}$$

This gives the system

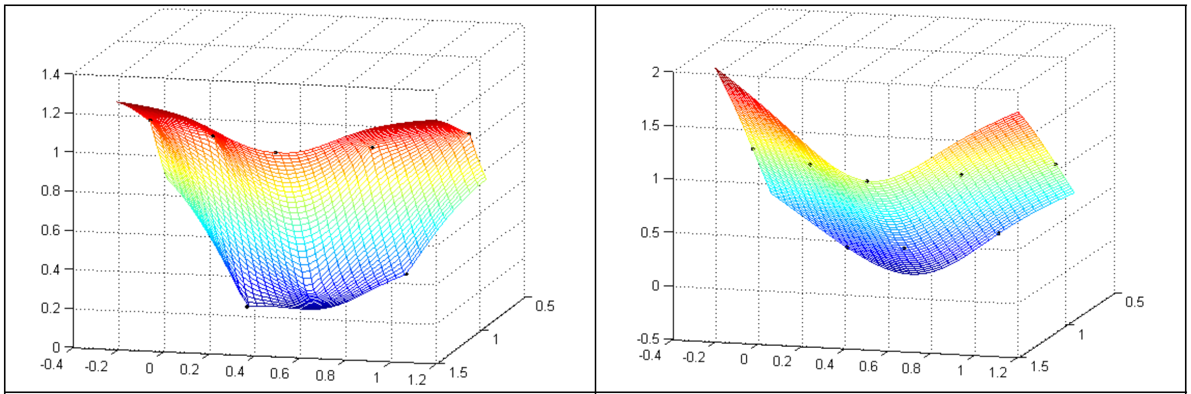
$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} w \\ c' \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{3.37}$$

where $A_{i,j} = \phi\left(\left\|x_i - x_j\right\|\right)$ for *i,j*=1,…,*m*, $P_{i,j} = p_j(x_i)$ for *i*=1,...*n*, *j*=1,...,*k*, and *k* is the number of polynomial coefficients in the basis representation [9,24].

The remaining question is how to determine the scalar *c* for the applicable RBFs. Hans Bruun Nielsen suggested a value of 1 in all cases [24]. In cases of large ranges of distances, this value has little impact. In general, there does not currently exist a best way to choose *c* [17]. For the multi-quadric RBF, Franke found that using the average distance between centers worked well [17]. Furthermore, as *c* becomes larger, accuracy increases. However, over a finite rectangular grid, as *c* increases, problems with conditioning of *A* from (3.37) are much more likely to occur. Therefore, for this research the average distance between centers is generally used, so as to provide some accuracy benefit, but also to prevent too much degradation in the condition of A. This is further justified by results presented in Section 4.11.

Baxter [17] provided an in-depth look at RBFs and the conditioning of A in the case of multi-quadric and Gaussian RBFs. In general, Baxter cautioned against the use of Gaussian RBFs, noting the inability to interpolate constants on a grid and sensitivity to c. Effective preconditioners for the preconditioned conjugate gradient method (PCG) were presented by Baxter [17], however, these matrices could not in all cases be generalized. Fortunately, using the identity matrix (*i.e.*, no preconditioning) still reduces error, although it requires more iterations of PCG. Unfortunately, PCG in either case does not always guarantee an efficient means of solving any near-singular system. For this research, if a system is identified as ill-conditioned, singular-value decomposition is applied to achieve better coefficients.

An example of how RBFs interpolate data is shown in Figure 3.4.3. The first plot depicts the interpolation using an underlying constant polynomial, while the second plot depicts how these interpolations are less pronounced with a different polynomial and kernel (the effect of the polynomial should be clear). This data is again Dias Γ2 Objective 1 data (z-axis is the objective).



(a) Bi-Harmonic Constant	(b) Tri-Harmonic Reduced Quadratic

Figure 3.4.3: RBF Surrogates

3.4.3. Nadaraya-Watson Estimator. Kernel regression or kernel smoothing is a nonparametric fitting method, the cornerstone of which is the Nadaraya-Watson estimator. This estimator is used to approximate a function at a point *x* according to

$$\hat{f}(x) = \frac{\sum_{i=1}^{N} \overline{F}_i K_h(x - X_i)}{\sum_{i=1}^{N} K_h(x - X_i)}, \tag{3.38}$$

where $\overline{F}_i$ is the function value of design site X_i, K_h is the kernel function which has the property $\int_{-\infty}^{+\infty} K_h(x) = 1$, $x - X_i$ is input as a 2-norm divided by *h*, and *h* is the smoothing parameter or bandwidth [63]. Viewing the approximation as a weighted sum, K_h determines the "shape" of the weights, and *h* determines the "size." The degree of nonlinearity is essentially determined by *h*, with smaller values of *h* allowing more curvature, but also allowing outliers to affect the estimation. A univariate example is shown in Figure 3.4.4.

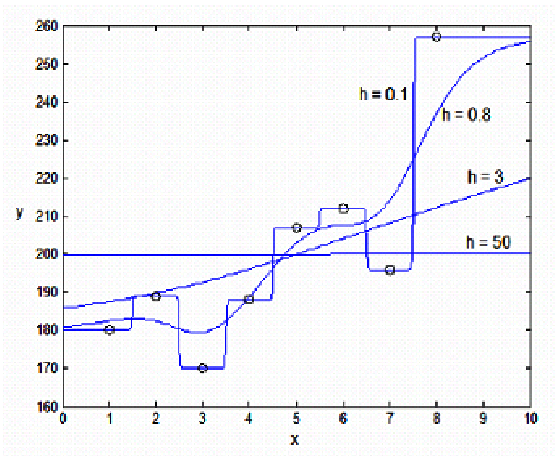


Figure 3.4.4. Effect of *h* on curvature [63]

Code developed by Abramson, Dunlap, and Sriver is used in this research to form the surrogate [4]. Kernels, $K(u)$, that are evaluated include: uniform, $0.5 \cdot I(|u| \leq 1)$; triangle, $(1 - |u|)I(|u| \leq 1)$; Epanechnikov, $0.75 \cdot (1 - u^2)I(|u| \leq 1)$; quartic,

$15/16 \cdot (1-u^2)^2 I(|u| \le 1)$; triweight, $35/32 \cdot (1-u^2)^3 I(|u| \le 1)$; Gaussian, $\frac{1}{\sqrt{2\pi}} \cdot \exp(-0.5u^2)$; and cosinus, $\frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right) I(|u| \le 1)$. I is an indicator function that returns a 0-1 value, depending on whether the expression given in the input argument is satisfied or not. Lower and upper bounds for h are input, with a golden section search optimizing h with respect to sum of squared errors in a cross-validation approach. For this research, a lower bound of 0.1 and an upper bound of 50 are used.

3.4.4. Artificial Neural Networks (ANN). Artificial Neural Networks (ANN) are also investigated in this research. An ANN is a structure of nodes, weights, and functions, which attempts to "learn" data, so that, it can yield a correct response output for any new data. Theoretically, ANNs work like the human brain, simulating biological information processing by processing data through neurons, or brain cells. As knowledge accumulates, connections between the neurons strengthen; *i.e.*, weights become more accurate. A drawback of ANNs is that they often over-train, or learn traits too well, and become bad predictors. Additionally, ANNs are typically not deterministic, and will train differently upon every instance.

Neural networks contain layers, with each layer containing neurons. A typical model consists of an input layer, output layer, and one or more hidden layers. The hidden layer(s) allow the network to learn non-linear relationships. The optimal number of neurons in the hidden layer(s) and the optimal number of hidden layers can be problem-dependent, but a rule of thumb is to start with one hidden layer with a number of neurons equal to half of the total number of variables in the input and output layers [18]. The number of neurons and layers should never exceed the total number of input and output variables. Activation functions, g, determine the response of each neuron and introduce the non-linear relationships.

Looking at a single node or neuron, as illustrated in Figure 3.4.5, input data is weighted and summed with a constant bias term θ_i, after which this sum is input to the activation function *g*.

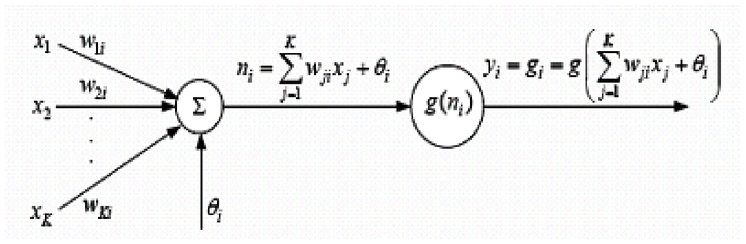

x_1
x_2
x_K
w_{1i}
w_{2i}
w_{Ki}
Σ
θ_i
$n_i = \sum_{j=1}^{K} w_{ji}x_j + \theta_i$
$g(n_i)$
$y_i = g_i = g\left(\sum_{j=1}^{K} w_{ji}x_j + \theta_i\right)$

Figure 3.4.5: Neuron in ANN [39]

Within MATLAB®, the activation function *g* can be chosen as the identity, hyperbolic tangent, or $\log\text{sig}(x) = \frac{1}{1+e^{-x}}$.

In this research, two ANNs are evaluated: a generalized regression network, and a feed-forward backpropagation network. In the generalized regression network, two layers are used. The first uses radial basis neurons with bias; the second uses identity neurons without bias. The radial basis neurons use an activation function of e^{-x^2}. A spread parameter allows the user to vary the smoothing. The feed-forward backpropogation network uses a user-defined number of layers, number of neurons within the layers, and activation functions. Weights and bias are determined by backpropogation, reducing the sum of squares of the differences between the generated outputs and the desired outputs. MATLAB® provides many options for backpropogation. In this research the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is used, due its better computational efficiency over other methods included in MATLAB®. The BFGS algorithm adjusts variables iteratively according to $x_{k+1} = x_k + \alpha_k p_k$, where α_k is selected to minimize error along the search direction p_k, using a backtracking line search. The initial search direction is the negative gradient of the error, and successive search directions are calculated by solving $B_k p_k = -\nabla f\left(x_k\right)$ [52]. The approximate Hessian B_{k+1} is updated using

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{3.39}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right)$.

3.5. Least Squares & Factor-Screening Methods

For this research, least squares regression is evaluated as a possible surrogate and method to screen factors. It must be noted here that although it is best to capitalize on the properties of the response surface methodology designs, this may not happen in practice. Runs of the design may lead to dominated points within SMOMADS. These points should not be used in fitting any surrogate as they negatively impact the model (*i.e.*, only want Pareto points), and thus are essentially removed from the design. Therefore, the runs that do not lead to dominated points will not always be the entirety of the design, and orthogonality, rotatability, etc. may be lost. It could be argued that this eliminates the usefulness of many of the designs evaluated in this research. However, because the number of runs in these designs differs, and because it did no harm to evaluate all designs, all designs are included. Furthermore, the coded design matrices are used to form the least squares models, in the event desirable properties such as orthogonality and rotatability remain intact.

It is assumed here for brevity that the reader is somewhat familiar with ordinary least squares (OLS), weighted least squares (WLS), generalized least squares (GLS), the general linear model (GLM), and corresponding techniques such as the Box-Cox method. Therefore not all terms or methods are explicitly defined. If this is not the case, a good review can be found in Montgomery, Peck, and Vining [49].

The typical linear regression model is of the form $y = X\beta + \varepsilon$, where the parameters (β) are linear, *X* is the design matrix, and *y* is the response. Ordinary least squares (OLS) makes the assumption that $\varepsilon \sim NID(0, \sigma^2)$ and the errors are uncorrelated. Using the least-squares normal equations, β can be estimated using $\hat{\beta} = (X^T X)^{-1} X^T y$, where X has a leading column of ones to estimate the intercept term. A variance inflation factor (VIF) is an indicator of multi-collinearity, and if larger than 10 indicates a regressor that is near linearly dependent to, and thus contributes similar information as, another regressor. Therefore $X^T X$ may become near singular, adversely affecting the coefficient estimates, and so a corresponding regressor may need to be removed from the model. The VIFs are calculated according to $VIF_j = (1 - R_j^2)^{-1}$, where R_j^2 is the $R^2 = 1 - SS_{residual} / SS_{Total}$ value (*SS* denoting sum of squares) obtained by regressing the *j*th predictor on the remaining predictors. The Box-Cox method is a procedure that identifies a best power transformation for the response to correct nonconstant variance.

Generalized least squares (GLS) is a more general regression method, of which OLS can be considered a special case. The variance may be nonconstant and the observations may be uncorrelated or correlated. GLS assumes some variance/covariance structure to account for this. Weighted least squares (WLS) is another case of GLS where errors are assumed uncorrelated but the variance is not assumed constant.

For this research, use of the GLM beyond the normal model is likely to be of little benefit. Many GLMs are problem-specific, but all assume the response variable distribution is a member of the exponential family and have a link function that provides the relationship between the linear predictor and the mean of the distribution function. For example, logistic regression requires a binary response variable, Poisson regression is used for count data of a rare event, and gamma regression requires a positive response (with log canonical link). This research requires a generalized method for regression, so GLMs other than the normal (response variable distribution is assumed to be normal and the canonical link is the identity) will typically be inappropriate, not to mention that the noise affects the response such that the data would likely have to be changed, not just transformed by some power method, to meet the needs of these models (*i.e.*, gamma must be non-negative). Additionally, these models can have dispersion problems, which if significant enough, are not as easy to correct as the normal distribution case.

3.5.1. Model Building Approach. Given that no prior knowledge of any model is assumed in this research, a computationally efficient approach for constructing good models is needed. Unfortunately, a "best" model can only be guaranteed if all possible regressions are used. In all possible regressions, all subsets of regressors are used to form models. These models are compared, and the best is chosen. This is clearly inefficient as the number of regressors and responses increases.

Alternatively, stepwise regression methods attempt to intelligently select variables for the model without evaluating all possible regressions. Forward selection inserts regressors one at a time into the model based upon largest correlation to the response, adjusting the correlations for the effect of the previously entered regresssors (partial correlations). Equivalently, the regressor with the largest partial F statistic (t-statistic) can be added. Forward selection is problematic in that, as regressors enter, other regressors previously entered may become insignificant. Stepwise regression attempts to correct the problems of forward selection by dropping regressors at each iteration if their t-statistic is less than some removal criterion.

Backward elimination begins with all candidate regressors in the model. The smallest t-statistic is compared with some preselected value as a removal criterion. Regressors are removed until no regressor's t-statistic exceeds the removal criterion. Backward elimination often serves as a good selection procedure [49].

In 1978, Berk noted that forward selection tends to agree with all possible regressions for small subset sizes, while backward elimination tends to agree for large subset sizes. No method typically works better than another, and so it may be best to fit multiple models. However, for multi-objective problems a model is required for each objective, and so as the number of objectives and samples increases, the time required to formulate the models becomes much larger. For the surrogate, only a good, not necessarily optimal, prediction or set of factors is needed, as the stochastic nature of the problem likely prevents a near-perfect model regardless. The approach taken to screen factors and build a regression surrogate is as shown in Figure 3.5.1.

1. Choose either OLS or WLS.
   Using coded variables, where the variables are on the range [-1,1] (Hi, Lo), with the exception of axial runs:
2. Fit the full model using up to cubic polynomial terms and 2FI terms, checking for available degrees of freedom.
3. Remove Multicollinear terms iteratively (VIF>10), removing the least significant regressor among the idenitifed regressors and those with correlation > 0.5 to the identified regressor.
4. Run Box-Cox.
5. Fit model, check for outliers using studentized residuals *r*. If $|r_i| > 3$, remove point *i* while fitting models.
6. Check significance of regression (using F-test) and regression coefficients (using T-tests). If all coefficients are significant, and regression is significant, return model. Otherwise remove least significant coefficient.
7. If only one regressor is left and the model with one regressor is insignificant, return all main effects.
8. Continue Steps 5-8 until model returned. Return significant terms, main effects (including those from significant interactions), and create natural model.

Figure 3.5.1: Factor Screening/Regression Method

Available degrees of freedom are used to choose the problem-dependent starting model. Box-Cox is run early so as to not transform the response too many times, or too late in the process, as nonconstant variance may affect which regressors appear significant. The model is mainly built using coded variables so as to take advantage of those designs that are orthogonal (assuming all design levels achieve non-dominated responses). This algorithm is by no means perfect, but it is an effective, quick method of reducing the factors that need to be sampled when attempting to fill gaps in the objective space, if it is possible.

R-squared metrics (adjusted and predicted) are all returned upon completion of the model. Unfortunately, DOE and regression is an art, and so it cannot be said that this method or another method is a definitive best approach. GLS is not included because a correlation would have to be assumed, and because DACE uses GLS for its underlying polynomial already.

3.5.2. Multivariate Adaptive Regression Splines. Multivariate Adaptive Regression Splines (MARS) do not assume that there is some singular relationship between regressors and the response. Instead, MARS partitions the input space into regions (which may overlap), each with its own regression equation [66] or in the pure sense, combination of basis functions. In other words, a set of common basis functions with different coefficients and knots where the regression equation changes, is used. MARS is an expansion of recursive partitioning regression, where the design space is partitioned into separate regressions to reduce error. Friedman [33] gives a comprehensive explanation and development of MARS.

MARS is an extremely powerful algorithm; reasonable models can be fit to noise. Unfortunately, it is also computationally inefficient. MARS is not an all-possible regressions approach of basis functions, partitions, and variables, but it can be close (especially dependent upon the code implementation). Computational shortcuts presented by Friedman [33] and smart coding can make the algorithm more efficient. However, as allowable interactions and observations for which a basis function is positive increase, the computation time becomes exceedingly costly. Friedman referenced 20 and 1000, respectively, for maximum allowable interactions and observations, relative to a SUN Microsystems Model 3/260) [33]. Of course, such a machine is extremely out-powered by modern computers, but is evident that these models are not cheap to compute.

The problem with computation relative to this research is that as the number of objectives or variables increase, so too do the interactions and likely the number of data points. Four objectives need 28 2FI for the aspiration and reservation levels, and at three objectives, with a full factorial there will likely be at least 1000 data points to optimally partition. Furthermore, the aspiration and reservation levels may not serve well as predictors, and so, using Dias $\Gamma1$ as an example, there are 435 2FI for the corresponding 30 decision variables without having prior knowledge of the significant interactions.

The final downfall is that there exist multiple responses for which to fit a model, meaning a computationally expensive model has to be formed many times. Thus, the general use (for example, 10 objectives) of MARS is questionable versus other surrogate methods when it comes to computational time.

Therefore, the MARS-inspired algorithm in this research is more of a recursive partitioning approach. The computational time was a concern, because no matter how powerful the algorithm, it could be possible that surrogates simply are not a good method to approximate the Pareto front, or that surrogates continually need to be formed. Therefore, too much time should not be spent on forming the model. Originally, the author intended to write code for a MARS implementation, but came to realize just how inefficient aspects of the algorithm could be given certain instances. The implementation of MARS/Recursive Partitioning developed for this research is shown in Figure 3.5.2.

The sub-models are only valid over the range of the design sites, so a pure distance method cannot be used when predicting. Similarly, the regions do not necessarily overlap as in the true MARS algorithm. In the event a point falls in between regions, the global model is used to predict. The primary drawback of this implementation is that an outlier can cause early termination and the partitioning algorithm may not find the best partition to fit a new model. Also, eventually a small subset of points may be used to fit the sub-regression model, and thus these models may give misleading information despite the better fit. Unfortunately, there is no easy way to choose knots or partitions for the model without either prior knowledge of the surface or doing something similar to the true and expensive MARS approach. As SMOMADS is computationally expensive enough, the use of a full MARS algorithm is not likely to be of value if the goal is to expand to any number of objectives, variables, etc.

1. Check available degrees of freedom and fit largest model, up to cubic terms, over the entire region using WLS, Box Cox, and backward elimination (Multicollinearity correction optional).
2. Use squared error as lack of fit measure (LOF).
3. Set maximum number of subregions to the floor of the number of samples divided by the number of predictor variables.
4. While the number of subregions is less than the maximum, partition each subregion approximately in half, using the point with the worst squared error from the previous model as the "center" of the new region.
5. Fit a new model to the new subregion and check for improvement in squared error. If there is improvement, continue partitioning, otherwise stop partitioning that region.

Figure 3.5.2: MARS/Recursive Partitioning Implementation

The truncated cubic functions from MARS were also not added, although it would be beneficial to have such functions, as they do not directly correspond to the implementation used here. An appropriate modification of those functions could probably be determined, but it is shown in Section 4.11 that the least squares approaches were not of great value in general.

A recursive approach for Kriging, RBFs, etc. could also be attempted. However, as these surrogates are designed to globally interpolate, it becomes much less straightforward.

3.6. *Using Single-Objective Formulations (BiMADS)*

Audet, Savard, and Zgahal [13] recently devised the BiMADS method for solving bi-objective problems using MADS. This method uses the ordering property of 2-dimensional space in conjunction with reference-point based single objective formulations to approximate the Pareto front, such that there are no gaps within some tolerance. Additionally, this method avoids limitations presented by other methods (excluding SMOMADS). The ordering property is the property that sorting data in two dimensions, specifically the Pareto front points, will result in properly placing points in the 2-dimensional space. Therefore, determining which points are neighbors in space is straightforward and the size of a gap in the Pareto front can be easily interpreted by Euclidean distance. However, this property does not generalize to more than two dimensions, which means that solving problems with more than two objectives is problematic.

BiMADS relies upon a series of single objective optimizations to solve for the Pareto front. These single-objective formulations rely on a reference point $r \in \mathbb{R}^p$ in the objective space (of dimension p) and are of two forms. The first form, the normalized formulation, is defined as

$$\hat{R}_r : \min_{x \in X} \hat{\psi}_r(x) = \hat{\phi}_r(f_1(x), f_2(x), \ldots, f_p(x)) = \max_{i \in \{1,2,\ldots,p\}} \frac{f_i(x) - r_i}{s_i}, \qquad (3.40)$$

where $s \in \mathbb{R}^n$. Figure 3.6.1 depicts the level sets of this formulation and the product formulation, and shows intuitively why these formulations and an appropriate reference point work to fill gaps in the Pareto space.

The second single objective formulation, the product formulation, is defined as

$$\tilde{R}_r : \min_{x \in X} \tilde{\psi}_r(x) = \tilde{\phi}_r(f_1(x), f_2(x), \ldots, f_p(x)) = -\prod_{i=1}^{p} ((r_i - f_i(x))_+)^2 , \qquad (3.41)$$

where $(r_i - f_i(x))_+ = \max\{r_i - f_i(x), 0\}$ for $i = 1, 2, \ldots, p$. This formulation is advantageous over the previous in that it preserves the differentiability of the orginal problem.

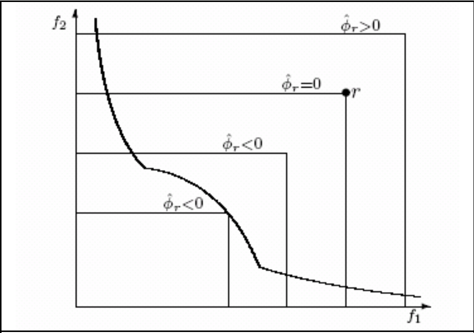
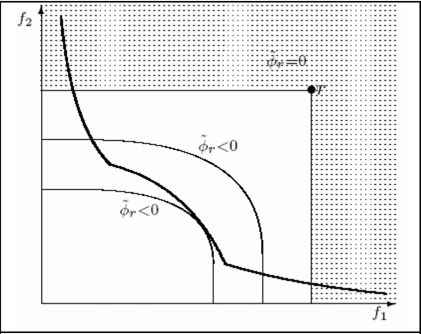
 f_2 $\hat{\phi}_r>0$ $\hat{\phi}_r=0$ r $\hat{\phi}_r<0$ $\hat{\phi}_r<0$ f_1	 f_2 $\tilde{\psi}_r=0$ r $\tilde{\phi}_r<0$ $\tilde{\phi}_r<0$ f_1
(a) Normalized Formulation	(b) Product Formulation

Figure 3.6.1: Level Sets [13]

Audet, Savard, and Zgahal [13] proved that the optimal solutions to these formulations for p objectives ($p \geq 2$) are Pareto optimal, and that the formulations preserve local Lipschitz continuity and a condition involving Clarke descent directions for all objectives and ψ_r. The actual BiMADS algorithm is shown in Figure 3.6.2. The initial points for this algorithm can be chosen by any means, but are recommended to be those found when solving for the utopia point to ensure the true spread of solutions.

BiMADS begins with some initial set of points and rapidly works towards the Pareto front. At each iteration, the algorithm searches for three points such that the distances between the three are maximal, while using the weighting so that a valid gap is not identified continually (if a discontinuous front). The starting iterate is then changed to match the solution to the middle point, and the reference point is built using the two endpoints. Solving the single-objective formulation generates Pareto points around the middle point and fills the two gaps, or works further towards the true Pareto front.

BiMADS is very fast and works extremely well in two objectives. With three or more objectives, the ordering property ceases to exist. The gap algorithm presented in Section 3.7 is used in this research to eliminate the need for the ordering property. Using the gap algorithm, or some visualization technique, a slightly different approach can be taken than that of BiMADS.

INITIALIZATION:

- Apply the MADS algorithm from x_0 to solve $\min_{x\in X} f_1(x)$ and $\min_{x\in X} f_2(x)$.
- Let $X_L = \{x^1, x^2, ..., x^J\}$ be an ordered list of pairwsie nondominated points such that $f_1(x^1) < f_1(x^2) < ...f_1(x^J)$ and $f_2(x^1) > f_2(x^2) > ...f_2(x^J)$. Initialize the weight $w(x) = 0$ for all $x \in X$ and let $\delta > 0$.

MAIN ITERATIONS: Repeat

- REFERENCE POINT DETERMINATION:
  - If $J > 2$, let $\hat{j} \in \arg\max_{j=2,\dots,J-1} = \dfrac{\left\|F(x^j) - F(x^{j-1})\right\|^2 + \left\|F(x^j) - F(x^{j+1})\right\|^2}{w\left(x^j\right)+1}$,

    and define the reference point $r = (f_1(x^{\hat{j}+1}), f_2(x^{\hat{j}-1}))$.
  - If $J = 2$, let $x^{\hat{j}} = x^2$, define the reference point $r = (f_1(x^2), f_2(x^1))$ and set $\delta^{\hat{j}} = \dfrac{\left\|F(x^2) - F(x^1)\right\|^2}{w\left(x^2\right)+1}$.
  - If $J = 1$, let $x^{\hat{j}} = x^1$, $\delta^{\hat{j}} = \dfrac{\delta}{w(x^{\hat{j}})+1}$ and apply the MADS algorithm from $x^{\hat{j}}$ to solve $\min_{x\in X} f_1(x)$ and $\min_{x\in X} f_2(x)$. Terminate MADS when the mesh size parameter Δ^m drops below $\Delta(\delta^{\hat{j}}) = O(\delta^{\hat{j}})$ and continue to the step UPDATE X_L.
- SINGLE-OBJECTIVE FORMULATION MINIMIZATION: Solve a single-objective formulation R_r using the MADS algorithm fom starting point $x^{\hat{j}}$. Terminate MADS when the mesh size parameter Δ^m drops below $\Delta(\delta^{\hat{j}}) = O(\delta^{\hat{j}})$ or if a maximal number of objective evaluations is attained.
- UPDATE X_L:

Add to X_L all nondominated points found in the current iteration, remove dominated points from X_L, and order the resulting list of points. Increase weights: $w(x^{\hat{j}}) \leftarrow w(x^{\hat{j}})+1$ for each $x \in X_L$.

Figure 3.6.2: BiMADS [13]

Instead of identifying three points, two bounds or endpoints of a single gap are identified and used to create the reference point. This single gap is relative to one or more objectives and their indifference values. These boundary solutions can then be used as starting iterates for the corresponding single-objective formulation and one or both starting iterates will likely work to fill the identified gap along the objective(s). This works in part because BiMADS makes no assumption about where the middle point is in reference to the other two. The use of this approach from here on will be termed nMADS.

Audet, Savard, and Zgahal also recommend two metrics for the uniformity distribution of Pareto solutions based upon the squared distance between nondominated points. These metrics do not add much information to the entropy, cluster, and number of distinct points metrics already introduced in Section 3.2, and were more important within the context of BiMADS.

3.7. Identifying Gaps in the Pareto Front

After the initial estimation of the Pareto front, it is critical to find any gaps that may exist, specifically with regard to a set of indifference values. Recall from Section 3.2 that indifference values form a grid of indifference regions over the objective space such that a decision-maker is indifferent between any two solutions within a single indifference region. These indifference values are used in nMADS to generate a required fidelity for the Pareto front. A *gap*, for the purposes of this research, consists of two endpoints that do not satisfy indifference values with respect to at least one objective, such that there are no other points between those endpoints on the current Pareto approximation in the unsatisfied objectives.

3.7.1. Limited Methods. First, as the entropy and distinct point metrics already build a grid of points, that grid (either projected or non-projected) could be checked quickly for which grid hypercubes lack points. This is problematic for several reasons. For one, the projected grid is less desirable, in that points along hyper-diagonals are projected to the same location in the projected space. Therefore, an empty projected hypercube has no direct meaning. Second, only those hypercubes on the Pareto front are of interest, and in some cases, distinguishing these from the other hypercubes could be difficult. Finally, if the points are projected, dimensionality is lost.

For instance, consider Figure 3.7.1 in which a front in two objectives is shown with indifference values ω_i for $i = 1, 2$, where there exist two points with no gap in the first objective, but a gap in the second. When the front is projected, this curve becomes a line, and the gap in the second objective may be lost.

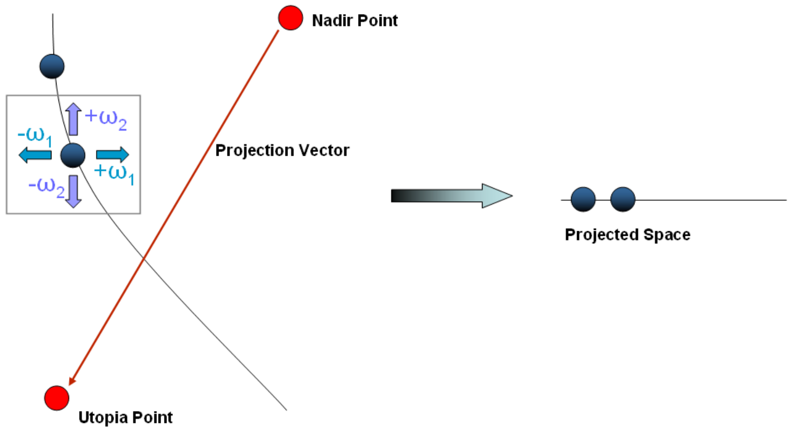


Figure 3.7.1: Projection

Further considering a hypercube grid based on the indifference values, hypercubes can be removed based on dominance and inferiority to the current Pareto approximation. This too has its problems. Two examples are shown in Figure 3.7.2 with the Pareto points in green and hypercube centers in blue. Figure 3.7.2(a) shows a front that is very narrow in all three objectives, while Figure 3.7.2(b) shows a front that is part of a sphere.

It is clear that there are many grid hypercubes that are neither dominated by, nor are inferior to, the current Pareto approximation, but the majority of them are not a part of the true Pareto approximation. It should be clear that further intensifying the criteria may validly elimate points for one front, but not another. Adding some distance criteria to ensure the grid hypercubes are near the current approximation is also problematic, in that a large gap may exist and will not be found.

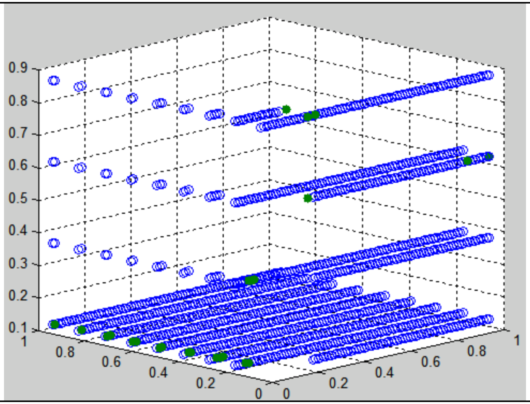
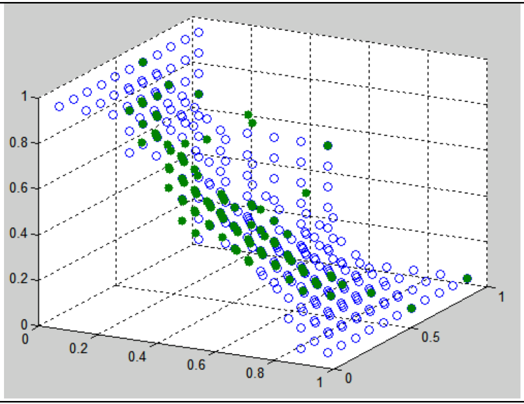
	
(a) Viennet3	(b) Tamaki

Figure 3.7.2: Removing Grid Hypercubes Based on Dominance

Therefore, using a pre-formed grid presents more serious disadvantages than advantages. This inferiority and dominance approach could also be implemented using randomly generated points within the bounds of the Pareto approximation. However, some of these points will lie off of the front, and it is uncertain whether or not randomly generated points will fall within true gaps. The method developed for this research deals solely with those points found by SMOMADS/nMADS in the objective space, and not a grid.

3.7.2. The Gap Algorithm. To identify gaps in the *m*-dimensional Pareto front, care was taken to make the algorithm as computationally efficient as possible. The efficiency is restricted by the fact that the points lie in *m*-dimensional space. The general notion behind the algorithm is to use indifference values to identify missing portions of the Pareto approximation and to determine when a point has other points surrounding it. Given a vector of indifference values, $\bar{\omega}$, each point should have another point within ω_i and $-\omega_i$ (above and below) in each objective *i*. The extreme points in each objective are a special case, requiring only a point above for a minimum, or below for a maximum (since the extreme points constitute the current bounds of the Pareto approximation). Euclidean distance is used to determine when points are near each other in the Pareto space.

The current Pareto approximation objective function values are sorted one objective at a time. The points corresponding to the maximum and minimum in each objective are identified as extreme points. Searching through the approximate Pareto solutions with respect to a particular objective, differences in objective function value are compared to the respective indifference value. This constitutes searching the objective space one-dimensionally.

For a given objective *i*, the search is first conducted in ascending fashion, and then later descending fashion, starting from each data point, proceeding through respective data points to look for gaps "above" and "below." This is to ensure each point is "surrounded", by having a point of greater and lesser objective function value within ω_i for each objective *i*. However, because of the one-dimensional sort, Euclidean distance must be used to determine if a point that is within ω_i in function value in objective *i*, is truly in the same part of the Pareto front as the starting data point. If a successive point in the search (with respect to the point the search is started from) is within the distance criteria, $d_{crit} = c \cdot \|\bar{\omega}\|$ (*c* recommended to be 0.5), and is within the particular indifference value ω_i, there is no gap.

If the difference in objective function value between points is larger than ω_i, a distance vector is checked and the closest point above or below (depending on the search being conducted) the current point is found. If the difference in objective function value for that point and the current also is larger than ω_i, then the gap is considered valid. Otherwise, it is ignored and will be found later with respect to another objective, since the distance criteria was not met. That is, if those points do represent a gap, their objective function values cannot be within every indifference value and so it will be found with respect to another objective. Due to the nature of sorting, these precautions are necessary, as illustrated by the parabola missing a piece of its curve, shown in Figure 3.7.3. It would be easy to accidentally identify no gaps by using sorting.

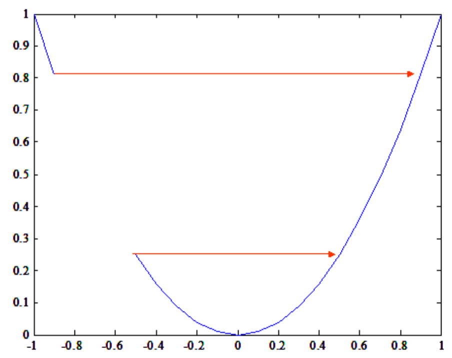


Figure 3.7.3: Parabola

Using the closest point "above" or "below" does not necessarily correspond to filling in empty space the fastest. Consider the portion of a Pareto front shown in Figure 3.7.4, where the grey box represents the indifference region and the red circle represents a set distance criteria. The current point, in green, has a point above and below in the first objective and only below in the second. However, in looking above on the y-axis (searching Objective 2), Point 1 is outside the distance criteria but is within ω_2. Therefore the algorithm moves on to Point 2. This point satisfies the distance criteria but does not satisfy ω_2. The algorithm would stop at this point and identify a gap using the current point and Point 2 because any future point will also be outside of ω_2, and Point 2 is closest.

In this two-dimensional view, it would appear that using Point 3 would be better than Point 2, and it in fact could be. However, Point 3 could be in an entirely different part of the Pareto front once a third objective is considered. This is the purpose of using the closest point above or below, so that the center point of any gap identified is as near as possible, or on, the true Pareto front. Adding other criteria to try and determine the "best" endpoint adds computational effort and may mistakenly move to other portions of the Pareto front (a large Euclidean distance could signify the best endpoint or another point that is in a very different part of the front).

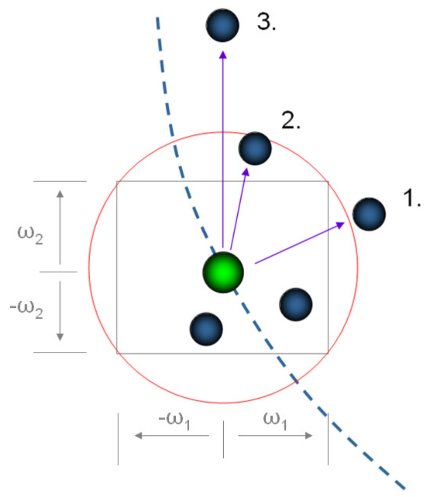


Figure 3.7.4: Searching Around a Point

There of course is the possibility the same gap is identified multiple times or similar gaps are identified. Using the Euclidean distance between center points of gaps comparative to d_{crit}, only distinct center points can be retained. If a gap is filled only with respect to one problem objective, that gap will be identifiable again in the other problem objectives if those are not simultaneously filled (if two endpoints constituted a gap for more than one objective, it is possible all of those objectives will be satisfied after one attempt to fill the gap; any added point adds a new value for all objectives). Gaps should then be sorted according to Euclidean distance between the endpoints, as filling larger gaps first is preferable.

In practice, the algorithm was relatively efficient even with as many as 3500 points, 8 objectives, and 10 indifference regions in each objective (~10 seconds on a 2.1 GHz, 1GB RAM machine). The algorithm is shown in Figure 3.7.5.

```
1. Pick some $c > 0$ and set $d_{crit} = c \cdot \|\bar{\omega}\|$, where $\bar{\omega}$ is a vector of indifference values.
2. For each objective $m$, sort the Pareto objective data ($n$ solutions) in ascending
   order of function value. Set $j = 1$.
    a. For each data point $j$, relative to the sorted data, search below:
        i. Let $i = 1$.
        ii. If $j = 1$ or $j = n$, that data point is an extreme point. Set $j = j + 1$
            or stop respectively.
        iii. If $|f_j^m - f_{j-i}^m| \le \omega_m$ and $\|f_j^m - f_{j-i}^m\| \le d_{crit}$, set $j = j + 1$.
        iv. Else, if $|f_j^m - f_{j-i}^m| > \omega_m$, find the closest point $k$ to $j$, from point 1
            to $j - 1$ using Euclidean distance.
            1. If $|f_j^m - f_k^m| \le \omega_m$, set $j = j + 1$ (will add in another
               objective; did not satisfy the distance previously).
            2. Else, add $(j, k)$ as a gap. Set $j = j + 1$.
        v. Else, $i = i + 1$.
    b. Search above using same process as (a), except using $j + i$ instead of $j - i$
       in (iii) and (iv), and also using points $j + 1$ to $n$ in (iv).
3. Remove gaps with a distance between their centers less than $d_{crit}$ (retaining one).
```

Figure 3.7.5: Gap Algorithm

3.7.3. Limitations of the Algorithm. There is an unavoidable drawback to this method when using more than two objectives. For example, Figure 3.7.6 depicts a Tamaki problem Pareto approximation in only two objectives, with Pareto points in blue, and identified gaps in green. Because the algorithm looks above and below each point, but using a distance criteria (in this case, 0.5 of the norm of the indifference regions: [0.1, 0.1, 0.1]), as long as a point has some other point within its vicinity with an acceptable higher or lower objective function value (although that point may be on a diagonal), no gap is found. In reality, the red circle is a gap, but since all of its surrounding points meet the criteria, no gap is stored. Hopefully, as the algorithm progresses, either new points will fill that gap, or the algorithm will be able to identify it (due to noise or the directions in GPS/MADS). Of course, if a user can visually identify the gap, dependence on this algorithm is not required. In practice, if a gap was not identifiable in a given iteration of SMOMADS or nMADS, it was identified in later iterations, due to new approximate Pareto points being added.

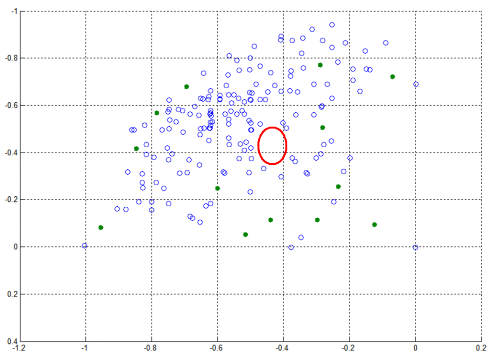


Figure 3.7.6: Identified Gaps

This drawback is further exemplified in Figure 3.7.7. In searching above and below, there is some chance that for any two objectives, two points may account for both the above and below points in both objectives (versus four points). This is shown in Figure 3.7.7(a), with a potential unidentified gap represented by the purple arrows and the indifference region shown as the grey box. The points each have another point within the indifference value above and below in each objective, and a string of such points can result in a circular gap, as in Figure 3.7.7.

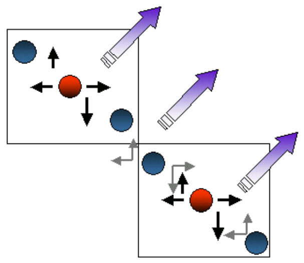
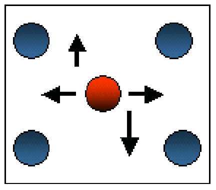
	
(a) Searching Above/Below	(b) Searching Every Diagonal

Figure 3.7.7: Searches

The correction for this would be partly combinatorial. Each point would need other points above in one objective and above in another, below in one objective and above in another, etc. Essentially points would be required in the diagonal regions illustrated in Figure 3.7.7(b). This, in fact, has more problems associated with it. First, this too allows gaps, as points may fall ever so slightly within these sub-hypercubes associated with the diagonals, which would still allow empty regions (rectangular gaps) to occur. Furthermore, consider Figure 3.7.8. It should be clear that many of the points on this front would be identified incorrectly as gaps because they have no points in a certain diagonal direction (the red arrow). Therefore, the algorithm given in Figure 3.7.5 is better, because it allows for any shape of curvature in the Pareto front.

A required tolerance could be used to ensure objective function values change by at least some amount, so that the unidentifiable circular or rectangular gaps do not occur. However, due to noise, this tolerance could be misleading, and choosing a value for the tolerance may not be straightforward. Furthermore, the increasing number of criteria to be met will make the algorithm less efficient as the number of objectives increase, without providing in practice a significant advantage over the algorithm given in Figure 3.7.5.

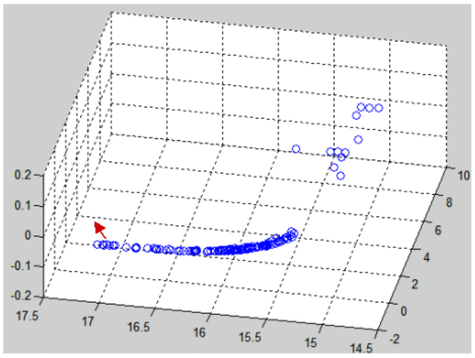
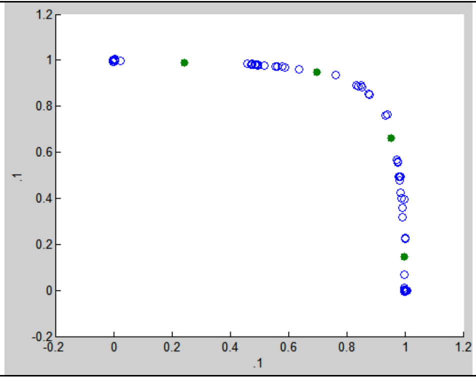
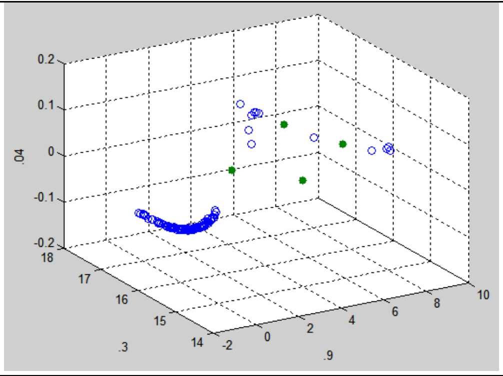


Figure 3.7.8: Viennet3

As was mentioned previously, the algorithm performed extaremely well in practice. Figure 3.7.9 shows both a two-objective and three-objective example, with Pareto points in blue, identified gaps in green, and indifference regions on the axes. In the three-objective problem all of the gaps (centers) were correctly identified, but one additional gap was falsely identified. However, identifying incorrect gaps is not a problem because it will not occur often, and Pareto points are still identified. It is more important that the true gaps were all identified.



(a) Fonseca F1



(b) Viennet3

Figure 3.7.9: Gap Examples

Intensifying the distance criterion identifies more gaps. In practice, a value of $c = 0.5$ seemed to work best. When looking at these plots, the reader should keep in mind that gaps can only be found inside the bounds of the Pareto points found thus far. The algorithm is limited in that it follows the current approximation and is not robust enough to interpret the surface the data represents.

3.8. Visualization of N-Dimensions

As the number of objectives increases beyond three, one can no longer visualize the Pareto front. Therefore, a decision-maker becomes entirely reliant upon metrics, indifference regions, and the gap algorithm. Fortunately, there has been some work done in this area so as to be able to visualize any number of objectives. This enables the method to catch any gaps that are not caught by the gap algorithm, or to determine that the current approximation is sufficient. There are, in fact, a variety of methods for visualizing *n*-dimensions, to include the obvious two- or three-objectives at-a-time approach, graph morphing, and physical programming visualization. The limitation in these methods is that the information can become overwhelming and difficult to piece together in one representation.

3.8.1. HSDC. Agrawal, Lewis, and Bloebaum first developed a method, called Hyper-Space Diagonal Counting (HSDC) and then a visualization, Hyperspace Pareto Frontier (HPF), so as to be able to visualize the entire Pareto space in two dimensions intuitively (*i.e.*, easily interpreted) [7].

HSDC is based on the premise of Cantor's counting method from complexity theory. Cantor's counting method is used to prove that the set of rational numbers is countable, by establishing a one-to-one correspondence between the rationals and the set of natural numbers. HSDC maps points to a line by counting along hyperdiagonals that move away from the origin. Figure 3.8.1 shows example hyperdiagonals for two objectives and three objectives, where in the two objective case counting is performed along the red diagonals, starting at bin (1,1).

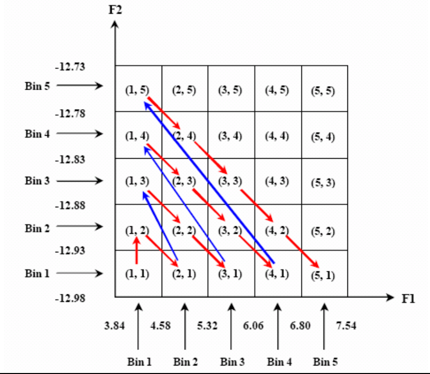
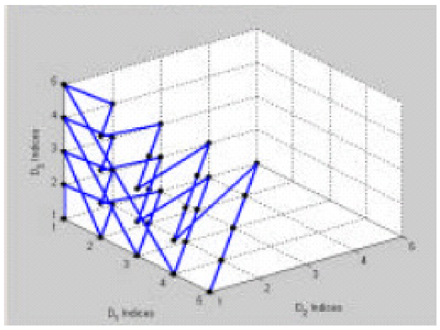
 F2 F1 -12.73 Bin 5 -12.78 Bin 4 -12.83 Bin 3 -12.88 Bin 2 -12.93 Bin 1 -12.98 (1, 5) (2, 5) (3, 5) (4, 5) (5, 5) (1, 4) (2, 4) (3, 4) (4, 4) (5, 4) (1, 3) (2, 3) (3, 3) (4, 3) (5, 3) (1, 2) (2, 2) (3, 2) (4, 2) (5, 2) (1, 1) (2, 1) (3, 1) (4, 1) (5, 1) 3.84 4.58 5.32 6.06 6.80 7.54 Bin 1 Bin 2 Bin 3 Bin 4 Bin 5	
(a) 2 Objectives	(b) 3 Objectives

Figure 3.8.1: Hyperdiagonals

First, note that the number of points on a level, or hyperdiagonal, is given by,

$$E_l^n = \frac{\prod_{k=0}^{k=n-2}(l+k)}{(n-1)!}, \tag{3.42}$$

where $n \in \{2,3,...\}$ is the number of objectives, and *l* is the level. The total number of elements up to a particular level or hyperdiagonal is given by

$$TE_l^n = \sum_1^l E_l^n \tag{3.43}$$

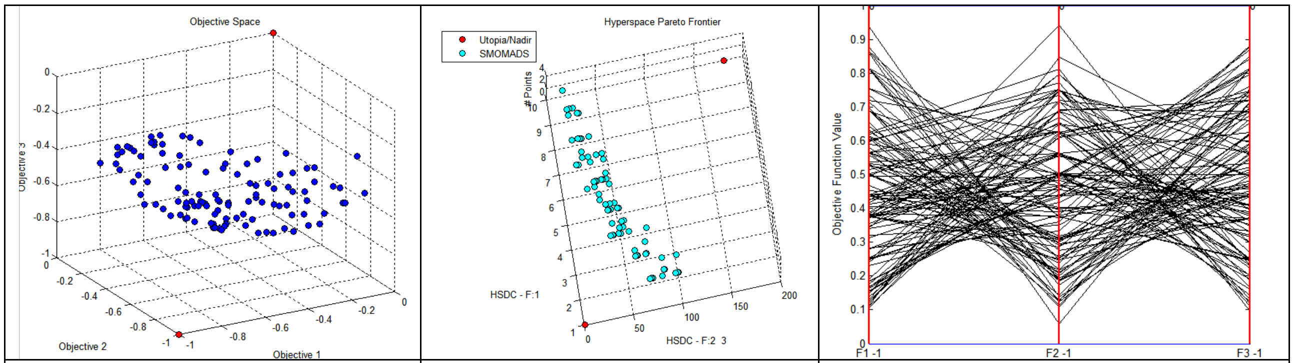
and the sum of the indices at a particular level is given by $S_l = n + l - 1$. Note that the size of the hyperdiagonals continually increases. For example, in Figure 3.8.1(a) the count at bin (2,5) is 17, at bin (3,5) is 24, at bin (4,5) is 32, and at bin (5,5) is 41.

Generating the HPF is done first by putting the objective function values into bins, with the objective functions grouped into two sets, counting each set using HSDC. These counts provide the linear indices for each point on the two-dimensional graph. The two-dimensional graph can then be interpreted as moving away from the origin, along the hyperdiagonals in the respective objectives, where a count of the number of points in a specific bin can also be added. Depending on the number of bins, the number of levels necessary for counting becomes $l = nb - n + 1$. The number of bins must be consistent in all objectives; otherwise, the counting becomes biased. In this research, the indifference values are used to find a common bin size, using the smallest number of resulting bins from all objectives (where the indifference values are used to bin each objective), for speed purposes.

The objective function values can also be grouped intelligently. With positive correlation of objective function values, the counts will be distributed across many levels. With negative correlation points will group on levels, and with zero correlation, pockets of bins develop. Therefore, grouping objectives based on the most positive correlation yields the most Pareto-like view.

For the implementation in this research, objectives are grouped in near-equal sized sets. Objectives are grouped according to correlation, with larger positive correlation meaning objectives are grouped together, and larger negative correlation meaning objectives are grouped separately. Specifically, the two objectives with largest positive correlation are grouped first, and then objectives are added to that group based on the maximum cumulative correlation (sum of the correlations) with the objectives already in the group until the maximum group size is reached. The event may occur where a particular objective has large positive and large negative correlations with other objectives, in which case its selection is not necessarily appropriate. However, any alternative automated method also has its drawbacks. An example of a three-dimensional Pareto front using HSDC is shown in Figure 3.8.2(b).

HSDC has the limitation that some neighborhoods are lost when forming the visualization. Furthermore, different objective grouping schemes cause different HPF visualizations.



(a) Pareto Front	(b) HSDC View	(c) PC

Figure 3.8.2: Example Views

3.*8.2. Parallel Coordinates*. The method of parallel coordinates plots each objective function on a tick of the x-axis, and connects the objective functions with lines [7,20,53]. For this research, the objective functions are normalized so that the y-axis of one objective function does not prevent data from another from being seen. The major drawback of using parallel coordinates is that as the number of solutions grows, the visualization can become too dense, and thus extremely difficult to interpret. Such an example is shown in Figure 3.8.2(c).

3.8.3. HRV. Chiu and Bloebaum developed Hyper-Radial Visualization (HRV) as a visualization that did not suffer any of the problems of other *n*-dimensional visualizations [20]. The specific goal of this visualization is to view the *n*-dimensional space in a straightforward manner, such that "good" regions of the performance space may be identified.

HRV uses the normalized objective function values, $\tilde{F}_i \in [0,1]$, for each objective $i=1,\ldots,n$. The Hyper-Radial Calculation (HRC) value is computed as:

$$HRC = \sqrt{\frac{\sum_{i=1}^{n} \tilde{F}_i^2}{n}}. \tag{3.44}$$

Because the objectives are normalized, $HRC \in [0,1]$. Part of the intent behind normalizing the objective function values is that the utopia point, or best estimate thereof, becomes the zero vector.

The objectives are split into two groups S_1 and S_2 such that $S_1 \cap S_2 = \{1,\ldots,n\}$. This gives one HRC value for each group, HRC1 and HRC2. The Hyper-Radial Value (HRV) is then $HRV = (HRC1)^2 + (HRC2)^2$. The HRV is truly the squared radius of the Pareto point from the utopia or minimum reference point. This value can be compared to indifference curves (developed from the indifference values, shown in Figure 3.8.3) to determine the quality of a point, with closer to the utopia point being better. This method is referred to as the Direct Sorting Method (DSM).

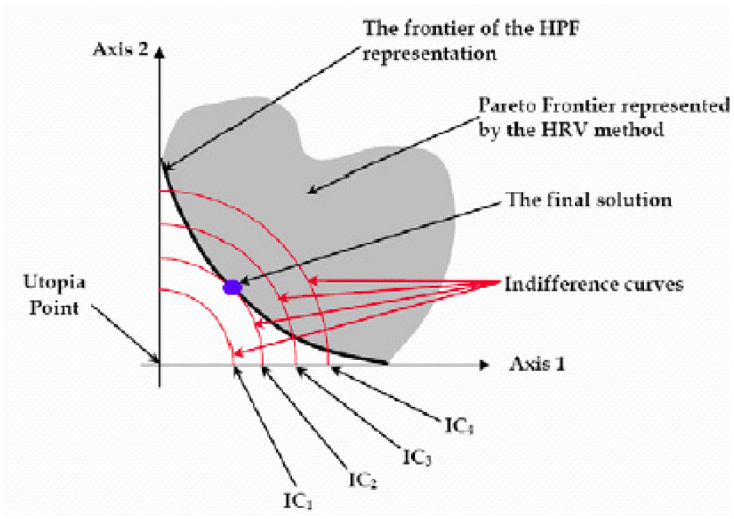

The frontier of the HPF representation
Axis 2
Pareto Frontier represented by the HRV method
The final solution
Utopia Point
Indifference curves
Axis 1
IC₄
IC₃
IC₂
IC₁

Figure 3.8.3: Indifference Curves [20]

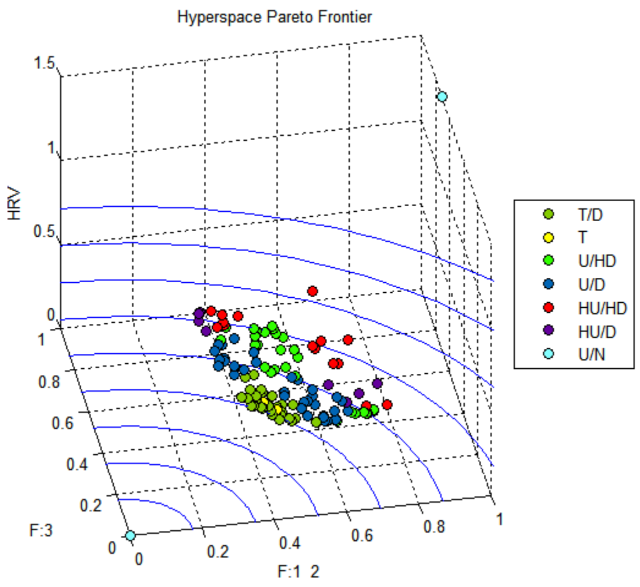


Figure 3.8.4: HRV Example

To maintain an unbiased representation, the two groups of objective functions must be equal in size, $|S_1| = |S_2|$. In the event of an odd number of objectives, a dummy objective is added, with a value of zero for all points. This maintains the unbiased representation, although it modifies the axes values. With the unbiased representation, the grouping of objectives becomes unimportant in relation to the indifference curves.

Pareto points may then be classified by preference. Chui and Bloebaum implement a hybrid preference structure that combines elitist and inclusive structures. This is shown in Table 3.8.1. Each coding number corresponds to a specific color. An example of the HRV representation is shown in Figure 3.8.4.

Table 3.8.1: Color-Coding for Hybrid Preference Structure [20]

Color-Coding	Preference Criteria
11	Pareto points with all Highly Desirable (HD: 0-20% from Lowest Value)
21	Pareto points with all Desirable (D: 20-40%) and at least one HD
22	Pareto points with all D
31	Pareto points with Tolerable (T: 40-60%) and better and at least one HD
32	Pareto points with only T and D
33	Pareto points with all T
41	Pareto points with Undesirable (U: 60-80%) and better and at least one HD
42	Pareto points with U and better (no HD) and at least one D
43	Pareto points with only U and T
44	Pareto points with all U
51	Pareto points with HU (Highly Undesirable: 80-100%) and better and at least one HD
52	Pareto points with HU and better (no HD) and at least one D
53	Pareto points with HU and better (no D or HD) and at least one T
54	Pareto points with only U and HU
55	Pareto points with all HU

3.8.4. Using Visualizations Computationally to Find Gaps. A short discussion on using these visualizations computationally to find gaps is warranted because it would be best to require no user interaction. Unfortunately, none of these visualizations can be used in the computational context to find gaps. Parallel coordinates have no new information, and so the gap algorithm would still be required. HSDC uses binning to represent the data, counting along hyperdiagonals that get further away from the origin. By binning, some local information is lost, and furthermore some bins correspond to non-existent areas in space because the hyperdiagonals continually get longer. Therefore, a gap that is found may not truly be a region in space, and gaps may exist that will not be found due to the effect of binning. HRV uses hyper-radials, and thus can map points from different regions to the same location. Therefore, any gap found in the HRV space will not have a singular meaning in three or more objectives. Instead, only radii can be identified that have no points, but this is easily done visually.

3.9. Final Dominance Check

Walston checked points for dominance as they were added to the approximate Pareto set, against those previously added. However, she suggested adding a final check, since points added after other points may, in fact, dominate. In some of the Chapter IV results, some dominated points were, in fact, retained. A check was eventually added in this research, but not before many plots were already done; thus many of them could contain a few dominated points.

Because all points have already been checked against those points prior to them, they simply need to be checked against those points following them. The combination of these two checks also saves some time versus a single final check, as points may be removed earlier in the process. An alogrithm such as BiMADS requires only Pareto points at any iteration, and so this savings in time is valuable.

In the stochastic case, there is a possibility that the maximum amount of noise is subtracted from each objective. The probability of this is very small, but it could occur. As shown in Figure 3.9.1, if this were to occur, many "valid" Pareto points already found would be dominated. Depending upon the shape of the front, and as the number of objectives or noise increases, this can become increasingly troublesome. As every possible solution, and not just the final, is checked for dominance using the BiMADS approach, there are opportunities in a localized area for this problem to occur, and "valid" Pareto solutions already found could be removed. This could also add a great deal of computational time, since the random number generation now becomes critical in achieving maximum noise (it is harder to get to the minimum curve consistently). Furthermore, this forces the approximation below the true Pareto front, which would cause the efficiency of BiMADS/nMADS to be lost.

SMOMADS and nMADS both concentrate on regions, and by using a mean response from R&S, the effect of noise is reduced. Therefore this case of domination becomes much less probable. Walston discussed a method that could be used within the R&S framework from the Multi-Objective Computing Budget Allocation algorithm (MOCBA) to help prevent this dominance problem, by using probabilities that a point is dominated. Such probabilities may be difficult to formulate and imply some sort of tolerance from a threshold, which will be discussed shortly. Fortunately, in practice, this dominance event did not seem to occur. Of course, that may change at large noise levels.

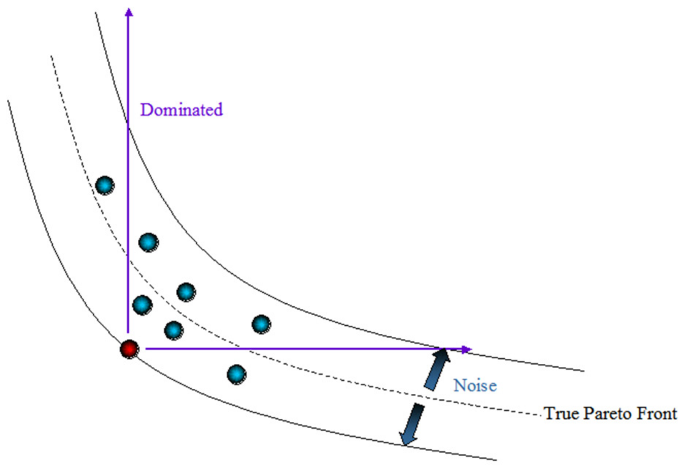


Figure 3.9.1: Noise Limitation in 2 Objectives

Any use of a tolerance could be difficult, as an estimate of noise would be required, and there would have to exist some notion of a cut-off. Interestingly, in the general case, no confident estimate of noise can ever be achieved in the black-box context because a deviation in noise may constitute a much larger or much smaller deviation in objective function value, such as in the case of a piece-wise or sensitive objective function. Tolerance could also be generated from indifference values, or in the case of a surrogate solution, an estimate of error. However, this too has its difficulties as error may vary by region and indifference values can be subjective. Additionally, in allowing for some tolerance, non-Pareto solutions may be accepted.

The concepts and proposals from this chapter are tested and analyzed in Chapter IV. Additionally, where necessary, the concepts are put together to form two new general algorithms. These algorithms are also tested and analyzed in Chapter IV.

IV. Results and Analysis

The analysis and results of the methodologies presented in Chapter III follow. Further development of some of those concepts is also developed as part of the analysis. A general approach is presented for each section, followed by the analysis.

4.1. Testing Approach

The initial SMOMADS algorithm used for this research was acquired directly from Walston [70]. The various runs were conducted on four computers, ranging from 2.19 to 3 GHz and 500 to 3GB RAM. Three were Pentium machines on the AFIT network, and thus performed slower than would be normally expected, courtesy of network patches, etc. None of the runs were conducted on high-performance machines for one of three reasons: 1) the code was experimental and thus had to be continually tweaked and modified whenever errors arose; 2) too much data had to be saved to too many locations to easily use Open Office on Linux; and 3) achieving a level of code and algorithm quality that could be used on a desktop machine was more desirable. The machines used for each set of runs are identified when time is presented as a metric. Each test problem during a set of runs (section of this thesis) was conducted using a single machine for consistency.

An identical suite of test problems was generally used in this research to compare with the results of Walston [70]. Specific test problems and the specific implementations tested by Walston [70] are listed in Table 4.1.1, where FF denotes full factorial, CCD denotes Central Composite, and BB denotes Box-Behnken.

Data analysis was conducted using appropriate statistical and design of experiment techniques, when appropriate. Results are often shown for only a representative subset of the complete test set, although analysis was done for every problem (to keep thesis length reasonable). The specific problem formulations used by

Walston [70] follow in Section 4.2. It is important to note that the approximations found by Walston [70] were dependent upon ranges determined from published Pareto fronts and their observed utopia and nadir points, where often the published front came from genetic algorithms [69].

Table 4.1.1: Problem Set (Walston)

Test Problem	# Vars	# Objs	Var Type	# Test Points	Experimental Design	Solver
Viennet4	2	3	Continuous	4209	FF,CCD, BB	MVPS-RS
Viennet3	2	3	Continuous	4096	FF	MVPS-RS
Poloni	2	2	Continuous	10272	FF	MVPS-RS
Tamaki	3	3	Continuous	145	FF,CCD	MVMADS-RS
Dias Γ1	30	2	Continuous	697	FF,CCD	MVPS-RS
Dias Γ2	30	2	Continuous	625	FF	MVPS-RS
Fonseca F1	2	2	Continuous	10036	FF,CCD	MVPS-RS
Schaffer F3	1	2	Continuous	11250	FF,CCD	MVPS-RS
Srinivas	2	2	Continuous	697	FF,CCD	MVMADS-RS
DTLZ7	2	2	Continuous	36	FF,CCD	MVPS-RS
Disk Brake	4	2	Mixed	108	CCD	MVMADS-RS

4.2. Test Problems

In general, uniformly distributed random noise was added to and subtracted from each objective function, so that the expected value of the noise was zero. In Walston's work [70], noise was simply added to the objectives, in essence raising the objective function values such that, with large amounts of noise, the optimization would become much easier for MADS/GPS (it is easier to find –5, than –10 for a minimization). Walston [70] added 1% of the maximum objective function value (nadir point component) to each objective, with the exception of Viennet4, where the noise was not scaled.

These test problems encompass a good variety with respect to the number of decision variables, types of constraints, and types of objectives, convexity and non-convexity, and discontinuity. All are re-formulated as minimization problems, as the code accompanying this research requires. However, in the experimental design results section, some of the problems may be shown as maximizations (the results multiplied by –1), while in later sections, they will be shown as minimizations. This was just a result of those batch files being based on Walston's original files [70] used to generate the plots.

The problem formulations follow, to include the starting iterates used. Walston's results [70] for these problems are not shown until Section 4.17.

4.2.1. Viennet4.

$$\min\ F_1(x_1,x_2)=\frac{(x_1-2)^2}{2}+\frac{(x_2+1)^2}{13}+3$$
$$F_2(x_1,x_2)=\frac{(x_1+x_2-3)^2}{175}+\frac{(2x_2-x_1)^2}{17}-13$$
$$F_3(x_1,x_2)=\frac{(3x_1-2x_2+4)^2}{8}+\frac{(x_1-x_2+1)^2}{27}+15$$

subject to

$$4x_1+x_2-4\le 0$$
$$-x_1-1\le 0$$
$$x_1-x_2-2\le 0$$
$$x_1,x_2\in[-4,4]^2$$

Walston tested this problem using a CCD, Box-Behnken, and full factorial design using 3 levels, with 5 replications for each design [70]. For this research, an initial starting iterate of $[0,0]$ was used.

4.2.2. Viennet3.

$$\min\ F_1(x,y)=0.5(x^2+y^2)+\sin(x^2+y^2)$$
$$F_2(x,y)=\frac{(3x-2y+4)^2}{8}+\frac{(x-y+1)^2}{27}+15$$
$$F_3(x,y)=\frac{1}{(x^2+y^2+1)}-1.1e^{(-x^2-y^2)}$$

subject to

$$-3\le x,y\le 3$$

Walston tested this problem using a full factorial with 4 levels and 5 replications [70]. An initial starting iterate of $[0,0]$ was used.

4.2.3. Poloni.

$$\min\ F_1(x,y) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2$$
$$F_2(x,y) = (x+3)^2 + (y+1)^2$$

subject to

$$-\pi \le x, y \le \pi$$

where,

$$A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2)$$
$$A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2)$$
$$B_1 = 0.5\sin(x) - 2\cos(x) + \sin(y) - 1.5\cos(y)$$
$$B_2 = 1.5\sin(x) - \cos(x) + 2\sin(y) - 0.5\cos(y)$$

Walston noted that for this problem the published solution contained obviously dominated points [70]. She also noted that aspiration and reservation levels generally resulted in points on the middle of the Pareto front, but after adjustment of the ranges, points on the lower right side of the curve were found. Poloni was originally a maximization problem. An initial starting iterate of $[0,0]$ was used.

4.2.4. Tamaki.

$$\min\ F_1(x,y,z) = -x$$
$$F_2(x,y,z) = -y$$
$$F_3(x,y,z) = -z$$

subject to

$$x^2 + y^2 + z^2 \le 1$$
$$x, y, z \ge 0$$

This problem was originally a maximization problem. An initial starting iterate of $[1,1,1]$ was used, even though it is infeasible with respect to the nonlinear constraint.

4.2.5. Dias Γ1.

$$\min\ F_1(\vec{x}) = x_1$$

$$F_2(\vec{x}) = \left[1+9\sum_{i=2}^{30}\left(\frac{x_i}{29}\right)\right]\left[1-\sqrt{\frac{F_1(\vec{x})}{1+9\sum_{i=2}^{30}\left(\frac{x_i}{29}\right)}}\right]$$

subject to

$$0 \le x_i \le 1, \quad i = 1,2,...,30$$

where,

$$\vec{x} = [x_1,...,x_{30}]$$

On this problem, Walston used confined ranges to fill gaps in the objective space [70]. When proper noise was added and subtracted, the second objective could yield imaginary numbers. The square root term was set to zero whenever this occurred. An initial starting iterate of $[0]^{30}$ was used.

4.2.6. Dias Γ2.

$$\min\ F_1(\vec{x}) = x_1$$

$$F_2(\vec{x}) = \left[1+9\sum_{i=2}^{30}\left(\frac{x_i}{29}\right)\right]\left[1-\left(\frac{F_1(\vec{x})}{1+9\sum_{i=2}^{30}\left(\frac{x_i}{29}\right)}\right)^2\right]$$

subject to

$$0 \le x_i \le 1, \quad i = 1,2,...,30$$

where,

$$\vec{x} = [x_1,...,x_{30}]$$

Dias Γ2 is nearly identical to Dias Γ1, with only a change in the second objective. An initial starting iterate of $[0]^{30}$ was used.

4.2.7. Fonseca F1.

$$\min\ F_1(x_1,x_2) = 1-\exp(-(x_1-1)^2-(x_2+1)^2)$$
$$F_2(x_1,x_2) = 1-\exp(-(x_1+1)^2-(x_2-1)^2)$$

subject to

$$-2 \le x_1 \le 2$$
$$-2 \le x_2 \le 2$$

An initial starting iterate of $[0,0]$ was used for Fonseca F1. This problem is nonconvex.

4.2.8. Schaffer F3.

$$\min\ F_1(x) = \begin{cases} -x, & x \le 1 \\ -2+x, & 1 < x \le 3 \\ 4-x, & 3 < x \le 4 \\ -4+x, & 4 < x \end{cases}$$
$$F_2(x) = (x-5)^2$$

subject to

$$-5 \le x \le 10$$

Here the first objective is a piece-wise function. Walston again created specific ranges in order to achieve her results [70]. An initial starting iterate of 1.5 was used. This problem is discontinuous.

4.2.9. Srinivas.

$$\min\ F_1(x,y) = (x-2)^2+(y-1)^2+2$$
$$F_2(x,y) = 9x-(y-1)^2$$

subject to

$$-20 \le x, y \le 20$$
$$x^2+y^2-225 \le 0$$
$$x-3y+10 \le 0$$

On this problem, Walston noted that the published solution contained many dominated solutions [70]. An initial starting iterate of $[10,10]$ was used.

4.2.10. DTLZ7.

$$\min\ F_1\left(x_1,x_2\right)=x_1$$

$$F_2\left(x_1,x_2\right)=(1+10x_2)\left(1-\left(\frac{x_1}{1+10x_2}\right)^2-\frac{x_1\sin(8\pi x_1)}{1+10x_2}\right)$$

subject to

$$0\le x_1,x_2\le 1$$

An initial starting iterate of $[0,0]$ was used for DTLZ7. This problem is discontinuous.

4.2.11. Disk Brake.

$$\min\ F_1(\vec{x})=4.9\times10^{-5}(x_2^2-x_1^2)(x_4-1)$$

$$F_2(\vec{x})=\frac{9.82\times10^6(x_2^2-x_1^2)}{x_3x_4(x_2^3-x_1^3)}$$

subject to

$$(x_2-x_1)-20\ge 0$$

$$30-2.5(x_4+1)\ge 0$$

$$0.4-\frac{x_3}{3.14(x_2^2-x_1^2)}\ge 0$$

$$1-\frac{2.22\times10^{-3}x_3(x_2^3-x_1^3)}{(x_2^2-x_1^2)^2}\ge 0$$

$$\frac{2.66\times10^{-2}x_3x_4(x_2^3-x_1^3)}{(x_2^2-x_1^2)}-900\ge 0$$

$$55\le x_1\le 80$$

$$75\le x_2\le 110$$

$$1000\le x_3\le 3000$$

$$2\le x_4\le 20$$

where,

$$\vec{x}=(x_1,x_2,x_3,x_4)$$

The discrete variable, x_4, represents the number of disks in the brake. An initial starting iterate of $[55,75,1000,2]$ was used. Due to constraints, values for the discrete variable may be reduced to $\{2,3,...,11\}$. Mixed variable problems in NOMADm require a discrete neighbor file. To be consistent with Walston [70], an iterate's discrete neighbors were set to be ± 1 from the current value, provided that a neighbor still had a value between 2 and 11.

4.3. Nadir Point Genetic Algorithm

The Nadir point genetic algorithm was tested using a DOE approach with a full factorial design with either two or three levels, dependent upon the factor. The specific levels tested are shown in Table 4.3.1. Replenishment refers to only keeping unique individuals in the population and inserting new randomly-generated individuals into the population to replace duplicates.

Table 4.3.1: GA Test Levels

Factor	Low	Center	High
Population	100	-	200
Generations	500	-	1000
Distribution Index	10	20	30
Probability of Crossover	0.5	-	0.9
Replenishment	Off	-	On

Again, the "true" nadir points were still considered the published points, as presented by Walston [70]. Additionally, 1% of the nadir component was added for noise. The levels were based on recommended settings for NSGA-II, and in the case of low generations, this setting was found by initial testing, running the GA on a few problems using different values.

Using runtime (in seconds) and Euclidean distance between the GA solution and "true" point as measures, results follow in Table 4.3.2. The best solution to use as a response, that is, either the overall maximums found (O) in each objective, the final first front maximum objective values (P), or final population objective maximums (F), was found using a paired t-test with a significance level of 0.05, using the Euclidean distance to the “true” nadir point as a measure. Note this is using all runs for the response. Significant factors are highlighted in gray by problem, with the best setting denoted.

The Disk Brake problem is intentionally not in Table 4.3.2 and will be discussed later in this section. For the Dias Γ1, DTLZ7, Poloni, Schaffer F3, Srinivas, and Viennet3 problems, the overall solution was significantly worse (statistically) than the other two solutions. For the Dias Γ2, Fonseca F1, and Viennet4 problems, the final population solution was statistically best. For the Tamaki problem, the final non-dominated front was statistically best, although the practical difference was minimal. However, for the DTLZ7, Fonseca F1, Schaffer F3, Srinivas, Tamaki, and Viennet3 problems, the final population and final non-dominated front solution averages were either identical or near-identical. In addition, for the Viennet4 problem, the final population solution was far better than the final non-dominated solution. Therefore, the best solutions listed in Table 4.3.2 should be considered appropriately.

Table 4.3.2: GA Results

Problem	Best Solution	Avg Distance	Measure	Population	Generations	Probability of Crossover
Dias Γ1	P	0.18	Time	Low	Low	
			Distance	High	High	
Dias Γ2	F	1.17	Time	Low	Low	Low
			Distance		High	
DTLZ7	P	0.37	Time	Low	Low	Low
			Distance			
Fonseca F1	F	0.007	Time	Low	Low	Low
			Distance			
Poloni	P	27.84	Time	Low	Low	Low
			Distance			
Schaffer F3	P	0.05	Time	Low	Low	Low
			Distance			
Srinivas	P	29.81	Time	Low	Low	Low
			Distance			
Tamaki	P	0	Time	Low	Low	Low
			Distance			
Viennet3	P	1.93	Time	Low	Low	Low
			Distance			
Viennet4	F	8.09	Time	Low	Low	Low
			Distance		Low	

Distibution index and replenishment are not shown in Table 4.3.2 because they were never significant for any measure or problem. The Dias Γ1 and Dias Γ2 results suggest using a higher number of individuals in the population and higher number of generations to minimize the distance measure, but obviously this is detrimental time-wise. The Viennet4 results suggest using a low number of generations, but the regression model was, in fact, a poor model, and the raw data did not necessarily support this finding. The crossover appears to be time-consuming (as a low probability of crossover is significantly faster than using a high probability of crossover) and it, replenishment, and the distribution index appeared to have no global impact on solution quality.

Clearly, a low number of crossovers, and if possible, small population size and number of generations is better for computational time. In addition, either the final population or non-dominated front should be used as the solution. At this point, it was important to look more closely at the raw data, both for settings to use, and to explain how the estimation could be so good for most problems, but very poor for the Poloni, Srinivas, and Viennet4 problems. It became clear at this point, that perhaps the "true" nadir points from published solutions were, in fact, **not** the true nadir points.

Looking at the raw data, it became apparent that using a low probability of crossover, low population size, high number of generations, and the final population estimation yielded the best overall solution quality among all problems. Replenishment was turned off, as doing so provided a slight advantage in a few problems, and a distribution index of 20 was used because it was the recommended value [27]. A lower or higher index yielded slight advantages in respective problems, but no clear advantage emerged as 20 also sometimes yielded an advantage. Recall that these factors did not have significantly effect results for time or distance. Furthermore, the DOE analysis did not indicate increasing generations beyond 1000 would be of any substantial benefit. With replenishment off, the final population should converge to the final non-dominated front, and if not, using the final population maximums allows for a higher value estimate of the nadir point components (in case the algorithm is not finished coverging).

At these settings, results follow in Table 4.3.3 for all but the Disk Brake problem. Here, 0.5% of the maximum objective function value was added and subtracted again as noise. The published deterministic nadir point, as well as that found by MADS-RS with noise (presented in the next section) are included. For the Srinivas and Viennet4 problems, the algorithm overestimated the nadir point compared to MADS-RS/GPS-RS. Overall, the algorithm performed reasonably well.

Table 4.3.3: GA Nadir Points at Chosen Settings

Problem	Time	Published	MADS-RS	GA
Dias Γ1	123	1, 1	1, 1.38	1.00, 1.16
Dias Γ2	120	1.1, 1.1	1, 1.6	1.01, 1.28
DTLZ7	118	0.85, 1.4	1, 1.7	0.81, 1
Fonseca F1	109	1.01, 1.01	1, 1	1, 1
Poloni	126	30, 50	18.41, 24.72	16.64, 25.01
Schaffer F3	124	1, 16	1, 16 *(GPS-RS)*	0.99, 15.95
Srinivas	126	250, 10	222.9, 21.83	278.19, 17.84
Tamaki	59	0, 0, 0	0, 0, -0.01	0, 0, 0
Viennet3	65	10, 18, 0.2	8.1, 17.24, 0.2	8.28, 17.13, 0.19
Viennet4	73	7.5, -11, 26	7.65, -12.47, 25.79	9.91, -11.47, 33.64

The Disk Brake problem is the only example of a mixed variable problem tested, and it was the problem for which the performance of the GA was the poorest. The published nadir point was [2.75, 33], whereas the MADS-RS solution was [2.8, 48.25]. Using random selection for the discrete variable mutation and crossover, the overall nadir point estimate (O) was consistently better than the other two estimates (P,F), often corresponding to individuals from the initial population. Only in a few cases did the other estimates have a reasonable solution for Objective 2; otherwise, they typically were on the range of 3-6 for Objective 2. One of the typical solutions is shown in Figure 4.3.1. The better solutions did not appear to correlate in any way to parameter settings. Using a lower probability of crossover and higher mutation rate could not, and did not, correct the problem. Generations and population size were significant factors with respect to time.

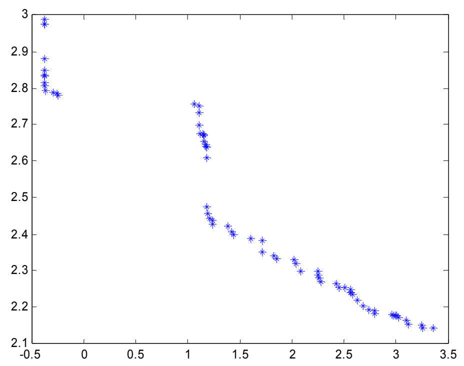


Figure 4.3.1: Typical Initial Disk Brake GA Final Non-Dominated Front

A plethora of crossover and mutation possibilities were tested with regard to the discrete variable, now without noise. Running with some of the prescribed settings found using the other ten problems, crossovers were performed by: 1) random selection from the discrete set; 2) doing Simulated Binary Crossover (SBX) and finding the nearest discrete neighbor; 3) randomly selecting from the parents (both children could have the same value); 4) switching between parents; and 5) taking the mean of the variable between the parents, and rounding up and down. Mutation was similarly done several ways: 1) random selection from the discrete set; 2) finding the nearest neighbor to the polynomial mutation; and 3) leaving the discrete variable untouched. Additionally, 2000 generations were evaluated.

Testing each combination of crossover and mutation, the true nadir component for Objective 1 was consistently found in the final population (2.793), with the exception of two combinations that came to 2.8. However, only four combinations managed to come close to the second objective component (an estimate >40, otherwise the estimate was 2.55). This could be partly random, as with all possible combinations no clear trend emerged. These four instances, shown in Table 4.3.4, were then replicated 10 times each. The results of the 40 runs for Objective 2 are shown in Table 4.3.5.

Table 4.3.4: Disk Brake Instances

Instance	Generations	Probability of Crossover	Replenishment	Crossover Type	Mutation Type
1	2000	0.9	Off	Random Selection	Random Selection
2	2000	0.9	Off	Nearest Neighbor to SBX	Nearest Neighbor to Polynomial Mutation
3	1000	0.9	Off	Parent Switch	Random Selection
4	2000	0.5	Off	Ceiling/Floor Mean Value	Random Selection

Table 4.3.5: Disk Brake Runs, Objective 2

1 (120)	2 (107)	3 (50)	4 (96)
2.56	3.13	2.56	2.56
43.47	3.13	2.56	42.92
2.57	3.13	2.56	2.56
2.56	3.13	42.78	2.56
2.56	2.56	2.56	2.56
2.56	3.13	2.56	2.56
2.56	2.56	2.56	45.05
42.68	3.13	2.56	2.56
2.56	3.52	2.60	2.56
2.56	2.56	2.56	2.56

None of the instances consistently achieved a desirable value, and they often converged to somewhere near 2.55. However, it is likely due to random number draws, selecting the particular discrete value, that the extreme solution is ever achieved. Therefore, random selection in the crossovers and mutations is likely suitable and a much larger population size may be of value. Leaving replenishment off is best nonetheless. Further, note that the probability of crossover may be left low, and in the end, generations and population size will likely need to be increased and replications conducted, to get the extreme solution. This may not be true in all MVP however, as this analysis is only based on a single problem.

Since the initial runs were done with noise added, each of the remaining problems was run an additional ten times using the determined settings without noise added, to determine the effect of the noise on the estimation. The average results follow in Table 4.3.6. Note the MADS results here are also without noise added.

Table 4.3.6: GA W/Out Noise

Problem	Time	GA	Published	MADS
Dias Γ1	46	1, 1.02	1, 1	1, 1
Dias Γ2	49	1, 1.01	1.1, 1.1	1, 1
DTLZ7	46	0.82, 1	0.85, 1.4	0.82, 1
Fonseca F1	46	1, 1	1.01, 1.01	1, 1
Poloni	46	16.77, 25.64	30, 50	16.77, 28.22
Schaffer F3	48	1, 16	1, 16	1, 16 *(GPS)*
Srinivas	45	277.65, 18	250, 10	225.55, 2.34
Tamaki	45	0, 0, 0	0, 0, 0	0, 0, -0.03
Viennet3	45	7.58, 17.04, 0.176	10, 18, 0.2	8.1, 17.04, -0.03
Viennet4	52	11.00, -11.34, 34.09	7.5, -11, 26	7.61, -12.22, 22.08

Over all problems, the solutions were extremely consistent for the GA, typically converging to a single solution each run, and were often of good quality. However, in the Viennet4 problem, the first component of the nadir point is high compared to MADS, although the GA was consistent in getting a value of 11. A similar event occurred with the first objective of Srinivas. The results for the GA without noise and with noise are reasonably similar.

In conclusion, the GA seems useful for getting an approximation on most problems, but performing replications is recommended so as to give the algorithm enough chances to converge to the correct solution. At the same time, it may be best to use MADS in the MVP case, although that conclusion is based off of only one problem. Furthermore, as the complexity and number of objectives increase, generations and replications should be increased. No ranking and selection procedure was used inside the algorithm because the effect of the noise should be somewhat mitigated by the large number of crossovers and mutations that take place, and because it would add time to the runs. As was seen with low noise, the mean response and a good nadir point approximation seemed to emerge from the algorithm.

4.4. MADS Nadir/Utopia Points

MADS, and GPS in some cases, was also used to find the utopia and nadir points. Runs were completed such that given a number of replications the best solution was taken from those replicates. Each replication number was itself replicated and Table 4.4.1 includes an average best point found using 5, 10, and 20 replications of MADS (or if asterisked, one run of GPS). The best point found overall was included in Table 4.3.3. Objectives used here were deterministic. The computational time was at most on the order of minutes; however, this was using a limit of 50000 function evaluations and would be faster otherwise. Note that using MADS or GPS is preferable to the GA, due to fewer function evaluations.

Five to ten replications are enough to find an accurate estimate of the utopia or nadir point. However, depending upon the fidelity required, even fewer replications may suffice (as in two or three). Many of these results were duplicated using different starting iterates in an attempt to make the results more robust. Additionally, in contrasting to those points found in Walston [70], either the same points, or perhaps better estimations of the nadir and utopia points, were found here.

Schaffer F3 was an interesting problem in that it is extremely sensitive to its only variable and for some unknown reason the implementation of MADS used in this research had difficulty accurately estimating the utopia and nadir points, while GPS did not. Further investigation is needed to explain this phenomenon.

Table 4.4.1: MADS Utopia and Nadir Points

Problem	# Reps	MADS Nadir	Published Nadir	MADS Utopia	Published Utopia
Dias Γ1	5	1, 1	1, 1	0, 0	0, 0
	10	1, 1		0, 0	
	20	1, 1		0, 0	
Dias Γ2	5	1, 1	1.1, 1.1	0, 0	0, 0
	10	1, 1		0, 0	
	20	1, 1		0, 0	
DTLZ7	5	0.818, 1	0.85, 1.4	0, -0.240	0, -0.6
	10	0.818, 1		0, -0.479	
	20	0.818, 1		0, -0.479	
Disk Brake	5	2.796, 49.965	2.75, 33	0.127, 2.071	0, 0
	10	2.793, 49.965		0.127, 2.071	
	20	2.793, 49.965		0.127, 2.071	
Fonseca F1	5	1, 1	1.01, 1.01	0, 0	0, 0
	10	1, 1		0, 0	
	20	1, 1		0, 0	
Poloni	5	16.7723, 25	30, 50	1, 0	0, 0
	10	16.772, 28.224		1, 0	
	20	16.772, 25.000		1, 0	
*Schaffer F3**	1	1, 16	1, 16	-1, 0	-1, 0
Srinivas	5	224.554, 2.167	250, 10	10.114, -217.555	0, -250
	10	221.829, 2.226		10.102, -217.611	
	20	224.400, 2.326		10.102, -217.500	
Tamaki	5	-0.012, 0, -0.004	0, 0, 0	-1, -1, -0.993	-1, -1, -1
	10	-0.033, -0.033, -0.059		-1, -0.999, -1	
	20	-0.016, -0.008, -0.016		-1, -0.998, -1	
Viennet3	5	6.515, 17.037, -0.035	10, 18, 0.2	0, 15, -0.1	1, 15, -0.2
	10	8.099, 17.037, -0.035		0, 15, -0.1	
	20	8.099, 17.037, -0.035		0, 15, -0.1	
Viennet4	5	7.611, -12.205, 21.846	7.5, -11, 26	3.324, -12.984, 15.009	3.3, -13, 15
	10	7.611, -12.204, 21.849		3.323, -12.984, 15.009	
	20	7.611, -12.221, 21.913		3.323, -12.984, 15.009	

Using MADS with a starting iterate of $x_0 = 1$ and eight LHS sites in the search step, a utopia point of [-1, 0.3906] and a nadir point of [0.375 16] were consistently achieved. Using starting iterates $x_0 < 1$, a utopia point of [-0.625, 0.3906] and nadir point of [0.375, 19.14] were consistently achieved. Furthermore, using $x_0 = 4.5$, a utopia point of [-0.625, 0.25] and nadir point of [0.5 19.14] were consistently achieved. However, by increasing the number of LHS sites to 40, in ten replications a utopia point of [-1, 0.0156] and a nadir point of [1.125, 16] were found. These estimates are very near the true points. Again, using GPS instead of MADS, the true utopia and nadir points were always found ([-1 0], [1 16]). The Viennet4 and DTLZ7 problems were also run using GPS, with the Viennet4 problem having extremely similar results to those shown for MADS, and the DTLZ7 problem surprisingly not doing as well as in the MADS case unless the number of LHS sites was increased to 10.

These MADS and GPS estimates were then used to create new noise levels equal to 1% of the nadir objective function value. The values used are included in Table 4.4.2. They were also used to create the indifference values for the problems, typically set at 0.1 times the difference in utopia and nadir components.

Table 4.4.2: Noise Values and Indifference Values

Problem	Noise	Indifference
Dias Γ1	0.01, 0.01	0.1, 0.1
Dias Γ2	0.01, 0.01	0.1, 0.1
DTLZ7	0.0082, 0.01	0.085, 0.2
Disk Brake	0.03, 0.49	0.275, 3.3
Fonseca F1	0.01, 0.01	0.1, 0.1
Poloni	0.17, 0.29	3, 5
Schaffer F3	0.01, 0.16	0.2, 1.5
Srinivas	2.25, 0.024	25, 26
Tamaki	0.01, 0.01, 0.01	0.1, 0.1, 0.1
Viennet3	0.08, 0.17, 0.001	0.9, 0.3, 0.04
Viennet4	0.076, 0.12, 0.22	0.42, 0.2, 1.1

Table 4.4.3: MADS-RS w/Noise

Problem	Utopia	Nadir
Dias Γ1	0, 0	1, 1.38
Dias Γ2	0, 0	1, 1.6
DTLZ7	0, 0	1, 1.7
Disk Brake	0.12, 2.08	2.8, 48.25
Fonseca F1	0, 0	1, 1
Poloni	1.04, 0	18.41, 24.72
*Schaffer F3**	-1, 0.03	0.97, 16
Srinivas	10.09, -217.68	222.9, 21.83
Tamaki	-1, -1, -1	0, 0, -0.01
Viennet3	-0.01, 15.01, -0.1	8.1, 17.24, 0.2
Viennet4	3.34, -12.93, 14.92	7.65, -12.47, 25.79

A set final set of runs was conducted using ten replications and +/-0.5 of the noise values from Table 4.4.2, to see the effect of noise on the estimations. The results are shown in Table 4.4.3. The introduction of noise begins to affect the estimation adversely (for example, the DTLZ7 utopia point), but most estimates are still reasonable. These estimations, with a set of replications, should probably be conducted multiple times on problems with unknown extreme points. Although GPS does not need to be replicated on a deterministic objective, GPS-RS does need to be replicated, in the event noise affects the optimization.

4.5. Exploration of SMOMADS Parameters

4.5.1. Test Approach. The use of MADS-RS and SMOMADS is not always straightforward. The implementation of MADS in the NOMADm software randomly selects a set of positive spanning directions in its poll step. Therefore, the objective function value found at the conclusion of a run is in no way deterministic, even without noise. This is true for the achievement scalarization function, and thus adds a random component into the true objective space. In addition, a CCD or LHS can be used to find points in the search step. Although using a CCD may provide more stability (in terms of a LHS design being random), the number of function evaluations required grows exponentially as the number of factors or variables increases. For example, the Dias Γ1 or Γ2 problems have 30 decision variables which results in a MATLAB® error due to the memory required. Because of this, a LHS was always used in this research.

The runs presented in this section followed a set of runs described in Appendix A that were done prior, which yielded similar results. The purpose of the initial runs was to determine the effect of the nadir point estimate, the number of replications, the level of noise, and the range over which aspiration and reservation levels were to be sampled. Published nadir points were found to be likely incorrect in some cases, and accurate nadir points yielded better results than overestimated nadir points (using maximum possible objective function values). The initial results also showed that there is little advantage to performing more than two replications of a design, with respect to generating unique Pareto points.

Table 4.5.1: Range Bounds

	Aspiration Levels Bounds	Reservation Levels Bounds
AR1	$\left[f_i^g, 0.99 \times mean(f_i^g, f_i^b)\right]$	$\left[1.01 \times mean(f_i^g, f_i^b), f_i^b\right]$
AR2	$\left[f_i^g, f_i^g + \frac{\lvert f_i^g - f_i^b \rvert}{3}\right]$	$\left[f_i^b - \frac{\lvert f_i^g - f_i^b \rvert}{3}, f_i^b\right]$
AR3	$\left[f_i^g - \frac{2}{5} \cdot \lvert f_i^g - f_i^b \rvert, f_i^g + \frac{2}{5}\lvert f_i^g - f_i^b \rvert\right]$	$\left[f_i^b - \frac{2}{5}\lvert f_i^g - f_i^b \rvert, f_i^b + \frac{2}{5}\lvert f_i^g - f_i^b \rvert\right]$

Within the following tables, AR refers to the design space used to create the aspiration and reservation levels, where Table 4.5.1 shows the lower and upper bounds used for the six ranges, in three combinations. Here, f_i^g denotes Objective *i* of the utopia point and f_i^b denotes Objective *i* of the nadir point. *NRI* refers to using *I* replications of the design. ND1 refers to using a good estimate of the nadir point, and ND2 refers to using an over-approximation based on maximum objective function values (except for the Tamaki and Fonseca F1 problems, where an overestimation is not possible). Both points are shown in Table 4.5.2, in that order. The words *bogus points* refer to the number of dominated points found.

All metrics were computed using true utopia and nadir points, so as to be comparable. All runs were with a CCD and 2 replications (unless NR3). AR1, AR2, AR3, ND1, and ND2 runs were all conducted with low noise. The three replicate run used AR1, low noise, and the true nadir point. The ND1 and ND2 runs were conducted using AR1. MADS-RS was used to perform the optimizations.

Table 4.5.2: Test Settings

Problem	f_1^g	f_2^g	f_3^g	f_1^b	f_2^b	f_3^b	μ_1	μ_2	μ_3
Viennet4	3.3	-13	15	7.5; 23	-11; -4	26; 90	0.42	0.2	1.1
Viennet3	1	15	-0.2	10; 10	18; 61	0.2; 1	0.9	0.3	0.04
Poloni	0	0		30; 32	50; 52		3	5	
Tamaki	-1	-1	-1	0	0	0	0.1	0.1	0.1
Dias Γ1	0	0		1; 1	1; 10		0.1	0.1	
Dias Γ2	0	0		1.1; 1.1	1.1; 10		0.1	0.1	
Fonseca F1	0	0		1.01	1.01		0.1	0.1	
Schaffer F3	-1	0		1; 4	16; 169		0.2	1.6	
Srinivas	0	-250		250; 687	10; 180		25	26	
DTLZ7	0	-0.6		0.85; 1	1.4; 11		0.085	0.2	
Disk Brake	0	0		2.75; 4	33; 50		0.275	3.3	

As stated previously, Walston [70] strictly added noise (no subtraction). In this research noise was added differently, but in a way that ensures an expected value of zero. Noise was added by multiplying a uniform random number on [-1,1] by 0.5%, 1%, 5%, and 10% (N1, N2, N3, N4 respectively) of the respective nadir component, yielding ranges of 1%, 2%, 10% and 20%. Noise level will be referred to using the +/- numbers, not the range. These runs were adequate because there was no evidence during the initial runs that interactions were significant, relative to main effects (the columns in the tables being main effects). All runs here were done with a limit of 50000 function evaluations.

4.5.2. Results. Again, only a representative subset of problems is shown for brevity, even though analysis was conducted on all problems.

Table 4.5.3: DTLZ7 Measures

Measure	AR1	AR2	AR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	36	38	43	36	44	39	41	36	55
Entropy	0.93	0.85	0.94	0.93	0.95	0.95	0.96	0.93	0.33
OS	5.25	0.95	2.62	5.25	2.72	8.37	1.64	5.25	0.56
OS1	1.00	0.99	1.01	1.00	1.01	1.08	1.05	1.00	0.08
OS2	5.23	0.96	2.60	5.23	2.70	7.78	1.57	5.23	6.82
NDC	12	7	12	12	14	16	15	12	6
CL	3.00	4.86	2.42	3.00	2.00	2.06	2.07	3.00	2.83
Time	1520	1551	3533	1520	1733	1785	1890	1520	1473
Largest Gap	6.10	0.52	2.35	6.10	1.25	3.77	0.41	6.10	5.04
Avg Gap	1.27	0.35	0.82	1.27	0.53	1.55	0.29	1.27	5.00
# Gaps	6	4	4	6	7	7	5	6	2

Table 4.5.3 gives results for DTLZ7, which has a discontinuous Pareto front. As expected, as noise increases so does computational time. Using the true nadir point provides a much better approximation of the front. Looking at OS2, the necessity for a final check for domination is apparent. Recall from Section 3.2 that any value above 1 implies either the utopia or nadir point is not estimated correctly, but correct points are being used here, so the high values have to be due to noise. AR1 and AR3 each had a point that single handedly caused such high OS2 values. Otherwise, AR1 and AR3 are relatively comparable, but the design levels used in AR3 cause a much higher run-time.

For the mixed variable Disk Brake problem, with results shown in Table 4.5.4, AR1 and AR3 are again comparable, except that AR3 has a bigger largest gap. Additionally, here AR1 requires more time. Furthermore, the over-estimated nadir point finds better extreme solutions. Note that three replicates provided no advantage over two, as was expected.

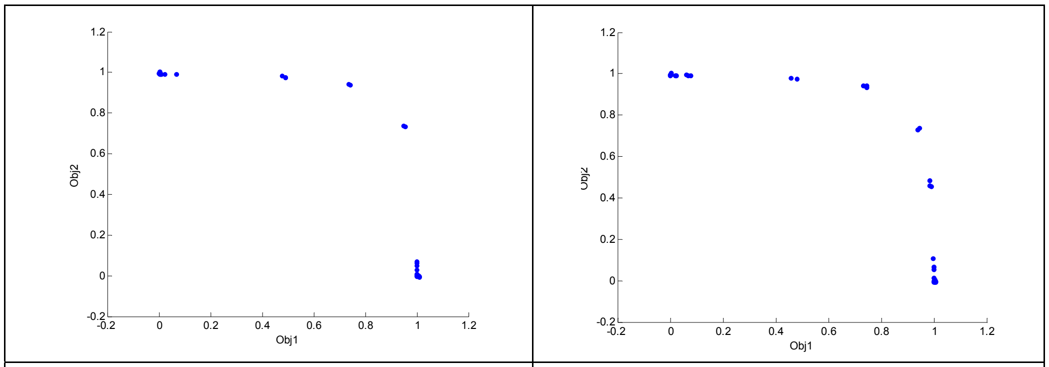
Table 4.5.4: Disk Brake Measures

Measure	AR1	AR2	AR3	NR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	10	16	9	10	10	11	18	22	10	14
Entropy	0.83	0.79	0.83	0.83	0.83	0.83	0.86	0.83	0.83	0.90
OS	0.16	0.07	0.18	0.18	0.16	0.12	0.27	0.23	0.16	0.53
OS1	0.47	0.32	0.48	0.48	0.47	0.46	0.65	0.37	0.47	0.98
OS2	0.34	0.23	0.38	0.36	0.34	0.27	0.41	0.64	0.34	0.54
NDC	9	7	9	9	9	9	8	10	9	9
CL	2.89	2.86	3.00	2.89	2.89	2.78	2.25	1.40	2.89	2.44
Time	406	331	290	395	406	431	1427	1431	406	1286
Largest Gap	1.47	3.74	3.59	6.67	1.47	3.64	5.57	16.77	1.47	9.15
Avg Gap	1.47	3.74	2.44	4.17	1.47	2.08	4.46	6.17	1.47	4.82
# Gaps	1	1	2	2	1	2	3	3	1	4

Results for the non-convex problem Fonseca F1 are shown in Table 4.5.5. As in most cases, AR2 did not perform as well as AR1 and AR3, which performed equally well, except that AR1 was much faster. This seemed to be because AR3 samples outside the utopia and nadir point ranges, requiring more time for the optimization to reach the Pareto front. One interesting finding is that increased noise did not correlate to increased time. Also, as can be seen in Figure 4.5.1, three replications provided minimal improvement over two.

Table 4.5.5: Fonseca F1 Measures

Measure	AR1	AR2	AR3	NR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	40	37	52	78	40	41	43	57	40	-
Entropy	0.85	0.71	0.81	0.94	0.85	0.94	0.84	0.90	0.85	-
OS	1.02	1.01	1.02	1.02	1.02	1.03	1.12	0.99	1.02	-
OS1	1.01	1.01	1.01	1.01	1.01	1.02	1.07	1.00	1.01	-
OS2	1.01	1.00	1.01	1.01	1.01	1.01	1.05	0.99	1.01	-
NDC	10	6	9	11	10	13	14	7	10	-
CL	3.20	5.83	2.22	2.73	3.20	2.38	2.07	2.14	3.20	-
Time	1261	1062	2752	1665	1261	1035	1101	1252	1261	-
Largest Gap	0.66	0.84	0.56	0.38	0.66	0.28	0.38	0.46	0.66	-
Avg Gap	0.40	0.81	0.41	0.30	0.40	0.20	0.23	0.29	0.40	-
# Gaps	4	2	4	5	4	7	6	5	4	-



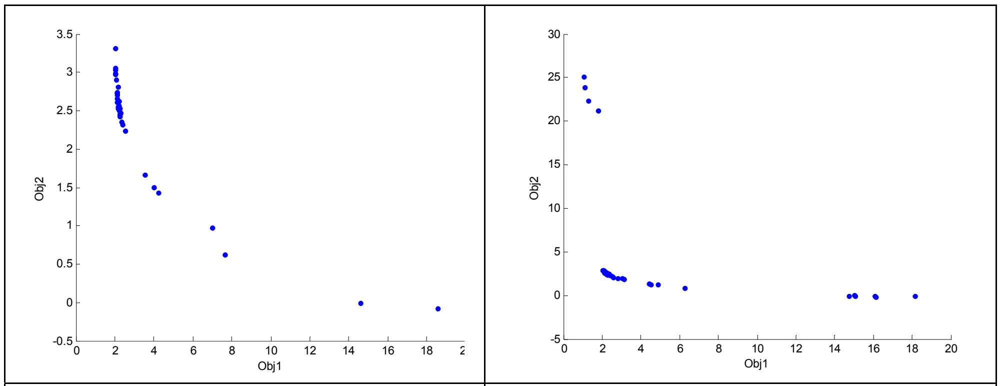
(a) 2 Reps (b) 3 Reps

Figure 4.5.1: Fonseca F1 Replications

Table 4.5.6: Poloni Measures

Measure	AR1	AR2	AR3	NR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	38	38	31	60	38	45	48	47	38	35
Entropy	0.58	0.52	0.71	0.59	0.58	0.68	0.78	0.81	0.58	0.68
OS	0.13	0.11	0.97	0.11	0.13	1.04	1.90	0.88	0.13	0.91
OS1	1.05	1.00	1.08	0.89	1.05	1.03	1.84	0.82	1.05	0.96
OS2	0.12	0.11	0.90	0.12	0.12	1.01	1.03	1.08	0.12	0.95
NDC	5	4	8	4	5	7	10	9	5	6
CL	6.80	8.50	5.13	12.00	6.80	3.86	2.40	2.78	6.80	6.17
Time	458	348	1287	815	458	1051	1957	2259	458	1388
Largest Gap	6.99	12.42	18.32	6.75	6.99	24.97	20.78	13.16	6.99	19.30
Avg Gap	6.99	12.42	13.43	6.75	6.99	14.73	11.62	11.03	6.99	12.34
# Gaps	1	1	2	1	1	2	4	2	1	2

The results for the Poloni problem are shown in Table 4.5.6. This is another discontinuous front. Here, the over-estimated nadir point performed better for AR1. However, using AR3 with the true nadir point, a better approximation was made. In fact, when looking at the raw data, AR1 was missing high values in the second objective. Those values in AR3 came from the axials of the CCD. This happened consistently, and indicates that axials are important. However, this only applies to CCDs. AR1 and AR3 are shown in Figure 4.5.2.



(a) AR1 (b) AR3

Figure 4.5.2: Poloni AR

Table 4.5.7: Srinivas Measures

Measure	AR1	AR2	AR3	NR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	22	16	17	49	22	25	24	30	22	27
Entropy	0.84	0.90	0.88	0.83	0.84	0.87	0.89	0.92	0.84	0.82
OS	0.97	0.96	0.97	0.93	0.97	0.97	0.91	0.81	0.97	0.88
OS1	0.96	0.94	0.97	0.98	0.96	0.95	0.90	0.88	0.96	0.97
OS2	1.01	1.02	1.00	0.95	1.01	1.02	1.00	0.91	1.01	0.92
NDC	9	11	12	8	9	9	14	13	9	6
CL	5.56	5.09	4.58	7.38	5.56	5.22	3.43	3.23	5.56	7.50
Time	222	191	367	189	222	263	420	559	222	157
Largest Gap	63.11	45.57	69.54	66.85	63.11	63.06	51.96	53.50	63.11	117.83
Avg Gap	59.53	40.12	68.73	59.19	59.53	55.21	51.67	44.17	59.53	99.16
# Gaps	4	5	2	4	4	4	2	2	4	2

The Srinivas problem results are shown in Table 4.5.7. Using the true nadir point is again advantageous. This was one of the rare cases where AR2 performed well.

However, AR3 performed just as well. Looking at the raw data, the points AR1 was missing often came from the center points, and sometimes the factorial-portion of the CCD for AR3.

The Tamaki problem results are shown in Table 4.5.8. Three replicates provide only a slight advantage, and similar to previous runs, AR3 is better than AR1. But again, those points not in AR1 that are in AR3 are relative to the CCD axials. Here, like the Fonseca F1 results, increased noise does not necessarily result in increased run time. The reason for this was unclear, except that with increased noise, MADS-RS could converge to a poor solution under the right circumstances (random number draws, etc.), or good solutions were found due to randomness.

Table 4.5.8: Tamaki Measures

Measure	AR1	AR2	AR3	NR3	N1	N2	N3	N4	ND1	ND2
Bogus Pts	2	1	5	2	2	5	29	51	2	-
Entropy	0.78	0.75	0.77	0.78	0.78	0.79	0.81	0.84	0.78	-
OS	0.27	0.15	0.81	0.34	0.27	0.56	0.60	0.38	0.27	-
OS1	0.64	0.55	0.96	0.71	0.64	0.84	0.84	0.74	0.64	-
OS2	0.68	0.48	0.93	0.68	0.68	0.81	0.89	0.80	0.68	-
OS3	0.63	0.58	0.91	0.71	0.63	0.84	0.80	0.65	0.63	
NDC	40	32	46	49	40	43	45	43	40	-
CL	2.90	3.66	2.46	3.57	2.90	2.63	1.98	1.56	2.90	-
Time	2662	1262	1331	4848	2662	3140	1206	572	2662	-
Largest Gap	0.25	0.17	0.78	0.20	0.25	0.78	0.88	0.29	0.25	-
Avg Gap	0.23	0.15	0.31	0.17	0.23	0.33	0.51	0.22	0.23	-
# Gaps	3	2	13	3	3	8	4	9	3	-

The trends were clear in these runs. Additional replications of a design beyond two still appear to have no real benefit. Furthermore, in general, it is best to use the true nadir point, or at least a good estimate thereof. AR1 generally performed better than AR3, and in all cases but one, most unique points from AR3 came from the CCD axials.

It is clear from the results that, in general, more noise means more computational time. However, +/-10% of the nadir objective function value appears to be too much noise for SMOMADS to generate reasonable solutions.

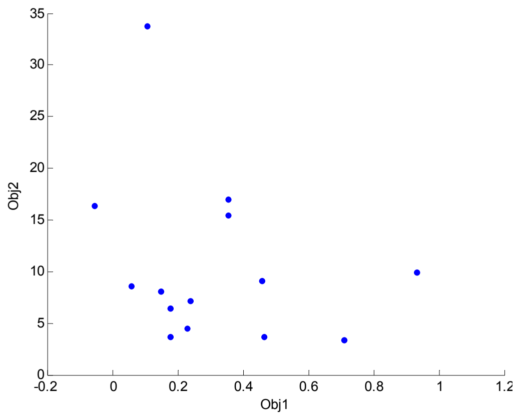


Figure 4.5.3: Disk Brake with 10% Noise

For example, consider the corresponding plot for Disk Brake in Figure 4.5.3. Clearly, the front has lost all shape, and many of the points are not on the true front. This may be more pronounced because Disk Brake is a mixed variable problem. The Pareto front for Poloni (see Figure 4.5.4a) has also lost most of its shape, while the front for Fonseca F1 (see Figure 4.5.4b) has not been too adversely affected by the high level of noise.

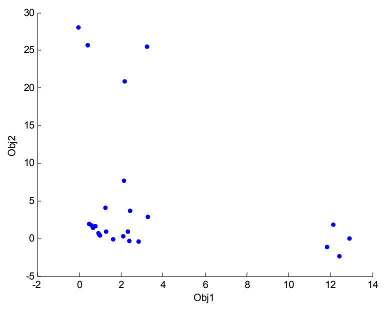
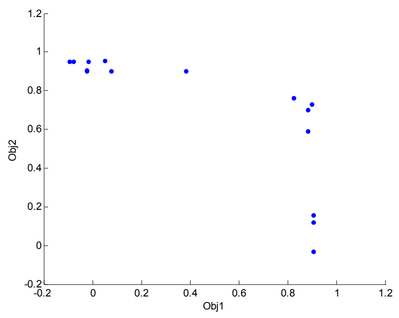
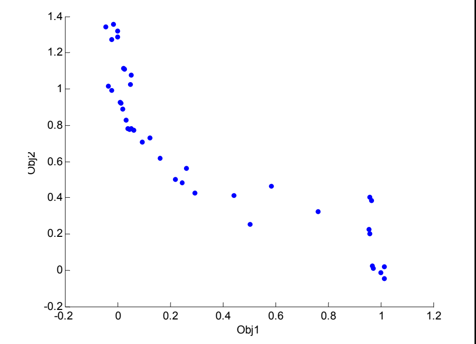
	
(a) Poloni	(b) Fonseca F1

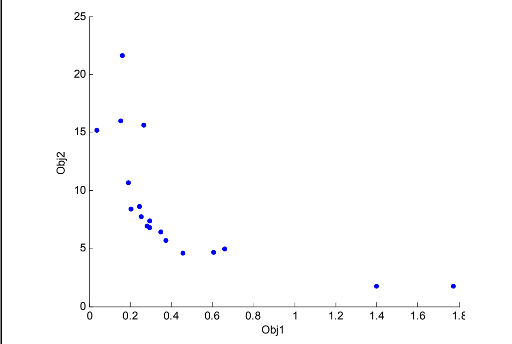
Figure 4.5.4: Problems with 10% Noise

Therefore, a single level of noise cannot be determined which may overwhelm MADS-RS or GPS-RS for all problems. However, a +/-5% noise level seems to retain most of the Pareto front shape and keep the front nearly correct across all problems. This level was applied to the Dias Γ1 and Disk Brake problems (see Figure 4.5.5), and to the

Viennet3 and Viennet4 problems (see Figure 4.5.6). Note the improvement specifically in the Disk Brake problem (Figure 4.5.5 versus Figure 4.5.3).

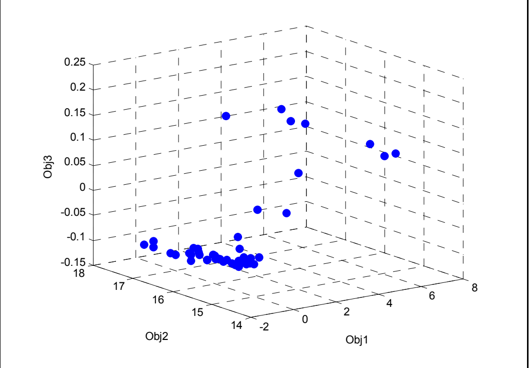


(a) Dias Γ1

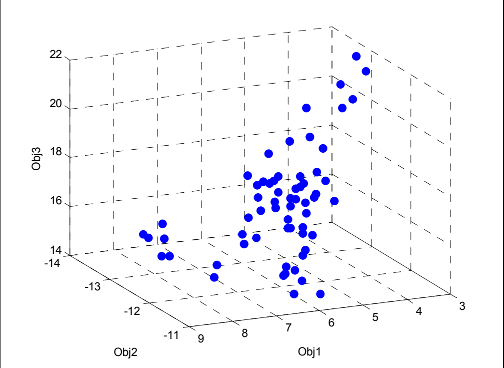


(b) Disk Brake

Figure 4.5.5: Two Objective Problems 5% Noise



(a) Viennet3



(b) Viennet4

Figure 4.5.6: Three Objective Problems 5% Noise

4.6. Experimental Design for Aspiration and Reservation Levels

4.6.1. Test Approach. Based on previous findings, these runs were done using two replications, 0.5% noise, MADS/GPS-estimated nadir points, and AR1. The noise was chosen to best represent Walston's [70] original intention to use 1% noise and so that the runs would not be too time consuming. Results from Section 4.5 showed that AR1, plus the axials from AR3, provided the best design range. However, the goal of these runs was to find a best experimental design, and to see if perhaps even using any sort of design levels involving AR3 could be avoided so that only one design would have to be used (instead of one design for AR1 and one for AR3).

All designs from Section 3.3 were evaluated, as appropriate. For those designs where a specific number of samples was required, a number equal to the number of runs for a CCD was used. Walston had noted that a CCD seemed to provide the best initial front during her experimentation [70]. In a few cases, designs were also tested with fewer points than the CCD. Additionally, Hammersley sequence sampling and near uniform designs were expanded to sample on a coded [-2,2] range, to include axial run space. The D-optimal design was tested using three levels with center points and axials added. However, as will be discussed, the D-optimal runs were accidentally, and fortunately, run using a different range.

In the case of the full factorial design, only three levels were used. Walston [70] used four or five levels; however, as the number of objectives increases, this becomes extremely intractable. In fact, at three objectives the five-level full-factorial is likely not a valuable option and becomes a brute force method by having an extremely large number of design levels. Certainly as the combinations of design levels and random number draws increase, more distinct points result. In the case of the Viennet3 and Viennet4 problems, the full-factorial at 3 levels was too much for the computers to handle. Therefore, these were evaluated with a limit of 500 on the number of function evaluations, rather than the 50000 function evaluation limit. All runs presented were conducted using MADS-RS.

Results follow for each problem in a table where metrics are columns and designs are rows. The key for the designs is shown in Table 4.6.1.

Table 4.6.1: Design Key

Design	Description
FF(i)	Full factorial with *i* levels
CCD(-)	C: Circumscribed, I: Inscribed, F: Face-centered
BB	Box-Behnken
LatinR(i)	Random Latin Hypercube with *i* samples
LatinL(i)	Lattice Latin Hypercube with *i* samples
LatinRC(i)	Random Latin Hypercube with reduced correlation and *i* samples
LatinLC(i)	Lattice Latin Hypercube with reduced correlation and *i* samples
OA(Multi)	Orthogonal Array using more than 2 levels
OA	Orthogonal Array using 2 levels
Hamm(i)	Hammersley sequence sampling using *i* samples
Hamm-A(i)	Hammersley sequence sampling using *i* samples taken over the [-2,2] coded range
Dopt(i)	D-Optimal design using *i* samples
Hybrid	Hybrid design
SCD	Small Composite design
Koshal1	Linear Koshal design
Koshal1+	Linear with interactions Koshal design
Koshal2	Quadratic Koshal Design
U(i)	Near-Uniform design with *i* samples
U-A(i)	Near-Uniform design with *i* samples taken over the [-2,2] coded range
MR5	Minimum Resolution V design

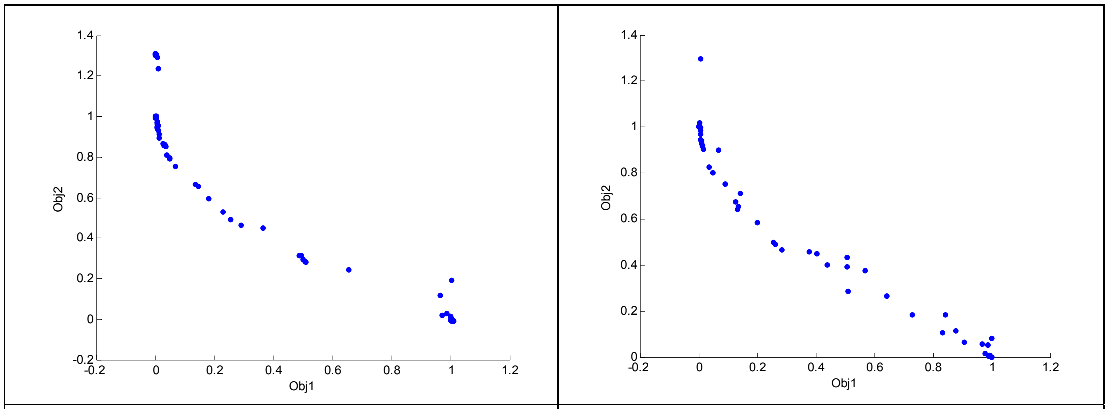
4.6.2. Results. Although the runs themselves were not replicated, the problems acted like replications, due to their large number. Therefore, consistent trends were noted.

The Dias Γ1 results are shown in Table 4.6.2. A few notable results emerged. All of the Latin Hypercube sampling methods performed extremely competitively, as did the Koshal1 and Koshal1+ designs. The D-optimal design, although only a 20 sample design, took a long time to complete as a result of the actual generation of the design. In some cases, the spreads are on the order of ~1.3, due to the lack of a final dominance check, and thus the gaps are really one less in number for such designs. In Figure 4.6.1, Hammersley(36) and FF(3) are shown. Hammersley sequence sampling may be able to achieve a better result in fewer points than other designs. This also held true for the near-uniform designs, although Hammersley(36) performed best here. Also, the face-centered and inscribed CCDs performed better than the circumscribed CCD.

.

Table 4.6.2: Designs for Dias Γ1

Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	88	0.87	1.33	1.01	1.32	20	3.70	6698	0.34	0.23	4
CCD(C)	32	0.83	1.32	1.01	1.31	16	2.50	3811	0.35	0.24	5
CCD(I)	21	0.91	1.32	1.01	1.31	20	2.55	724	0.30	0.19	4
CCD(F)	23	0.93	1.33	1.01	1.32	19	2.58	4018	0.40	0.23	8
BB	16	0.93	1.33	1.01	1.32	18	2.11	3056	0.24	0.19	6
LatinR(36)	32	0.94	1.01	1.00	1.00	17	2.35	247	0.28	0.18	5
LatinL(36)	29	0.92	1.02	1.00	1.01	16	2.69	193	0.25	0.22	4
LatinRC(36)	27	0.91	1.00	1.00	1.00	16	2.81	297	0.25	0.21	5
LatinLC(36)	34	0.96	1.01	1.01	1.00	15	2.53	199	0.27	0.20	4
OA(Multi)	4	0.93	0.95	0.99	0.96	8	1.75	47	0.42	0.21	6
OA	5	0.84	0.86	1.00	0.86	7	1.57	42	0.46	0.26	5
Hamm(36)	24	0.96	1.30	1.00	1.30	19	2.53	773	0.28	0.20	2
Hamm(20)	13	0.92	1.24	1.00	1.24	13	2.08	371	0.35	0.21	7
Dopt(20)	35	0.80	1.33	1.01	1.32	12	2.58	6116	0.69	0.41	6
Hybrid	9	0.93	1.33	1.01	1.32	13	1.92	1769	0.31	0.24	6
SCD	11	0.85	1.33	1.01	1.32	12	2.25	1845	0.41	0.25	5
Koshal1	2	0.84	0.90	1.00	0.90	8	1.50	36	0.43	0.31	4
Koshal1+	8	0.91	0.95	1.00	0.95	10	1.80	77	0.44	0.30	3
Koshal2	9	0.95	1.00	1.00	1.00	12	2.08	632	0.30	0.19	5
U(36)	29	0.85	1.00	1.00	1.00	14	3.07	193	0.29	0.21	4
U(20)	12	0.89	1.00	1.00	1.00	13	2.15	236	0.31	0.23	4
U-A(36)	32	0.93	1.03	1.00	1.03	16	2.50	190	0.28	0.22	3
U-A(20)	14	0.95	0.99	1.00	0.99	14	1.86	243	0.29	0.22	3
Hamm-A(36)	27	0.93	0.98	1.00	0.98	11	4.09	334	0.17	0.16	3
Hamm-A(20)	12	0.91	0.98	1.00	0.98	13	2.15	110	0.30	0.20	5



(a) FF(3) | (b) Hammersley(36)

Figure 4.6.1: Dias Γ1 FF and Hammersley Results

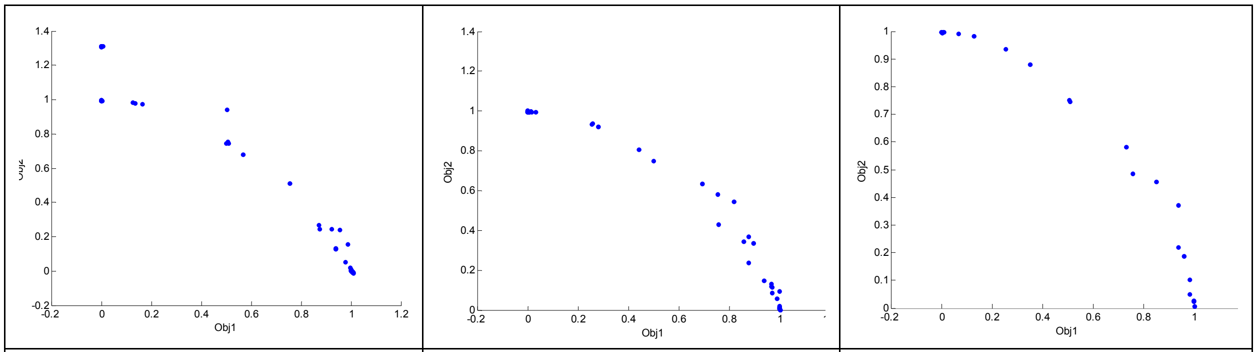
The Dias Γ2 results are shown in Table 4.6.3. The impracticality of using the full factorial design is evident in the time required. It provided some benefits versus the CCDs, but this should be expected, due to the larger number of design levels. The Box-

Behnken design provided no advantage over the CCDs. Further, the face-centered and circumscribed CCDs performed nearly the same, while the inscribed did better in the high (larger objective function value) Objective 1 region. Again, Latin hypercube sampling performed very well according to the Pareto quality metrics as did Hammersley sequence sampling and uniform design over both ranges. To show the good overall spreads and spread of points, the FF(3), LatinLC(36) and U(20) designs are shown in Figure 4.6.2.

Table 4.6.3: Designs for Dias Γ2

Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	110	0.83	1.34	1.01	1.33	15	3.47	32772	0.35	0.25	6
CCD(C)	39	0.84	1.33	1.01	1.32	12	2.75	9279	0.39	0.32	5
CCD(I)	40	0.90	0.99	1.00	0.99	12	2.67	1889	0.43	0.23	4
CCD(F)	39	0.72	1.34	1.01	1.33	12	2.75	10632	0.75	0.45	5
BB	27	0.79	1.34	1.01	1.32	13	2.08	8253	0.61	0.36	5
LatinR(36)	38	0.95	1.01	1.01	1.00	15	2.27	465	0.30	0.18	5
LatinL(36)	42	0.95	1.02	1.01	1.01	14	2.14	476	0.37	0.25	5
LatinRC(36)	40	0.90	1.00	1.00	1.00	11	2.91	459	0.35	0.26	4
LatinLC(36)	30	0.91	1.01	1.00	1.00	15	2.80	440	0.23	0.20	4
OA(Multi)	7	0.80	0.99	1.00	0.99	7	1.57	106	0.57	0.48	3
OA	6	0.89	1.00	1.00	1.00	6	1.67	100	0.57	0.34	4
Hamm(36)	30	0.92	1.33	1.01	1.32	16	2.63	1144	0.31	0.23	5
Hamm(20)	15	0.82	1.30	1.00	1.30	11	2.27	968	0.62	0.28	5
Dopt(20)	38	0.78	1.34	1.01	1.33	12	2.33	14927	0.46	0.37	4
Hybrid	13	0.78	1.33	1.01	1.32	11	1.91	4919	0.66	0.27	6
SCD	16	0.80	1.33	1.01	1.32	12	1.83	4361	0.50	0.37	4
Koshal1	5	0.86	0.98	1.00	0.98	6	1.50	99	0.55	0.33	4
Koshal1+	13	0.84	0.99	1.00	0.99	6	2.17	161	0.79	0.39	4
Koshal2	14	0.80	1.01	1.01	1.00	11	1.82	1703	0.56	0.32	4
U(36)	48	0.88	0.99	1.00	0.99	11	2.18	437	0.34	0.22	5
U(20)	16	0.95	1.00	1.00	0.99	13	1.85	250	0.28	0.19	4
U-A(36)	41	0.88	1.01	1.01	1.00	13	2.38	440	0.34	0.23	5
U-A(20)	16	0.92	1.01	1.01	1.01	14	1.71	258	0.30	0.22	5
Hamm-A(36)	39	0.80	1.02	1.01	1.01	13	2.54	454	0.55	0.33	3
Hamm-A(20)	17	0.86	1.02	1.01	1.01	11	2.09	263	0.38	0.31	3

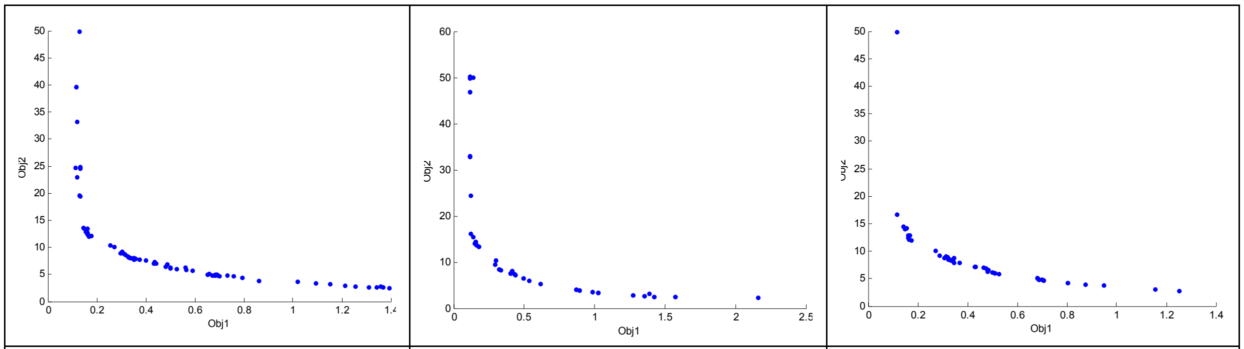
The Disk Brake results are shown in Table 4.6.4. A difference in the spread metrics among designs is clear. Looking at Figure 4.6.3, Dopt(20) has better extreme points in both objectives in only 20 points sampled, while also having a good distribution of points. Hamm(36) had only one point near the extreme of Objective 2, while FF(3), with its abundance of runs, managed to do as well as Dopt(20) in Objective 2, but not in Objective 1.



(a) FF(3) (b) LatinLC(36) (c) U(20)

Figure 4.6.2: Designs for the Dias Γ2 Problem

Some of what is seen in Disk Brake can be attributed to randomness. As expected, Latin Hypercubes and other designs performed well with respect to the distribution of Pareto points. However, these designs did not generate the extremes in either objective. In fact, they often yielded values between 0 and 1.4 in Objective 1, and between 0 and 20 in Objective 2.



(a) FF(3) (b) Dopt(20) (c) Hamm(36)

Figure 4.6.3: Designs for the Disk Brake Problem

DTLZ7 results are shown in Table 4.6.5. The CCD designs performed similarly (keeping in mind the spread values were affected by dominated, or bogus, points). However, the CCD(I) takes much less time to complete. Furthermore, the Hammersley and uniform designs again perform well, as does Latin Hypercube sampling. The reader should note that the OA designs, Hybrid, SCD, and Koshal designs are dominated by the other designs. Figure 4.6.4 depicts the full-factorial, inscribed CCD, and U-A(36) designs. Note the U-A design is clearly competitive with the full-factorial design. Also the full factorial design plot has a different scale in Objective 2 due to dominated points being retained because of noise in Objective 1.

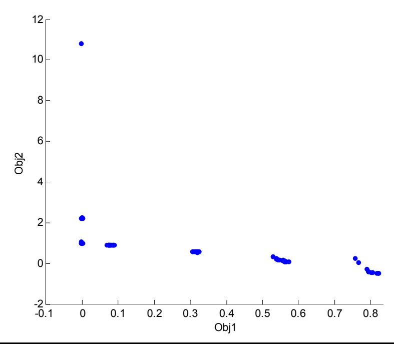
Table 4.6.4: Designs for Disk Brake

Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	89	0.89	0.47	0.48	0.99	15	4.87	4837	10.15	6.83	5
CCD(C)	26	0.81	0.17	0.47	0.36	10	4.60	806	1.37	1.37	1
CCD(I)	28	0.76	0.05	0.22	0.22	6	7.33	272	0.00	0.00	0
CCD(F)	27	0.84	0.14	0.48	0.29	9	5.00	613	0.00	0.00	0
BB	15	0.81	0.13	0.47	0.28	9	4.33	203	1.09	1.09	1
LatinR(36)	22	0.81	0.13	0.40	0.32	9	5.56	264	0.00	0.00	0
LatinL(36)	25	0.80	0.11	0.36	0.31	8	5.88	269	0.00	0.00	0
LatinRC(36)	22	0.80	0.18	0.44	0.40	9	5.56	269	5.33	3.57	2
LatinLC(36)	24	0.79	0.09	0.37	0.25	8	6.00	261	0.00	0.00	0
OA(Multi)	5	0.77	0.04	0.25	0.17	6	2.17	65	0.00	0.00	0
OA	1	0.78	0.06	0.24	0.24	6	2.50	60	4.03	4.03	1
Hamm(36)	25	0.81	0.42	0.43	0.98	9	5.22	584	33.23	33.23	1
Hamm(20)	6	0.81	0.33	0.38	0.86	8	4.25	449	26.70	26.70	1
Dopt(20)	32	0.95	0.76	0.76	1.00	16	2.13	5329	13.89	7.79	4
Hybrid	13	0.83	0.11	0.41	0.27	7	3.00	481	3.02	3.02	1
SCD	6	0.82	0.12	0.40	0.31	9	3.56	469	0.00	0.00	0
Koshal1	2	0.73	0.03	0.17	0.18	4	3.00	53	3.65	3.65	1
Koshal1+	8	0.73	0.02	0.16	0.14	4	4.50	117	0.00	0.00	0
Koshal2	8	0.74	0.04	0.21	0.18	6	4.33	128	0.00	0.00	0
U(36)	24	0.81	0.11	0.39	0.29	9	5.33	275	0.00	0.00	0
U(20)	7	0.81	0.09	0.34	0.27	7	4.71	145	0.00	0.00	0
U-A(36)	17	0.83	0.13	0.47	0.29	9	6.11	263	0.00	0.00	0
U-A(20)	9	0.84	0.13	0.47	0.27	8	3.88	153	0.00	0.00	0
Hamm-A(36)	26	0.82	0.13	0.46	0.28	9	5.11	270	0.00	0.00	0
Hamm-A(20)	5	0.81	0.12	0.43	0.29	10	3.50	151	0.00	0.00	0

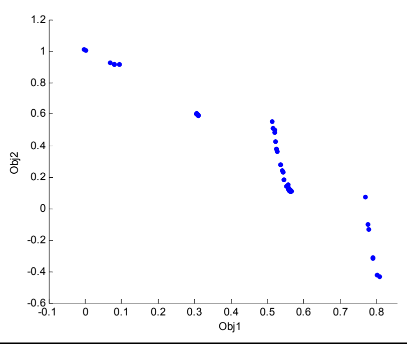
The results for the Fonseca F1 problem are shown in Table 4.6.6. General design performance trends repeated. However, the CCD(I) , Latin Hypercube designs, Hammersley designs, and uniform designs truly outperformed the full factorial design. To illustrate this, FF(3), Hamm(36) and U-A(36) are shown in Figure 4.6.5, where the space-filling designs filled gaps that the full-factorial design did not, and uniformly distributed points along the Pareto front.

Table 4.6.5: Designs for DTLZ7

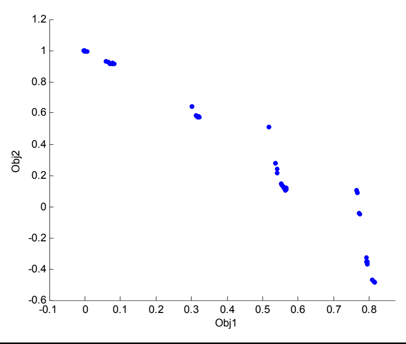
Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	93	0.95	7.72	1.01	7.65	14	4.93	7764	8.57	1.83	6
CCD(C)	37	0.95	1.22	1.01	1.21	12	2.92	2480	0.34	0.25	5
CCD(I)	23	0.92	0.97	0.99	0.97	11	4.45	371	0.37	0.26	3
CCD(F)	41	0.92	3.58	1.01	3.54	13	2.38	2744	1.96	0.89	6
BB	24	0.95	3.04	0.99	3.06	9	3.33	1621	2.96	1.02	4
LatinR(36)	14	0.94	0.92	0.98	0.94	10	5.80	96	0.34	0.26	4
LatinL(36)	32	0.95	1.04	1.12	0.93	11	3.64	96	0.53	0.31	5
LatinRC(36)	28	0.95	0.76	0.90	0.84	9	4.89	95	0.37	0.27	3
LatinLC(36)	21	0.95	0.93	0.98	0.95	10	5.10	95	0.35	0.26	3
OA(Multi)	3	0.97	0.72	0.95	0.75	8	1.88	24	0.38	0.27	3
OA	1	0.94	0.69	0.86	0.80	6	2.50	23	0.38	0.28	4
Hamm(36)	19	0.94	4.28	0.98	4.38	11	4.82	410	5.09	1.48	4
Hamm(20)	13	0.94	3.97	0.95	4.17	8	3.38	341	4.99	1.20	5
Dopt(20)	33	0.95	1.05	1.00	1.05	10	3.30	4099	0.50	0.37	4
Hybrid	16	0.90	2.64	0.99	2.67	9	2.00	1178	2.48	0.79	5
SCD	9	0.91	1.37	0.98	1.39	10	2.90	1184	0.59	0.43	4
Koshal1	2	0.95	0.82	0.89	0.92	5	2.40	19	0.40	0.36	4
Koshal1+	4	0.86	0.83	0.90	0.92	6	3.67	33	0.60	0.47	3
Koshal2	7	0.90	5.08	0.98	5.18	8	3.38	347	6.25	1.82	4
U(36)	22	0.95	0.91	0.94	0.97	9	5.56	101	0.28	0.25	3
U(20)	8	0.97	0.80	0.87	0.92	9	3.56	53	0.29	0.26	3
U-A(36)	28	0.97	1.00	1.00	1.00	11	4.00	97	0.35	0.26	4
U-A(20)	8	0.96	0.99	0.99	1.00	10	3.20	53	0.35	0.30	3
Hamm-A(36)	24	0.98	0.99	1.00	1.00	11	4.36	93	0.25	0.24	3
Hamm-A(20)	10	0.96	0.84	0.96	0.87	6	5.00	52	0.39	0.32	3



(a) FF(3)



(b) CCD(I)

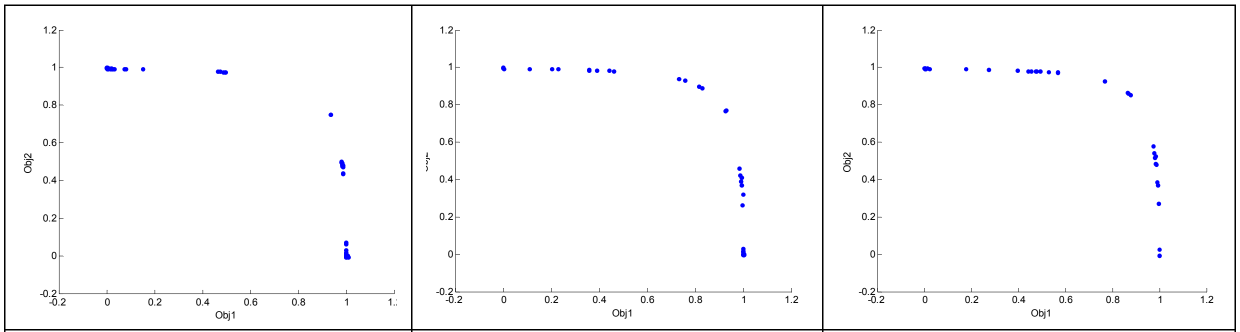


(c) U-A(36)

Figure 4.6.4: DTLZ7 Designs

Table 4.6.6: Designs for Fonseca F1

Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	104	0.88	1.02	1.01	1.01	12	4.83	9040	0.49	0.36	4
CCD(C)	43	0.91	1.01	1.01	1.00	11	2.64	5063	0.44	0.33	5
CCD(I)	41	0.94	1.00	1.00	1.00	13	2.38	289	0.38	0.21	6
CCD(F)	42	0.82	1.00	1.00	1.00	8	3.75	4704	0.69	0.49	3
BB	28	0.86	1.01	1.00	1.00	9	2.89	4414	0.66	0.48	3
LatinR(36)	33	0.93	1.00	1.00	1.00	11	3.55	286	0.35	0.25	4
LatinL(36)	32	0.94	1.00	1.00	1.00	14	2.86	280	0.42	0.25	5
LatinRC(36)	33	0.96	1.00	1.00	1.00	11	3.55	287	0.36	0.29	4
LatinLC(36)	36	0.94	1.01	1.00	1.00	15	2.40	286	0.31	0.19	6
OA(Multi)	8	0.88	0.99	0.99	1.00	5	2.00	71	0.56	0.41	4
OA	5	0.89	0.97	0.99	0.98	6	1.83	63	0.52	0.40	4
Hamm(36)	37	0.97	1.01	1.00	1.00	15	2.33	961	0.31	0.20	6
Hamm(20)	20	0.96	1.00	1.00	1.00	12	1.67	910	0.29	0.23	6
Dopt(20)	44	0.93	1.01	1.01	1.00	11	2.00	6743	0.45	0.28	6
Hybrid	16	0.91	1.01	1.00	1.01	9	2.00	2937	0.47	0.33	5
SCD	13	0.88	1.01	1.01	1.00	13	1.92	2597	0.53	0.25	6
Koshal1	5	0.85	0.99	0.99	1.00	5	1.80	58	0.70	0.52	3
Koshal1+	6	0.84	1.00	1.00	1.00	7	2.86	201	0.50	0.40	4
Koshal2	16	0.85	1.00	1.00	1.00	7	2.57	167	0.67	0.52	3
U(36)	40	0.96	1.00	1.00	1.00	13	2.46	286	0.38	0.27	4
U(20)	13	0.96	1.00	1.00	1.00	10	2.70	161	0.41	0.36	4
U-A(36)	37	0.97	1.00	1.00	1.00	15	2.33	302	0.29	0.20	5
U-A(20)	15	0.97	1.00	1.00	1.00	12	2.08	165	0.39	0.28	5
Hamm-A(36)	41	0.91	1.00	1.00	1.00	11	2.82	293	0.51	0.26	5
Hamm-A(20)	17	0.86	1.01	1.01	1.00	11	2.09	159	0.64	0.36	4



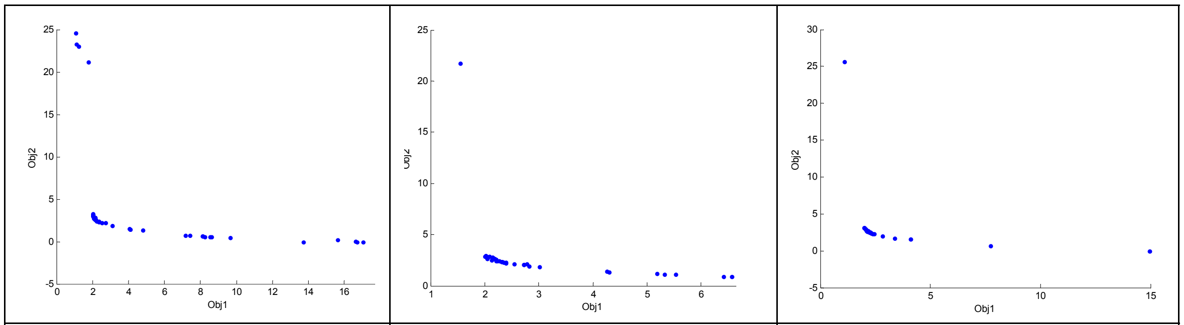
(a) FF(3) (b) Hamm(36) (c) U-A(36)

Figure 4.6.5: Fonseca F1 Design Comparison

The results for the Poloni problem are shown in Table 4.6.7. Other designs, such as the D-Optimal and Hybrid, again outperformed the full-factorial design. Many of the designs had trouble finding the points near the maximum objective function values for both objectives. Only the Dopt(20) design performed well, although the space-filling designs showed promise. The Hybrid and Hamm-A(36) designs each only had one point near the maximum in Objective 2, although Hamm-A(36) also had one point near the maximum in Objective 1. These three designs are shown in Figure 4.6.6.

Table 4.6.7: Designs for Poloni

Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	97	0.64	0.12	1.04	0.12	7	9.29	8580	5.72	5.72	1
CCD(C)	37	0.61	0.11	0.91	0.12	5	7.00	3216	7.92	7.92	1
CCD(I)	40	0.47	0.00	0.08	0.05	1	32.00	309	0.00	0.00	0
CCD(F)	39	0.58	0.11	0.95	0.12	5	6.60	4299	6.73	6.73	1
BB	22	0.61	0.11	0.95	0.11	5	6.40	2390	7.66	7.66	1
LatinR(36)	34	0.56	0.03	0.36	0.08	3	12.67	309	0.00	0.00	0
LatinL(36)	37	0.51	0.02	0.27	0.09	2	17.50	301	0.00	0.00	0
LatinRC(36)	32	0.54	0.02	0.27	0.08	2	20.00	309	0.00	0.00	0
LatinLC(36)	38	0.53	0.02	0.27	0.08	2	17.00	294	0.00	0.00	0
OA(Multi)	4	0.52	0.01	0.18	0.08	2	7.00	81	0.00	0.00	0
OA	4	0.48	0.00	0.09	0.05	1	12.00	70	0.00	0.00	0
Hamm(36)	45	0.53	0.02	0.23	0.07	2	13.50	310	0.00	0.00	0
Hamm(20)	14	0.51	0.10	0.90	0.11	3	8.67	529	11.46	11.46	1
Dopt(20)	27	0.84	0.89	1.01	0.88	10	3.90	3520	17.94	11.02	2
Hybrid	10	0.61	0.80	0.88	0.91	5	4.80	1208	22.51	13.11	2
SCD	14	0.58	0.13	1.08	0.12	5	4.80	1262	3.87	3.87	1
Koshal1	5	0.46	0.00	0.05	0.04	1	9.00	61	0.00	0.00	0
Koshal1+	12	0.46	0.00	0.05	0.04	1	14.00	119	0.00	0.00	0
Koshal2	14	0.51	0.09	0.84	0.11	2	10.00	589	11.77	11.77	1
U(36)	38	0.59	0.21	0.30	0.71	4	8.50	308	17.92	17.92	1
U(20)	16	0.52	0.02	0.24	0.07	2	12.00	173	0.00	0.00	0
U-A(36)	32	0.60	0.22	0.31	0.73	4	10.00	313	18.25	18.25	1
U-A(20)	14	0.56	0.03	0.34	0.09	3	8.67	178	0.00	0.00	0
Hamm-A(36)	37	0.61	0.23	0.32	0.74	4	8.75	308	18.82	18.82	1
Hamm-A(20)	18	0.53	0.02	0.25	0.08	2	11.00	172	0.00	0.00	0



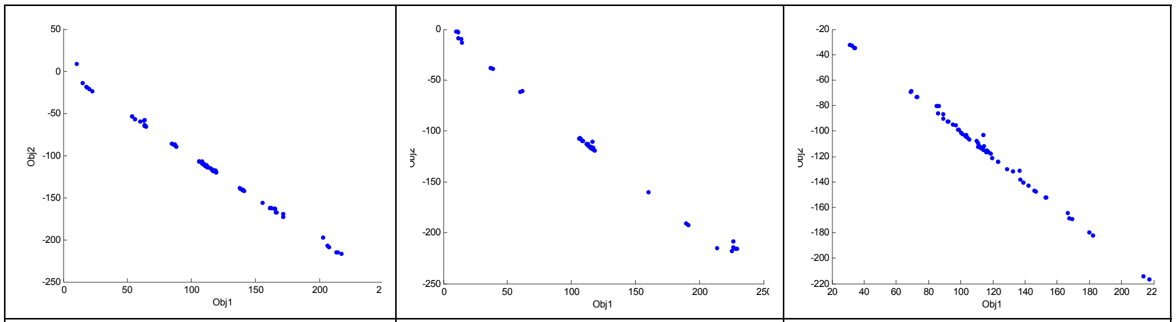
(a) Dopt(20) (b) Hybrid (c) Hamm-A(36)

Figure 4.6.6: Poloni Designs

The results for the Srinivas problem are shown in Table 4.6.8. Latin Hypercube sampling did not do as well in terms of finding extreme values. Dopt(20) performed well again in terms of spreads and entropy, however, in looking at the raw data, it could be seen that this was mainly a result of the added axial points, which corresponded to most of the more extreme values. The same extreme values were those missing from U-A(36) and Hamm-A(36), as these designs fill space and thus did not have levels at the edges of the design space. The inscribed CCD did not perform as well here. The other CCDs did generate good spreads and entropy, but points are still rather non-uniform on the front. FF(3), Dopt(20), and U-A(36) are shown in Figure 4.6.7. The full-factorial design did well likely in part due to its abundance of design levels.

Table 4.6.8: Designs for Srinivas

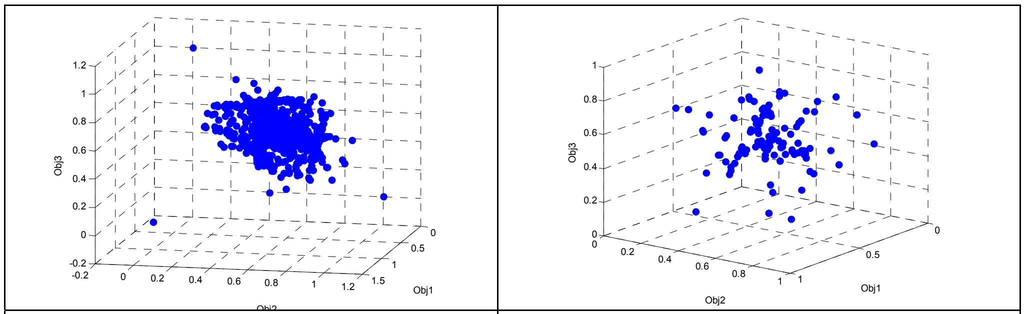
Design	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	77	0.94	0.99	0.96	1.03	12	7.08	1788	43.08	41.25	2
CCD(C)	27	0.85	0.83	0.92	0.90	8	5.63	548	62.90	60.77	3
CCD(I)	22	0.83	0.49	0.71	0.69	8	6.25	492	39.57	39.57	1
CCD(F)	25	0.83	0.90	0.96	0.94	7	6.71	573	70.08	61.56	4
BB	14	0.83	0.22	0.46	0.47	6	6.67	394	0.00	0.00	0
LatinR(36)	13	0.82	0.46	0.69	0.66	12	4.92	518	36.13	36.13	1
LatinL(36)	7	0.91	0.64	0.81	0.79	13	5.00	513	0.00	0.00	0
LatinRC(36)	8	0.88	0.49	0.71	0.69	13	4.92	523	0.00	0.00	0
LatinLC(36)	9	0.90	0.64	0.81	0.79	13	4.85	512	0.00	0.00	0
OA(Multi)	6	0.83	0.21	0.46	0.44	6	2.00	132	51.99	46.39	2
OA	3	0.67	0.06	0.25	0.24	3	4.33	114	0.00	0.00	0
Hamm(36)	16	0.90	0.60	0.78	0.76	14	4.00	518	0.00	0.00	0
Hamm(20)	7	0.88	0.52	0.73	0.71	11	3.00	287	48.21	41.69	2
Dopt(20)	20	0.83	1.00	1.02	0.98	10	4.60	1017	63.94	55.04	3
Hybrid	4	0.86	0.47	0.71	0.66	8	3.75	244	62.42	53.17	2
SCD	7	0.89	0.56	0.77	0.73	8	3.88	285	68.35	68.35	1
Koshal1	1	0.79	0.20	0.46	0.45	4	3.25	104	61.44	60.18	2
Koshal1+	6	0.78	0.21	0.47	0.45	5	4.00	274	0.00	0.00	0
Koshal2	4	0.85	0.55	0.70	0.78	7	4.29	838	89.41	89.41	1
U(36)	7	0.89	0.59	0.78	0.76	13	5.00	525	35.82	35.55	2
U(20)	5	0.86	0.33	0.58	0.57	12	2.92	298	0.00	0.00	0
U-A(36)	13	0.89	0.73	0.86	0.84	13	4.54	523	48.89	46.96	2
U-A(20)	6	0.89	0.34	0.59	0.58	11	3.09	302	0.00	0.00	0
Hamm-A(36)	12	0.90	0.70	0.85	0.83	15	4.00	552	43.20	41.54	2
Hamm-A(20)	6	0.90	0.71	0.84	0.84	10	3.40	311	67.43	59.54	2



(a) FF(3) (b) Dopt(20) (c) U-A(36)

Figure 4.6.7: Srinivas

The results for the Tamaki problem are shown in Table 4.6.9. The full-factorial design appears to have performed best according to the metrics, but when looking at the raw data it appears to have generated extreme points by chance. In looking at the levels and their responses, the same levels do not necessarily correspond to the same objective function values and there is a large mssing portion of the Pareto front in every objective. Additionally, the run time was rather large compared to its counterparts. Dopt(40) performed well, generating a reasonable approximation, but here the extreme values did not correspond to the axials. The SCD's metrics can be attributed to one extreme point, and thus it did not do as well as many of the other designs. The Hamm-A(59) and U-A(40) designs performed better than the SCD but failed to generate the extreme values. It can be argued that the CCD(C) did even better than the full-factorial. The FF(3) and CCD(C) are shown in Figure 4.6.8, while Dopt(40) and Hamm-A(59) are shown in Figure 4.6.9.

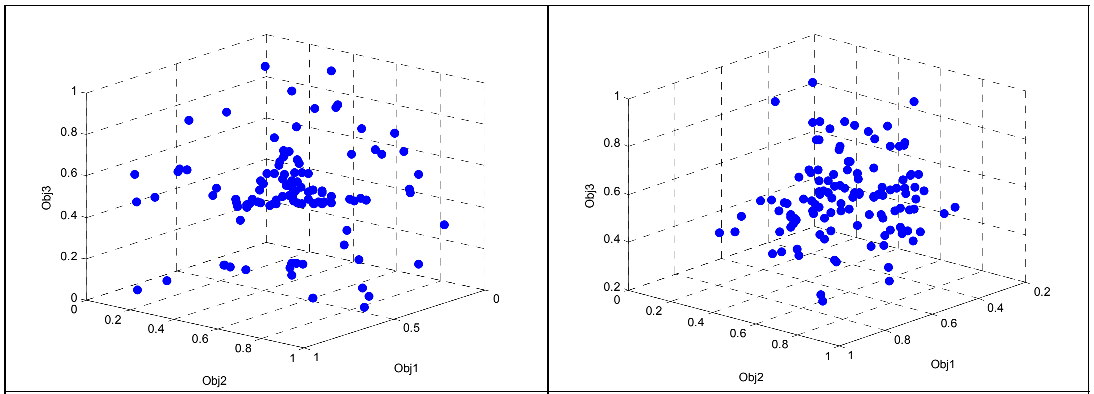


(a) FF(3) (b) CCD(C)

Figure 4.6.8: Tamaki Designs

Table 4.6.9: Designs for Tamaki

Design	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)	284	0.77	1.01	1.00	1.00	1.01	73	16.08	50490	0.77	0.51	8
CCD(C)	7	0.79	0.35	0.72	0.69	0.71	47	2.36	5276	0.22	0.18	4
CCD(I)	6	0.66	0.04	0.38	0.32	0.32	20	5.60	2660	0.00	0.00	0
CCD(F)	1	0.72	0.10	0.45	0.50	0.43	29	4.03	4262	0.23	0.23	1
BB	2	0.75	0.11	0.40	0.54	0.50	32	3.31	2526	0.13	0.13	1
Latin(R)	5	0.71	0.10	0.52	0.39	0.48	32	3.53	2238	0.23	0.19	2
Latin(L)	1	0.70	0.11	0.47	0.49	0.47	27	4.33	2580	0.16	0.15	2
Latin(RC)	2	0.72	0.11	0.39	0.52	0.56	32	3.63	2047	0.15	0.15	1
Latin(LC)	0	0.71	0.12	0.47	0.49	0.54	31	3.81	1827	0.19	0.19	3
OA(Multi)	2	0.71	0.09	0.43	0.44	0.47	27	4.67	2915	0.00	0.00	0
OA	12	0.67	0.08	0.48	0.45	0.37	24	6.17	2708	0.16	0.16	1
Hamm(59)	2	0.73	0.11	0.57	0.45	0.45	36	3.22	2169	0.18	0.18	1
Hamm(40)	0	0.72	0.13	0.57	0.46	0.50	30	2.67	1658	0.20	0.20	1
Dopt(40)	1	0.87	0.72	0.90	0.87	0.93	57	1.98	5762	0.38	0.20	15
Hybrid	2	0.82	0.17	0.62	0.51	0.53	31	1.74	2622	0.22	0.18	6
SCD	0	0.76	0.43	0.82	0.75	0.70	31	2.26	3315	0.84	0.84	1
Koshal1	0	0.68	0.03	0.32	0.29	0.29	12	1.50	419	0.00	0.00	0
Koshal1+	1	0.69	0.03	0.32	0.29	0.29	23	2.04	920	0.00	0.00	0
Koshal2	0	0.71	0.04	0.36	0.33	0.33	22	2.73	954	0.00	0.00	0
U(59)	0	0.71	0.04	0.36	0.33	0.33	22	2.73	954	0.00	0.00	0
U(40)	2	0.71	0.06	0.40	0.34	0.47	26	3.00	1811	0.00	0.00	0
U-A(59)	1	0.81	0.26	0.72	0.60	0.60	49	2.39	4111	0.30	0.18	5
U-A(40)	0	0.81	0.18	0.55	0.50	0.67	38	2.11	3062	0.13	0.13	1
Hamm-A(59)	0	0.80	0.29	0.64	0.66	0.69	45	2.62	4042	0.55	0.29	5
Hamm-A(40)	0	0.80	0.18	0.63	0.51	0.56	35	2.29	2427	0.21	0.17	2
MR5	1	0.79	0.17	0.50	0.57	0.60	34	2.09	2114	0.20	0.19	4



(a) Dopt(40) (b) Hamm-A(59)

Figure 4.6.9: More Tamaki Designs

For both Viennet3 and Viennet4, the full-factorial design with a limit of 50000 function evaluations was extremely time-consuming. Therefore, full-factorials for these two problems were run with only 500 function evaluations allowed on each design level.

The Viennet3 are shown in Table 4.6.10. Approximately 80% of the resulting solutions found were, in fact, dominated. In comparison, the CCD(C) only had approximately 42% dominated points. However, the full factorial did the best in terms of spread, and the reduction in time from only 500 function evaluations is extremely beneficial. Objective 1 proved to be elusive on this problem, with only the circumscribed CCD getting a single point near the maximum in that objective (outside of the full factorial), perhaps randomly. Designs with true axials nonetheless generated better extreme points in Objective 1. The CCD(C) and full factorial designs are shown in Figure 4.6.10.

Table 4.6.10: Designs for Viennet3

Design	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)*	1160	0.66	4.73	1.02	1.08	4.29	22	13.55	2183	7.91	3.18	5
CCD(C)	49	0.66	4.15	1.00	1.05	3.96	15	4.60	7616	6.49	2.66	3
CCD(I)	54	0.63	0.88	0.20	0.97	4.50	7	9.14	1005	1.46	1.13	2
CCD(F)	62	0.63	2.58	0.57	1.07	4.28	15	3.73	8756	2.79	1.48	3
BB	45	0.66	0.93	0.20	1.06	4.41	15	4.20	8031	1.10	1.03	2
Latin(R)	49	0.67	0.00	0.03	0.28	0.40	3	23.00	218	0.00	0.00	0
Latin(L)	43	0.66	0.00	0.03	0.27	0.42	4	18.75	219	0.00	0.00	0
Latin(RC)	58	0.65	0.00	0.03	0.26	0.39	4	15.00	219	0.00	0.00	0
Latin(LC)	43	0.66	0.00	0.03	0.32	0.39	4	18.75	219	0.00	0.00	0
OA(Multi)	65	0.66	0.00	0.03	0.29	0.38	3	21.00	235	0.00	0.00	0
OA	72	0.67	0.00	0.03	0.26	0.34	3	29.33	304	0.00	0.00	0
Hamm(59)	47	0.66	0.02	0.04	0.69	0.56	6	11.83	495	0.67	0.67	1
Hamm(40)	33	0.67	0.80	0.19	1.06	3.95	6	7.83	414	1.46	1.16	2
Dopt(40)	48	0.71	0.89	0.19	1.07	4.27	16	4.13	8440	0.77	0.57	2
Hybrid	20	0.70	0.94	0.20	1.06	4.35	13	2.77	4039	0.98	0.87	2
SCD	27	0.64	1.03	0.23	1.06	4.29	12	3.58	4535	1.53	1.20	2
Koshal1	6	0.62	0.00	0.02	0.15	0.30	3	4.00	33	0.00	0.00	0
Koshal1+	16	0.65	0.00	0.02	0.17	0.31	3	10.67	89	0.00	0.00	0
Koshal2	28	0.63	0.02	0.04	0.73	0.52	7	4.57	962	1.10	1.10	1
U(59)	28	0.63	0.02	0.04	0.73	0.52	7	4.57	962	1.10	1.10	1
U(40)	32	0.67	0.00	0.03	0.28	0.43	4	12.00	146	0.00	0.00	0
U-A(59)	57	0.71	0.01	0.03	0.37	0.52	4	15.25	222	0.00	0.00	0
U-A(40)	27	0.70	0.01	0.04	0.41	0.55	5	10.60	150	0.00	0.00	0
Hamm-A(59)	48	0.70	0.01	0.04	0.37	0.54	5	14.00	215	0.00	0.00	0
Hamm-A(40)	30	0.70	0.01	0.04	0.41	0.56	4	12.50	150	0.00	0.00	0
MR5	30	0.66	2.36	0.52	1.05	4.35	12	3.50	5049	2.86	1.60	3

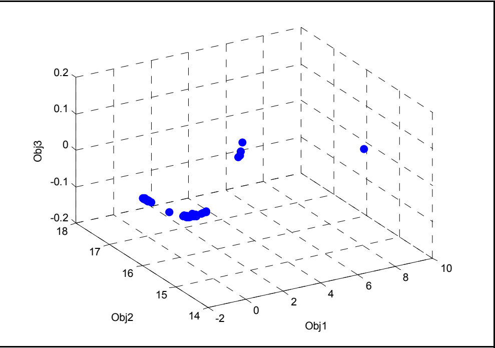
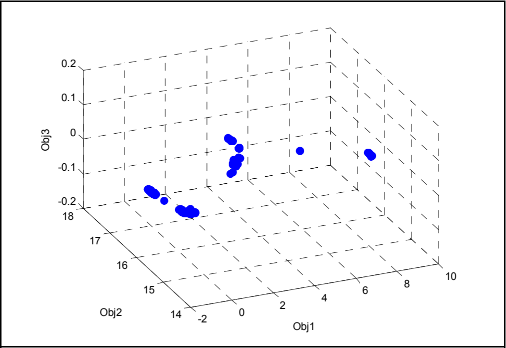
	
(a) CCD(C)	(b) Full Factorial

Figure 4.6.10: Viennet3 Results

Viennet4 results are shown in Table 4.6.11. Approximately 70% of the full factorial points (with 500 function evaluation limit) were dominated, versus 27% for the CCD(C). The full factorial, CCD(C), Dopt(40), and MR5 designs performed well. As shown in Figure 4.6.11, the full factorial design had a general area where no points were found, while the CCD points were more spread out, and the D-optimal design was not necessarily clustered in any region. The uniform and Hammersley designs did not perform as well in the three objective problems, but this may be in part because they did not include points at the exact extremes of the aspiration and reservation levels, or at the exact axials, and therefore, a change in the range over which they are conducted may improve their results (and did, as will be shown in Section 4.10).

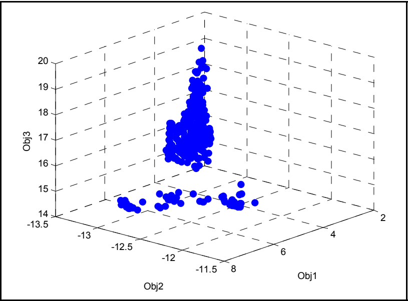
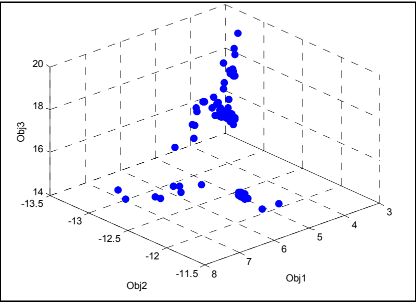
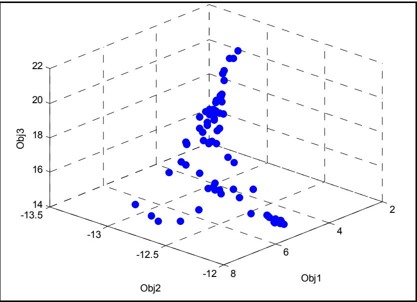
		
(a) Full-Factorial	(b) CCD(C)	(c) D-Optimal

Figure 4.6.11: Viennet4 Results

In conclusion, the D-optimal, CCD(C), Hammersley, and uniform designs appear to be good alternatives to a full-factorial for the initial design. The CCD(C) is representative, as no CCD type emerged conclusively better than another in all cases. However, the D-optimal designs were accidentally run using an entirely different range. This modified range, in conjunction with 3 levels, performed extremely well across most problems. This motivates Section 4.10.

Table 4.6.11: Designs for Viennet4

Design	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
FF(3)*	1016	0.83	0.86	0.92	1.41	0.67	53	8.34	2263	0.65	0.63	2
CCD(C)	32	0.80	1.11	0.93	1.69	0.70	28	3.07	3582	0.51	0.50	2
CCD(I)	30	0.79	0.20	0.39	1.22	0.43	15	5.87	746	1.68	1.68	1
CCD(F)	39	0.79	0.58	0.89	1.30	0.50	24	3.29	7434	1.01	0.77	2
BB	22	0.82	0.56	0.86	1.31	0.50	27	3.19	6123	1.57	1.21	2
Latin(R)	24	0.82	0.07	0.33	0.50	0.43	18	5.22	193	0.00	0.00	0
Latin(L)	23	0.81	0.07	0.33	0.45	0.50	15	6.33	197	0.00	0.00	0
Latin(RC)	18	0.83	0.08	0.30	0.60	0.41	21	4.76	197	0.00	0.00	0
Latin(LC)	22	0.82	0.06	0.33	0.44	0.42	18	5.33	194	0.00	0.00	0
OA(Multi)	36	0.83	0.06	0.29	0.61	0.37	20	4.60	214	0.00	0.00	0
OA	43	0.82	0.04	0.27	0.48	0.29	14	8.36	266	0.00	0.00	0
Hamm(59)	22	0.82	0.23	0.40	1.32	0.43	20	4.80	498	1.20	1.20	1
Hamm(40)	10	0.83	0.17	0.53	0.70	0.46	17	4.12	432	1.09	1.09	1
Dopt(40)	37	0.87	1.20	0.97	1.30	0.95	34	2.26	7565	1.17	0.84	3
Hybrid	13	0.81	0.67	0.87	1.30	0.59	17	2.53	3436	0.95	0.78	2
SCD	13	0.81	0.59	0.76	1.33	0.58	22	2.59	4348	0.78	0.78	1
Koshal1	2	0.77	0.02	0.26	0.27	0.27	5	3.20	33	0.62	0.62	1
Koshal1+	11	0.80	0.04	0.33	0.35	0.37	14	2.64	81	0.00	0.00	0
Koshal2	17	0.78	0.44	0.74	1.28	0.47	12	3.58	673	1.48	1.48	1
U(59)	17	0.78	0.44	0.74	1.28	0.47	12	3.58	673	1.48	1.48	1
U(40)	14	0.83	0.08	0.31	0.56	0.48	19	3.47	136	0.00	0.00	0
U-A(59)	24	0.86	0.21	0.46	0.81	0.56	26	3.62	200	0.00	0.00	0
U-A(40)	8	0.86	0.19	0.42	0.83	0.55	22	3.27	135	0.00	0.00	0
Hamm-A(59)	13	0.86	0.18	0.48	0.72	0.51	24	4.38	193	0.00	0.00	0
Hamm-A(40)	12	0.85	0.16	0.45	0.66	0.52	22	3.09	133	0.00	0.00	0
MR5	15	0.80	0.86	0.90	1.31	0.72	23	2.48	4429	1.84	1.52	2

Up to this point in the analysis, a Hammersley or uniform design, in conjunction with axials and/or AR3 type range, appeared to outperform everything else in approximating the Pareto front, saving a large amount of time and runs. In fact, for a larger number of objectives, these space-filling designs make SMOMADS tractable, in terms of the number of required runs (recall the number of factors is two times the number of objectives). Further, these designs provide uniform points along the Pareto front, which is desirable and would be expected if MADS did not have its random element and no noise was present. Unfortunately, since this is not the case and although uniform fronts appear, it cannot necessarily be said that a certain set of levels will give a specific point on the Pareto front. The axials are necessary as they sometimes force the algorithm to find the more extreme values.

4.7. Limiting Function Evaluations

Clearly, the number of function evaluations within MADS-RS/GPS-RS affects the run-time of SMOMADS. However, the effect of limiting these evaluations may or may not result in premature termination at a poor solution, and additional replications may be necessary to compensate. A few initial runs of the DTLZ7, Disk Brake, and Viennet4 problems were conducted to look at this. The AR1 type range was used. The Tamaki problem was then run using the CCD/Near Uniform design combination suggested by results from Section 4.10 to show the number of function evaluations used during each design level when having a limit of 500, to see if all 500 evaluations were being used.

First, the Viennet4 problem was run using a full factorial design with three levels, using both two and four replications, and with a limit of 500 function evaluations. Of course, there is no comparison to a 50000 function evaluation result, as this was extremely computationally expensive. As shown in Table 4.7.1, the additional two replications provided no benefit. In fact, many of the points were redundant. The graphs of these points supported this finding, but are not shown.

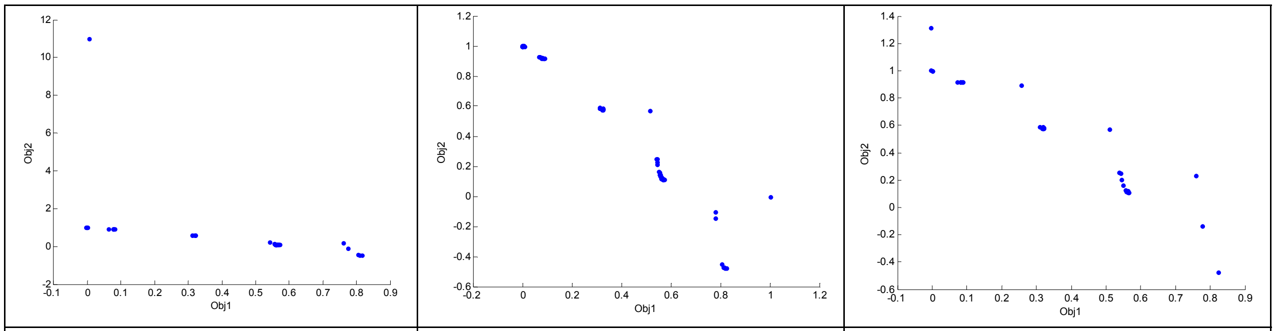
Table 4.7.1: Viennet4 Full Factorial

Reps	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
2	1016	0.83	0.86	0.92	1.41	0.67	53	8.34	2263	0.65	0.63	2
4	2221	0.83	0.75	0.91	1.40	0.59	61	11.39	4549	0.61	0.57	2

The DTLZ7 problem was run using the 500 function evaluation limit, CCD(C) and two and five replications. In Table 4.7.2, a 50000 evaluation run is in italics. Figure 4.7.1 also contains the plots of all three solutions. Ignoring the obvious outlier in the first approximation (an overall spread of 7.78), five replications provided little improvement over two (as seen with 50000 evaluations in Section 4.5 and Appendix A), and furthermore, 50000 evaluations does not seem to provide much of an advantage over using 500, other than reducing the clustering. Obviously, the improvement in run time after limiting evaluations is also beneficial.

Table 4.7.2: DTLZ7 CCD

Reps	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
2	28	0.90	7.78	1.00	7.76	10	4.4	104	10.00	2.28	5
5	115	0.92	1.23	1.22	1.00	13	5	263	0.39	0.28	4
2	*37*	*0.95*	*1.22*	*1.01*	*1.21*	*12*	*2.92*	*2480*	*0.34*	*0.25*	*5*



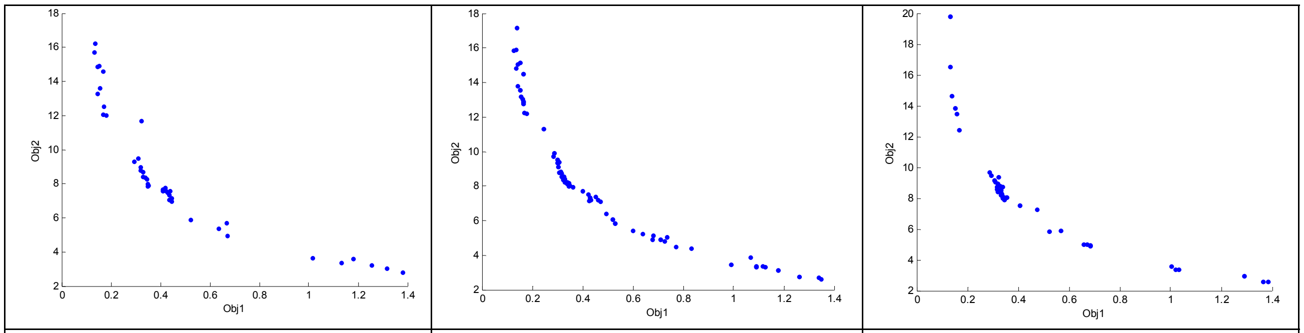
(a) 2 Reps	(b) 5 Reps	(c) 2 Reps, 50,000 Fn Evals

Figure 4.7.1: DTLZ7 Comparison

Finally, the Disk Brake problem was run similarly to DTLZ7, with results shown in Table 4.7.3. Again, using a limit of 500 evaluations does not seem to affect the results. Here, however, the five replications do provide a slight advantage, as visually depicted in Figure 4.7.2.

Table 4.7.3: Disk Brake CCD

Reps	Bogus	Entropy	OS	OS1	OS2	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
2	30	0.83	0.13	0.47	0.28	9	4.67	119	1.39	1.39	1
5	107	0.83	0.14	0.46	0.30	9	8.11	300	0.00	0.00	0
2	*26*	*0.81*	*0.17*	*0.47*	*0.36*	*10*	*4.60*	*806*	*1.37*	*1.37*	*1*

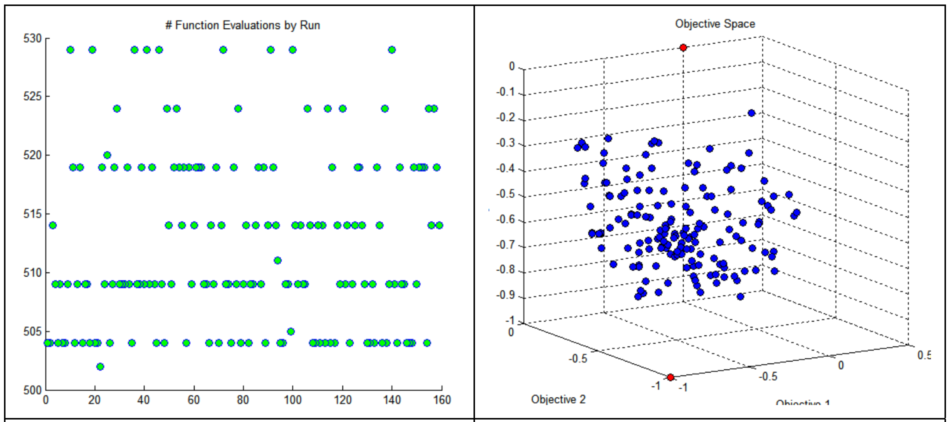


(a) 2 Reps (b) 5 Reps (c) 2 Reps, 50,000 Fn Evals

Figure 4.7.2: Disk Brake Comparison

Figure 4.7.3 shows a Pareto approximation for the Tamaki problem (b) and the corresponding number of function evaluations used for those points (a). The approximation is good, and clearly all 500 evaluations were consistently used for each design level (or sub-problem in SMOMADS).

In conclusion, a limit of 500 function evaluations appears to be worth the savings in computational time, and using two replications may be just as advantageous as using more.

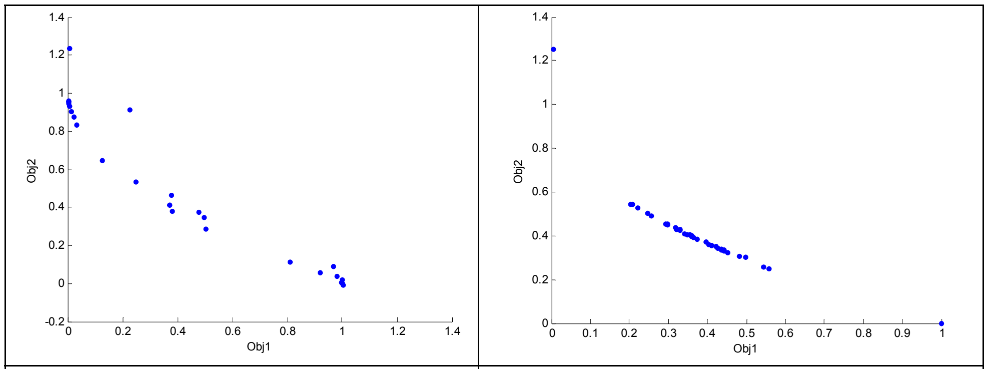


(a) # Function Evaluations (b) Pareto Approximation

Figure 4.7.3: Tamaki

4.8. MADS-RS vs. GPS-RS on Linearly Constrained Problems

As both GPS and MADS can be used on linearly constrained problems (here the sub-problems in SMOMADS), a comparison seemed warranted. A different design was chosen for each linearly constrained problem in the test set, and run with a 50000 function evaluation limit, two replications, 0.5% noise, and the aspiration and reservation ranges presented at the conclusion of Section 4.6. The results are shown in Table 4.8.1, with previous MADS-RS results italicized. The qualities of the front approximations are similar and GPS-RS is either much faster or comparable, except in the case of the Dias Γ1 and Dias Γ2 problems. The approximate Pareto fronts for these two problems are shown in Figure 4.8.1 and Figure 4.8.2. GPS-RS took considerably longer on both problems, but also resulted in far fewer dominated points. As can be seen for Hamm(20)/Dias Γ1 in Figure 4.8.1, the GPS-RS result appears to have a worse distribution of points on the front. However, those points in the MADS-RS solutions not near the center of the front may be in part due to noise. Dias Γ2 helps confirm this, as the GPS-RS solution shown in Figure 4.8.2 has a better distribution, with those levels that previously corresponded to dominated points helping to fill gaps along the front. On the remaining problems, GPS-RS is faster and converges to the same quality of solution as MADS-RS.

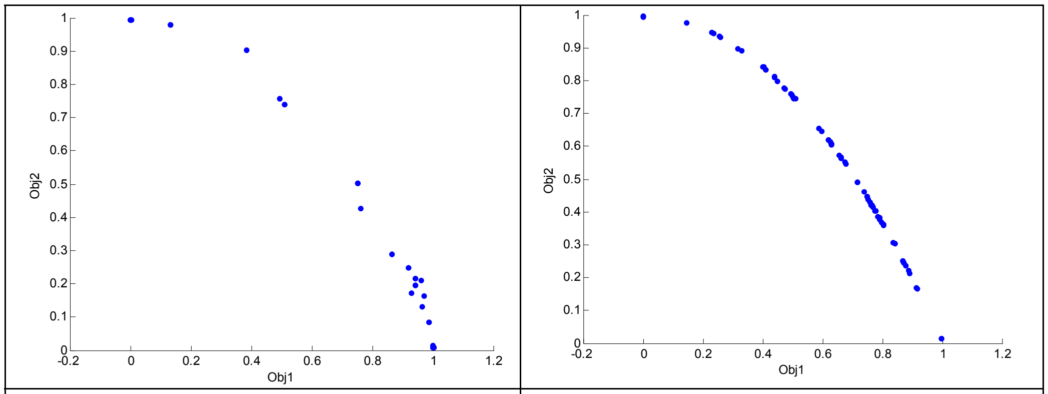


(a) Dias Γ1 MADS-RS Hamm(20) (b) Dias Γ1 GPS-RS Hamm(20)

Figure 4.8.1: MADS-RS vs. GPS-RS Dias Γ1

Table 4.8.1: MADS-RS vs. GPS-RS

Problem/ Design	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
Dias Γ1/ Hamm(20)	*13*	*0.92*	*1.24*	*1.00*	*1.24*	-	*13*	*2.08*	*371*	*0.35*	*0.21*	*7*
Dias Γ1/ Hamm(20)	3	0.73	1.25	1.00	1.25	-	8	4.63	999	0.73	0.62	2
Dias Γ2/ U(36)	*48*	*0.88*	*0.99*	*1.00*	*0.99*	-	*11*	*2.18*	*437*	*0.34*	*0.22*	*5*
Dias Γ2/ U(36)	8	0.97	0.98	1.00	0.98		16	4	1083	0.17	0.16	2
DTLZ7/ CCD(C)	*37*	*0.95*	*1.22*	*1.01*	*1.21*	-	*12*	*2.92*	*2480*	*0.34*	*0.25*	*5*
DTLZ7/ CCD(C)	43	0.90	1.01	1.01	1.00		9	3.22	1893	0.64	0.38	5
Fonseca F1/ BB	*28*	*0.86*	*1.01*	*1.00*	*1.00*	-	*9*	*2.89*	*4414*	*0.66*	*0.48*	*3*
Fonseca F1/ BB	25	0.83	1.01	1.01	1.00	-	10	2.9	1699	0.49	0.39	4
Viennet4/ U(40)	*14*	*0.83*	*0.08*	*0.31*	*0.56*	*0.48*	*19*	*3.47*	*136*	*0.00*	*0.00*	*0*
Viennet4/ U(40)	16	0.82	0.04	0.28	0.48	0.32	13	4.92	198	0.00	0.00	0



(a) Dias Γ2 MADS-RS U(36) | (b) Dias Γ2 GPS-RS U(36)

Figure 4.8.2: MADS-RS vs. GPS-RS Dias Γ2

MADS-RS was still used for the majority of the test runs on these linearly constrained problems, excluding the final runs, with the intended caveat that results would likely improve with respect to time if using GPS-RS. This provided some needed consistency.

4.9. Using Combinations of Component Functions

The achievement scalarization function uses the minimum of all component achievement functions at a point as its response. In this section, the benefit of using two component achievement functions at-a-time for three-objective problems is explored. Using a circumscribed CCD, 50000 function evaluations, and two replications, all three three-objective problems were tested. The results follow in Table 4.9.1, where *Old* refers to the original CCD using all component achievement functions, the numbers in parentheses refer to the specific component achievement functions, and *Total* refers to three two-component approximations put together.

Table 4.9.1: Three Objective Results

Problem	Bogus	Entropy	OS	OS1	OS2	OS3	NDC	CL	Time	Largest Gap	Avg. Gap	# Gaps
Tamaki (Old)	7	0.79	0.35	0.72	0.69	0.71	47	2.36	5276	0.22	0.18	4
Tamaki (12)	22	0.68	0.11	0.46	0.47	0.51	22	4.36	4314	0.34	0.22	4
Tamaki (23)	26	0.69	0.13	0.50	0.44	0.58	23	4.00	7000	0.44	0.27	4
Tamaki (13)	27	0.65	0.04	0.44	0.22	0.40	17	5.35	3630	0.00	0.00	0
Tamaki (Total)	75	0.87	0.72	0.91	0.89	0.89	62	4.50	14944	0.38	0.26	10
Viennet3 (Old)	49	0.66	4.15	1.00	1.05	3.96	15	4.60	7616	6.49	2.66	3
Viennet3 (12)	60	0.22	4.46	0.92	1.08	4.51	21	2.76	24693	2.44	1.88	4
Viennet3 (23)	55	0.69	4.08	0.96	1.07	3.97	15	4.20	7031	7.26	3.04	5
Viennet3 (13)	85	0.25	0.00	0.01	0.16	0.02	6	5.50	6815	0.00	0.00	0
Viennet3 (Total)	200	0.70	4.68	0.96	1.08	4.52	28	5.50	38539	2.44	1.24	5
Viennet4 (Old)	32	0.80	1.11	0.93	1.69	0.70	28	3.07	3582	0.51	0.50	2
Viennet4 (12)	45	0.66	0.29	0.32	0.71	1.27	20	3.65	21081	1.50	1.50	1
Viennet4 (23)	55	0.76	0.66	0.90	1.28	0.57	17	3.71	24514	0.90	0.90	1
Viennet4 (13)	54	0.83	0.56	0.89	1.27	0.49	16	4.00	16601	1.50	1.13	3
Viennet4 (Total)	154	0.87	2.09	1.03	1.30	1.56	42	4.76	62195	1.50	0.97	3

Table 4.9.1 shows that using two component functions at-a-time focuses in on specific regions of the Pareto front, here for the Tamaki problem. In terms of spread and entropy, combining the three pairs of component functions may only provide marginal improvement over using all three functions at-a-time. Figure 4.9.1 shows the

corresponding plots for the Tamaki problem. Each pair of component functions focuses on a corresponding region, and excludes the center of the Pareto front. The best approximation to the front is therefore obtained by using all component functions simultaneously, allowing generation of the entire front with only one design. A limit of 500 function evaluations was also evaluated. Similar results followed, except that a better spread of points resulted in part because MADS-RS was not as accurate (which, in this case, was a good thing).

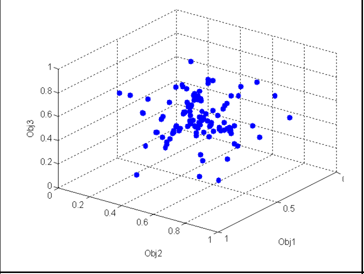
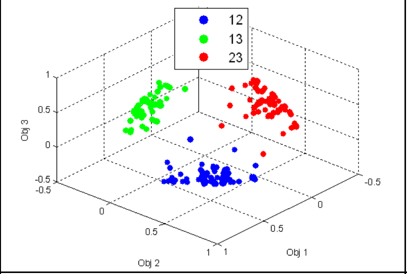
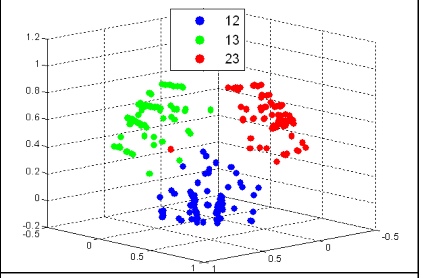
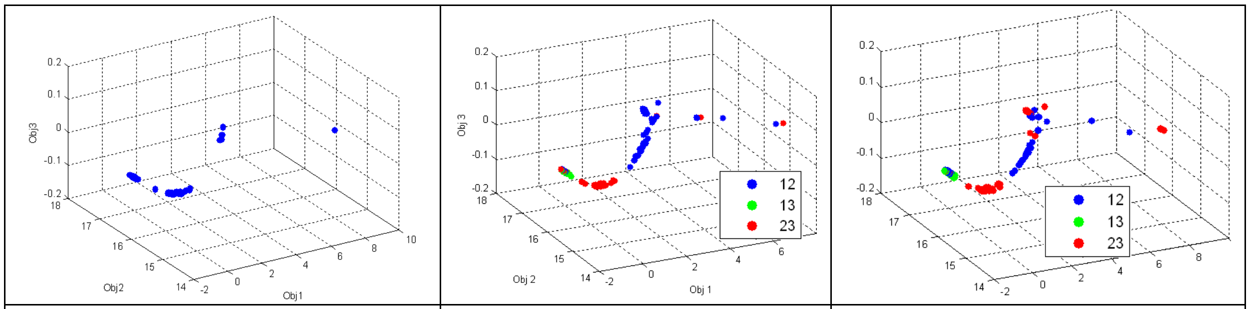
		
(a) All	(b) 2 at a Time	(c) 500 Fn Eval Limit

Figure 4.9.1: Tamaki

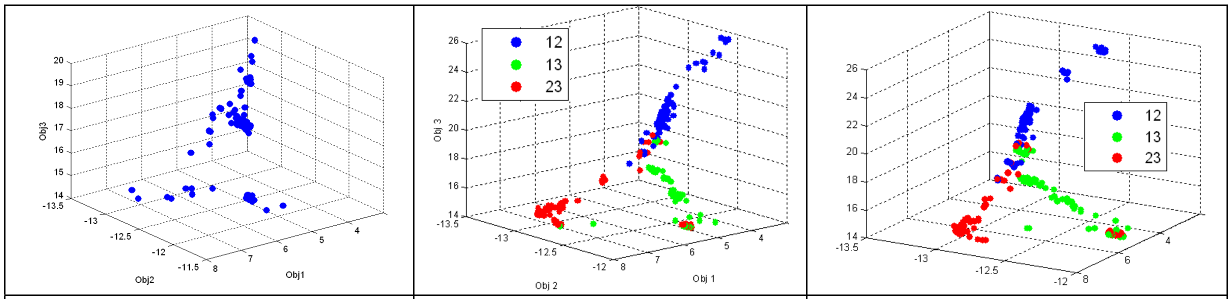
Approximate Pareto fronts for the Viennet3 problem are shown in Figure 4.9.2. In this case, using pairs does provide improvement. Portions of the Pareto front in Objective 1 and Objective 3 that were more difficult to get using all three component functions are found. However, if the three component run was replicated three times (same number of runs as using the three two at-a-time designs), a few points in those regions would likely be found, due to randomness, and so the advantage of using pairs may be less remarkable. Furthermore, in using 500 function evaluations again, nearly the same approximation results.

Finally, Figure 4.9.3 displays the resulting approximate Pareto fronts for Viennet4. Remarkably, using pairs provides almost no benefit, except for generating the extreme values in Objective 3. Both exclude the center portion of Objective 1 and Objective 2 on the Pareto front. The results shown here for Viennet4 can be more or less duplicated simply by selecting a correct range and design, using far fewer runs and all component functions. This was shown in Section 4.6. Again, looking at 500 function evaluations a nearly identical approximation is found.



(a) All	(b) 2 at a Time	(c) 500 Fn Eval Limit

Figure 4.9.2: Viennet3



(a) All	(b) 2 at a Time	(c) 500 Fn Eval Limit

Figure 4.9.3: Viennet4

Figure 4.9.4 shows a computed Pareto front using only the first component for the Viennet4 problem; *i.e.*, Pareto solutions with a minimum value in the first objective (maximum in the third).

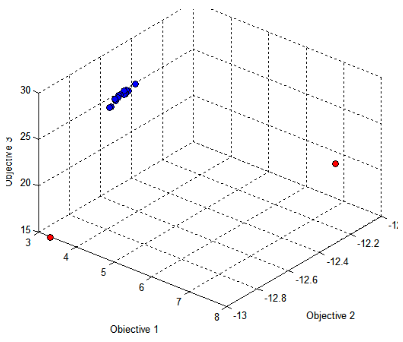


Figure 4.9.4: Viennet4(1)

Based solely on the three three-objective problems in this research, only on difficult regions like that in Viennet3 and its first objective is the pair-wise (or less-than-all) component method advantageous (in generating the Pareto front efficiently). This advantage is based mainly on the ability to generate points in specific regions. Otherwise, it is not a benefit because it is preferable to get a good front in fewer runs, and less expensive methods are able to fill any gaps that result. Of course, more objectives may increase this advantage. This method also appears to be useful if trying to estimate the utopia quickly, however, a design consisting of an optimization for each level was used here to get these regions of the Pareto fronts. Therefore, it is still faster to perform single-objective minimizations.

4.10. Final Aspiration/Reservation Level Range Analysis

4.10.1. Test Approach. Based on previous runs, further analysis on the aspiration and reservation ranges, as well as the effect of limiting function evaluations, was justified. Both CCD and near uniform designs were evaluated, where the near uniform design had an equivalent number of points to the CCD (36 for two objectives, 59 for three objectives) so as to have a valid comparison. These designs were chosen because they are two of the best ones found in Section 4.6, and because the CCD is a factorial-based design (like the full-factorial) and the near uniform is a space-filling design (similar to Hammersley sequence sampling). Asterisks in the results tables denote a limit of 500 function evaluations; otherwise a limit of 50000 was used.

New ranges were constructed with the success of the D-Optimal design from Section 4.6 in mind and the range it used. Here, Range 1 uses 0.495 of the difference between utopia and nadir components, both added and subtracted from a) the utopia point component, for the aspiration range, and b) the nadir point component, for the reservation range. Range 2 uses the utopia and nadir points as the bounds for the aspiration and reservation ranges. Range 3 uses the entire utopia and nadir point range in addition to subtracting 0.495 of the difference between points from the utopia component, for the aspiration range, and adding 0.495 of the difference between points, for the reservation range. To further clarify, these ranges are depicted in Figure 4.10.1, where the red points are the utopia and nadir points, respectively.

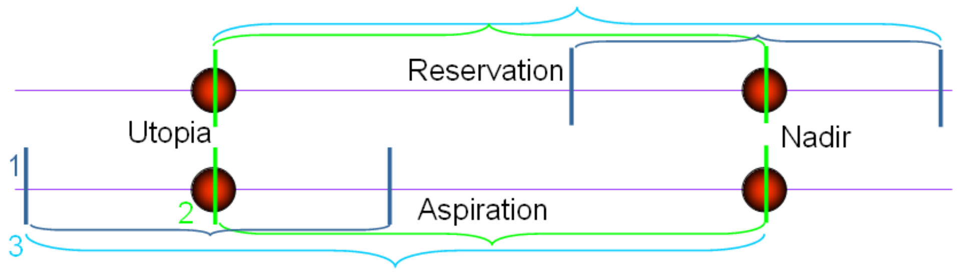


Figure 4.10.1: Aspiration and Reservation Level Ranges

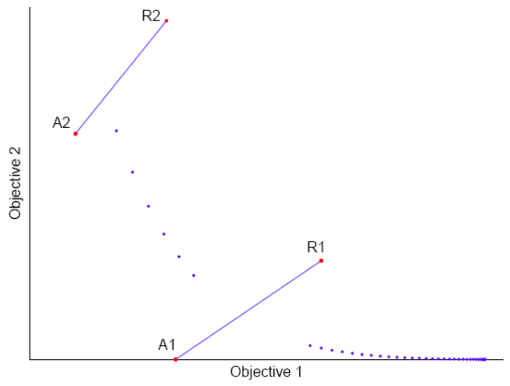


Figure 4.10.2: Aspiration and Reservation Levels Intersecting the Front

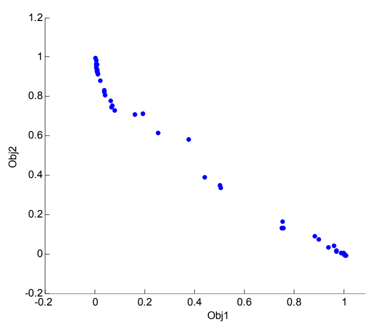
One further method could have been to sample over the entire utopia and nadir point range, adding the additional two 0.495 pieces in both the aspiration and reservation levels, effectively doubling the space in Range 2. In thinking about the levels visually, hypothetically (without noise or any MADS limitations), any point on the Pareto front may be found just by using the entire range between utopia and nadir components. Recall, given an aspiration and reservation level as shown in Figure 4.10.2, SMOMADS finds the point on that ray formed by the aspiration and reservation levels closest to the aspiration level. In performing Range 3 as done here, each ray still intersects the Pareto front even though sometimes the levels are outside the utopia and nadir components. However, if the doubled space were used for both aspiration and reservation, there could be design levels where both the aspiration and reservation levels were outside the utopia and nadir components, meaning that hypothetically the ray would not cross the Pareto front. Of course, in reality, SMOMADS will result in some point for any design level, but it should be of no added value. Furthermore, the CCD2 design is effectively using that range when it generates its axial points.

4.10.2. Results. Results are presented by problem for a majority of the test set. Only a few problems are not included, as they added little to the findings.

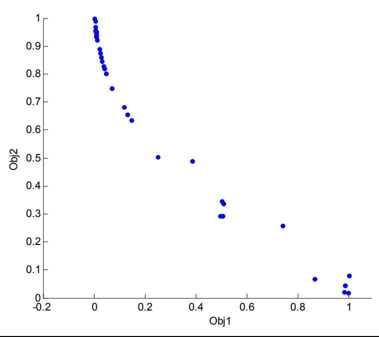
The Dias Γ1 results are shown in Table 4.10.1. All ranges and both function evaluation limits performed well according to the metrics (noting again that some spreads were influenced by obvious dominated points). This was also validated graphically. The best solution was found by using U3*, which is illustrated in Figure 4.10.3, relative to the U1* and U2* runs. For this problem, the gaps were a good indication of quality.

Table 4.10.1: Dias Γ1 Results

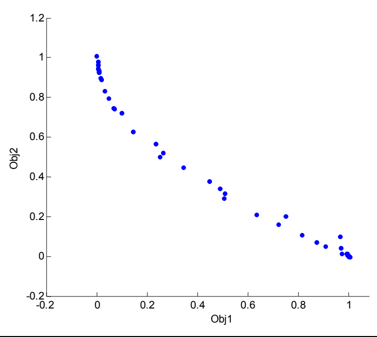
Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	38	38	38	41	39	35	20	26	33	35	25	27
Entropy	0.89	0.82	0.87	0.93	0.90	0.84	0.92	0.89	0.89	0.88	0.87	0.94
OS	1.33	1.33	8.52	5.31	1.01	4.69	1.02	1.00	1.00	0.98	1.01	1.02
OS1	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.00	1.00	1.00	1.01
OS2	1.32	1.31	8.46	5.27	1.00	4.65	1.01	1.00	1.00	0.98	1.01	1.01
NDC	15	11	18	18	14	16	20	14	16	14	15	19
CL	2.27	3.09	1.89	1.72	2.36	2.31	2.60	3.29	2.44	2.64	3.13	2.37
Time	8891	457	4923	358	9015	354	492	371	475	339	742	365
Largest Gap	0.43	0.49	3.76	3.96	0.27	3.65	0.31	0.30	0.21	0.25	0.27	0.17
Avg. Gap	0.24	0.27	1.21	0.78	0.21	0.93	0.21	0.19	0.15	0.19	0.22	0.15
# Gaps	6	5	7	7	4	5	3	4	5	5	4	2



(a) U1*



(b) U2*



(c) U3*

Figure 4.10.3: Dias Γ1 Results

The Dias Γ2 results are shown in Table 4.10.2. It was evident in the metrics and plots (not shown) that Ranges 2 and 3 performed better than Range 1. Such a result is perhaps intuitive but is also interesting, as it did not necessarily show in Dias Γ1. Again the uniform design performed better.

Table 4.10.2: Dias Γ2 Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	49	39	49	44	42	44	42	43	45	47	38	34
Entropy	0.81	0.71	0.94	0.94	0.90	0.80	0.68	0.83	0.88	0.92	0.87	0.91
OS	1.02	1.33	1.33	5.09	1.03	1.02	1.02	1.02	1.00	1.01	1.02	1.03
OS1	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.01	1.01	1.01
OS2	1.01	1.32	1.32	5.04	1.02	1.01	1.01	1.01	0.99	1.00	1.02	1.02
NDC	9	10	16	17	15	13	7	11	11	12	14	17
CL	2.56	3.30	1.44	1.65	2.00	2.15	4.29	2.64	2.45	2.08	2.43	2.24
Time	3314	458	4938	372	9363	361	191	383	174	350	347	367
Largest Gap	0.56	0.56	0.31	3.46	0.32	0.54	0.84	0.53	0.50	0.38	0.28	0.39
Avg. Gap	0.32	0.34	0.25	0.64	0.21	0.40	0.40	0.24	0.29	0.22	0.18	0.22
# Gaps	4	5	8	8	6	5	3	5	4	5	6	4

The Disk Brake results are shown in Table 4.10.3. CCD2* is depicted in Figure 4.10.4. The near uniform designs attained essentially the same approximations, except for the extreme points. These points occurred in the factorial portion of the CCD. The near uniform design never tests levels precisely at their maximum or minimum values as the CCD does. The second and third ranges performed comparably, but again better than Range 1. Interestingly, limiting the number of function evaluations does not seem to hamper the approximation. In fact, allowing "worse" solutions may introduce more Pareto points in some cases. Ranges 2 and 3 take more time, but only if the function evaluations are not limited to 500.

Table 4.10.3: Disk Brake Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	26	26	38	41	38	33	26	22	26	26	25	27
Entropy	0.81	0.83	0.89	0.89	0.89	0.86	0.81	0.81	0.84	0.85	0.88	0.87
OS	0.16	0.19	1.01	1.00	0.80	0.61	0.11	0.13	0.18	0.17	0.28	0.20
OS1	0.56	0.67	1.08	1.07	0.80	0.61	0.41	0.45	0.59	0.57	0.62	0.60
OS2	0.29	0.29	0.93	0.93	1.00	1.00	0.27	0.28	0.31	0.30	0.45	0.33
NDC	10	13	13	14	11	14	10	9	10	11	11	11
CL	4.60	3.54	2.62	2.21	3.09	2.79	4.60	5.56	4.60	4.18	4.27	4.09
Time	1363	287	4923	292	4911	292	295	283	277	287	546	285
Largest Gap	1.21	0.86	13.68	23.08	22.57	26.48	0.76	0.00	0.52	0.62	6.52	0.00
Avg. Gap	1.21	0.86	8.28	6.99	8.28	14.49	0.76	0.00	0.52	0.62	3.61	0.00
# Gaps	1	1	4	5	5	2	1	0	1	1	2	0

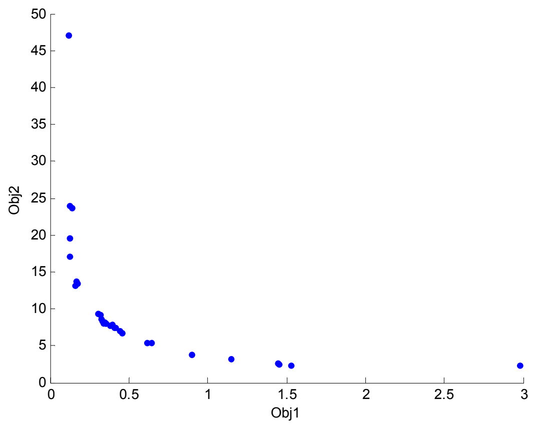


Figure 4.10.4: Disk Brake CCD2*

DTLZ7 results are shown in Table 4.10.4. The 50000 evaluation limit took less time than the 500 for Range 1 on the uniform design (this occurred on Dias Γ2 as well). This was likely a product of random number draws and polling directions. The uniform design performed better in general when looking at the plots; however, no range really outperformed another.

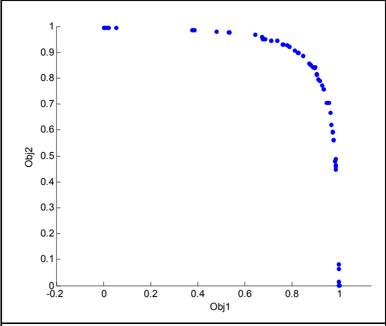
Table 4.10.4: DTLZ7 Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	52	41	44	49	46	45	30	28	39	40	37	31
Entropy	0.89	0.91	0.92	0.94	0.90	0.84	0.91	0.96	0.95	0.96	0.96	0.94
OS	4.35	1.23	9.46	1.08	5.22	1.00	1.24	1.23	0.96	0.85	1.01	1.22
OS1	1.23	1.23	1.22	1.08	1.00	1.00	1.23	1.23	0.99	0.91	1.01	1.23
OS2	3.54	1.00	7.76	1.00	5.23	1.00	1.00	1.00	0.98	0.94	1.00	0.99
NDC	10	9	13	10	9	8	11	13	8	7	11	12
CL	2.00	3.44	2.15	2.30	2.89	3.38	3.82	3.38	4.13	4.57	3.18	3.42
Time	7219	289	4850	250	8349	223	99	209	329	238	687	223
Largest Gap	3.33	0.52	5.94	0.53	6.26	0.49	0.52	0.52	0.34	0.32	0.41	0.50
Avg. Gap	0.94	0.37	1.67	0.38	1.56	0.38	0.37	0.29	0.28	0.27	0.27	0.31
# Gaps	6	5	7	4	5	4	4	5	4	4	5	5

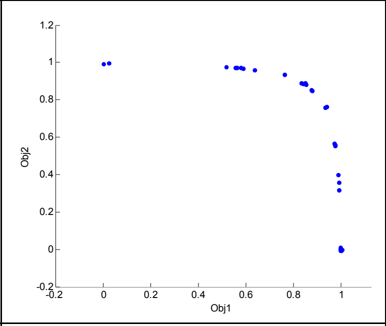
The Fonseca F1 results are shown in Table 4.10.5. Range 2 outperformed both of the other ranges in both the CCD and the uniform design. Furthermore, the uniform design fared better than the CCD. It is interesting to note, that, without a limit on function evaluations, the relative time for each range varies according to the problem. The limited uniform runs of Range 2 and 3, as well as the limited run of Range 2 for the CCD are shown in Figure 4.10.5. Surprisingly, the number of dominated (bogus) points is somewhat unaffected by limiting the function evaluations, in general, but this also may change as the number of objective functions increases.

Table 4.10.5: Fonseca F1 Results

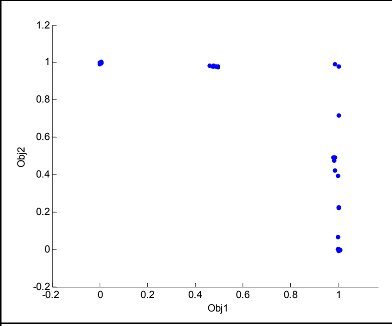
Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	48	50	48	37	58	52	53	53	24	17	38	39
Entropy	0.74	0.83	0.92	0.94	0.71	0.63	0.89	0.91	0.94	0.94	0.96	0.95
OS	1.02	1.01	1.01	1.02	1.01	1.01	1.01	1.01	1.00	1.00	1.00	1.01
OS1	1.01	1.01	1.00	1.01	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.01
OS2	1.01	1.00	1.01	1.01	1.00	1.01	1.00	1.01	1.00	1.00	1.00	1.00
NDC	9	9	10	13	7	6	6	7	19	15	12	12
CL	2.67	2.44	2.40	2.69	2.00	3.33	3.17	2.71	2.53	3.67	2.83	2.75
Time	6918	269	4838	268	6341	349	163	319	1799	257	1790	281
Largest Gap	0.71	0.70	0.57	0.69	1.24	0.94	0.49	0.50	0.31	0.37	0.49	0.49
Avg. Gap	0.52	0.48	0.32	0.35	0.79	0.93	0.41	0.39	0.23	0.27	0.27	0.26
# Gaps	3	3	6	7	4	2	4	4	3	3	5	5



(a) U2*



(b) U3*



(c) CCD2*

Figure 4.10.5: Fonseca F1 Results

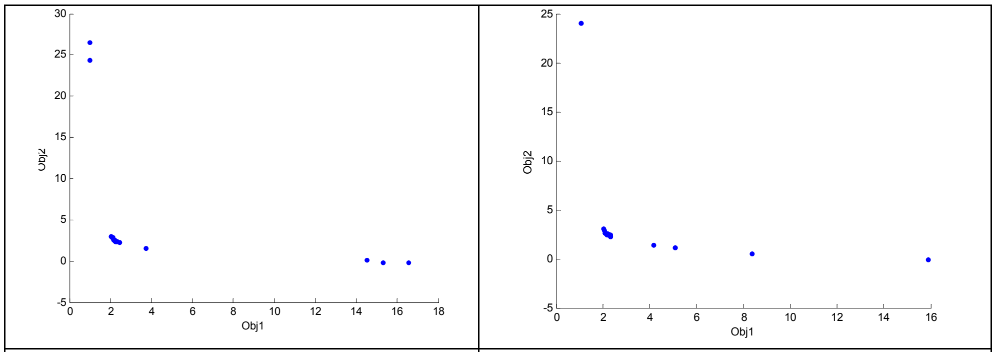
The Poloni results are shown in Table 4.10.6. This is the first problem where the function evaluation limit results in a drop-off in quality. For this problem in general, when the function evaluations are limited in number, the spreads and entropy decrease, while the cluster metric, largest gap, and average gap slightly increase. However, Figure 4.10.6 shows that there is truly little difference. Furthermore, Ranges 2 and 3 again generally perform slightly better. For the uniform designs, Range 3 performed best but did not achieve the extreme value in Objective 2.

Table 4.10.6: Poloni Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	36	34	48	51	48	46	27	35	25	27	36	31
Entropy	0.78	0.69	0.62	0.64	0.72	0.64	0.58	0.57	0.70	0.68	0.79	0.75
OS	1.21	0.96	0.93	0.80	0.61	0.57	0.25	0.31	0.55	0.61	0.66	0.73
OS1	1.37	1.06	0.99	0.94	0.62	0.65	0.34	0.39	0.64	0.64	0.75	0.87
OS2	0.88	0.90	0.95	0.85	0.99	0.87	0.72	0.79	0.85	0.96	0.88	0.83
NDC	10	7	5	5	5	4	4	4	6	6	7	7
CL	3.60	5.43	4.80	4.20	4.80	6.50	11.25	9.25	7.83	7.50	5.14	5.86
Time	3681	276	4448	279	9271	269	138	280	127	286	240	279
Largest Gap	19.01	21.23	21.41	21.03	20.31	20.51	17.66	19.63	17.73	17.88	18.54	18.27
Avg. Gap	9.42	12.37	16.14	12.18	14.06	14.81	17.66	19.63	17.73	17.88	11.92	12.88
# Gaps	3	2	2	2	2	2	1	1	1	1	2	2

The Srinivas results are shown in Table 4.10.7. The uniform design significantly outperformed the CCD, and again Range 2 and Range 3 outperformed Range 1, with Range 3 doing the best. This problem highlights the advantage of a space-filling design.

As shown in Figure 4.10.7, the uniform design places points near-uniformly on the Pareto front. Once again, limiting function evaluations does not seem to hurt the approximation.

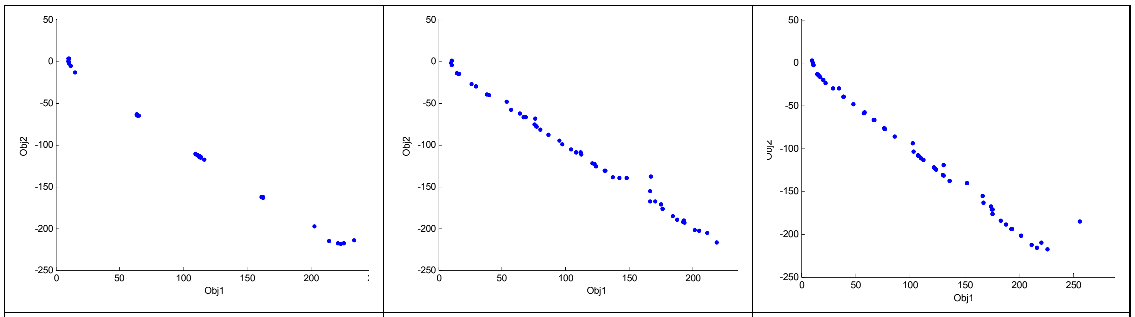


(a) CCD2 (b) CCD2*

Figure 4.10.6: Poloni CCD

Table 4.10.7: Srinivas Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	26	31	36	42	35	32	10	14	11	17	16	16
Entropy	0.73	0.79	0.74	0.82	0.93	0.89	0.98	0.97	0.99	0.99	0.99	1.00
OS	0.99	0.99	0.99	1.05	1.10	1.05	0.92	0.92	0.96	0.95	1.14	1.04
OS1	0.98	1.00	1.01	1.02	1.09	1.04	0.97	0.96	0.98	0.98	1.14	1.06
OS2	1.00	0.99	0.98	1.03	1.01	1.01	0.95	0.96	0.98	0.97	1.00	0.99
NDC	4	6	7	7	8	9	15	15	18	16	20	19
CL	11.50	6.83	5.14	4.29	4.63	4.44	4.13	3.87	3.39	3.44	2.80	2.95
Time	932	262	2913	263	5962	266	189	262	962	266	2555	262
Largest Gap	122.57	118.10	129.00	114.92	74.16	69.56	35.28	34.95	0.00	0.00	44.35	0.00
Avg. Gap	121.14	107.20	108.81	75.38	70.84	62.39	35.28	34.95	0.00	0.00	44.35	0.00
# Gaps	2	2	2	3	4	4	1	1	0	0	1	0



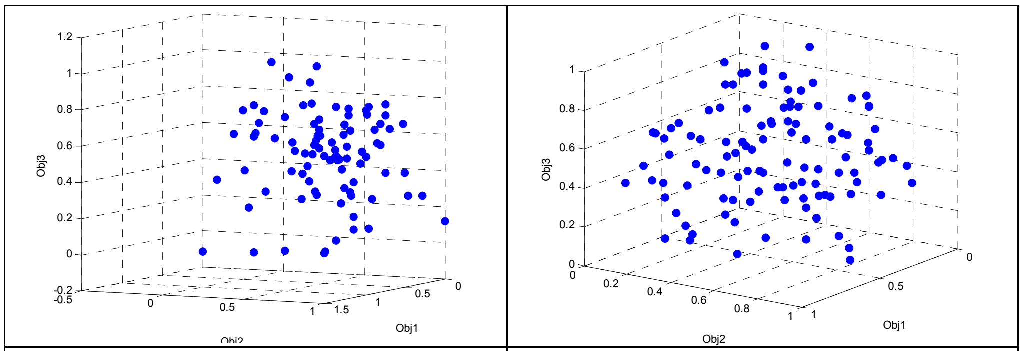
(a) CCD3* (b) U3* (c) U3

Figure 4.10.7: Srinivas Results

The Tamaki results are shown in Table 4.10.8. Limiting the number of function evaluations did not cause a decrease in quality of the approximation. Furthermore, Range 3 performed better than Range 2, and Range 2 better than Range 1. However, in looking at the plots, the uniform design for Range 2 and Range 3 are fairly comparable. The uniform design outperforms or is nearly equivalent to the CCD, as evidenced by the NDC metrics and Figure 4.10.8. Clearly, the AR1 range used for the designs influenced the previous findings in reference to Hammersley sequence sampling and uniform designs in Section 4.6. Note that a relatively good approximation was found in only 118 points, rather than the thousands that would be required using more replications or a design, such as the full factorial.

Table 4.10.8: Tamaki Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	0	10	45	51	30	29	1	10	0	9	1	7
Entropy	0.79	0.86	0.82	0.85	0.90	0.93	0.79	0.85	0.92	0.90	0.94	0.93
OS	0.64	0.57	0.96	0.95	0.98	0.98	0.25	0.41	0.68	0.66	0.65	0.64
OS1	0.84	0.80	0.98	0.95	0.98	1.00	0.63	0.76	0.90	0.80	0.84	0.88
OS2	0.86	0.87	1.00	1.01	1.00	1.00	0.67	0.78	0.85	0.91	0.91	0.82
OS3	0.89	0.82	0.99	1.00	1.01	0.98	0.60	0.68	0.88	0.90	0.85	0.88
NDC	45	55	32	43	50	64	42	53	70	64	68	77
CL	2.62	1.96	2.28	1.56	1.76	1.39	2.79	2.04	1.69	1.70	1.72	1.44
Time	5261	425	12151	430	4332	430	2945	433	8095	432	3768	434
Largest Gap	0.49	0.39	0.47	0.54	0.47	0.36	0.14	0.27	0.29	0.24	0.18	0.26
Avg. Gap	0.27	0.20	0.29	0.27	0.24	0.22	0.13	0.20	0.22	0.19	0.14	0.19
# Gaps	9	7	21	17	16	17	4	5	6	9	6	3



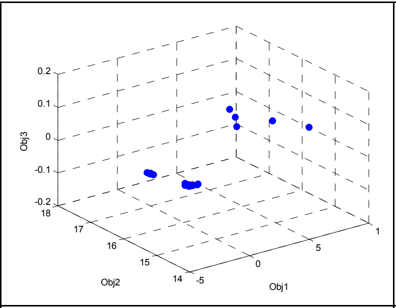
(a) CCD3* (b) U3*

Figure 4.10.8: Tamaki Designs

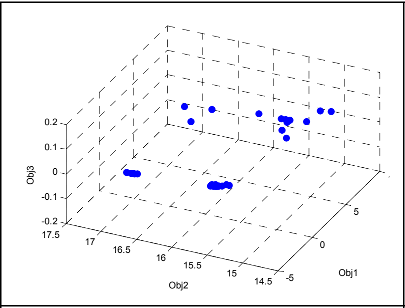
The Viennet3 results are shown in Table 4.10.9. Range 2 and Range 3 again were better than Range 1, but relatively comparable with one another. Furthermore, the CCD proved to have some advantage over the uniform design in reaching a certain portion of the Pareto front. In the case of Range 2, limiting the number of function evaluations affected the approximation, although increased replications may have forced out some of the dominated points, and the same thing did not happen in Range 3. The CCD2, CCD2*, and U2 designs are shown in Figure 4.10.9 to support these findings.

Table 4.10.9: Viennet3 Results

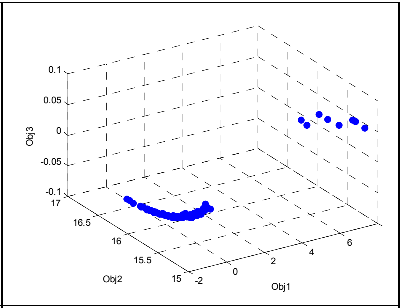
Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	50	49	74	74	70	63	35	45	31	44	36	37
Entropy	0.69	0.70	0.60	0.60	0.66	0.65	0.70	0.68	0.73	0.73	0.74	0.74
OS	4.27	0.90	4.64	4.81	4.62	4.53	1.93	1.94	2.06	2.01	2.09	2.24
OS1	1.02	0.20	1.02	1.02	1.02	1.00	0.63	0.56	0.94	0.93	1.02	1.03
OS2	1.05	1.07	1.07	1.04	1.01	1.00	0.96	1.06	0.79	0.81	0.88	0.91
OS3	3.97	4.29	4.27	4.52	4.51	4.51	3.20	3.26	2.77	2.64	2.33	2.39
NDC	16	13	10	19	13	12	8	10	11	9	12	14
CL	4.25	5.31	4.40	2.32	3.69	4.58	10.38	7.30	7.91	8.22	6.83	5.79
Time	7937	429	27939	425	4577	427	1059	437	968	422	225	438
Largest Gap	7.89	1.14	7.94	7.99	6.43	6.47	4.69	3.06	7.10	5.55	7.59	6.97
Avg. Gap	3.17	0.88	3.34	3.12	2.56	2.61	2.60	1.69	3.47	3.53	7.59	6.97
# Gaps	5	2	5	5	3	3	2	3	4	2	1	1



(a) CCD2



(b) CCD2*



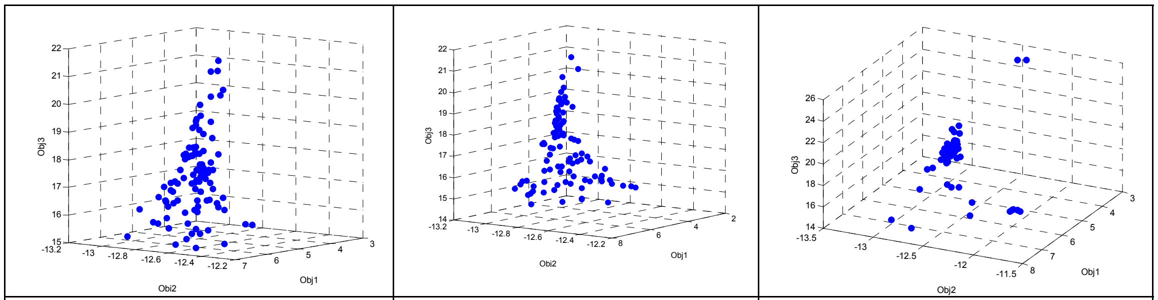
(c) U2

Figure 4.10.9: Viennet3 Results

The Viennet4 results are shown in Table 4.10.10. The near uniform designs performed much better than the CCD designs, with Range 2 and 3 performing better than Range 1 again, and themselves being relatively comparable, although Range 3 was metrically superior. Again, limiting function evaluations showed no significant degradation in the approximation. The U2*, U3*, and CCD2 designs are shown in Figure 4.10.10 as examples, since the metrics are not necessarily straightforward here.

Table 4.10.10: Viennet4 Results

Metric	CCD1	CCD1*	CCD2	CCD2*	CCD3	CCD3*	U1	U1*	U2	U2*	U3	U3*
Bogus	52	50	59	56	62	67	26	26	21	22	16	23
Entropy	0.85	0.84	0.78	0.78	0.78	0.79	0.81	0.81	0.86	0.86	0.89	0.89
OS	0.11	0.20	1.93	1.90	1.15	1.06	0.06	0.06	0.40	0.41	0.72	0.54
OS1	0.39	0.86	0.98	0.99	0.96	0.79	0.33	0.33	0.68	0.77	0.88	0.85
OS2	0.48	0.46	1.31	1.29	0.81	1.15	0.47	0.42	0.70	0.62	0.88	0.74
OS3	0.60	0.50	1.50	1.49	1.47	1.16	0.40	0.47	0.85	0.86	0.94	0.86
NDC	20	23	25	25	20	20	16	18	31	32	37	33
CL	3.30	2.96	2.36	2.48	2.80	2.55	5.75	5.11	3.13	3.00	2.76	2.88
Time	7380	450	26194	453	3758	439	490	455	412	470	154	458
Largest Gap	0.00	2.08	10.69	4.13	10.99	8.40	0.00	0.00	0.00	0.72	0.56	0.00
Avg. Gap	0.00	2.08	6.13	1.80	3.06	3.89	0.00	0.00	0.00	0.72	0.56	0.00
# Gaps	0	1	3	6	6	3	0	0	0	1	1	0



(a) U2* (b) U3* (c) CCD2

Figure 4.10.10: Viennet4 Results

In general, limiting the number of function evaluations does not have an overwhelmingly negative impact on the Pareto approximations, with regard to quality or dominated points. Furthermore, allowing less precision may, in fact, help get more points on the front. Using the entire region, or more, for both the aspiration and reservation levels works far better than any other range technique. This may seem odd, in that aspiration levels are supposed to be "good" values and reservation "bad," but in visualizing the process, as in Figure 4.10.2, it becomes clear (access to the entire range is necessary for both the aspiration and reservation levels). Furthermore, the analysis supports the fact that using even more samples within a uniform design or Hammersley sequence sampling will result in even better approximations.

Except in rare cases, space-filling designs outperform CCDs. The only advantage of the CCD is that it may find the extreme values. The space-filling designs have been shown to outperform full-factorial designs with respect to putting points uniformly along the Pareto front in a fewer number of samples. The spread metrics given here are sometimes biased by dominated points, as that check was not yet added. However, other metrics such as the NDC metric (considering the exclusion of dominated points), and even the plots themselves, support use of space-filling designs, due to their ability to generate more distinct points.

It is clear the ranges used here outperform those used in Section 4.5. In general, a combination of two replications of CCD2* and two replications of U3* should provide a quick, best initial (and perhaps even final) approximation of the Pareto front. The CCD may not even be necessary in some cases. Additionally, with the time saved by limiting function evaluations, more replications could be run, and the number of points in the uniform design or Hammersley design could be increased dramatically depending on time constraints. By limiting function evaluations, the full-factorial design becomes a viable alternative again, but the space-filling designs achieve equal or better approximations in far fewer runs.

4.11. Quality of Surrogate Types

One of the first things to evaluate in considering the use of surrogates to fill in the Pareto front is what to use as the response and how to do so. Figure 4.11.1 shows example AS function values for the Tamaki problem using Hammersley sequence sampling. It is clear that different areas of the Pareto front are indistinguishable in terms of the AS function values. This is more obvious for single-objective formulations.

Furthermore, each AS function and single-objective formulation can have very different behavior. This implies that for each single-objective optimization, of which there will be many, a different surrogate would either have to be known beforehand, or a cross-validation approach has to be used. The former is highly unlikely, and the latter could be more expensive than just fitting surrogates to the objectives themselves.

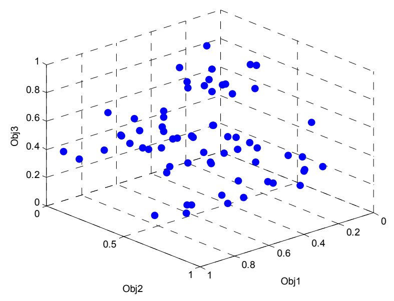
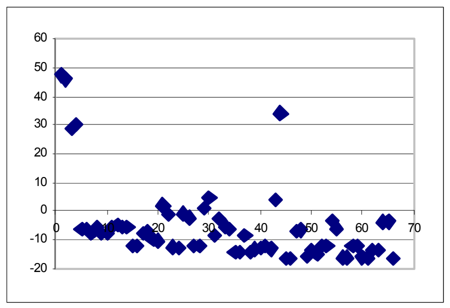
	
(a) Pareto Front	(b) Corresponding AS Function Values

Figure 4.11.1: AS Function Values

Therefore, actual objective function values should serve as the response, implying the need to form a surrogate for each objective. Soo and Bates [61] give an example of a surrogate that can simultaneously fit multiple functions, a univariate spline regression with fixed levels. Unfortunately, it is not applicable here because it is univariate. However, surrogates can be fit "simultaneously" by using basis functions, and fitting each response with different weights or knots. The advantage of this as opposed to fitting each response independently is not immediately clear, other than some small savings in memory and time.

4.11.1. Test Approach. In this research, both the design variables and the aspiration and reservation levels are compared as possible predictor variables. Furthermore, coded values of the aspiration and reservation levels are considered for those surrogates that use a factor-screening or backward-elimination process (the least squares models). The goal of this analysis was to determine a subset of possible models, and a cross-validation technique, with which to approximate the Pareto front or the objectives. The cross-validation of the subset yields the best surrogate type to use, as well as an accompanying measure of quality. The surrogates were built using the CCD2* and U3* approximate Pareto solutions from Section 4.10.

For Artificial Neural Networks (ANNs), Radial Basis Functions (RBFs), and DACE (Kriging), a 10-fold cross validation was used. This is done by generating a random permutation of the rows of data and dividing the data into 10 sets. Each set, or fold, is used to validate a model fit to the complementary 90% of the entire data. The permutation was re-generated for every model instance evaluated. This may seem counterintuitive, but in doing this, general trends should emerge that are not dependent upon the specific permutation. Root mean squared error (RMSE: the square root of sum-squared-error divided by the number of points used for the sum-squared-error) was recorded cumulatively for the 10 folds along with the maximum squared error over all of the 10 folds, for each objective.

RMSE and maximum squared error were suggested by Srivastava et. al [62] as appropriate measures. Using 10 folds was supported by a bias and variance study by Kohavi [38] which stated that variance of *k*-fold cross-validation is not dependent on *k*, and that 10 is a reasonable minimum number of folds for bias and stability. Holdout, where a model is trained on some portion of data and validated on the remaining data once, is not superior in any aspect to *k*-fold cross-validation. Boot-strapping, where each fold would not be mutually exclusive, could present a large bias. Furthermore, one study included by Kohavi [38] indicated that 10-fold cross-validation provided better model selection than leave-one-out cross-validation, the case of *k*-fold cross validation where *k* is the number of samples (this could not be used in this research anyways, due to its excessive time consumption).

Cross-validation was used for these surrogates because RBFs and Kriging are designed to interpolate (and thus error is near zero), and ANNs use random initial weights. Clearly, for least squares models, sum-squared error is a valid metric, and so no cross-validation is required. For Nadaraya-Watson, a sum-squared-error (literally the sum of the squared errors) is also found when minimizing the smoothing parameter and thus was used here. Nadaraya-Watson *attempts* to interpolate as well, so the numbers for this surrogate that follow will be slightly misleading. However, in the cross-validation scheme used as a product of this research, a *k*-fold approach is taken also with Nadaraya-Watson. The mean response at a level is used for those surrogates that interpolate. For RBFs, responses are supposed to be unique to allow for interpolation, but a model can still be formed using more than one response at a design level, and so both response types are evaluated.

4.11.2. OLS/WLS Surrogates. The various least squares models that were presented in Section 3.5 are not included in the main result tables for this section. In general, no combination of settings for these surrogates performed very well using either the mean reponse or a reduced reponse set built by accepting +/-10% of the range from the mean reponse at a design level. The foundation of the methods used to form these surrogates did, however, prove to be sound, in that using all of the options provided better models.

There is obviously some explanation as to the poor performance of these models. The factor-screening algorithm removed outliers to improve its fit, but few points should be outliers. Further, the presence of noise makes it more difficult for these surrogates to fit the data. Backward elimination of variables in the factor-screening algorithm did improve the R^2 metrics (in many cases, to values greater than 0.8 or 0.9 for all) and the other methods used seemed to serve their purpose well. Unfortunately, it was difficult for a least squares approach to fit the Pareto front, in part because an approximation to the Pareto front is generally more complicated than a polynomial.

Partitioning the data in the recursive-partitioning model sometimes improved the fit, but also at the expense of prediction capability in some partitions. Although these models were formed with minimal data, other surrogate types consistently had better fits using the same data. All of the above statements were true for both the decision variable (DV) and aspiration and reservation level (AR) models. For the factor-screening models, using coded values had no impact generally, partly because not all levels resulted in MADS finding a Pareto optimal point, and thus the properties of the design matrix were lost. Weighted Least Squares (WLS) had only a modest impact because there were only two replications, but it did fare better than Ordinary Least Squares (OLS).

For the Dias Γ1 and Dias Γ2 problems, neither the AR nor DV models performed well. The DV model had problems in part because there were so many variables. For the Disk Brake problem, only the factor-screening models did remotely well, where WLS with Box-Cox yielded $R^2_{adj} > 0.8$. For the DTLZ7 problem, only the full factor-screening model did well. The best Fonseca problem model, and one of the best overall (in terms of correct scaling, shape, prediction, etc.), was the full factor-screening model shown in Figure 4.11.2. However, even this had significant error in prediction, as evidenced by predicted solutions that dominate known true Pareto solutions.

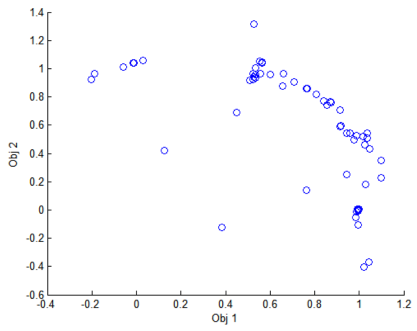


Figure 4.11.2: Best Fonseca F1 DV Predictions

The full factor-screening models performed best for the Poloni problem, with the DV model having a final $R_{adj}^2 = 0.99$, relative to the final data (outliers removed). Interestingly, on this problem, only the second decision variable was used to fit the surrogate. The AR model was decent as well and is shown in Figure 4.11.3. Overall, the factor-screening algorithm would eliminate decision variables used to fit the model, but not AR levels.

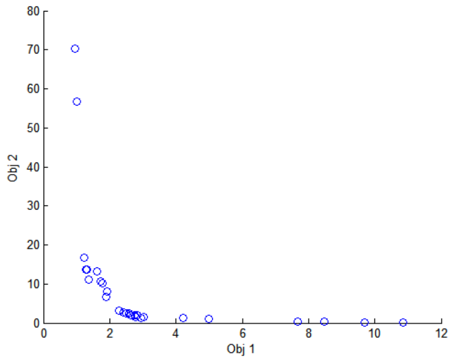


Figure 4.11.3: Poloni AR Model Predictions

Only the full factor-screening model (AR levels) worked well for the Srinivas problem, while only a recursive-partitioning without backwards elimination and multi-collinearity check model worked well for the Tamaki problem (R^2 metrics all ~0.98). This was a little surprising because Tamaki's objective functions are so simple (although there was noise). No model did very well on the Viennet3 or Viennet4 problem data. The maximum squared error on the best model for the Viennet4 problem data was 227 for Objective 3.

Any attempt at screening factors is questionable, as even with step-wise or forward regression methods, the results will likely be similar. Least squares approaches did not do well on most problems, and had large error. However, one good thing that came of looking at these surrogates was that the factor-screening algorithm confirmed that all AR level columns were required (typically).

4.11.3. Other Surrogate Results. The results for the remaining surrogate types follow, in order of objective, and where for RBFs *c* denotes some constant times the mean distance between sites. Neural networks were run with a target MSE of 0.001 and 10000 training epochs.

Data is presented in tables specifically in order of objective, where Objective 1 is the first two columns, Objective 2 the next two, and so on. This data represents the best for that surrogate type. The surrogates were run in an all-possible-combination manner. The number following *Poly* represents the order, with *R* meaning reduced (no interactions). A limitation to this analysis is that the mean response is a product of only two responses, due to the two replications used in generating the data. However, results should only get better with increased replication. Not all of the test problems are included here, for purposes of brevity, but those shown are representative of the entire set. Any plots shown for a problem used the same prediction points to provide a valid comparison.

The results for the Dias Γ1 problem using AR levels are shown in Table 4.11.1, and the results using DV are shown in Table 4.11.2. MATLAB® RBFs were not evaluated for the design variables on the Dias Γ1 problem because thirty variables are computationally prohibitive. Cubic and full quadratic polynomials performed worst (in terms of error metrics) for Kriging using AR levels, over all correlation functions. No correlation function performed best over all polynomials. Furthermore, linear, constant, and reduced quadratic polynomials were mixed in how they performed relative to one another, although they performed similarly across correlation functions.

When using design variables to create the Kriging surrogate, the error increased dramatically with quadratic and cubic polynomials, likely because of the number of design variables, 30, and the degrees of freedom (dof) needed to estimate such large polynomials (only 71 dof available). Only constant polynomials were "reasonable" however, as linear polynomials and higher had absolute errors of anything from 20 to $7 \cdot 10^{11}$. Unfortunately, the constant polynomials often resulted in only a single point.

Table 4.11.1: Dias $\Gamma 1$ using AR Levels

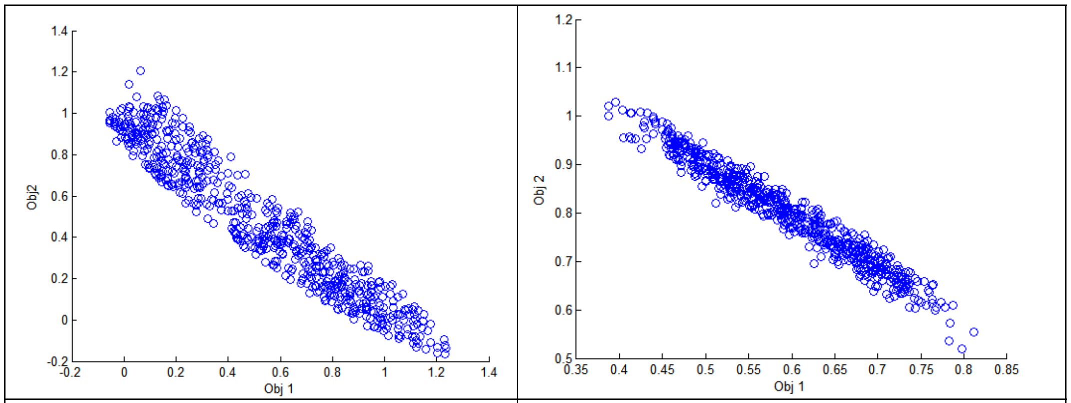
		Max Abs		Max Abs	
Surrogate Type	RMSE	Sq Error	RMSE	Sq Error	Params
DACE	0.342	0.438	0.272	0.300	Poly2R, Spline
(Kriging)	0.378	0.705	0.304	0.320	Poly0, Spherical
	0.341	0.700	0.347	0.518	Poly0, Linear
RBFs	0.344	0.846	0.404	1.38	Poly0, Bi-Harmonic, c=1
(Mean	0.364	0.897	0.406	1.43	Poly1, Bi-Harmonic, c=1
Response)	0.372	1.026	0.401	1.40	Poly1, Bi-Harmonic, c=1
RBFs	0.252	0.656	0.333	1.352	Poly0, Bi-Harmonic, c=1
(All Data)	0.274	0.862	0.317	1.378	Poly1, Bi-Harmonic, c=1
	0.284	0.836	0.315	1.215	Poly2R, Bi-Harmonic, c=1
N-W	0.284	0.352	0.321	0.681	Gaussian
	0.306	0.374	0.334	0.813	Triweight
	0.311	0.376	0.341	0.817	Triangle
FFNN	0.367	1.087	0.475	1.942	10 Neurons
GRNN	0.310	1.001	0.366	1.680	Spread=0.1
	0.390	0.425	0.338	0.627	Spread=1
RBFNN	0.397	1.009	0.367	1.579	Spread=0.1
	1.174	30.285	1.181	25.001	Spread=1

Table 4.11.2: Dias $\Gamma 1$ using Design Vars

		Max Abs		Max Abs	
Surrogate Type	RMSE	Sq Error	RMSE	Sq Error	Params
DACE	0.390	0.347	0.421	0.648	Poly0, Gaussian
(Kriging)	0.415	0.352	0.421	0.668	Poly0, Exp
	0.424	0.373	0.427	0.649	Poly0, Spline
RBFs	0.079	0.111	0.115	0.254	Poly0, Bi-Harmonic, c=1
(Mean	0.082	0.119	0.115	0.269	Poly0, Bi-Harmonic, c=1
Response)	0.086	0.135	0.112	0.206	Poly0, Bi-Harmonic, c=1
RBFs (All Data)	0.072	0.111	0.091	0.191	Poly0, Bi-Harmonic, c=1
	0.026	0.040	0.305	2.870	Poly1, Bi-Harmonic, c=1
	0.215	1.264	2.663	328.041	Poly1, Thin-Plate Spline, c=1
N-W	0.307	0.041	0.305	0.175	Gaussian
	0.419	0.391	0.400	0.490	Triweight
	0.417	0.376	0.401	0.518	Triangle
FFNN	0.064	0.051	*	*	10 Neurons
GRNN	0.068	0.224	0.129	0.693	Spread=0.1
	0.330	0.257	0.335	0.532	Spread=1

The bi-harmonic models without interactions performed best (in terms of the error metrics) for the RBF surrogates, followed by the thin-plate spline and tri-harmonic kernel function models. Using the design variables, constant and linear polynomials were best, with the error rapidly increasing due to ill-conditioning (despite using singular-value decomposition) and the 30 design variables.

In trying to generate prediction points for the design variables, the Dias Γ1 problem is problematic in that using *gridsamp* from DACE would require some number to the 30th power. Therefore, something similar to the initial population function from the GA of Section 3.1 can be used to generate test points. Predictions using all data, Poly0, bi-harmonic RBF models are shown in Figure 4.11.4. Although the AR model (a) has larger error than the DV model (b), it does map points near to the true Pareto front. This particular DV model would likely be chosen in any cross-validation scheme for this problem and appears to be accurate.



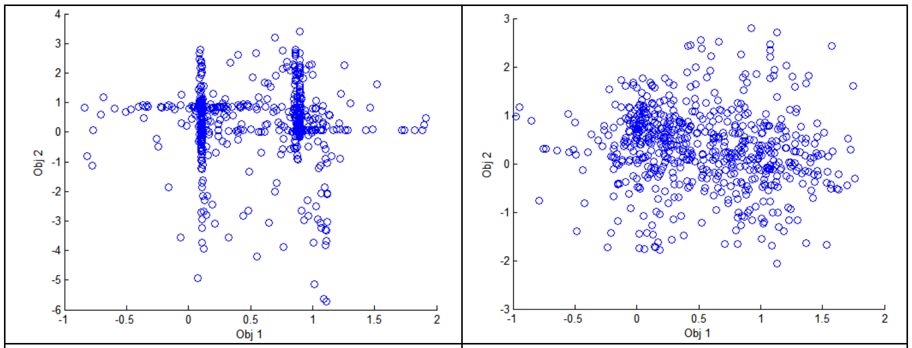
(a) AR Level Predictions (b) DV Predictions

Figure 4.11.4: Dias Γ1 RBF Predictions

Changing c for the Gaussian, multiquadric, and inverse multiquadric kernels was also tested using 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, and 2 (the c value in the tables) times the average distance (the standard is the average distance). Any changes provided no benefit, and typically did worse. Note that other kernels are shown in the tables with $c = 1$ even though they do not use this parameter. Using the mean response at each design level provided no real advantage or disadvantage for RBFs. However, the mean response model did not require singular-value decomposition to correct ill-conditioning.

For Nadaraya-Watson, using all of the data consistently provided a slightly better surrogate than using a mean response. The Gaussian kernel performed best in all instances. Something that must be kept in mind when using this surrogate type is that the sum of the kernel evaluations must not be zero; otherwise the denominator of (3.38) is zero. This problem occurred often when generating prediction points for this problem. In fact, every kernel but the Gaussian consistently had this problem (although very rarely the Gaussian kernel would have a point or two result in a sum of zero), but fortunately the Gaussian kernel was almost always the best fit throughout all problems. To account for when the sum of the kernel evaluations is zero, the code was modified to replace the function value with *NaN* (not a number) so that the particular point is ignored and does not affect the scaling of a resulting plot.

No FFNN model did particularly well. Varying the number of neurons in the layers and changing from AR levels to design variables did not make a large impact, although using all of the data did slightly better (in terms of error metrics) than using the mean response. Figure 4.11.5 and Figure 4.11.6 showcase these findings. The RMSE and maximum squared error for the second objective of the design variable data were not recorded due to a computer glitch.



(a) 5 Neurons | (b) 15 Neurons

Figure 4.11.5: Dias Γ1 FFNN All AR Data, Predictions

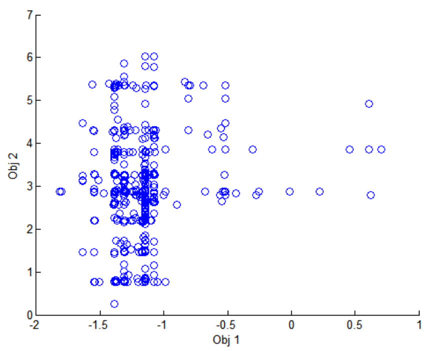


Figure 4.11.6: Dias Γ1 FFNN 10 Neurons DV, Predictions

For the GRNN, smaller spreads as small as 0.1 provided better models (but very small, such as 0.01, provided a bad model). Three spreads are depicted in Figure 4.11.7. Using the mean response consistently gave a negligibly worse model than using all of the data. This was expected, as neural nets generally perform better with more data. The design variables also did better than the AR levels according to the metrics; however, with a 0.1 spread, the GRNN predicted only a few distinct points (over-trained). It is important to note that a best RMSE did not necessarily correlate to a best maximum absolute error and that the metrics have obvious limitations as to their interpretation.

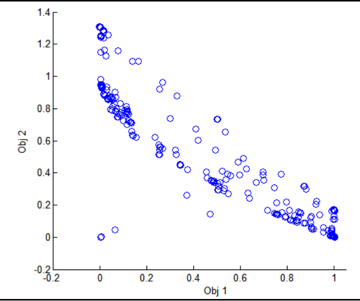
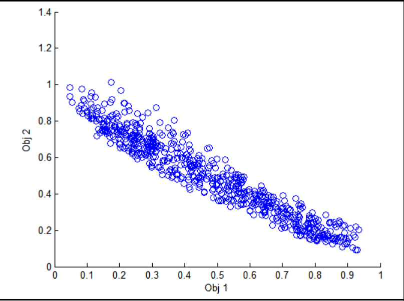
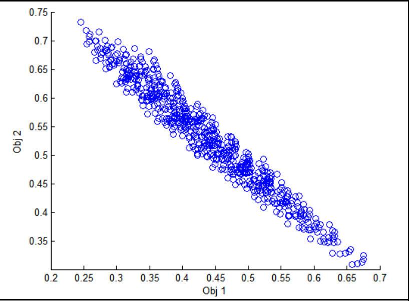
		
(a) Spread 0.1	(b) Spread 0.5	(b) Spread 1

Figure 4.11.7: Dias Γ1 GRNN AR Levels Mean Response, Predictions

The built-in MATLAB® RBFs were evaluated using spreads of 0.1, 1, and 10. Again the smaller spread performed better and using all data to form the model was slightly better. It can be clearly seen from the maximum squared errors in Table 4.11.1 that the surrogates are of poor quality.

The remaining results include the single best model for each surrogate type by problem, followed by the findings from the results and any conclusions that could be made.

The results for Disk Brake are shown in Table 4.11.3 for AR level models, and in Table 4.11.4 for DV models. The constant and linear polynomials typically did best (in terms of the error metrics) for Kriging on the AR levels, while for the design variables, the reduced quadratic and reduced cubic polynomials typically did best. The design variable models were better than the AR level models, particularly in terms of the maximum squared error. For RBFs, no *c* outperformed another, with metrics negligibly different. For the RBFs using design variables, the reduced quadratic polynomial models did best, with the Gaussian, inverse multi-quadric, and bi-harmonic kernels performing well. For the AR levels, the best models were somewhat mixed, with the inverse multi-quadric and Gaussian kernels with linear and reduced quadratic polynomials, being best.

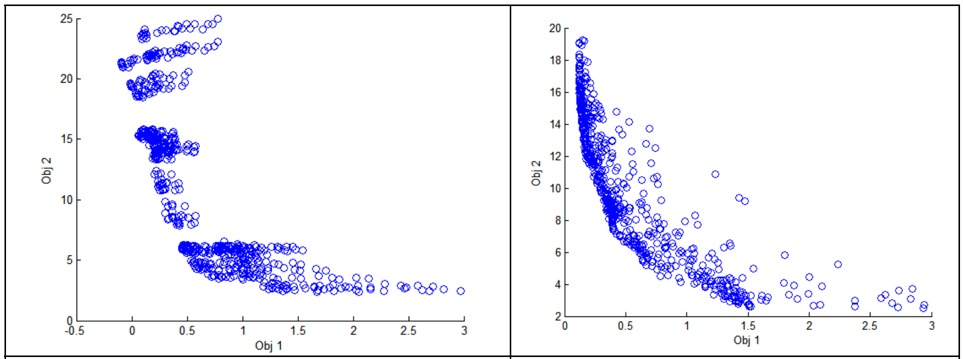
Table 4.11.3: Disk Brake AR Levels

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	0.317	1.879	2.503	69.367	Poly0, Exp
RBFs (Mean Response)	0.449	3.157	0.430	232.786	Poly1, Bi-Harmonic, c=1
RBFs (All Data)	0.345	1.859	5.019	862.684	Poly1, Bi-Harmonic, c=1
N-W	0.284	0.334	4.957	0.125	Cosinus
FFNN	0.535	3.604	6.594	428.664	10 Neurons
GRNN	0.580	5.892	6.193	251.064	Spread=100
RBFNN	0.742	8.096	9.500	514.310	Spread=0.1

The Nadaraya-Watson results were similar using the mean response data and all of the data. The MATLAB® RBF models did not do well, and neither did any of the ANNs. The FFNN with design variables had a very high squared error in the second objective (~1493). In practice, increasing the number of neurons dramatically increased the run time. Predictions from two models using the AR levels are shown in Figure 4.11.8.

Table 4.11.4: Disk Brake Design Variables

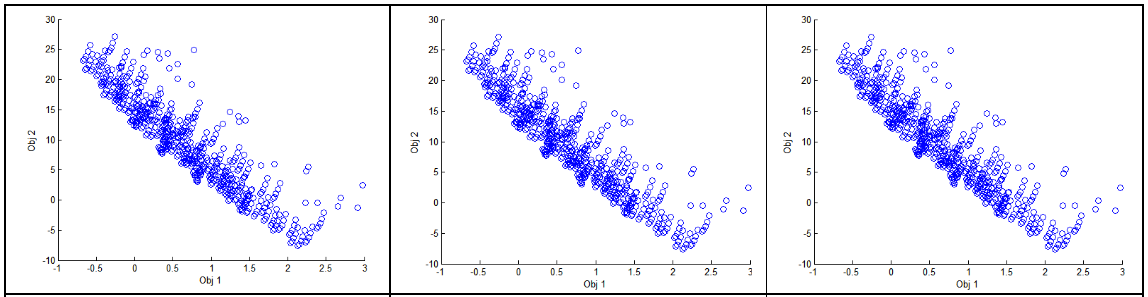
		Max Abs		Max Abs	
Surrogate Type	RMSE	Sq Error	RMSE	Sq Error	Params
DACE (Kriging)	0.163	1.752	0.542	3.997	Poly2R, Spline
RBFs (Mean Response)	0.147	1.411	1.262	59.986	Poly2R, Gaussian, c=0.5
RBFs (All Data)	0.144	1.358	1.298	64.156	Poly2R, Inv Multi-Quadric, c=1.25
N-W	0.179	0.018	2.899	3.778	Gaussian
FFNN	0.510	6.051	6.653	1483.728	10 Neurons
GRNN	0.498	5.821	6.817	2164.623	Spread=100
RFNN	0.660	8.108	9.206	2000.696	Spread=0.1



(a) Kriging, Poly0, Exp (b) Nadaraya-Watson Gaussian

Figure 4.11.8: Disk Brake AR Level Models, Predictions

Figure 4.11.9 shows the similarity between inverse multi-quadric RBF models generated on the AR level data from Disk Brake, using different values for *c*. The three models are nearly identical in their predictions. This result was seen on the other problems as well for both DV and AR level models.



(a) Mean Response (c=1.5)	(b) Mean Response (c=1)	(c) All Data (c=0.25)

Figure 4.11.9: Comparison of *c* in RBF, Disk Brake Predictions

The results for Fonseca F1 using AR levels are given in Table 4.11.5, and using DV are given in

Table 4.11.6. The mean response was slightly better for RBFs, and polynomials without interactions did best. Further, bi-harmonic and thin-plate spline were the top kernels, while the *c* value made little impact (changes in the thousandths for RMSE and hundredths or thousandths for maximum squared error).

Table 4.11.5: Fonseca F1 AR Levels

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	0.294	0.722	0.335	0.433	Poly2R, Gauss
RBFs (Mean Response)	0.279	0.636	0.304	0.454	Poly0, Bi-Harmonic, c=1
RBFs (All Data)	0.220	0.865	0.282	0.691	Poly2, Tri-Harmonic, c=1
N-W	0.269	0.017	0.337	0.108	Gaussian
FFNN	0.338	1.021	0.354	1.011	10 Neurons
GRNN	0.354	0.649	0.425	0.319	Spread=10
RBFNN	0.559	0.997	0.654	1.006	Spread=0.1

The ANNs seemed to perform well on this problem, but, in reality, only the DV neural network did well for the GRNN (although there is randomness in the forming of the model). Predictions made using this GRNN are shown in Figure 4.11.10.

Table 4.11.6: FonsecaF1 Decision Vars

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	0.009	0.003	0.013	0.003	Poly3R, Spherical
RBFs (Mean Response)	0.007	0.001	0.007	0.001	Poly3R, Thin-Plate Spline, c=1
RBFs (All Data)	0.008	0.001	0.007	0.001	Poly3R, Thin-Plate Spline, c=1
N-W	0.022	0.008	0.044	0.006	Triangle
FFNN	0.005	$4 \cdot 10^{-4}$	0.052	0.125	10 Neurons
GRNN	0.022	0.013	0.049	0.048	Spread=0.1
RBFNN	0.006	$7 \cdot 10^{-4}$	0.006	0.001	Spread=1

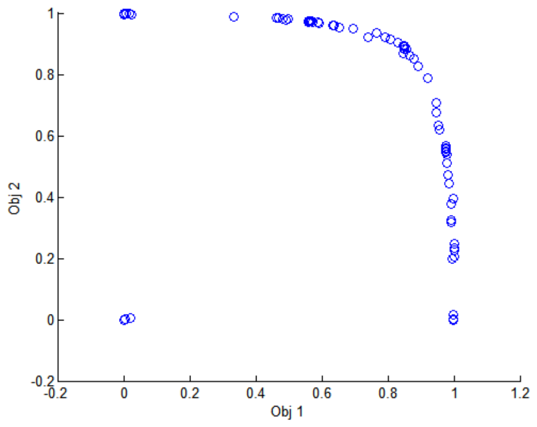


Figure 4.11.10: Fonseca F1 GRNN Model, Predictions

Poloni results using AR levels are shown in Table 4.11.7, and using DV are shown in Table 4.11.8. Polynomials with cubic terms did best for Kriging using DV, and for the AR levels, linear and reduced quadratics did best when using Kriging. No correlation function was dominant. The DV Kriging model had θ-values in the hundreds. An additional model was formed limiting these parameters to an upper bound of 30, but the model showed no improvement.

Again, the value for *c* had little impact. Bi-Harmonic and thin-plate splines with cubic polynomials did best on the DV models. The ANNs on this problem did not perform well.

Table 4.11.7: Poloni AR Levels

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	2.559	85.102	4.946	211.550	Poly1, Gauss
RBFs (Mean Response)	2.234	82.011	5.439	341.117	Poly0, Bi-Harmonic, c=1
RBFs (All Data)	1.649	56.671	6.741	407.942	Poly1, Inv Multi-Quadric, c=0.75
N-W	2.212	111.394	0.337	42.343	Epanechnikov
FFNN	2.696	78.648	5.967	297.830	10 Neurons
GRNN	2.553	120.116	5.973	373.954	Spread=10
RBFNN	3.768	145.295	6.541	423.341	Spread=10

Table 4.11.8: Poloni Decision Vars

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	0.071	0.052	0.059	0.038	Poly3, Exp
RBFs (Mean Response)	0.069	0.058	0.079	0.042	Poly3, Bi-Harmonic, c=1
RBFs (All Data)	0.062	0.051	0.076	0.037	Poly3, Bi-Harmonic, c=1
N-W	2.310	0.298	0.271	0.196	Tri-weight
FFNN	0.243	1.184	0.094	0.217	10 Neurons
GRNN	0.363	3.011	0.191	0.543	Spread=0.1
RBFNN	0.536	11.086	0.603	20.487	Spread=10

Results for Viennet3 using AR levels are shown in Table 4.11.9, and for using DV values are shown in Table 4.11.10. The Kriging models using quadratic polynomials did best on design variables while low order polynomials did best on AR levels. Interestingly, the θ-values for the DV model were on the order of thousands. The mean response again did best for the Nadaraya-Watson surrogates. The ANNs continued to perform poorly with the exception of the GRNN.

In conclusion, several findings could be made across all problems. The ANNs were preferable when training on all the data rather than the mean response, but were still not competitive with other surrogate types in either case. However, the GRNN provided a suitable alternative on some problems, although it never provided the best model.

Table 4.11.9: Viennet3 AR Levels

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	1.725	56.929	0.385	1.869	0.051	0.064	Poly0, Exp
RBFs (Mean Response)	1.691	59.323	0.677	4.034	0.092	0.086	Poly0, Gaussian, c=1
RBFs (All Data)	1.543	59.198	0.603	4.395	0.744	0.087	Poly0, Gaussian, c=0.75
N-W	1.662	57.523	0.481	1.292	0.076	0.053	Gaussian
FFNN	6.235	2133.564	0.511	1.500	0.164	2.522	10 Neurons
GRNN	1.494	61.202	0.506	1.496	0.069	0.067	Spread=10
RBFNN	1.767	68.591	1.070	15.985	0.102	0.064	Spread=0.1

Table 4.11.10: Viennet3 Decision Vars

Surrogate Type	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	RMSE	Max Abs Sq Error	Params
DACE (Kriging)	0.254	2.873	0.050	0.089	0.036	0.077	Poly2, Linear
RBFs (Mean Response)	0.256	3.208	0.077	0.373	0.011	0.009	Poly0, Bi-Harmonic, c=1
RBFs (All Data)	0.236	3.212	0.078	0.389	0.012	0.011	Poly0, Bi-Harmonic, c=1
N-W	0.370	2.141	0.079	0.045	0.011	$1 \cdot 10^{-4}$	Gaussian
FFNN	0.371	8.046	0.882	46.484	0.035	0.092	10 Neurons
GRNN	0.404	2.589	0.449	1.193	0.060	0.071	Spread=1
RBFNN	3.028	677.442	0.577	34.935	0.160	2.492	Spread=10

Using the mean response for a model was typically better or nearly equivalent to forming a model on all data in terms of the error metrics. For RBFs, varying the *c* parameter did not have a large impact, and so it is likely that using a $c = 1$ (the mean distance between design sites) is adequate. Furthermore, models using AR levels tended to use low order polynomials. The DV models generally had lower error, but they also had an advantage in that there were always more unique design sites (more unique DV levels than AR levels when using a mean response).

With regard to the Nadaraya-Watson surrogates, the uniform kernel always generated a model with the most error, while the Gaussian kernel typically generated a model with the least error. If the Gaussian was not best, it was nearly equivalent to the best. During Kriging, some models with high θ-values performed well. In the event of models with high θ-values, limiting these values to 30 did not necessarily provide any improvement in predicting.

In general, a quality AR surrogate appears to be much harder to achieve than a quality DV surrogate. However, some of the surrogate types are of high enough quality with a small number of points, that they may be used in lieu of the true function. This is important in the event of expensive function evaluations. The RMSE and maximum squared error metrics also appear adequate for use in selecting models, but both should be used. The results across all problems suggest that Kriging, RBFs, Nadaraya-Watson (with Gaussian kernel), and GRNN are the best set of models to use for the cross-validation. The GRNN is included because it is cheap to compute, and for some problems provided what appeared to be a quality model. Although bi-harmonic and thin-plate spline kernels were typically among the best kernels for the RBF models, this result was not definitive.

4.11.4. Method for Selecting a Surrogate. The cross-validation scheme proposed as a result of Section 4.11.3 uses a *k*-fold cross-validation including all RBFs, Kriging models, Nadaraya-Watson (gaussian kernel only), and a GRNN with spread equal to the mean distance of sample sites. The average RMSE and maximum squared error for each polynomial/kernel combination is ranked within each surrogate type, and the combination with the lowest sum of ranks is picked. By using both RMSE and maximum squared error, the rankings take into account that one surrogate may have a better local, or global, fit than another. These polynomial/kernel combinations are then ranked against each other, with the lowest sum of ranks picked for each objective. The data permutations used for the folds are consistent among surrogate types, with the knowledge that an awkward selection of sample sites (sites very near one another) could result in a poor choice. However, in practice this approach appeared to work well and efficiently, although the DACE surrogates are much more expensive than the others (due to the optimization of the θ-values; this was evident during the course of the runs).

The number of folds is limited by how many data points exist. Therefore, it may not be possible to use 10 folds. Different random permutations of the data yield different folds, which can obviously give different best surrogates, and so it is important to have enough data to be able to do more than just a few folds, if possible. With a large amount of data, this will not necessarily be the case.

4.12. Surrogate Uses

Section 4.11 developed a set of surrogate types, and a method to select a best surrogate from that set, for an objective. However, depending upon the required level of accuracy, there are three ways a surrogate can be used to approximate the Pareto front.

First, prediction points can be generated and used to fill gaps in the current approximation. To do so, a function, such as *gridsamp* from DACE or one that resembles the initial population generator from the GA in Section 4.3, can be used. Again, considering a gap to only consist of two endpoints, the endpoint values can be used as lower and upper bounds from which to sample. In the case of a mixed variable problem and DV, values likely have to be selected from the the discrete values at the endpoints (or perhaps including those values inbetween as well). Second, the surrogates can be used to generate a surface that approximates the Pareto front. Obviously, neither the first or second methods guarantee that the resulting approximation is truly Pareto optimal. The third method is to use the surrogates within the context of optimization, where they may be used as a search (such as is found in the NOMADm software for the search step of GPS or MADS), or where they can replace the simulation entirely, until true function values are needed.

4.12.1. Using Generated Predictions to Fill Gaps. The point generation approach makes the assumption that variable values are near one another along the Pareto front and that exact solutions are not required. There exist a few difficulties when using generated points and surrogates to fill gaps on the Pareto front. First, values may be generated that are not feasible relative to constraints. It could conceivably become computationally expensive to generate a large number of feasible prediction points. Second, if a gap is large, non-Pareto solutions may be predicted. If the surrogates have some level of error, predictions can also be made to areas where the Pareto front should be discontinuous. Finally, the use of the categorical variables is restricted. If values other than those from the endpoints of a gap are allowed, there is a very real risk of predicting non-Pareto points or points nowhere near the gap of interest. Therefore, care has to be taken when using surrogates in this manner.

In the plots for this section, the point generation methods are demonstrated. Gap endpoints are shown in red, Pareto points in dark blue, and predicted points from surrogates in respective colors. In the case of AR levels, the prediction points were generated using *gridsamp* with 625 points for 2 objectives and 729 for 3 objectives. The GA method was used for the DV models. AR levels are better in a way because feasibility is not an issue, and a predicted point should be Pareto optimal. No feasibility or dominance check was included for these plots. Such a check considers the surrogate points against themselves for dominance, returning the surrogates' best estimate of Pareto points. An example of this is shown in Figure 4.12.1 for the gap with center (0.2, 0.76).

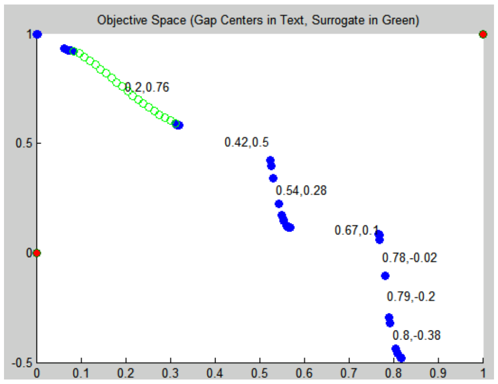
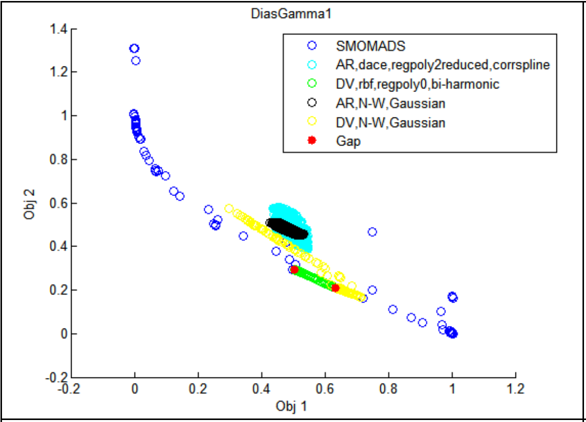
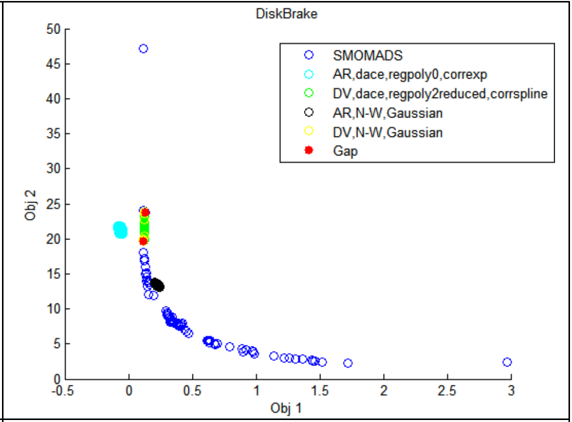


Figure 4.12.1: DTLZ7 Dominance Check Result

Figure 4.12.2(a) shows a gap in the Dias Γ1 Pareto front. The RBF and DACE DV models do well in terms of filling the gap accurately, while the DACE AR model gives the correct shape of the missing portion of the front, but error causes the predictions to lie away from the true front. The Nadaraya-Watson DV model predicts points on the front, but also overshoots the gap. Disk Brake is the only mixed variable problem evaluated, and is shown in Figure 4.12.2(b). In this case, using only those discrete variable values from the endpoints seems to work well.



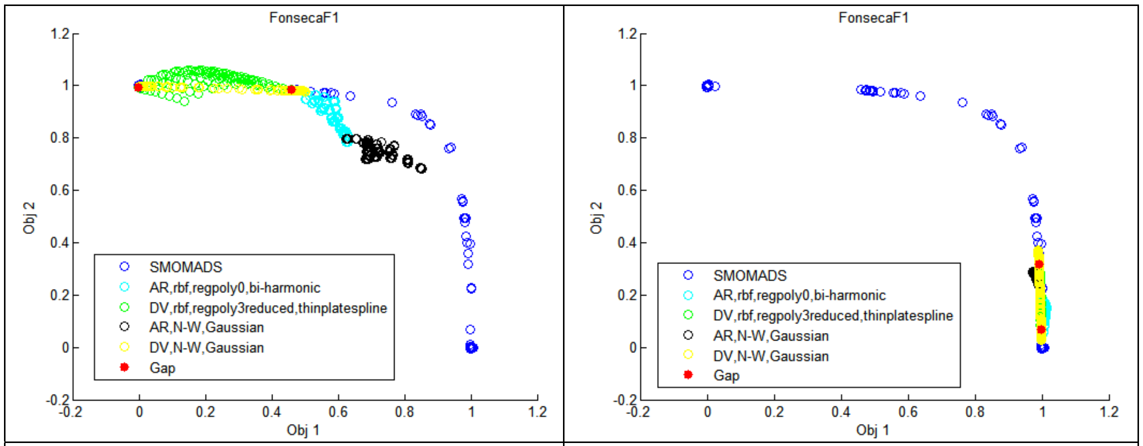
(a) Dias Γ1



(b) Disk Brake

Figure 4.12.2: Prediction Comparisons for Dias Γ1 and Disk Brake

Figure 4.12.3 shows results for two gaps in the Fonseca F1 Pareto front. The DV models always do well, while the AR models fit the second gap (b) well, but not the first (a). Figure 4.12.4 shows results for the Poloni problem, where the identified gap should be a discontinuity in the Pareto front (*i.e.*, a valid gap). All surrogates except the DV Nadaraya-Watson incorrectly place points within the gap, although the Nadaraya-Watson AR model comes closest to avoiding doing so. Interestingly, the DV DACE model used all feasible points, and a dominance check would have maintained some of those points.



(a) Gap 1 (b) Gap 2

Figure 4.12.3: Prediction Comparisons for Fonseca F1

Figure 4.12.5 shows results for the Tamaki and Viennet3 problems. All models provide fairly accurate predictions for the Tamaki gap. The Viennet3 gap presents a larger challenge for the models and only the DV models predict in the correct area of the Pareto front.

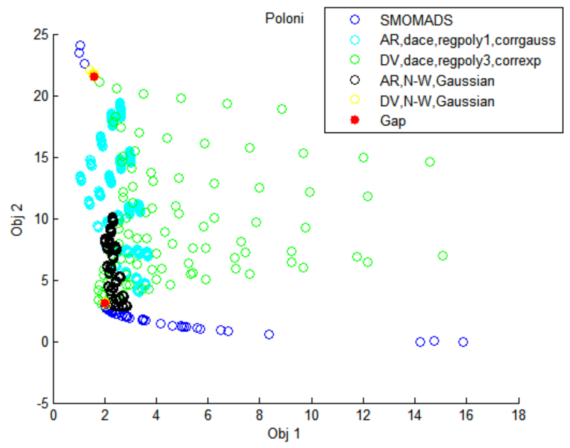
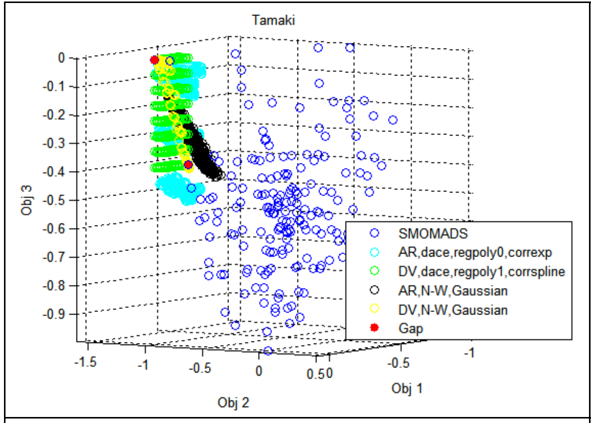
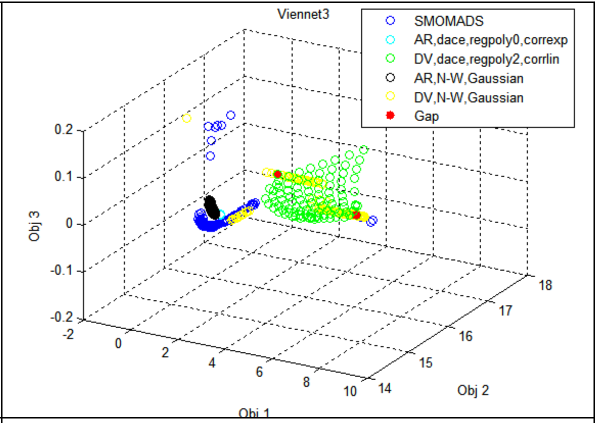


Figure 4.12.4: Prediction Comparisons for Poloni

Clearly, a dominance check is not a good idea if true Pareto points are included in the same set as the surrogate predictions (error could lead surrogate predictions to dominate true solutions), however, it appears a check could be used in relation to only the surrogate points. In the case of a feasibility check, Tamaki was the only problem for which infeasible DV values were used when predicting. In Figure 4.12.6, the first plot shows those values predicted using the DV models, and the second plot shows those predictions corresponding to the infeasible points. If only feasible points are used, a valid approximation results.



(a) Tamaki



(b) Viennet3

Figure 4.12.5: Prediction Comparisons for Tamaki and Viennet3

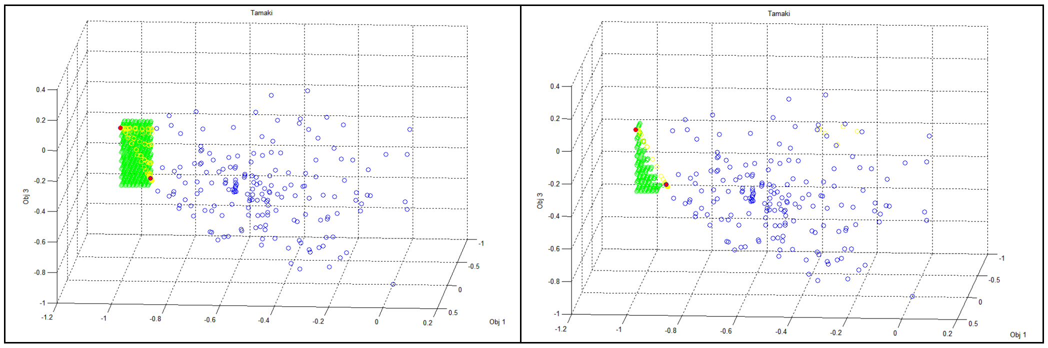


Figure 4.12.6: Tamaki Gap

4.12.2. Generating a Surface. Assuming continuity, current Pareto approximate solutions, by themselves or with newly generated solutions can be used to form a surface for the Pareto front. Cubic splines can be used over the new predictions (green) and Pareto solutions (blue), as shown for the Tamaki problem in Figure 4.12.7. Depending upon the shape of the front, this may not always yield an entirely correct surface.

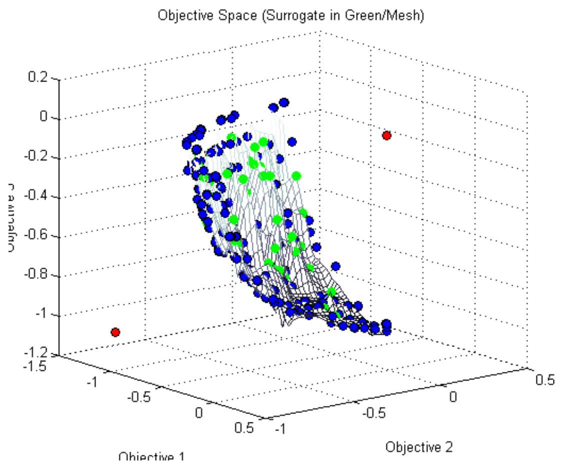


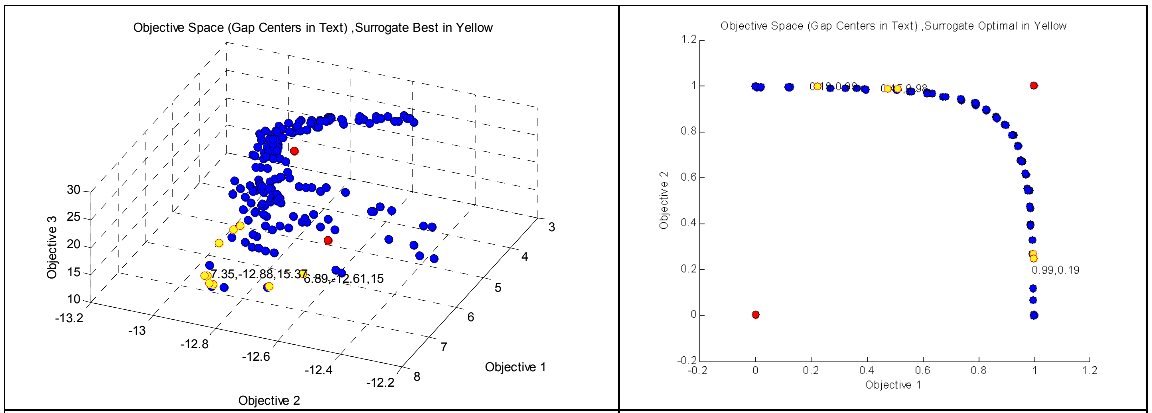
Figure 4.12.7: Tamaki Pareto Front Surface

4.12.3. Surrogates within an Optimization Framework. If surrogates are used to fit the objective functions or the single-objective formulation, they may be used to perform the optimization without using expensive function evaluations. Once the surrogate's solution is found, then the real function is checked using that solution. This may work well with a one-objective problem, but there are problems associated with doing so in the multi-objective case.

First, the nMADS (BiMADS) algorithm tries to use all function evaluations to find Pareto solutions. Using this approach with surrogates, each evaluation would have to be repeated using the true functions (and then also repeated like R&S to eliminate the effect of noise). This, of course, eliminates its advantage over just using the true functions in the first place, that is, unless only the optimal solutions of each single-objective formulation are checked using the true functions. In this latter case, some efficiency is still lost.

Additionally, the deterministic dominance check can be a problem, whether surrogates are fit to the objectives or to the single-objective formulations. A resulting optimal point or set of near-optimal points of the surrogate can be evaluated by the true functions, but in practice surrogate-found points are rarely retained due to slight error in the models, since the feasible space is continuous, and because there is noise present. The complications involved when trying to account for this were detailed in Section 3.9.

For example, Figure 4.12.8(a) depicts a result using this approach (optimizing the surrogate) with nMADS for the Viennet4 problem and the resulting non-dominated responses over all function evaluations, evaluated using the true response. Here, the surrogates (fit to the objectives) were of high quality, and the solutions appear to be true Pareto solutions in the two shown gaps (yellow points are true responses corresponding to surrogate solutions). However, a deterministic dominance check would remove these points due to surrogate error or noise. In two objectives, there is a chance the surrogates will find correct solutions. Figure 4.12.8(b) shows single-objective formulations solved using the surrogates for the Fonseca F1 problem, where the yellow points are the final surrogate solutions assessed with the real functions. The solutions are reasonable and would be retained by the dominance check.



(a) Non-Dominated Solutions from nMADS | (b) Final Solutions Only

Figure 4.12.8: Optimizing the Surrogate

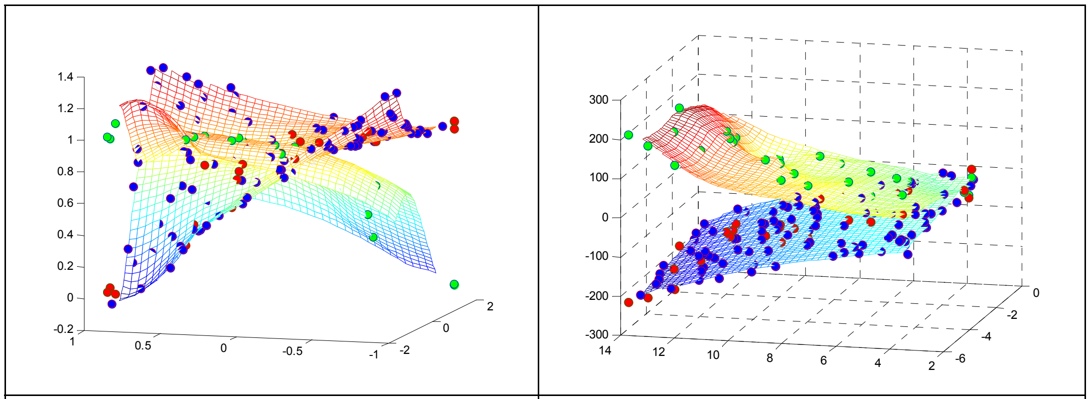
Surrogates can also be fit directly to the single-objective formulations, but the surrogates have the opportunity to continually become better by approximating the objectives. There are many, many single-objective formulations to be performed and the responses corresponding to decision variables will vary according to the reference point in use.

NOMADm has the capability to use surrogates within the search step of GPS and MADS algorithms. Surrogates have not been tested in this manner for the multi-objective case, though they have been tested on very difficult single-objective problems [16]. Further, there is again the problem of having to either know a best model prior to each optimization, or using a cross-validation repeatedly. Therefore, it is more useful to replace the objective functions with the surrogates; but again, this is where the dominance check becomes an issue. Further investigation into the use of surrogates for optimization is suggested for future research.

4.13. Other Surrogate Considerations

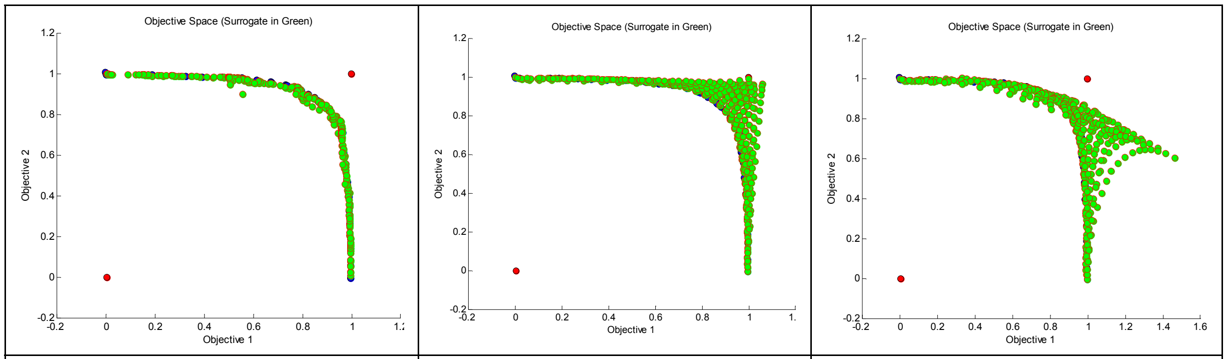
There are two other considerations with repect to surrogates that need to be mentioned. First, smoothing could be added to Kriging or RBFs to reduce the effect of noise. In the nMADS approach, this is somewhat unnecessary because mean responses are used, and the distribution of points tends to be somewhat uniform. The plots in Figure 4.13.1 show actual Pareto solutions in green (Objective 1) and red (Objective 2), for the Fonseca F1 (a) and Srinivas (b) problems versus the decision variables. The mesh and blue points represent predicted values for the surrogates (here RBFs with tri-harmonic, reduced quadratic). Although the noise level in the true functions was high, the RBFs generated a fairly smooth surface without having to modify the weights.

The second consideration is solely with respect to the Nadaraya-Watson surrogate. The Nadaraya-Watson surrogate seemed to always be of high quality in Section 4.11, however, the metrics were somewhat misleading.



(a) Fonseca F1 5% Noise | (b) Srinivas 20% Noise

Figure 4.13.1: Surrogates on Pareto Front



(a) N-W | (b) RBFs | (c) DACE

Figure 4.13.2: Fonseca F1 Surrogates

The plots in Figure 4.13.2 depict three surrogates formed over the same set of Pareto solutions, and then used to predict values over the entire range of decision variable values. A subset of the predictions is given in Table 4.13.1. The Nadaraya-Watson surrogate determines the shape of the Pareto front very well. However, any data points added to the surrogate will not be able to deviate too far away from that main shape due to the nature of (3.38). The denominator is just some constant, and the numerator prohibits values from deviating far away from function values that created the surrogate (the kernel realizations are small). For example, consider the first few points of Table 4.13.1. Although the RBF is very accurate and follows the true objective function value, Nadaraya-Watson gives approximately the same, inaccurate, solution. Consider also the objective space point (1,1) in the RBF plot; no such point exists for Nadaraya-Watson, it was mapped to the Pareto front incorrectly.

Table 4.13.1: Fonseca F1 Objective 2 Values

x_1	x_2	True Obj Value at (x_1, x_2)	N-W	RBF Tri-Harmonic Poly3R	DACE Spline Poly3R
-1	-0.8947	0.972399	0.7109	0.9726	0.6869
-1	-0.7895	0.959332	0.7109	0.9492	0.7004
-1	-0.6842	0.941371	0.7109	0.9197	0.7078
-1	-0.5789	0.917332	0.7109	0.8839	0.7119
-1	-0.4737	0.886027	0.7109	0.8417	0.7065
-1	-0.3684	0.846264	0.7109	0.7928	0.6913
-1	-0.2632	0.797228	0.7108	0.7369	0.6731
-1	-0.1579	0.738346	0.7104	0.6742	0.6384
-1	-0.0526	0.66977	0.7045	0.6049	0.5891
-1	0.0526	0.59244	0.6111	0.53	0.5286
-1	0.1579	0.507929	0.2848	0.4511	0.4567
-1	0.2632	0.418924	0.1897	0.3703	0.3729
-1	0.3684	0.328955	0.1686	0.2905	0.2902
-1	0.4737	0.241939	0.1595	0.2145	0.2121
-1	0.5789	0.162493	0.1536	0.1463	0.1398
-1	0.6842	0.094918	0.1392	0.0897	0.083
-1	0.7895	0.043343	0.0827	0.0464	0.0452
-1	0.8947	0.011027	0.0216	0.0157	0.017
-1	1	0	0.0065	-0.005	-0.005

4.14. Using New Utopia/Nadir Points to Fill Gaps

Another method to fill gaps is to use SMOMADS designs with utopia and nadir points based on gap endpoint objective function values. A few exploratory runs were done to see how effective this method could potentially be, given that the design ranges were becoming more restrictive to fill gaps, but only 500 function evaluations are still allowed. In the following plots, original Pareto points are in blue, the new utopia and nadir points are in red, and the resulting SMOMADS points are in green. First, a gap in the Disk Brake problem was tested using a uniform design with 30 points, two replications, and sampled over the entire utopia/nadir range using the original starting iterate. It is clear in Figure 4.14.1 that the gap is filled.

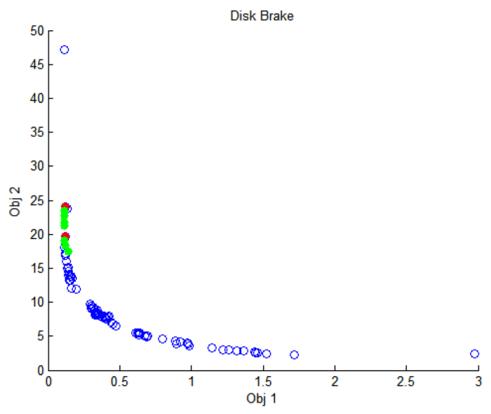


Figure 4.14.1: Disk Brake Gap

To test the effect of the starting iterate, a gap for the Fonseca F1 problem was run, using the original starting iterate and a starting iterate corresponding to a gap endpoint, but again with a uniform design with 30 runs (two replications) and sampling over Range 3 from Section 4.10. It can be seen in Figure 4.14.2 that both methods performed equally well.

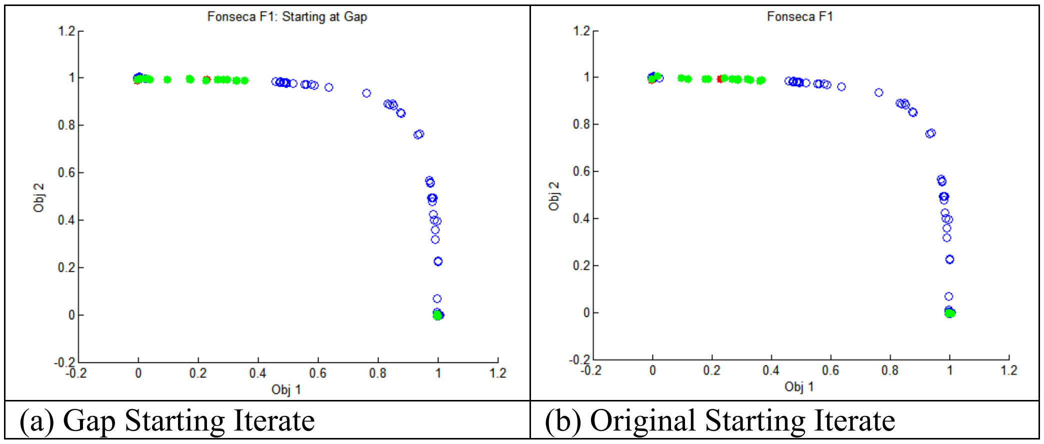
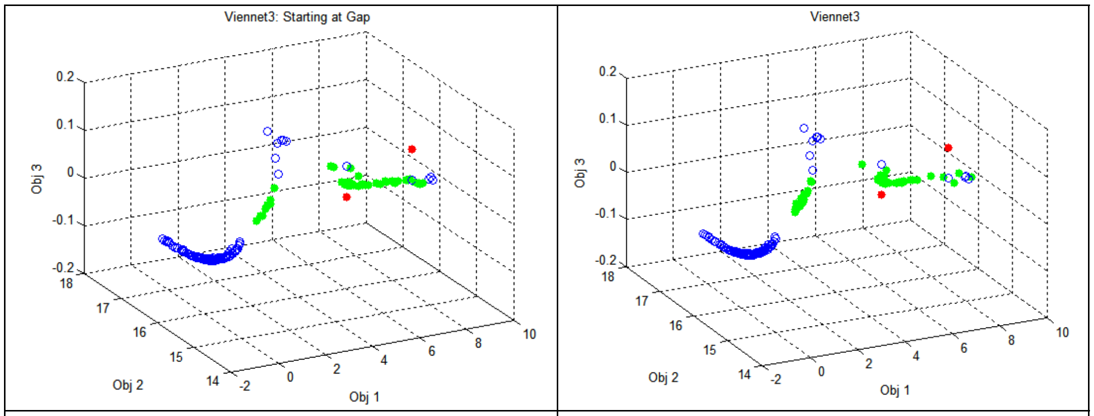


Figure 4.14.2: Fonseca F1 Gap

Viennet3 was also evaluated using the original starting iterate, and an iterate corresponding to one of the gap endpoints. As shown in Figure 4.14.3, the performance is again similar. This was important to verify, as the original designs had problems in that area of the objective space. For this problem, 50 design levels (two replications) were used.



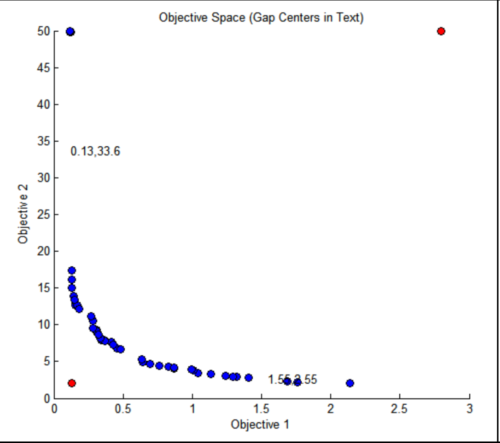
(a) Gap Starting Iterate (b) Original Starting Iterate

Figure 4.14.3: Viennet3

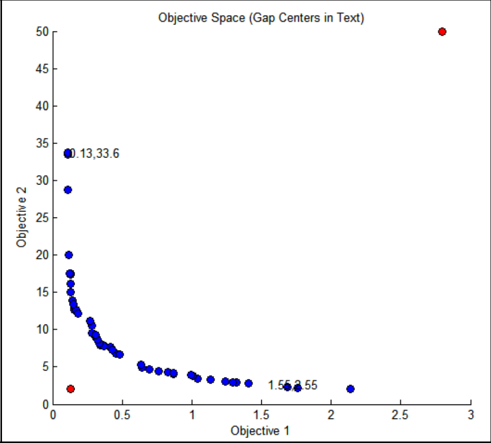
It is somewhat safe to conclude that the starting iterate is not necessarily important, and that this method is capable of filling in any gaps that are identified, so long as the gaps are not excessively small. In general, for all results throughout this research, as the number of replications increase, the results improve but computational time increases. Experimental design results should also follow those from Section 4.6 and Section 4.10 when used to fill the gaps, in terms of space-filling and range development.

4.15. Single Product Formulations

Although the single product formulations from Section 3.6 are well developed, it must be ensured that the minor modifications made still allow them to be used in the *n*-dimensional case (although convergence is proven for more than 2 objectives [13]). An initial Pareto front for the Disk Brake mixed variable problem is given in Figure 4.15.1(a), with two gaps labeled. Using the normalized formulation (3.40), with $c = 1$ (since no objective is necessarily more important than another), a point near the center of the gap is achieved. This was the goal and required only one replication, starting from both gap endpoint iterates. Also, probably due to noise, the original extreme solution is dominated. A function evaluation limit of 500 was used. This is high compared to the limit of about 30 used by the BiMADS authors [13], but it must also be considered that this is the stochastic case, and so R&S evaluates each point four times. Therefore, this is more like 120 function evaluations.



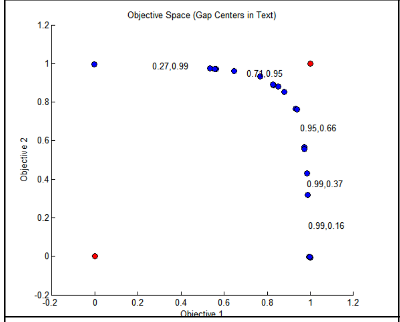
(a) Initital Front



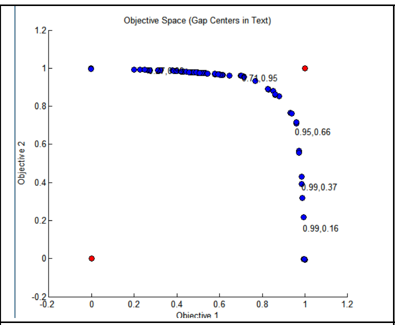
(b) Normalized

Figure 4.15.1: Disk Brake

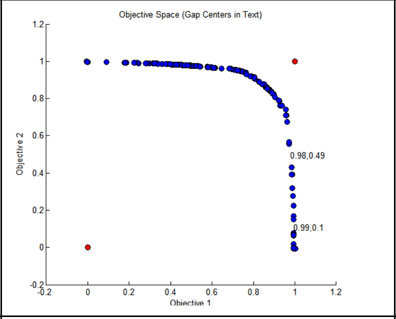
Looking at Fonseca F1 in Figure 4.15.2, it is clear that the formulations are working as intended. Using one replication with the normalized formulation provided at least some improvement on all gaps. Then, applying two replications to those resulting gaps either completely filled the gaps, or provided more improvement in terms of filling the respective gaps.



(a) Initial Front



(b) Plus 1 Replicate Normalized



(c) Plus 2 Replicates Product

Figure 4.15.2: Fonseca F1

In two objectives, nMADS is really just a slight modification of BiMADS (that is, everything in the objective formulation is the same except that the endpoints of a gap are used as starting iterates). Therefore, these results were to be expected. However, perhaps the more important test is in three objectives. The nMADS approach takes advantage of the fact that the single-objective formulations from Section 3.6 have convergence results when the formulations are generated using more than two original objectives [13].

Consider the Pareto front shown in Figure 4.15.3, where curvature effect is not depicted with a great deal of accuracy. Assume the two endpoints of a gap (in blue) do not satisfy the indifference value in at least one objective (the gap is shown here as the grey rectangle, although gaps do not have to be a specific shape). A reference point is constructed using the maximum objective function values from these endpoints (shown in purple). The single-objective formulation solutions will fill or reduce the gap by moving away from at least one of the endpoints, and away from the reference point, into the Pareto front. This may result in a path being followed, or just some portion of the gap being filled; however, it should be clear that a gap can be filled with respect to multiple objectives at once, depending upon the other current Pareto solutions. nMADS constitutes a "simple" way to fill identifiable gaps on the Pareto front.

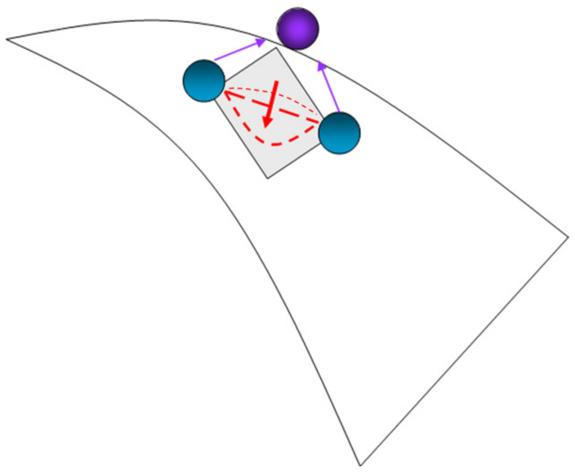
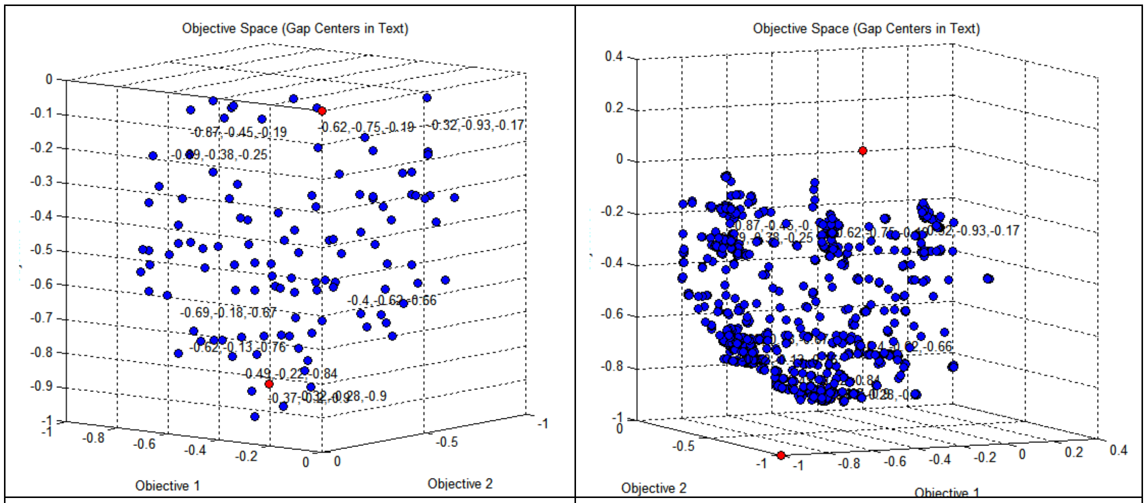


Figure 4.15.3: Gap-Filling in More Than 2 Objs

Because both endpoints are used as starting iterates, some improvement will be found (if the optimum for the single-objective formulation is near an endpoint, the replication using the other endpoint will move towards the original). Additionally, GPS and MADS provide additional improvement due to their poll step and search step, which allows function evaluations to deviate from any path.

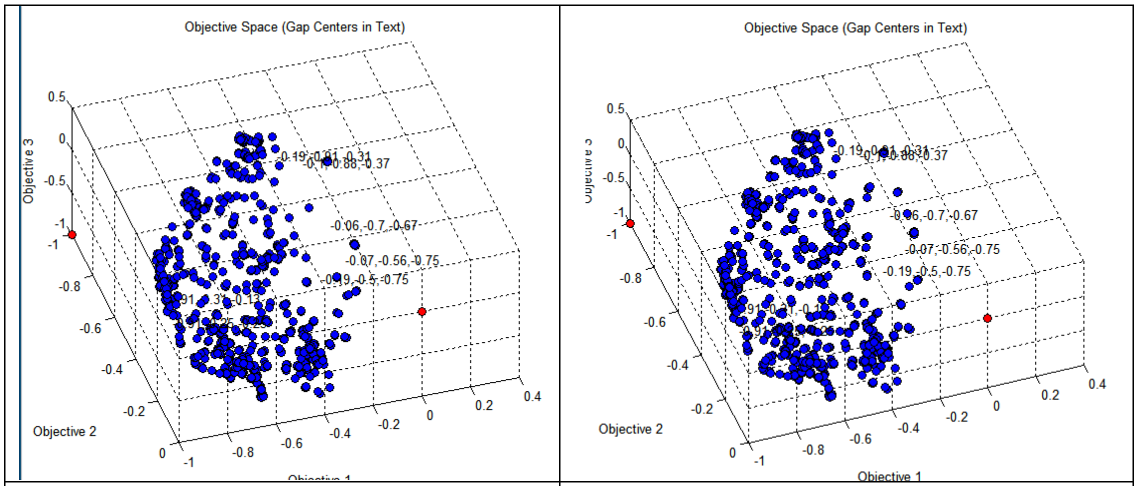
Figure 4.15.4 shows Pareto fronts for the Tamaki problem. The first plot depicts the results from a near-uniform design with 60 runs and two replications, and gaps found by the gap algorithm (see Figure 3.7.5). The second plot shows the results from the nMADS approach using one replication, with normalized formulation (and $c = 1$) on each gap, and the gaps not re-identified. Each gap in (a) has had a point added in its center and other points added that reduced the gap.

Figure 4.15.5(a) shows a gap at the point [-0.06, -0.7, -0.67]. Using one replication of the nMADS product formulation (3.41), a few points including the center are added to reduce, or fill, the gap. This is shown in Figure 4.15.5(b).



(a) Initial Front | (b) Plus 1 Replicate Normalized

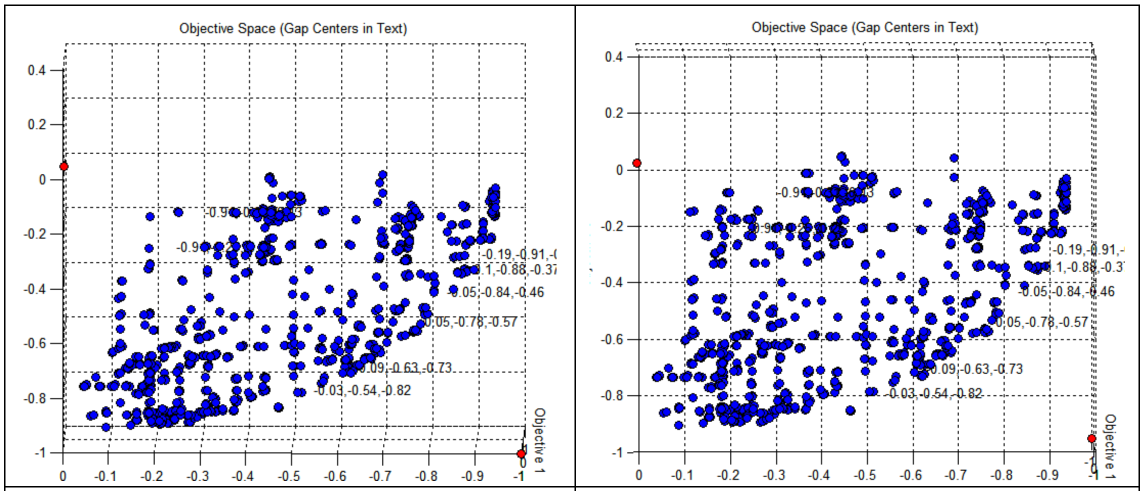
Figure 4.15.4: Tamaki Example



(a) Before nMADS (b) After nMADS

Figure 4.15.5: Tamaki Specific Gap Using Product Formulation

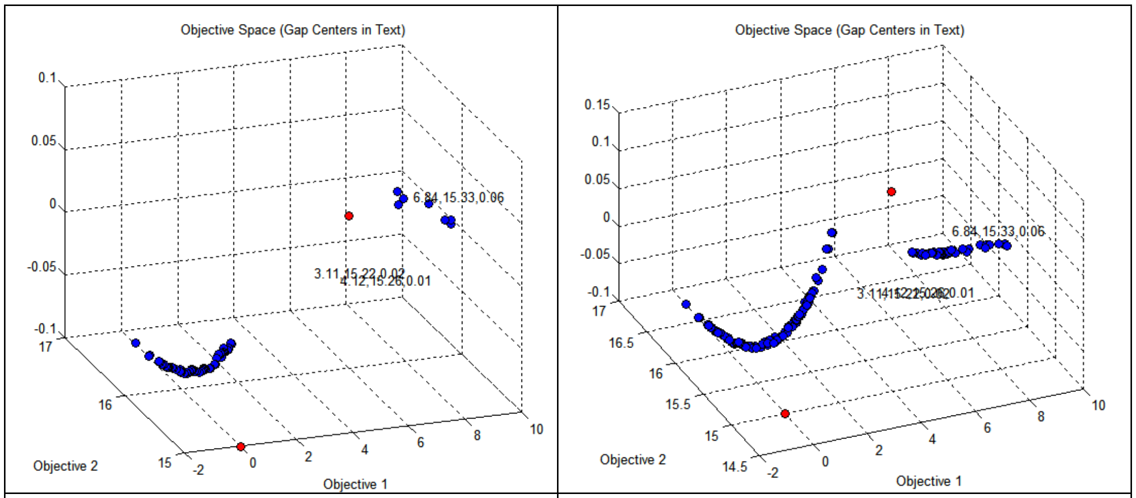
Figure 4.15.6(a) shows a gap at the point [-0.9, -.2.3, -2.7], where Objective 3 is the z-axis. Using one replication of the nMADS normalized formulation, a few points are added to fill the gap, to include the center of the gap. This is shown in Figure 4.15.6(b). Both formulations seemed to work well on all problems.



(a) Before nMADS (b) After nMADS

Figure 4.15.6: Tamaki Specific Gap Using Normalized Formulation

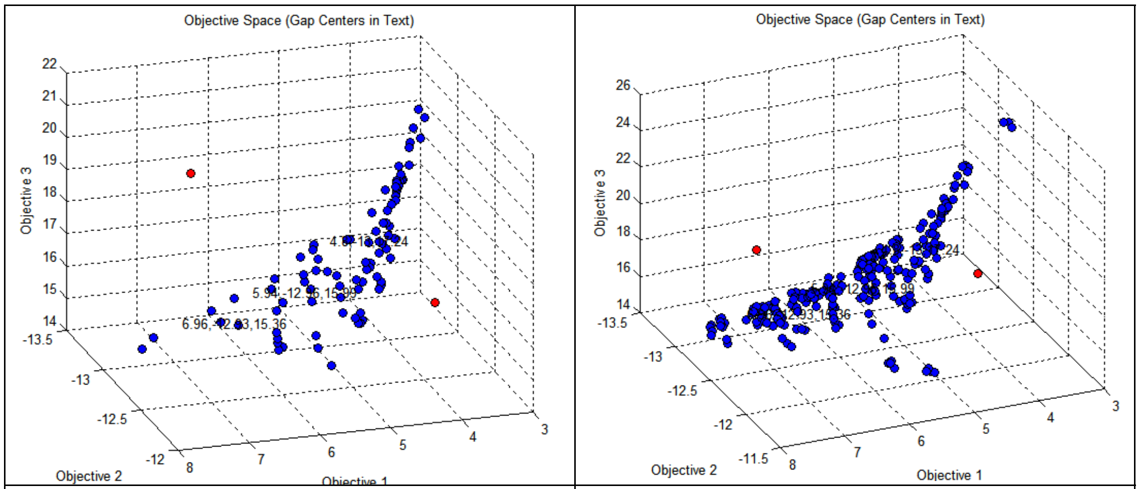
Again, using the indifference values as defined in Section 4.4, the gaps in an initial approximation (near-uniform with 60 runs, 2 replicates) of the Viennet3 front are shown in Figure 4.15.7(a). The second plot shows the approximation following one replication of nMADS (or really using GPS in the nMADS framework here) with product formulation on the gaps.



(a) Initial Front | (b) After 1 nMADS Rep

Figure 4.15.7: Viennet3 Example

Figure 4.15.8(a) shows an initial front for Viennet4 resulting from a near uniform design with 60 runs, replicated twice. The second plot shows nMADS (again, truly with GPS here) easily filling the gaps with only one replication of the normalized formulations. A few points on the right side appear to have unidentified gaps on the right side, but in fact, these points satisfy both the gap and indifference criteria.



(a) Initial Front | (b) nMADS Normalized

Figure 4.15.8: Viennet4

Both the normalized and product formulations work well on gaps, and the slight modifications made to BiMADS appear to generalize the approach to *n*-dimensions, even with noise (the noise level was still set at 1% of the nadir component value). Furthermore, nMADS in combination with the gap algorithm works fairly well in identifying and filling gaps. Here, a limit of 500 function evaluations was used for each formulation, but this can be restricted even further in some cases. This will be explored in the automated form of nMADS (see Section 4.17).

4.16. Extreme Points

An additional problem that needs to be addressed is to find or guarantee the extreme solutions of the Pareto front. These solutions are usually discarded, as they are not interesting from a tradeoff standpoint; however, they can be important in generating a complete front. Finding these extreme solutions is typically difficult, as even multi-objective methods, such as normal-boundary intersection, can fail in this regard [25]. Furthermore, when the true front is unknown, it can be difficult to verify if the extreme solutions have been obtained.

nMADS is dependent upon the spread of the initial solutions, as nMADS takes a gap-filling approach. Therefore, if the solutions corresponding to the utopia point are used as the initial solutions, there must be some level of confidence in the estimation of the utopia point. Problems may exist where there is little confidence in the estimation. An example could be Schaffer F3 with an added non-linear constraint. This may require a larger set of LHS samples within the search step than the eight typically used in this research, in order to get a good estimation of the utopia point. In this event, the SMOMADS aspiration and reservation level approach allows for an over or under estimation of the utopia point, and is able to find extreme solutions. However, for a large number of objectives, sub-designs specifically built to find these solutions may be necessary. Another approach could be to use some subset (larger than just one) of the component functions. As shown in Section 4.9, this approach finds extreme solutions, but could be computationally prohibitive for problems with a large number of objectives. Additionally, false reference points for nMADS could be specified outside of current estimations (axials), to see if in fact, the extremes have been found.

4.17. nMADS

4.17.1. Comparison to SMOMADS. It is clear from Section 4.15 that single-product formulations are useful. Furthermore, they are also generally faster, requiring fewer function evaluations than SMOMADS because they take advantage of each function evaluation, checking each possible solution for dominance. For a single gap, nMADS can potentially add many points to the Pareto approximation, while SMOMADS only uses the final point from a design level. Therefore, an automated strategy with a good initial spread of points is likely to perform better than SMOMADS in terms of computational effort. In fact, this approach can perform better than a heuristic, such as NSGA-II from Section 4.3.

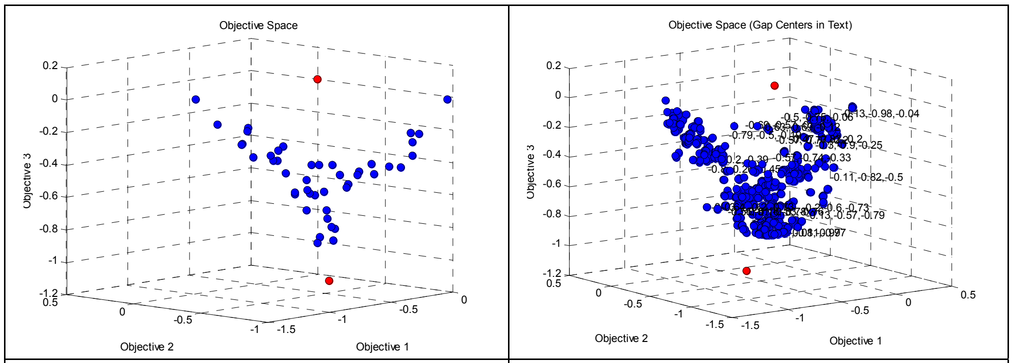
4.17.2. nMADS Algorithm. The automated nMADS algorithm, as it is termed in this research, is presented in Figure 4.17.1. The algorithm uses the utopia point to find an initial set of points, with maximal spread, and iteratively fills gaps in the Pareto front. Gaps are weighted according to how many times they, or a very similar gap, have been identified, but such that similar gaps will not be identified too many times. This assumes that the single-objective formulations show a reduction in gap size within a few attempts. The algorithm concludes when the size of the largest weighted gap is below some specified criteria. Other approaches developed in this research may be used upon its conclusion if necessary.

```
INITIALIZATION:
Let size(g) denote the Euclidean distance between the two endpoints for a gap g. Let $\bar{\omega}$
be a vector of indifference values for the objectives.
•  Apply the MVMADS-RS/MVPS-RS algorithm from starting point $x_0$ to solve
   $\min_{x \in X} f_i(x)$ for each objective i=1,...,M.
•  Run gap algorithm (see Figure 3.7.5) to identify a set of gaps G, given some $c > 0$.
•  Initialize the weights w(g) to size(g) for all gaps $g \in G$. Initialize the weights v(g) to
   1 for all $g \in G$.
MAIN ITERATIONS: Repeat while $G \neq \varnothing$ and $\max\{w(g)\} > c \cdot \|\bar{\omega}\|$.
1. For each $g \in G$
      o  If $w(g) < c \cdot \|\bar{\omega}\|$, G=G\g. Go to 1.
      o  Else:
            ▪  Build reference point r by using maximum objective values from the
               endpoints of g.
            ▪  Solve a single-objective formulation using the MVMADS-RS/ MVPS-
               RS algorithm from the starting iterates corresponding to the two
               endpoints of g.
2. Remove dominated points and run gap algorithm with resulting set of gaps $G'$.
      o  If any center of $g' \in G'$ is within $c \cdot \|\bar{\omega}\|$ of any center of $g \in G$ (according to
         Euclidean distance), set $v(g') = 2v(g)$, $w(g') = size(g') / v(g')$.
      o  Else, set $w(g') = size(g')$ and $v(g') = 1$.
3. Set $G = G'$.
```

Figure 4.17.1: nMADS

The weighting scheme for recurring or similar gaps, $v(g') = 2v(g)$, or *double*, can also be $v(g') = v(g) + 1$, or *add-one*. This is really dependent upon the fidelity and speed required, or if the front has large "true" gaps. This scheme is necessary, however, so that the algorithm does not stagnate on “true” gaps. If gaps persist at the conclusion of the algorithm, nMADS can be run further, or another technique can be used as discussed in this research. All that is required is a notion of indifference in each objective. This should not be too difficult to establish assuming utopia and nadir points are estimated, there is some knowledge of the system, or a predetermined percentage of the utopia and nadir point range can be selected for implementation (so that when MADS estimates the utopia and nadir points, indifference values can be automatically generated).

For clarification, Figure 4.17.2 shows exactly what nMADS is doing. The first iteration uses the solutions corresponding to the utopia point to identify and fill gaps in the next iteration. nMADS is then able to fill in more of the Pareto front and identify more missing areas. The gaps are filled by essentially moving along some path between the endpoints, where that path meets the Pareto front. Therefore, gaps that do not satisfy many objective indifference values could require multiple nMADS paths to be filled. However, noise and the poll and search steps in MADS and GPS also help to possibly get points not on a path, aiding in filling in the gap. The algorithm does not re-check for gaps until a set of gaps have been processed by the algorithm. The intent of this is to save time on large problems (because the gap algorithm and dominance check can become expensive with a very large dataset).



(a) First Iteration | (b) Second Iteration

Figure 4.17.2: NMADS Iterations

nMADS results follow for all test problems, as well as two additional problems with four objectives and eight objectives. The automated algorithm is used first, followed by a single nMADS application on any remaining gaps.

4.18.3. Results. For these runs, nMADS was conducted using a $c = 0.5$ in the gap algorithm, a noise level set to +/-0.5% of the nadir point components, and two replications with a limit of 500 function evaluations used to find the utopia point. Recall that the limit on the number of function evaluations has to be somewhat high because ranking and selection is used (4 replications of each point), and too low a limit prevents MVPS-RS/ MVMADS-RS from evaluating enough points. To fill gaps, a limit of 150 function evaluations was used, unless noted otherwise. To reduce computational time and the effect of noise, the nMADS-RS algorithm uses the mean found by ranking and selection (in this case, the mean of 4 evaluations). The reader should keep in mind that results are varied, but those shown here were representative of many runs.

Walston's [70] results are shown next to the nMADS results as a point of comparison. A summary of the nMADS results and settings is shown in Table 4.17.1. An asterisk denotes GPS-RS was used (although for generality the approach is still termed nMADS), the *Formulation* letter denotes the type of single-objective formulation used (*N*: normalized, *P*: product), and *FEvals* denotes the number of total function evaluations used, where SMOMADS is an estimate based on the number of design levels used. CPU time is included for nMADS, for a 3GHz, 3GM RAM machine on the AFIT network, to show the effiency of the algorithm.

The nMADS approximation (using GPS-RS here) for Viennet4 is shown in Figure 4.17.3. The estimated utopia and nadir points are shown in red. Once the automated nMADS algorithm finished, a value of $c = 0.25$ was used to identify four remaining gaps that were clear visually, and normalized formulations were used to fill those gaps, as well as three new gaps in a second additional iteration.

Table 4.17.1: nMADS Results

Problem	Formulation	Weighting Scheme	nMADS FEvals	SMOMADS FEvals (≤)	CPU Time (s)
Viennet4	N*	Add-1	8037	2104500	47
Viennet3	N*	Add-1	7852	2048000	52
Tamaki	N	Add-1	35306	72500	266
Poloni	N	Double	3998	5136000	26
Dias Γ1	N*	Double	27548	348500	95
Dias Γ2	N*	Double	27348	312500	151
Fonseca F1	P*	Add-1	5668	5018000	37
Schaffer F3	N*	Double	3153	5625000	22
Srinivas	N	Add-1	2278	348500	13
DTLZ7	N*	Double	5708	18000	41
Disk Brake	N	Add-1	9708	54000	45

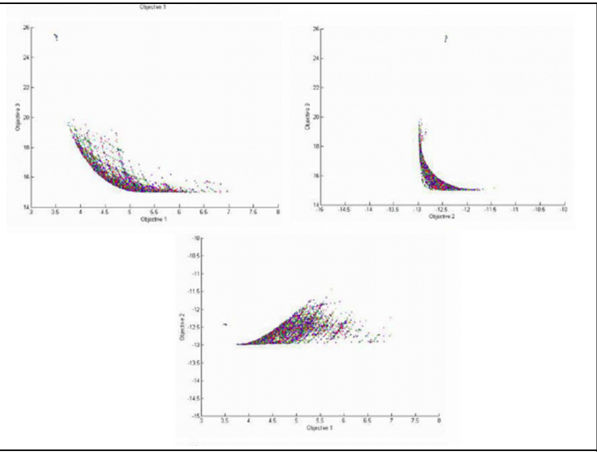
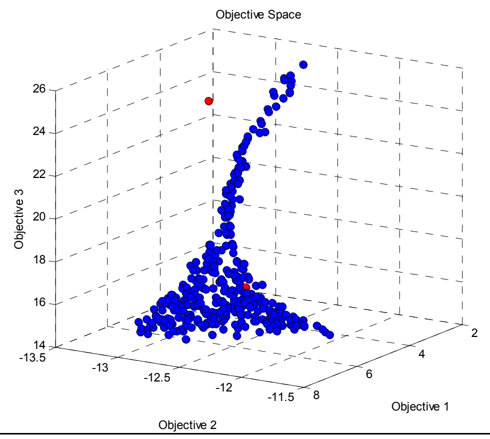
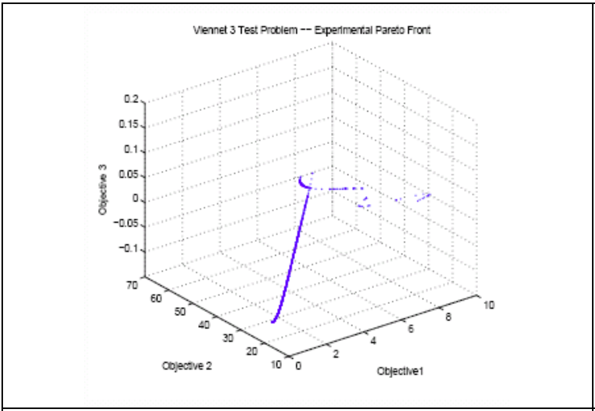
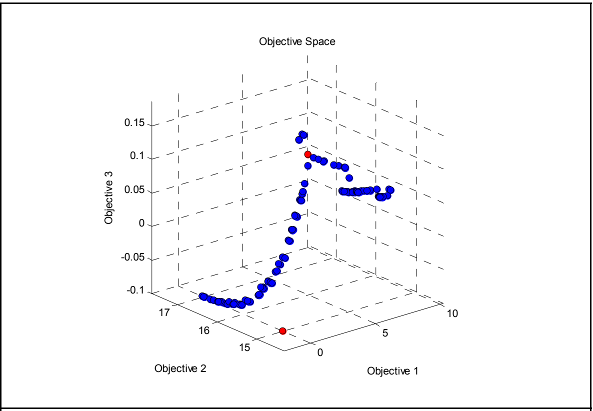
	
(a) Walston	(b) nMADS

Figure 4.17.3: Viennet4

The Viennet3 nMADS approximation is shown in Figure 4.17.4. In some cases on this problem, a gap near the high in Objective 3 is only very gradually filled. On this particular run it was filled quickly. Again, the reader should keep in mind, for three objectives, approximately 3000 of the evaluations were used to find the utopia point (3 objectives, two replications, 500 function evaluations limit).



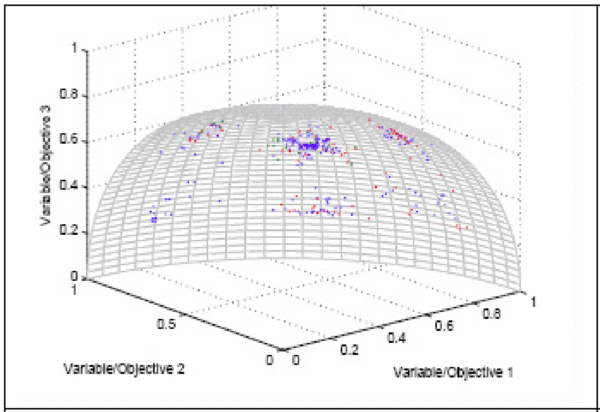
(a) Walston



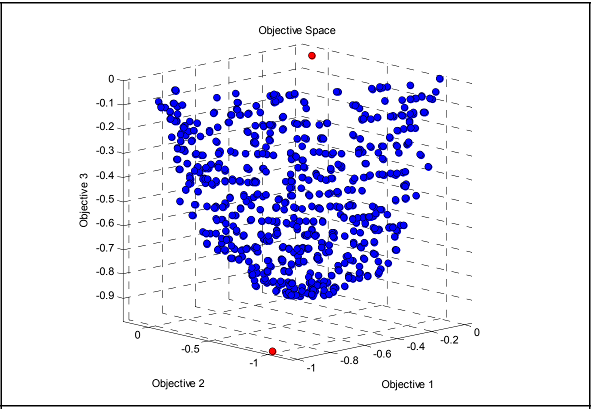
(b) nMADS

Figure 4.17.4: Viennet3

The Tamaki nMADS approximation is shown in Figure 4.17.5. Here, a limit of 250 evaluations was used for the gaps. This higher limit seemed to save evaluations in the long run on this problem. Two applications of nMADS were required after the automated algorithm. nMADS resulted in a near-perfect spread and distribution of solutions, while previous results had many portions missing (Recall, the plot in Figure 4.17.5(a) was shown as maximizations, and the same is true of Figure 4.17.6(a)).



(a) Walston



(b) nMADS

Figure 4.17.5: Tamaki

The nMADS approximation for the Poloni problem is shown in Figure 4.17.6. Approximately 2000 of these evaluations were used to estimate the utopia point (2 objectives, 2 replications, 500 function evaluations). All gaps were filled by the automated algorithm.

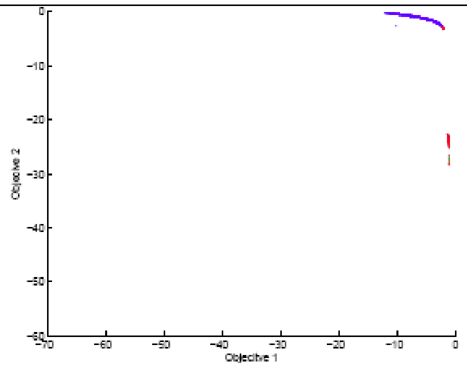
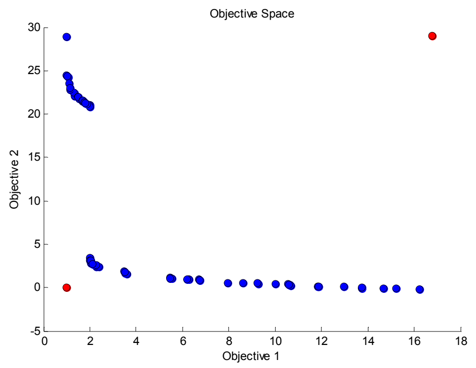
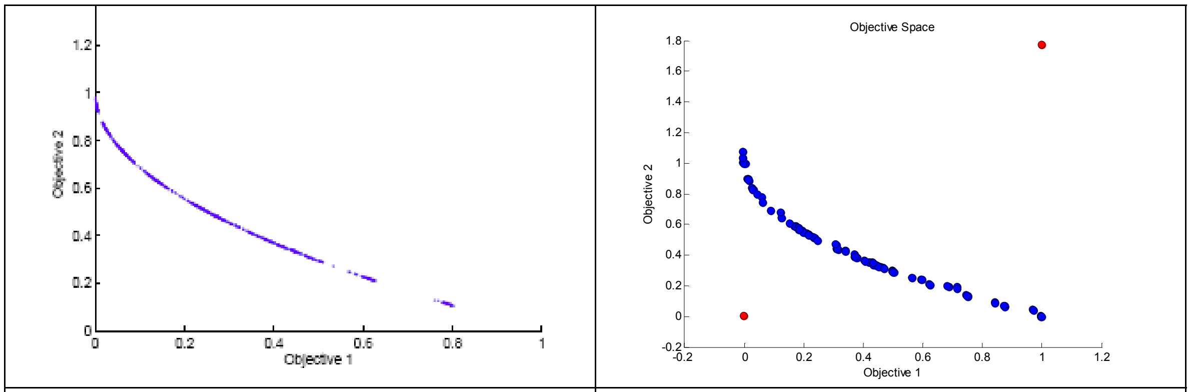
	
(a) Walston	(b) nMADS

Figure 4.17.6: Poloni

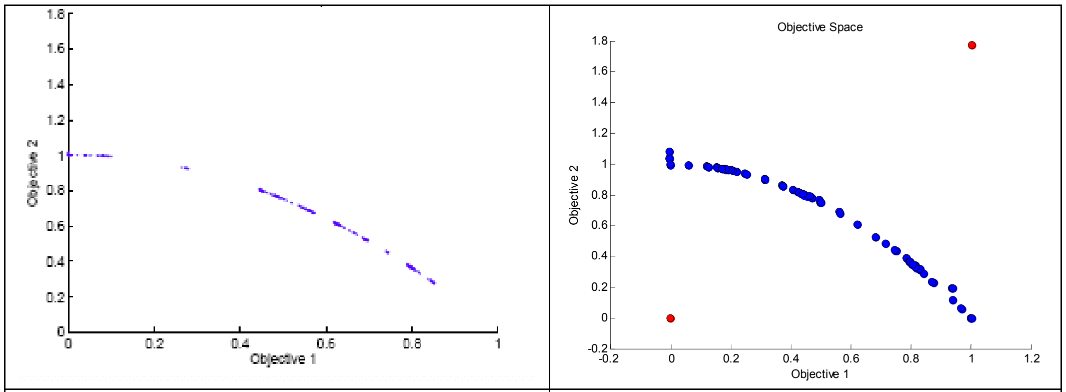
The Dias Γ1 nMADS approximation is shown in Figure 4.17.7. Here a 2000 function evaluation limit was used for each gap because there are 30 decision variables, and 500 evaluations are not enough to search appropriately. This large number of evaluations was also the reason for using the *double* weighting scheme. Clearly the nadir point was over-estimated due to noise, but the algorithm was indifferent to that fact. There is an interesting additional observation with respect to this problem. Considering the uniform design with AR levels and 36 samples replicated twice, and assuming a similar estimation of the utopia point being performed, SMOMADS would require 38000 function evaluations. Performance of nMADS may be variable due to noise, such that in a very large number of variables the SMOMADS approach, although still likely not as good as nMADS, becomes more reasonable.

The Dias Γ2 nMADS approximation is shown in Figure 4.17.8. Again, a 2000 function evaluation limit was used for each gap. All gaps were filled by the automated algorithm. In a few runs, a very high estimate for the second objective nadir component was found (~6). This caused the algorithm to try and fill a gap that was really not present.



(a) Walston | (b) nMADS

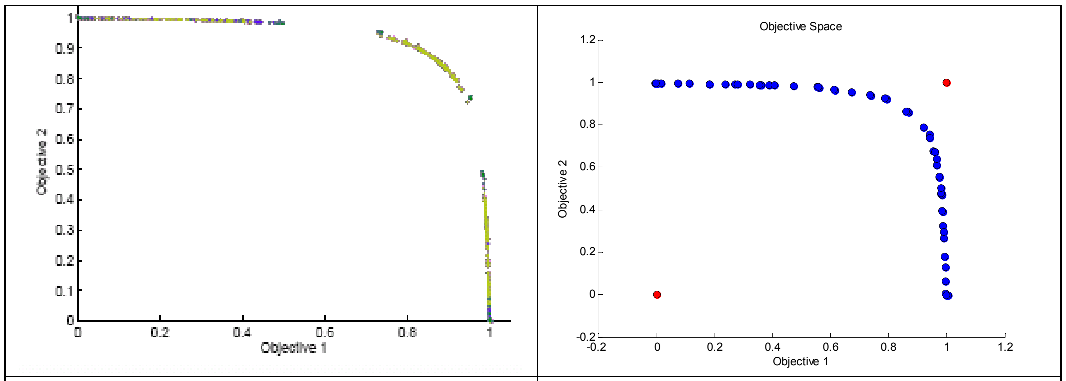
Figure 4.17.7: Dias Γ1



(a) Walston | (b) nMADS

Figure 4.17.8: Dias Γ2

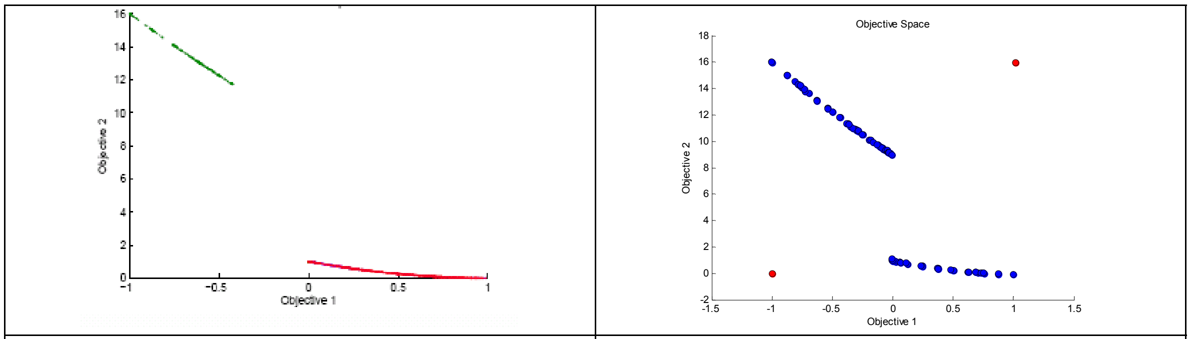
The Fonseca F1 nMADS approximation is shown in Figure 4.17.9. A limit of 150 function evaluations was used to fill gaps. Walston's plot, as shown in Figure 4.17.9(a), may be incomplete. In Walston's work [70], there were many, many data points and the portions that appear to be missing may have simply not loaded due to insufficient computing resources. All but three gaps were filled by the automated algorithm for this problem. Running this problem five times, a standard deviation of approximately 900 function evaluations occurred. This difference occurs due to noise in the objectives.



(a) Walston | (b) nMADS

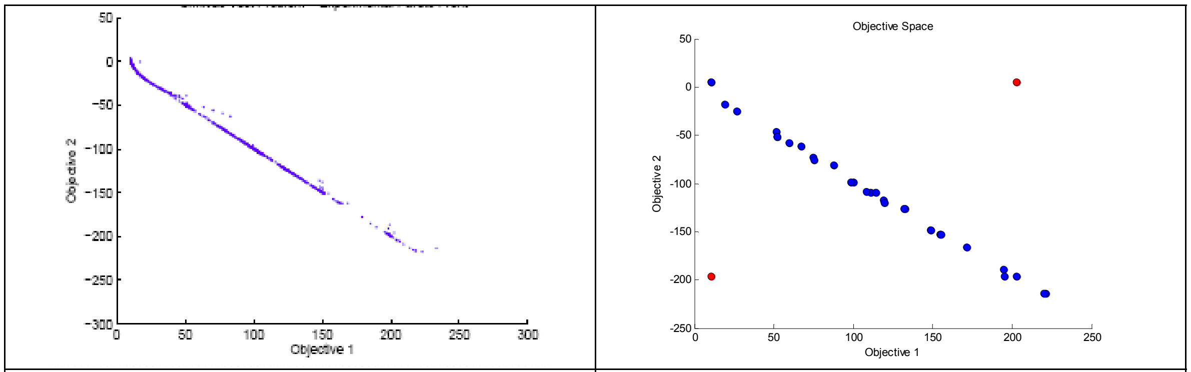
Figure 4.17.9: Fonseca F1

The Schaffer F3 nMADS approximation is shown in Figure 4.17.10. The *double* weighting scheme was used due to a known gap. All gaps but one, excluding the valid gap, were filled by the automated algorithm.



(a) Walston | (b) nMADS

Figure 4.17.10: Schaffer F3



(a) Walston | (b) nMADS

Figure 4.17.11: Srinivas

The Pareto approximation for Srinivas is shown in Figure 4.17.11. All gaps were filled by the automated algorithm. The indifference values, (25,26), are clearly satisfied and so a front with more points would require finer indifference values.

The DTLZ7 nMADS approximation is shown in Figure 4.17.12. All gaps, not including the true, were filled by the automated algorithm. Those points high in Objective 2 in Figure 4.17.12(b) are not dominated, rather they are ever so slightly less in Objective 1. Not shown in Figure 4.17.12(b) is a point very high in Objective 1 that was generated due to noise. These high noise points often occurred. However, on many runs of this problem, the DV values with which these points occurred were also unique, meaning that any kind of check to eliminate such points using objective function value or DV values could falsely remove points in the general case.

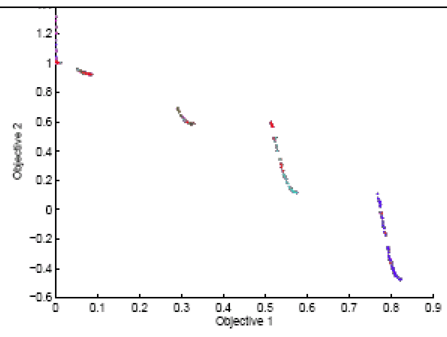
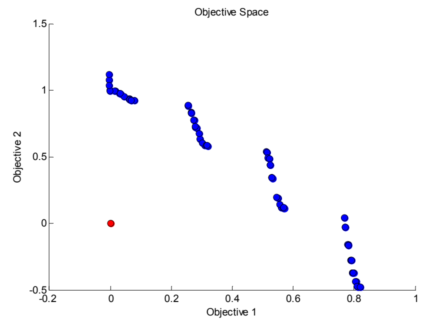
	
(a) Walston	(b) nMADS

Figure 4.17.12: DTLZ7

The Disk Brake approximation is shown in Figure 4.17.13. A limit of 250 function evaluations was used to fill gaps, taking into account that this is a mixed variable problem. 9,708 function evaluations were used versus 108 SMOMADS test points (54,000) from Walston's work. All but one gap was filled by the automated algorithm. This gap was near the high in Objective 1, and probably could have been filled faster by allowing more function evaluations. It took several tries to completely fill this gap, although improvement was always achieved. Note the better spread in the nMADS solution than Walston's previous work [70].

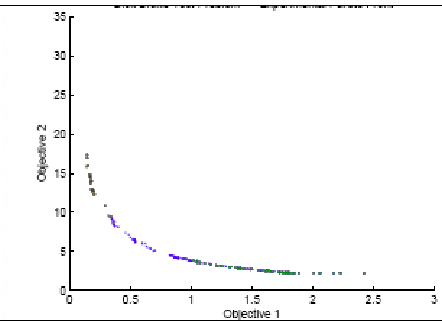
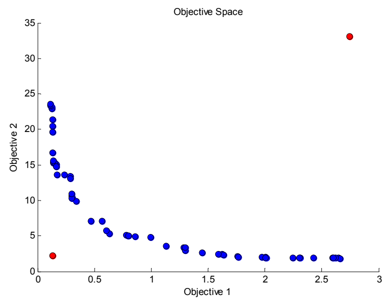
	
(a) Walston	(b) nMADS

Figure 4.17.13: Disk Brake

4.18.4. More Than 3 Objectives. Also tested here are a four-objective problem and eight-objective problem from [20]. Such large problems are likely impractical using SMOMADS. The eight-objective problem is shown in Equation 4.1. The four objective problem is the same but without the final four objectives. Indifference values and noise are shown in Table 4.17.2. These were based on utopia and nadir points from [20].

$$
\begin{aligned}
\min\ F_1(x,y) &= \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3 \\
F_2(x,y) &= \frac{(x+y-3)^2}{175} + \frac{(2y-x)^2}{17} - 13 \\
F_3(x,y) &= \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} + 15 \\
F_4(x,y) &= \frac{(3x+y+9)^2}{34} + \frac{(x+1)^2}{15} + 29 \\
F_5(x,y) &= \frac{(4x-y-4)^2}{22} - \frac{(y-1)^2}{5} - 17 \\
F_6(x,y) &= \frac{(y+14)^2}{8} + \frac{(x+y)^2}{10} + 64 \\
F_7(x,y) &= \frac{(17-x-y)^3}{995} + \frac{(8y-5x)}{65} \\
F_8(x,y) &= \frac{(7+2x+5y)}{5} + \frac{(y-3x)^3}{235}
\end{aligned}
\tag{4.1}
$$

subject to

$$4x + y - 4 \le 0$$

$$-1 - x \le 0$$

$$x - y - 2 \le 0$$

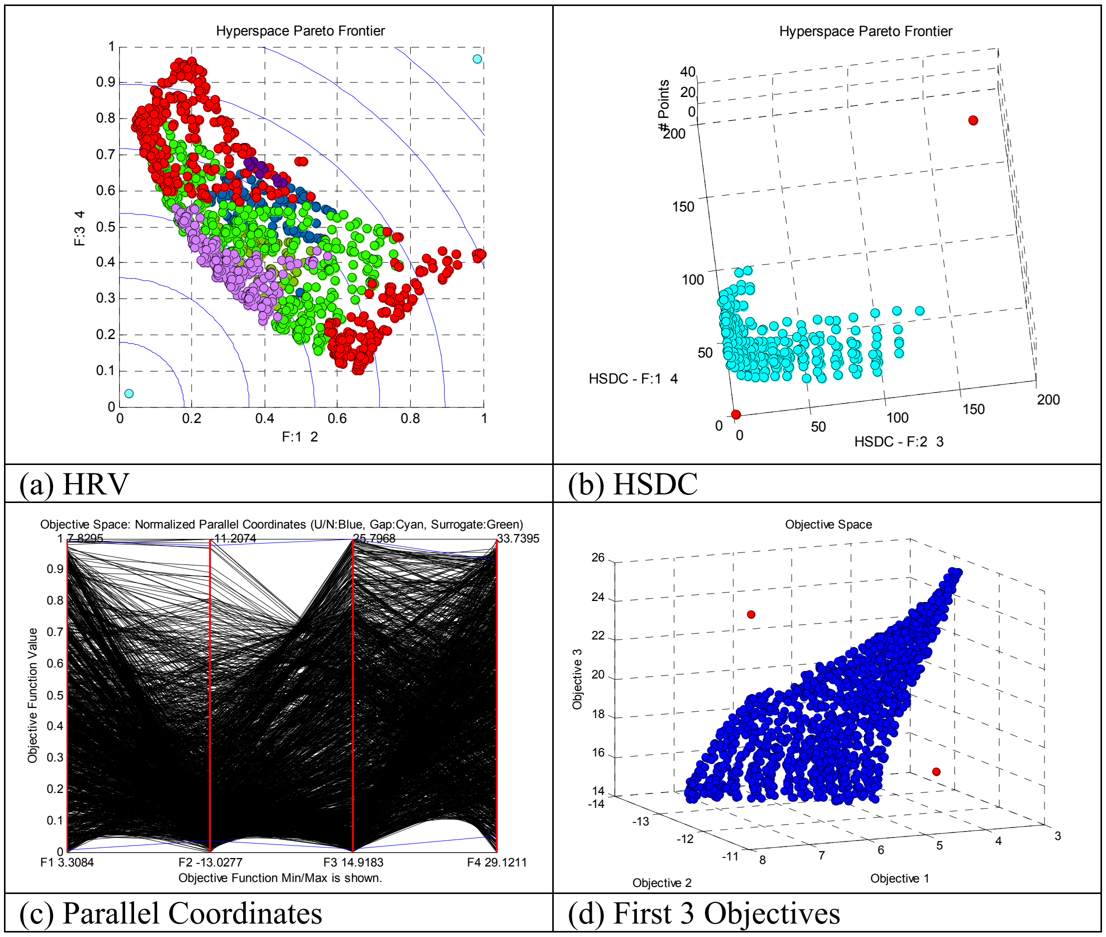
$$-4 \le x, y \le 4$$

Table 4.17.2: 4 and 8 Objective Problem

	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5	Obj 6	Obj 7	Obj 8
Indifference	0.42	0.14	1	0.44	0.73	2.4	0.58	0.74
Noise	0.08	0.12	0.25	0.34	0.09	1.05	0.09	0.06

On the four-objective problem, GPS-RS, the *double* weighting scheme, and normalized formulation were used. A limit of 120 function evaluations was used to fill gaps. Initially, approximately 11286 function evaluations were used by the automated nMADS algorithm, but 9 gaps remained according to the gap algorithm. Filling in the front until no gaps remained, a total of 15246 evaluations (including the 11286) were used. This number is higher than previous problems because of the increase in the number of objectives. Computational time was 201 seconds for the initial approximation and 347 seconds total. The *n*-dimensional visualizations from Figure 4.17.14 show that if any gaps do remain, they are relatively small. In looking at the objectives three at a time, as in Figure 4.17.14(d), no obvious gaps were noted.

The eight objective problem was run using GPS-RS, a limit of 120 function evaluations, the *double* weighting scheme, $c = 0.5$, and both normalized and product formulations. Both nMADS approximations, shown as Figure 4.17.15(b) and Figure 4.17.16(b) are clearly better than the published [20], deterministic solution found by a genetic algorithm, shown as Figure 4.17.15(a) (1902 Pareto points found versus 625 published [20]).



(a) HRV

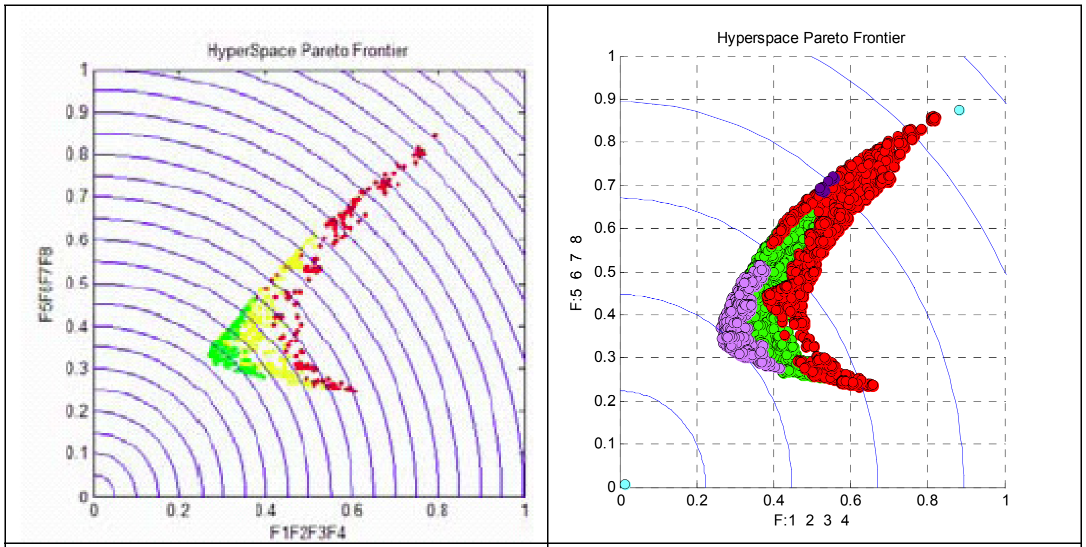
(b) HSDC

(c) Parallel Coordinates

(d) First 3 Objectives

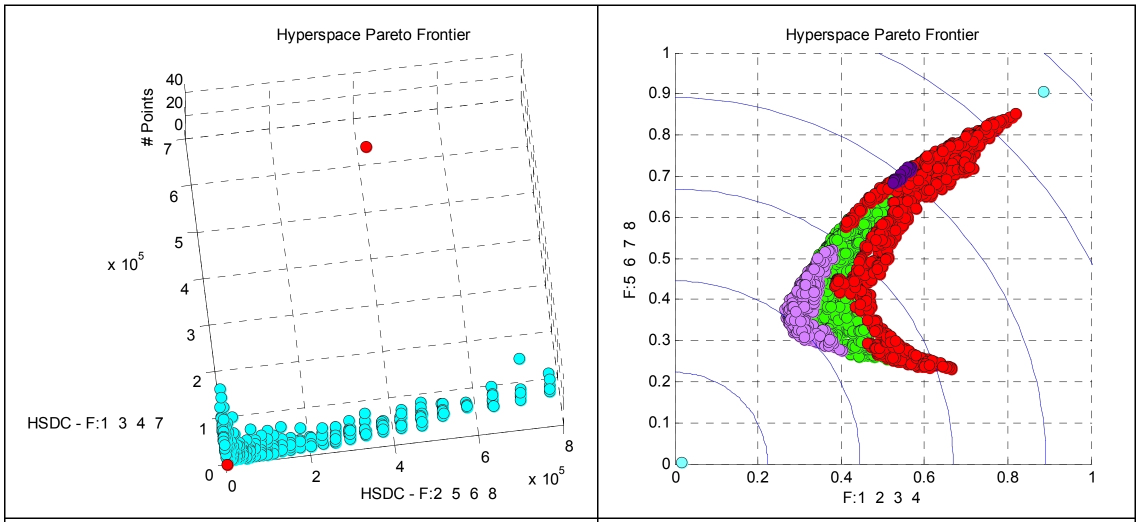
Figure 4.17.14: 4 Objective Problem

The product formulation was run using five replications to find the utopia point and used a total of 25726 function evaluations (the five utopia replications contributed significantly). Only two iterations were required to complete the front after the automated algorithm. The normalized formulation used only two replications to find the utopia point and finished in 498 seconds total. Using three iterations of formulations after the automated algorithm, a total of 14342 function evaluations were used. Again, no gaps had to be identified visually, and the other visualizations confirmed the completeness of the front. Four visualizations are shown in Figure 4.17.16 and Figure 4.17.17.



(a) Published Solution | (b) Automated NMADS HRV Product

Figure 4.17.15: 8 Objective Problem



(a) HSDC | (b) HRV

Figure 4.17.16: 8 Objective Problem (Normalized)

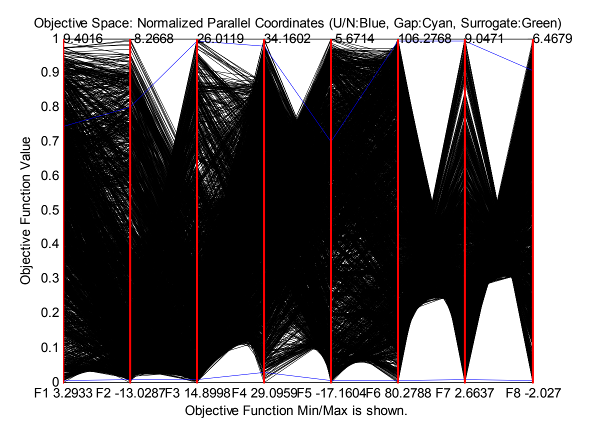
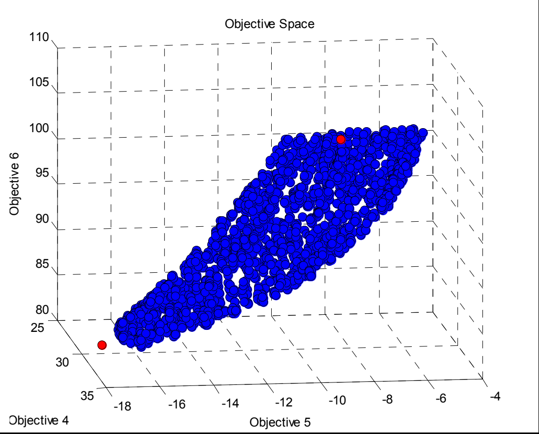
	
(a) Parallel Coordinates	(b) Objectives 4,5,6

Figure 4.17.17: 8 Objective Problem (Normalized)

4.18.4. Increased Noise. To test the performance of nMADS in an instance of increased noise, Tamaki was chosen because of its apparent difficulty during the course of the analysis.

Using +/-5% noise (5 times the noise value from Table 4.4.2), five replications to find the utopia point, and the same parameters as previously used on Tamaki, the initial approximation finished in 26112 function evaluations and in 169 seconds. Approximately 7500 of these were used to find the utopia point. Two replications would likely have been sufficient. The approximation, which is shown in Figure 4.17.18, is of relatively high quality. It appears, based upon this and results from Section 4.5, that SMOMADS and nMADS, and the dominance check, will not cause serious problems until the noise level is 10% or higher.

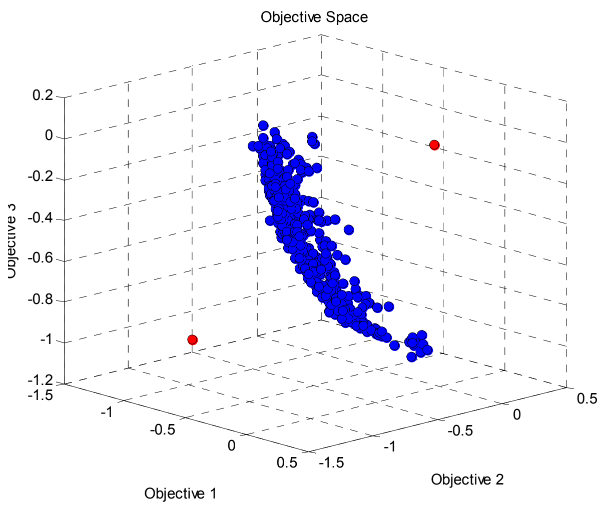


Figure 4.17.18: Tamaki

4.18. Final SMOMADS Algorithm

The algorithm shown in Figure 4.18.1 is presented as pseudo-code and includes a majority of the concepts covered in this thesis. Not all concepts actually need to be implemented. The sub-algorithms were previously shown in their complete mathematical detail, either in Chapter II, Chapter III, or previously in Chapter IV.

This version of SMOMADS allows estimation of the entire Pareto front, and in a more efficient manner. The nMADS algorithm can, in fact, be run as a sub-algorithm of SMOMADS to fill gaps after an initial aspiration and reservation level design completes.

1. Choose some number of LHS samples for the search step, and a limit for the number of function evaluations, for MVMADS-RS or MVPS-RS.
2. Estimate the utopia point and/or the nadir point using MVMADS-RS or MVPS-RS as appropriate. For confidence purposes, also estimate the nadir point using the GA. If the true points are known, simply input the utopia and nadir points.
3. Create an initial sequence of designs and ranges over which to sample aspiration and reservation levels, and choose indifference values.
4. The following is to be done iteratively:
    - a. Calculate metrics and visualizations.
    - b. Find gaps using the gap algorithm or *n*-dimensional visualization.
        - i. If there are no gaps found, no identifiable gaps in the visualizations, the entropy is approximately greater than or equal to 0.95, and the spread metrics are near 1, the approximation is finished.
        - ii. Else, for each gap, choose one:
            1. Run a sub-design using aspiration and reservation levels.
            2. Use a surrogate, selected by k-fold cross-validation to fill the gap or to form a surface.
            3. Use nMADS as a sub-algorithm to fill gaps.
        - iii. In the case of 1) or 3), choose some number of LHS samples and a limit for the number of function evaluations for MVMADS-RS or MVPS-RS.

Figure 4.18.1: SMOMADS

Many conclusions can be drawn from the Chapter IV results. Furthermore, these conclusions provide more broad observations. These conclusions and observations are presented in Chapter V, as are recommendations for future research.

V. Conclusions & Recommendations

Due to the depth of this research, and the length of this document, it may not have been all that straightforward to the reader how the results and analysis come together. General conclusions follow. After the conlusions, recommendations for future research are presented.

5.1. Conclusions

5.1.1. Utopia/Nadir Points. Finding the utopia and nadir points is not trivial. Using points from published results is not necessarily a good approach if those results come from heuristics like genetic algorithms. When solving for the utopia point using MVMADS-RS or MVPS-RS, those solutions that constitute the components of the utopia also correspond to the nadir point, albeit not the same components. This is clear because no point can dominate a component of the utopia. Therefore, there is no need to estimate the nadir point outright. Finally, the approaches taken in this research appear to be generally adequate when approximating these points, although with large noise levels, estimation could become difficult. Finding the utopia point can be expensive in terms of function evaluations.

5.1.2. MVMADS-RS/MVPS-RS. The algorithms presented in this research that use MVMADS-RS and MVPS-RS should be generally insensitive to the original starting iterate, and there was some indication that MVPS-RS is preferable for linearly constrained multi-objective problems. The number of LHS sites used within MVPS-RS or MVMADS-RS in the search step might have an impact on the success of the approximation, although using eight was typically fine on the problems tested here.

It was shown that a noise level of +/-5% of the nadir point should be acceptable when trying to approximate the Pareto front. However, +/-10% is probably too high to get the appropriate shape of some fronts with the current implementation of a dominance check. Increasing the noise also significantly increases computational time. Limiting the function evaluations was highly useful in reducing computational time, and did not adversely affect the Pareto front approximation. However, for some problems, most runs used close to the limit of 500 function evaluations. Therefore, the effect of reducing this limit further is unclear.

5.1.3. SMOMADS. For an initial Pareto front approximation, certain designs and aspiration/reservation level ranges were shown to be more useful than others. Specifically, design ranges should cover all Pareto objective function values that are desired, for both aspiration and reservation levels. Near-uniform and Hammersley sequence sampling designs not only provide an alternative to a full-factorial design, but also provide better approximations than using a full-factorial, with a large reduction in runs. CCDs showed the most promise in getting extreme solutions, but only because the space-filling designs do not include design levels at axial points or at the bounds of the range.

The nadir point and utopia point estimation play a large role only if the estimations are poor. If the estimations are approximate, the algorithms should be indifferent. There is no apparent advantage to using more than two replications of any design, unless surrogates are used. In this case, more data should result in better surrogates. Finally, using subsets of the component functions can generate extreme solutions, but selecting designs intelligently prevents the need to use the subsets, which can greatly increase the number of required runs. On problems with a large number of decision variables, SMOMADS may be a reasonable altenative, provided there is not a large number of objectives.

5.3.4. Surrogates. Surrogates can be a quick approach for finding remaining portions of the Pareto front after an initial approximation and the *k*-fold cross validation approach is performed. However, optimizing the surrogates in MADS requires more research from the perspective of the dominance check. Although the deterministic dominance check seems suitable when using optimizations performed entirely on true functions, it is less suitable in the case of surrogate data. Since surrogates have inherent error, optimizing these models is not guaranteed to lead to an exact Pareto solution. This implies a tolerance or some subjective, or probabilistic, measure to determine when a resulting solution is “close enough.”

The point generation method for surrogates does work well, however. Because the error of the models is low, generating a large number of points, inexpensively running them through the surrogates, and checking for dominance probably results in an accurate approximation. The only problem with this approach is that true gaps can be filled and solutions are not guaranteed to be Pareto optimal. Surrogate points from a single-objective formulation optimization should not be checked for dominance, however, because a formulation with noise could generate, for example, a cubic surface, that would eliminate the true optimal (recall the product formulation, which uses the maximum of zero and the negative squared difference of the reference component and objective function; noise can incorrectly generate zero values).

Decision variable surrogates have less error (even with fewer runs) than aspiration and reservation levels. However, aspiration and reservation level surrogates do improve with more data, and provide an advantage in that instead of fitting the true objective functions, they should fit the Pareto front. There is no apparent advantage to using coded values versus natural values for the predictors, and ordinary/weighted least-squares based surrogates appear to be of little value. Kriging and RBFs were most promising.

For the c parameter in RBFs, the mean distance between sites seemed to be the best value (there was no evidence to the contrary). Additionally, forcing a limit of $\bar{\theta} < 30$ in Kriging was of no value. Typically, those Kriging models that had large θs were either good models, or models such that enforcing the limit provided no advantage. Due to the success of other surrogate types in approximating the objective functions and/or Pareto front, there appears to be little value in pursuing a MARS surrogate. However, MARS could become more advantageous in the case of high noise levels.

5.3.5. Termination Criteria. The quality metrics presented are not necessarily the best choice to use as termination criteria. HD and AC are extremely expensive to compute once a moderate number of points are in the Pareto approximation. The spread metrics are a good way to determine if extreme solutions are achieved, but do not contain any information about the rest of the front. NDC and CL can be used to compare approximations but reveal nothing of the completeness. Furthermore, entropy runs into problems on discontinuous fronts, as the metric assumes that the entire projected space contains Pareto solutions. However, if the front is continuous and will fill the projected space (*e.g.*, Tamaki will, Viennet3 will not), and the estimates of the utopia and nadir point are good, then the metrics can be used in combination to determine a suitable termination criteria.

Alternatively, a notion of indifference values was used as the basis for an algorithm that iteratively attempts to find gaps in the n-dimensional objective space. nMADS then fills those gaps or helps to verify that the gap is a true gap, either standalone or as a sub-algorithm to SMOMADS. Further, the n-dimensional visualization can be used to identify any gaps that may go undetected by the gap algorithm. In practice, these gaps may not be identifiable using the *n*-dimensional visualizations if they are relatively small compared to the objective space, but large gaps in the Pareto front can be avoided.

5.3.6. Single Objective Formulation and nMADS. The expansion of BiMADS to nMADS worked extremely well across all test problems. Neither single-objective formulation proved better than another. The nMADS algorithm from Figure 4.17.1 proved efficient and robust in solving up to eight objectives and shows promise as a useful algorithm in practice.

5.3.7. General Conclusions. In general, as the number of objectives and points increase, algorithms slow noticeably. This is likely unavoidable, although some areas of improvement in efficiency exist. However, on the problems in this research, efficiency was not necessarily a problem. For deterministic problems, the nMADS algorithm would effectively halve the number of evaluations and CPU times (or better) shown here for stochastic problems.

5.2. Recommendations for Future Research

Some areas still need to be investigated with repect to the SMOMADS and nMADS algorithms. Further, computational efficiency could be improved, both in coding and in reducing the number of function evaluations.

5.2.1. Algorithm Efficiency. There are areas of the algorithms that could be improved so as to decrease CPU time and the number of function evaluations. The dominance check begins to become expensive with thousands of data points. R&S should be tested with fewer than four replications per point. nMADS should be tested using only one gap endpoint starting iterate, although both are likely necessary.

Estimating the utopia point effectively using fewer function evaluations would be of great value. This estimation constituted a large portion of the nMADS function evaluations. Further, perhaps not all gaps have to be assessed in a given iteration during nMADS. However, the gap algorithm and dominance check should not be run after each gap assessment either (in an effort to reduce the number of gaps), unless the time to do so is less than the time to complete an iteration of nMADS.

5.2.2. Mixed Variable Nadir Point GA. The nadir point GA had trouble on the Disk Brake problem. This could be in part because the extreme solution in one objective is hard to achieve. However, it is also because the crossovers and mutations evaluated in this research were clearly not sufficient. Although MADS and GPS provide suitable alternatives (perhaps making the GA unnecessary), this area could be improved. One possibility is to use a nearest-neighbor approach like that found in MV-MADS/MVPS, or to ensure all discrete values are in the intial population.

5.2.3. The Gap Algorithm. The gap algorithm presented in Section 3.7 has its limitations, as were discussed. Some improvements were proposed that may be too time-consuming to be of value. In future research, an algorithm that can identify gaps in *n*-dimensional space, likely based on indifference values, in an efficient and *complete* manner, would be of great value and could greatly increase the effectiveness of nMADS. In a blackbox context, indifference values are not as easy to determine aside from: 1) using a predetermined percentage of the utopia and nadir points (which would be used once the utopia and nadir points are estimated to create the indifference values); 2) assuming some knowledge of the system; or 3) Estimating the utopia and nadir point prior to the mutli-objective optimization. An algorithm not based on indifference values would be useful as well.

5.2.4. NMADS Efficiency. An initial weighting scheme and strategy were created for nMADS. In future research, this scheme and the algorithm in general may be made more effective by selecting the gaps to fill in some other manner, perhaps in part by further limiting the number of times a recurring gap is selected.

5.2.5. Surrogates. Small amounts of error in multiple objectives sometimes prevent the optimizations from being able to perform well in a nMADS approach with surrogates. That is, the resulting points when evaluated by the true objective function are dominated. Specifically, the dominance check would require a method to accept reasonable dominated solutions into the true Pareto set, which is currently not done. A tolerance value could be based on the error estimates provided by cross-validation, but it is not clear how to account for noise that could potentially be very difficult to estimate and how to prevent the algorithm from accepting a bad solution in the general case. It may very well be that the decision-maker would have to accept some number of non-Pareto solutions.

5.2.6. Noise. In her recommendations for future research, Walston [70] discussed using a probability scheme to determine if a point is dominated, based on one present in MOCBA. From the present research, this may only be necessary if there are high levels of noise. It would be worthwhile to investigate using such a scheme, versus the current dominance check.

5.2.7. SMOMADS. Using all function evaluations from SMOMADS, like the nMADS algorithm, should be evaluated. This may or may not be useful.

5.2.8. MVMADS-RS. Convergence results for the stochastic, multi-objective, nonlinearly constrained case have not yet been rigorously proven and depend on Conjecture 3.3.10. from Walston [70]. A rigorous analysis of this conjecture is recommended.

Appendix A. Initial Analysis

A.1. Initial SMOMADS Runs

A.1.1. Test Approach. The runs done for this research were evolutionary. This was the first true set of batch runs, with the intent of determining how many replications might be required of a design, what design space to use on the aspiration and reservation levels, what impact different magnitudes of noise may have, and how sensitive the levels may be to using the true nadir point versus an overestimation (assuming the ranges are based off of utopia and nadir points).

Unfortunately, there were many issues with these runs for various reasons, to include finding the noise error and a rounding error within the entropy metric. Therefore, the most useful information to come from these runs was with repect to the number of replications of a design. Results follow in tables for a representative subset of the problems, again so that the length of this document would be reasonable.

These runs were conducted using a CCD and a uniform design with 20 samples. Times shown here include the time required to calculate metrics, although that time was minimal (as it was also recorded). Additionally, the metrics and gap information can be misleading if taken at face value, as there is no guarantee the front found contained extreme values. However, it can be noted that if there is a good estimation of the utopia and nadir points, the spread metrics in conjunction with the entropy metric and NDC are a good measure of the front.

A DOE approach was taken with the full set of data to test factors for significance using items in the first column of the table as a response. Significant factors, using an alpha of 0.05, are highlighted in gray. Unless otherwise mentioned, gray factors were significant in the coded and natural space. The data in the tables are averages of the runs that completed. The ranges, nadir point estimation, and noise often tested as significant.

However, again, these results are not shown due to duplication elsewhere in this thesis and because of the debugging issues in these particular runs.

"True" nadir points and utopia points for these runs were taken from Walston [70]. It is also necessary to mention that the entropy values in this section used a σ of 1/6. Gaps are presented in terms of Euclidean distance and time is in seconds. HD and AC are not presented due to their computational time for large numbers of points. AR refers to the design space used to create the aspiration and reservation levels where AR1 used $[f_i^g, 0.99\times mean(f_i^g, f_i^b)]$, $[1.01\times mean(f_i^g, f_i^b), f_i^b]$ (the mean was set to 10^{-5} here in the case of zero so that two levels could not be identical and result in an error inside the component achievement functions), AR2 used $\left[f_i^g, f_i^g + \left|f_i^g - f_i^b\right|/3\right]$, $\left[f_i^b - \left|f_i^g - f_i^b\right|/3, f_i^b\right]$, and AR3 used $\left[f_i^g - \left(2\cdot\left|f_i^g - f_i^b\right|\right)/5, f_i^g + \left(2\cdot\left|f_i^g - f_i^b\right|\right)/5\right]$, $\left[f_i^b - \left(2\cdot\left|f_i^g - f_i^b\right|\right)/5, f_i^b + \left(2\cdot\left|f_i^g - f_i^b\right|\right)/5\right]$. N refers to noise, where N*I* refers to adding *I* times 1% of the "true" nadir component as noise. NR*I* refers to using *I* replicates of the design. ND1 refers to using the "true" nadir point and ND2 refers to using the over-approximation.

A.1.2. Results.

Dias Γ1 & Disk Brake

	Dias Γ1				Disk Brake			
Measure	NR2	NR3	NR4	NR5	NR2	NR3	NR4	NR5
Bogus Pts	30.2	49	71	96.25	32.67	53.67	79.67	111.33
Entropy	0.96	0.97	0.97	0.97	0.95	0.95	0.95	0.95
OS	0.97	0.96	0.96	0.96	0.16	0.16	0.16	0.17
OS1	1.01	1.01	1.01	1.01	0.42	0.41	0.43	0.44
OS2	0.95	0.95	0.95	0.95	0.37	0.38	0.38	0.39
NDC	14.6	17.8	17.75	20.75	8.33	8.67	9.33	9.33
CL	2.92	3.35	4.23	4.13	4.78	6.24	6.93	7.41
Time	2942	4280	5168	6246	638	773	1032	1276
Largest Gap	0.31	0.26	0.33	0.20	1.05	0	0	0
Avg Gap	0.22	0.21	0.20	0.15	1.05	0	0	0
# Gaps	3.6	2.8	4.25	3.5	1	0	0	0

For Dias Γ1, the number of replicates provided a statistically significant difference in overall spread for Objective 1 (raw data was not rounded to two decimal places). However, practically, this difference was minimal. Note that spread is greater than 1 due to noise. Also, increasing the replicates provided more distinct points, but also increased clustering and time (obviously). In coded space, NDC was not significantly different. Although not shown, the interaction of AR range and number of replicates was significant positively, but was not as large as either AR type or the number of replicates as main effects.

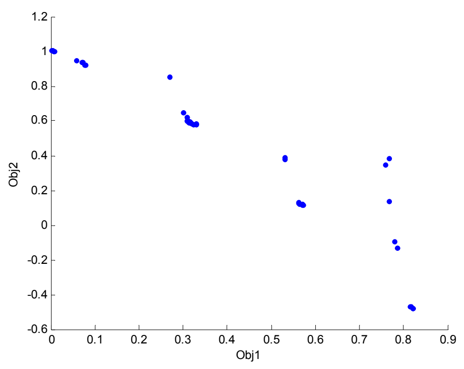
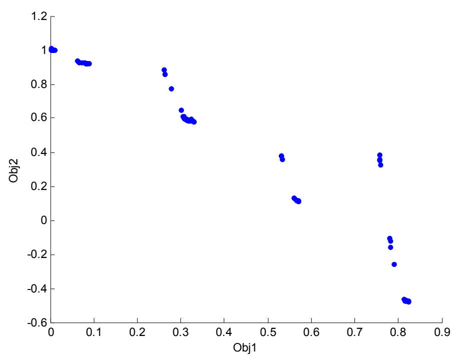
For Disk Brake, increasing replications provides a statistically, but not practically, significant increase in OS. Additionally, this increase in spread also causes an increase in clustering and time. Of course, zero gaps only means that there are no gaps within the bounds of the points found. For clustering, noise and number of replicates had a significant negative interaction. This was also true for NDC and time.

DTLZ7 & Fonseca F1

	DTLZ7				Fonseca F1			
Measure	NR2	NR3	NR4	NR5	NR2	NR3	NR4	NR5
Bogus Pts	30.67	59	83.33	105.67	36.08	57.92	82	105.08
Entropy	0.99	0.99	0.99	0.99	0.92	0.94	0.95	0.96
OS	0.73	0.72	0.72	0.73	0.99	1.00	1.00	1.00
OS1	0.98	0.96	0.97	0.98	1.00	1.00	1.00	1.00
OS2	0.74	0.75	0.75	0.75	1.00	1.00	1.00	1.00
NDC	11.33	12.67	12.67	12.33	7.08	7.67	8.58	9.08
CL	3.68	3.89	4.97	6.03	3.27	3.63	3.98	4.43
Time	75	114	151	191	228	343	467	566
Largest Gap	0.37	0.33	0.31	0.29	0.71	0.61	0.58	0.55
Avg Gap	0.26	0.25	0.24	0.22	0.27	0.31	0.33	0.42
# Gaps	4.33	4	3.67	3.67	3.58	4.58	4.42	4.5

On DTLZ7, clustering increased with more replicates and the number and size of gaps did not decrease practically. For average gap size, there was a small, significant, negative interaction between noise and number of replicates. The plots for low noise with two and five replicates follow. The additional replications provided no marked benefit. The same was true at any level of noise and between levels of noise.

Increasing replicates for Fonseca F1 provided more distinct points and a reduction in gap size, but did not improve spread or entropy, or even the number of gaps (statistically speaking). In the case of average gap size, the interaction between AR type and number of replicates was significant, but small.

	
(a) 2 Replicates	(b) 5 Replicates

DTLZ7 Pareto Fronts

Poloni & Srinivas

	Poloni				Srinivas			
Measure	NR2	NR3	NR4	NR5	NR2	NR3	NR4	NR5
Bogus Pts	32.04	54.21	74.38	97.04	17.33	31.08	45.83	62.79
Entropy	0.93	0.94	0.94	0.94	0.98	0.98	0.98	0.98
OS	0.09	0.11	0.11	0.12	0.35	0.36	0.36	0.35
OS1	0.24	0.24	0.25	0.27	0.52	0.52	0.53	0.52
OS2	0.34	0.43	0.44	0.41	0.59	0.59	0.59	0.59
NDC	4.29	4.75	4.88	5.08	10.25	10.17	11.00	10.92
CL	6.03	6.49	7.95	8.79	3.90	5.40	6.18	7.27
Time	234	340	456	567	358	570	780	952
Largest Gap	13.64	17.62	17.63	16.39	67.70	68.27	65.52	62.05
Avg Gap	11.83	16.11	16.78	14.81	57.95	58.82	58.72	55.02
# Gaps	1	1.17	1.08	1.13	1.75	1.5	1.46	1.58

For Poloni, increasing the number of replicates provides no real advantage. On Srinivas, increasing the number of replicates beyond two did not clearly provide benefit. For NDC, only the nadir point and number of replicates were significant in coded variables. For time, AR type and number of replicates had a large, positive, significant interaction.

Using any more than two replicates seems to provide no true advantage, although it is true that in brute forcing a large number of design levels through SMOMADS, there is some probability that points will be found on the front that would not have been with fewer runs. Future runs took this into account so as to save time and computer resources. Additionally, future runs did not take the “every possible combination” approach. All of the forementioned runs were completed on the Intel machines. There were more runs conducted between those just presented and those presented in Chapter IV, however, they had to be excluded for purposes of brevity.

Bibliography

1. M. A. Abramson, C. Audet, and J.E. Dennis, Jr., "Filter pattern search algorithms for mixed variable constrained optimization problems," *Pacific Journal of Optimization*, to appear. Also appears as Technical Report TR04-09, Department of Computational and Applied Mathematics, Rice University; Houston, Texas, 2004.

2. M.A. Abramson, "NOMADm version 4.02." Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA, 2006.

3. M.A. Abramson, "Radial basis functions estimator version 1.0." Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA, 2006.

4. M.A. Abramson, "Nadaraya-Watson estimator version 3.3." Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA, 2005.

5. M.A. Abramson, *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, August 2002.

6. M.A. Abramson, "Second Order Behavior of Pattern Search." *SIAM Journal on Optimization*, vol. 16, no.2, pp315-330, 2005.

7. G. Agrawal, K.E. Lewis, and C.E. Bloebaum, "Hyperspace Pareto Frontier for Intuitive Visualization of Multiobjective Optimization Problems," [www.gautamagrawal.com/PDFs/ASME_Revised_Version_Aug2006.pdf; accessed 6-Dec-2007].

8. T. Allen and L. Yu, "Low cost response surface methods for and from simulation optimization," *Proceedings of the 2000 winter simulation conference*, 2000.

9. ARANZ, "Using RBFs," 2002. [http://www.aranz.com/research/modelling/theory/rbffaq.html; accessed: 8 Oct 2007].

10. AT&T, "Orthogonal Arrays," 2007. [http://www.research.att.com/~njas/oadir/index.html; accessed 23-Sept-2007].

11. C. Audet and J. Dennis, Jr., "Pattern search algorithms for mixed variable programming." *SIAM Journal on Optimization*, vol. 11 no. 3 pp. 573-594, 2000.

12. C. Audet and J. Dennis, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 188-217, 2006.

13. C. Audet, G. Savard, and W. Zghal, "Multiobjective Optimization through a Series of Single-Objective Formulations," *Les Cahiers du GERAD*, G-2007-05, 2007. *SIAM Journal on Optimization*, to appear.

14. C. Audet and J.E. Dennis, Jr., "Analysis of Generalized Pattern Search." *SIAM Journal on Optimization*, vol. 13, no. 3, pp889-903, 2003.

15. C. Audet and J.E. Dennis Jr., "A Pattern Search Filter Method for Nonlinear Programming without Derivatives." *SIAM Journal on Optimization*, vol. 14, no. 4, pp980-1010, 2004.

16. D. Bethea, *Improving Mixed Variable Optimization of Computational and Model Parameters Using Multiple Surrogate Functions*. Master's Thesis. Air Force Institute of Technology, March 2008.

17. B. Baxter, *The interpolation theory of radial basis functions*. PhD Dissertation. Cambridge University, August 1992.

18. L. Berke, S. Patnaik, and P. Murthy, "Application of Artificial Neural Networks to the design optimization of aerospace structural components." NASA Technical Memorandum 4389, 1993.

19. ChemicalProcessing.com, "Rethink experiment design," 2006. [http://www.chemicalprocessing.com/articles/2006/166.html?page=2; accessed 27-Sept-2007].

20. P. Chiu and C. Bloebaum, "Hyper-Radial Visualization for decision-making in Multi-objective Optimization," in *46th AIAA Aerospace Sciences Meeting and Exibit*, 2008.

21. H. Chung and J. Alonso, "Comparison of approximation models with merit functions for design optimization" *8th AIAA/USAF/NASA/ISSMO Symposium on multidisciplinary analysis and optimization*, AIAA 2000-4754, 2000.

22. F.H. Clarke, *Optimization and Nonsmooth Analysis*, Philadelphia, PA, USA: Classics in Applied Mathematics Vol. 5, SIAM, 1990.

23. A. R. Conn, N.I.M. Gould, and P.L. Toint, "A Globally Convergent Augemented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds." *SIAM Journal on Numerical Analysis*, vol. 28, no. 2, pp545-572, 1991.

24. DACE, "Focus on Kriging and radial basis functions," 2004. [http://www2.imm.dtu.dk/~hbn/dace/surrogate.pdf; accessed: 8 Oct 2007].

25. I. Das and J. Dennis, “Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems.” *SIAM Journal on Optimization*, vol. 8, pp631-637, 1998.

26. C. Davis, “Theory of Positive Linear Dependence.” *American Journal of Mathematics*, vol. 76, no. 4, pp733-746, 1954.

27. K. Deb, S. Chauhuri, and K. Miettinen, “Towards estimating nadir objective vector using evolutionary approaches,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, 2006.

28. C. de Boor and A. Ron, “On multivariate interpolation,” *Constr. Approx.*, vol. 6, pp. 287-302, 1990.

29. M. Ehrgott and D. Tenfelde-Podehl, “Computation of ideal and nadir values and implications for their use in MCDM methods.” *European Journal of Operational Research*, vol. 151, pp. 119-139, 2003.

30. A. Farhang-Mehr and S. Azarm, “An information-theoretic entropy metric for assessing multi-objective optimization solution set quality,” *Journal of Mechanical Design*, vol. 125, no. 4, pp. 655-663, 2003.

31. R. Fletcher and S. Leyffer, “Nonlinear Programming Without a Penalty Function.” *Mathematical Programming*, vol. 91, pp239-269, 2002.

32. A. Freitas, “A critical review of multi-objective optimization in data mining: a position paper,” *SIGKDD Explorations Newsletter*, vol. 6, no. 2, pp.77-86, 2004.
33. J. Friedman, “Multivariate Adaptive Regression Splines,” *The Annals of Statistics*, vol. 19, no. 1, pp 1-67, 1991.

34. L. Gámiz-Gracia and et. al, “Use of highly efficient Draper-Lin small composite designs in the formal optimisation of both operational and chemical crucial variables affecting a FIA-chemiluminescence detection system,” *Talanta*, vol. 60, no. 2, pp. 523-534, 2003.

35. A. Giunta, S. Wojtkiewicz Jr., and M. Eldred, “Overview of modern design of experiments methods for computational simulations,” AIAA Paper 2003-0649, 2003.
36. L. Imhof, “Optimum designs for a multiresponse regression model,” *Journal of Multivariate Analysis*, vol. 72, pp. 120-131, 2000.

37. S. Jeong, S. Obayashi, and K. Yamamoto, “A Kriging-based probabilistic optimization method with an adaptive search region,” *Engineering Optimization*, vol. 38, no.5, pp. 541-555, 2006.

38. R. Kohavi, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Validation,” in *International Joint Conference on Artificial Intelligence*, 1995.

39. H. Koivo, “Neural Networks: Basics using MATLAB neural network toolbox,” [http://www.control.tkk.fi/Kurssit/AS-74.3115/Materiaali/Material2007/NN_Basics_2006.pdf; accessed 17-Oct-2007].

40. R.M. Lewis and V. Torczon, “Rank Ordering and Positive Bases in Pattern Search Algorithms.” Technical Report ICASE 96-71, NASA Langley Research Center, 1996.

41. R.M. Lewis and V. Torczon, “Pattern Search Methods for Bound Constrained Minimzation.” *SIAM Journal on Optimization*, vol. 9, no.4, pp1082-1099, 1999.

42. R.M. Lewis and V. Torczon, “Pattern Search Methods for Linearly Constrained Minimzation.” *SIAM Journal on Optimization*, vol. 10, no. 3, pp917-941, 2000.

43. R. M. Lewis and V. Torczon, “A Globally Convergent Augmented Lagrangian Pattern Search Algorithm for Optimization with General Constraints and Simple Bounds.” *SIAM Journal on Optimization*, vol. 12, no. 4, pp1075-1089, 2002.

44. J. Li and et. al, “Comparing prediction variance performance for variations of the central composite design for 6 to 10 factors,” Working Paper, Arizona State University, 2006.

45. S. Lophaven, H. Nielsen, and J. Sondergaard, “DACE: A MATLAB Kriging toolbox,” [http://www2.imm.dtu.dk/~hbn/dace/], 2002.

46. C. Ma and K. Fang, “A new approach to construction of nearly uniform designs,” *International Journal of Materials and Product Technology*, vol. 20, pp. 115-126, 2004.

47. M. Makowski, “Methodology and a modular tool for multiple criteria analysis of lp models.” Working Paper 94-102, International Institute for Applied Systems Analysis. Laxonburg, Austria. 1994.

48. D. Montgomery, *Design and Analysis of Experiments*, 6th ed. Hoboken, NJ, USA: John Wiley & Sons, 2005.

49. D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*, 4th ed., Hoboken, NJ, USA: John Wiley & Sons, 2006.

50. R. Myers and et. al, “Response surface methodology: a retrospective and literature survey,” *Journal of Quality Technology*, vol. 36, no. 1, pp. 53-77, 2004.

51. R. Myers and D. Montgomery, *Response Surface Methodology*, 2nd ed., New York, NY, USA: John Wiley & Sons, 2002.

52. J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., New York, NY, USA: Springer Science+Business Media, 2006.

53. "Parallel Coordinates," Cornell University, 2001. [http://www.nbb.cornell.edu/neurobio/land/PROJECTS/Inselberg/; accessed 17-Jan-2008].

54. M. Perry, "Response Surface Methodology Lecture Notes," Florida International University, Miami, FL, 2007.

55. Politecnico di Torino, "Computer experiments: Kriging methodology and sequential designs," 2007. [http://web.econ.unito.it/deinde07/downloads/contributed/g_vicario.pdf; accessed: 7-Oct-2007].

56. R. Regis and C. Shoemaker, "Parallel radial basis function methods for the global optimization of expensive functions," *European Journal of Operational Research*, vol. 182, pp. 514-535, 2007.

57. K.G. Roquemore, "Hybrid Designs for Quadratic Response Surfaces," *Technometrics*, vol. 18, no. 4, pp. 419-423, 1976.

58. A. Seshadri, Matlab Central File Exchange, "NSGA - II: A multi-objective optimization algorithm," 2006. [Online; accessed 14-September-2007].

59. A. Siem, D. den Hertog, and J. Kleijnen, "The correct Kriging variance estimated by bootstrapping," *Operations Research 2004-International Conference*, [http://center.uvt.nl/phd_stud/siem/; accessed: 8 Oct 2007].

60. T. Simpson et. al, "Kriging models for global approximation in simulation-based multidisciplinary design optimization," *AIAA Journal*, vol. 39, no.12, pp. 2233-2241, 2001.

61. Y. Soo and D. Bates, "Multiresponse spline regression," *Computational Statistics & Data Analysis*, vol. 22, pp619-631, 1996.

62. A. Srivastava et. al, "Development of a Kriging based surrogate approximation method for large-scale systems," [http://does.eng.buffalo.edu/publications/publications/askh_2000.pdf; accessed 7-Oct-2007].

63. T. Sriver, *Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems*. PhD thesis. Air Force Institute of Technology, September 2004.

64. T. Sriver and J. Chrissis, "Combined pattern search and Ranking and Selection for Simulation Optimization," in *Proceedings of the 2004 Winter Simulation Conference*, 2004.

65. Statlib, "OA Shell Archive," 1995. [http://lib.stat.cmu.edu/designs/oa.c; accessed 26-Sept-2007].

66. StatSoft Inc., "Multivariate Adaptive Regression Splines," 2003. [http://www.statsoft.com/textbook/stmars.html; accessed: 24 Sep 2007].

67. V. Torczon and M. Trosset, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1-25, 1997.

68. L. Trutna, National Institute of Standards and Technology, "Plackett-Burman Designs," 2006. [http://www.itl.nist.gov/div898/handbook/pri/section3/pri335.htm; accessed 26-Sept-2007].

69. D. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD Dissertation. Air Force Institute of Technology, 1999.

70. J. Walston, *Search Techniques for Multi-Objective Optimization of Mixed-Variable Systems Having Stochastic Responses*. PhD Dissertation. Air Force Institute of Technology, September 2007.

71. K. Won and T. Ray, "A framework for Design Optimization using Surrogates," *Engineering Optimization*, vol. 37, no. 7, pp. 685-703, 2005.

72. J. Wu and S. Azarm, "Metrics for quality assessment of a multiobjective design optimization solution set," *Transactions of the ASME*, vol. 123, no. 1, pp. 18-25, 2001.

Vita

Captain Todd Paciencia graduated from Kenmore East High School in Tonawanda, NY in 2000. He graduated from Binghamton University in Binghamton, NY in May 2003 with a Bachelor of Arts degree in Mathematical Sciences. He was commissioned through Air Force Officer Training School in February 2004.

Captain Paciencia served his first assignment as a Lead Analyst for the Joint GPS Combat Effectiveness Joint Test & Evaluation. Upon completion of that assignment, he became the Analysis Branch Chief of the newly-formed Air Force Joint Test & Evaluation Group Quick Reaction Test Force and following worked for both AFOTEC, as a Systems Analyst, and AFRL, as a member of one of its SPACE-CHOP missions. In August 2007, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to Rome Labs.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 03-30-2008	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Sep 2007 - Mar 2008

4. TITLE AND SUBTITLE MULTI-OBJECTIVE OPTIMIZATION OF MIXED-VARIABLE, STOCHASTIC SYSTEMS USING SINGLE-OBJECTIVE FORMULATIONS	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Paciencia, Todd, J, Captain, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GOR/ENS/08-17
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Two new algorithms are presented for multi-objective optimization of mixed-variable, stochastic systems. Both are based off of prior algorithms, but combine those pre-exsting algorithms with several other methods, to include single-objective formulations, surrogates, n-dimensional visualizations, aspiration an reservation levels, and direct search methods. Results are shown for a test set of 13 problems, ranging from 2 to 8 objectives, and including non-convex, mixed-variable, and discontinuous problems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. James W. Chrissis
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	236	(937) 255-3636, ext 4606; e-mail: james.chrissis@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18