A Pareto Following Variation Operator for Fast-Converging Multiobjective Evolutionary Algorithms

A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya
Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3053, Australia
{akmkat,mkirley,raj}@csse.unimelb.edu.au

ABSTRACT

One of the major difficulties when applying Multiobjective Evolutionary Algorithms (MOEA) to real world problems is the large number of objective function evaluations. Approximate (or surrogate) methods offer the possibility of reducing the number of evaluations, without reducing solution quality. Artificial Neural Network (ANN) based models are one approach that have been used to approximate the future front from the current available fronts with acceptable accuracy levels. However, the associated computational costs limit their effectiveness. In this work, we introduce a simple approach that has comparatively smaller computational cost and we have developed this model as a variation operator that can be used in any kind of multiobjective optimizer. When designing this model, we have considered the whole search procedure as a dynamic system that takes available objective values in current front as input and generates approximated design variables for the next front as output. Initial simulation experiments have produced encouraging results in comparison to NSGA-II. Our motivation was to increase the speed of the hosting optimizer. We have compared the performance of the algorithm with respect to the total number of function evaluation and Hypervolume metric. This variation operator has worst case complexity of $\mathcal{O}(nkN^3)$, where N is the population size, n and k is the number of design variables and objectives respectively.

Categories and Subject Descriptors

I.2.m [**Artificial Intelligence**]: Miscellaneous—*Evolutionary computing and genetic algorithms* ; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

Keywords

Evolutionary Multiobjective Optimization, Variation Operator, Dynamic System Identification, Function Evaluation

1. INTRODUCTION

Multiobjective Evolutionary Algorithms (MOEA) are now a well-established technique, both in terms of methodologies and algorithm development [10] [11] for tackling multiobjective optimization problems. However, one of the major difficulties when applying evolutionary algorithms to real-world problems is the computational costs associated with the large number of function evaluations necessary to obtain a range of acceptable solutions. Often, the function evaluations are time-consuming and are obtained by solving numerical models using methods such as finite-differences or finite-elements. The use of distributed systems, where each fitness evaluation is performed on a separate processor, does offer one approach for helping to reduce the computational time. Such models typically require a large number of networked computers (scaling in size from local clusters to full Grid deployment) and an adequate parallelization of the numerical code [19] [26]. However, a parallel approach *per se* does not necessarily reduce the number of function evaluations.

The development of techniques enabling a reduction in the number of function evaluations, without reducing solution quality, is an important goal of MOEA research [18, 25]. An on-going challenge, therefore, is to develop good approximate methods that can be used to solve multi-objective problems while considering the number of objectives and the possible interaction between them.

The use of approximate (or surrogate) models when using evolutionary algorithms to solve single objective problems is well documented (see [17, 21] for a survey). Several approaches have been used successfully to approximate objective functions in single objective optimization problems, such as the Kriging/Gaussian Process [8], Radial Basis Function(RBF) [34] [27] [28] [29], Polynomial Regression [17] and Artificial Neural Networks [21].

In the MOEA domain, there have been relatively few papers reporting the use of surrogate models. For example, [12] and [22] have incorporated a Gaussian Random Field Metamodel into the algorithm and [20] has adopted an approximation strategy. Gradient based and/or a directional local search strategies have also been used as a surrogate assisted MOEA for problems with differentiable objectives [4, 7, 6, 3, 15].

Artificial Neural Networks (ANN) based approximate models have also been used with some success [13, 14, 1, 2, 32, 31]. Typically, the ANN model approximates the design variables from the current front of the objective space by treating the objective vectors as inputs to the ANN and the design variables as output. Determining the best network structure (specifically, the number of hidden layers) and the total learning costs must be factored into the computational costs of the model. Other techniques such as the Quadratic Approximation Model [9] and Weighted Ranking Model [5] have also been reported in the literature. Clearly, the balance between computational cost and convergence speed is a major challenge that must be addressed.

In this study, we propose a novel approximation model for MOEA, which has a comparatively smaller computational cost than other surrogate models. We have developed this model as a variation operator that can be used in any multiobjective optimizer to speed up the search process by reducing the number of function evaluations and thus encourage the evolving population to follow the right trajectory towards a range of good trade-off solutions.

When desiging and analysing our model, we have considered the whole search procedure as a dynamic system [23], which takes the available objectives values in the current nondominated front as inputs and generates approximated design variables for the next front as the output. As such, this approach could be thought of as a form of "Response Surface Approximation" [13]. The ANN techniques described above are representative examples of this technique. However, in our approach we have replaced the ANN with a simple Linear Time Invariant system (LTI) [24, 23] and we attempt to model the system using the linear least square method (a widely used approach in dynamic system identification). We then use the model to approximate the next front from the current front and create the so called "Mirage Solutions" [2]. The Mirage Solutions are then used as input to the dynamic system to approximate the corresponding design variables. The resulting solutions (individuals) are then added to the current population of the hosting optimizer. Although this sort of naive technique does not guarantee a 100% correct mapping from objective space to design variable space, this technique is capable of approximating the solutions that resides in close vicinity of the Pareto-front relatively quickly.

In the following sections of the paper, a brief introduction to multiobjective optimization problems and a general framework for solving such problems is presented. Section 3.1 and 3.2 introduce the concept of dynamic system identification in MOEA. In section 3.3 we describe our model in detail. This is followed by a description of the design of the Pareto following operator and a discussion of computational complexity. Section 4 presents experimental results. The paper then concludes with a discussion of the findings and future research directions.

2. MULTIOBJECTIVE OPTIMIZATION

In this section, we will focus on the background notion suitable for a multiobjective optimization problem (MOP) and the design of a general MOEA.

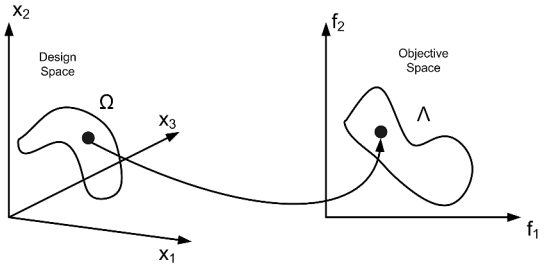

Figure 1: Representation of the decision variable space and the corresponding objective space

2.1 Definitions

A multiobjective optimization problem (MOP) consists of multiple criteria, more often conflicting, that need to be optimized simultaneously. (See Figure 1) Formally, a MOP can be defined as follows :
Find the vector, $\vec{x}^* = [{x_1}^*, {x_2}^* \dots {x_n}^*]$ which satisfies m inequality constraints:

$$g_i(\vec{x}) \geq 0 \qquad i = 1, 2 \dots m \tag{1}$$

$$h_i(\vec{x}) = 0 \qquad i = 1, 2 \dots p \tag{2}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}) \dots f_k(\vec{x})]^T \tag{3}$$

In other words, the aim is to determine from among the set of all values which satisfy (1) and (2) the particular set ${x^*}_1, {x^*}_2 \dots {x^*}_n$ which yields the optimum values of all the objective functions. In MOP's there is no single solution rather we have to find all compromising (**Pareto-optimal**) solutions. A solution $\vec{x}^* \in \Omega$ is **Pareto-optimal** if for every $\vec{x} \in \Omega$ and $I = \{1, 2 \dots k\}$ either,

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

or, there is at least one $i \in I$ such that

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{5}$$

The constraints given by (1) and (2) define the **feasible region** Ω and any point $\vec{x}$ in Ω defines a **feasible solution**. The vector function $\vec{f}(\vec{x})$ is a function which maps the set Ω into the set Λ which represents all possible values of the objective functions.

For a given MOP $\vec{f}(x)$, the **Pareto-optimal Set** ($\mathcal{P}^*$) is defined as

$$\mathcal{P}^* := \{x \in \Omega | \neg \exists x' \in \Omega : \vec{f}(x') \preceq \vec{f}(x)\} \tag{6}$$

Here the sign $\preceq$ refers to **Pareto-dominance**. A vector $\vec{u} = (u_1, u_2 \dots u_k)$ is said to dominate $\vec{v} = (v_1, v_2 \dots v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, i.e. $\forall i \in \{1, 2, \dots k\} : u_i \leq v_i \wedge \exists i \in \{1, 2, \dots k\} : u_i < v_i$.

Our goal is to find the set of all **Pareto-optimal** solutions and the corresponding objective values of this set is defined as **Pareto-front**. The **Pareto-front** ($\mathcal{PF}^*$) can be mathematically defined as,

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x) \dots f_k(x)) | x \in \mathcal{P}^*\} \tag{7}$$

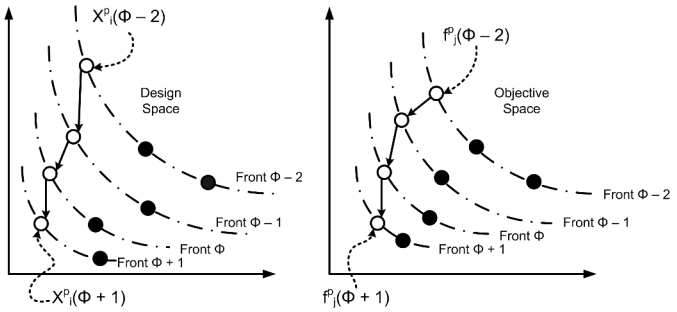

Figure 2: After nondominated sorting, individuals in each front are sorted with respect to one objective. Individual ($\circ$) is the same individual moving from front $\phi - 2$ to front ϕ

2.2 A Typical Nondominated Sorting MOEA

A typical nondominated sorting MOEA employing an elitist model has the following functionality: Firstly, the algorithm starts with a randomly generated population P_t, and then after evaluation, the individuals are sorted according to the nondomination criteria and divided into ϕ fronts.

$$P_t := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1\}$$

Then, from the best front $\mathcal{F}_\phi$, mutation, crossover and other genetic operators are applied to expand the population to the next best front $\phi + 1$ to create the next population P_{t+1}. Different algorithms uses different techniques to expand this population to the next front. In this paper, we have chosen NSGA-II as the base (or host) algorithm to be combined with our Pareto following operator for illustration purposes. NSGA-II uses nondominated sorting, an elitist selection scheme and a crowding distance based selection strategy and thus can provide a good spread of solutions throughout an evolutionary run. It is important to note, that our model can be plugged into any nondominated sorting-based MOEA for numerical optimization problems.

3. THE MODEL

3.1 System Identification

In this subsection, we present the core idea behind the design of our variation operator used to approximate the individuals for the next front.

In any typical nondominated sorting MOEA, a number of individuals are generated randomly, evaluated and then sorted according to nondomination criteria. Figure 2 provides a schematic view of the the individuals in objective space and design variable space after sorting.

Here, we are considering M individuals with n design variables and k objectives. Suppose, $x_i^p(\phi)$ and $f_j^p(\phi)$ denote the i^{th} design variable and j^{th} objective value of an individual p in front ϕ. If we sort the individuals in every front with respect to one objective, we can assume that the p^{th} individual of each front as the same individual that is moving from front $\phi - 2$ to ϕ. Here, decreasing values of ϕ represent a worse front. Now if we could somehow extrapolate this trajectory, we can infer that this individual will eventually reach the next front $\phi + 1$ (refer to Figure 2). Moreover, if the distance between two consecutive fronts is small, then we can also assume that this trajectory is piece-wise linear.

Considering only one individual p, we can also say that this search algorithm takes $x_i^p(\phi - 1)$ as input and generates $x_i^p(\phi)$ as output. Instead of considering the search algorithm as a procedure *per se*, let us consider it as a dynamic system [23, 24] (with transfer function H), which takes the series $\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2)\ldots\}$ as input and generates $\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2)\ldots\}$ as output (refer to Figure 3). Again, if we consider an inverse system (with transfer function H^{-1}), then it will generate $\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2)\ldots\}$ as output when $\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2)\ldots\}$ is input. Therefore, if we could somehow approximate the system's parameters, then we can easily approximate the design variable of the next front $x_i^p(\phi+1)$ (when $f_j^p(\phi+1)$ is given as input).

Our next task is to generate the so-called "Mirage Solutions" $f_j^p(\phi+1)$. Approximation of $f_j^p(\phi+1)$ is quite straightforward since in any type of MOP, we have to maximize (or minimize) multiple objectives. In the case of a minimization problem, the objective value in the next front $\phi + 1$ will be smaller then that of the current front ϕ by some amount of Δf. So for a minimization problem,

$$f_j^p(\phi + 1) = f_j^p(\phi) - \Delta f \tag{8}$$

The preceding discussions dictate that our Pareto following variation operator depends on the following formulations:

- approximating the parameters of the dynamic system
- approximating $f_j^p(\phi+1)$ from $f_j^p(\phi)$ by choosing a suitable Δf
- from $f_j^p(\phi+1)$, approximate the design variable $x_i^p(\phi+1)$ of the next front $\phi + 1$

3.2 Model Formulation

First we construct a simple model for the i^{th} design variable and j^{th} objective:

$$x_i(\phi) + a_0 x_i(\phi - 1) = b_0 f_j(\phi) + b_1 f_j(\phi - 1) + \epsilon(\phi) \tag{9}$$

Here, f_j are input and x_i are considered as output. Therefore,

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \begin{bmatrix} a_0 \\ b_0 \\ b_1 \end{bmatrix} + \epsilon(\phi) \tag{10}$$

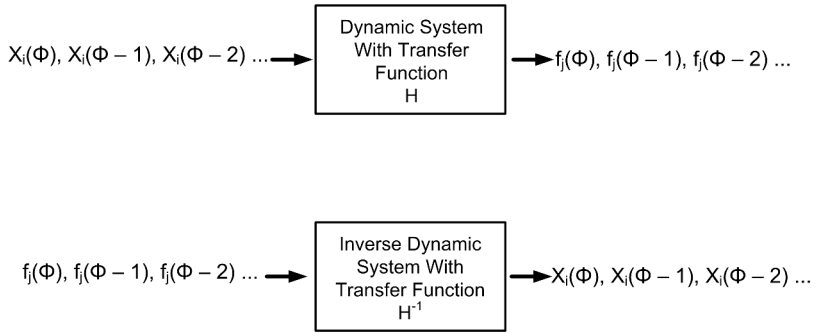

Figure 3: The forward and inverse dynamic system

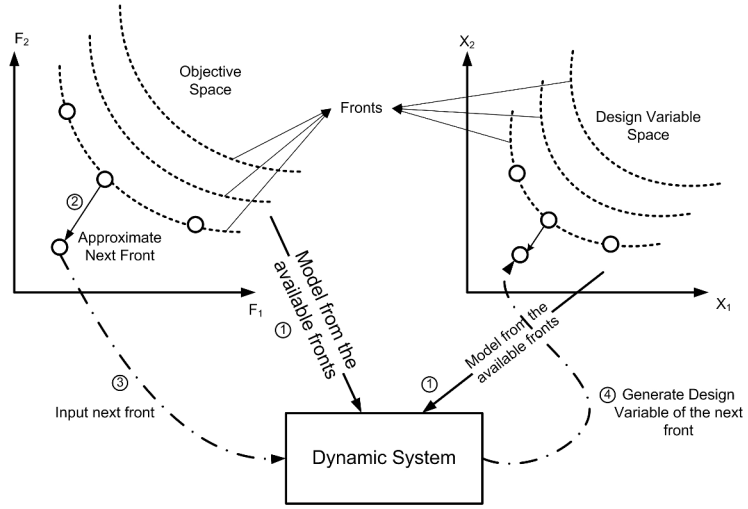

Figure 4: Working steps of the Pareto following variation operator

Now, we can construct a matrix formation as:

$$\underbrace{\begin{bmatrix} x_i(\phi) \\ x_i(\phi-1) \\ x_i(\phi-2) \\ \vdots \\ x_i(2) \end{bmatrix}}_{\text{y}} = \underbrace{\begin{bmatrix} x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \\ x_i(\phi-2) & f_j(\phi-1) & f_j(\phi-2) \\ x_i(\phi-3) & f_j(\phi-2) & f_j(\phi-3) \\ \vdots & \vdots & \vdots \\ x_i(1) & f_j(2) & f_j(1) \end{bmatrix}}_{\Phi} \cdot \underbrace{\begin{bmatrix} a_0 \\ b_0 \\ b_1 \end{bmatrix}}_{\beta_{ij}}$$

$$+ \underbrace{\begin{bmatrix} \epsilon(\phi) \\ \epsilon(\phi-1) \\ \epsilon(\phi-2) \\ \vdots \\ \epsilon(2) \end{bmatrix}}_{\epsilon}$$

Or we can rewrite

$$y = \Phi \cdot \beta_{ij} + \epsilon \tag{11}$$

Here, β_{ij} denotes parameter of the dynamic system for i^{th} design variable and j^{th} objective. β_{ij} can be approximated using least squares. Let us denote $\hat{\beta_{ij}}$ as approximated β_{ij}:

$$\hat{\beta}_{ij} = \underbrace{(\Phi^T\Phi)^{-1}}_{\text{pseudo-inverse}} \Phi^T y \tag{12}$$

So,

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \cdot \hat{\beta}_{ij} \tag{13}$$

Now we have $\hat{\beta}_{ij}$, from equation 14 and 8 we can easily approximate the i^{th} design variable of the next front $\phi+1$:

$$x_i(\phi+1) = \begin{bmatrix} -x_i(\phi) & f_j(\phi+1) & f_j(\phi) \end{bmatrix} \cdot \hat{\beta}_{ij} \tag{14}$$

$$= \begin{bmatrix} -x_i(\phi) & f_j(\phi) - \Delta f & f_j(\phi) \end{bmatrix} \cdot \hat{\beta}_{ij} \tag{15}$$

3.3 The Pareto Following Variation Operator

Based on the description above, we now illustrate the functionality of our Pareto following variation operator. A requirement for using this variation operator, is that the population has been sorted into nondominated fronts (in fact, there must be more than one front).

Now, let us suppose population P has M individuals.

$$P := \{I^1(\vec{f}^1, \vec{x}^1), I^2(\vec{f}^2, \vec{x}^2), \ldots, I^M(\vec{f}^M, \vec{x}^M)\}$$

Each individual has k objectives and n design variables. Therefore, individual p can be represented as

$$I^p(\vec{f}^p, \vec{x}^p) = \{\{f_1^p, f_2^p, \ldots, f_k^p\}, \{x_1^p, x_2^p, \ldots, x_n^p\}\}$$

If they are evaluated and sorted according to nondomination, ϕ fronts will be created, i.e.

$$P := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1\}$$

So, individual p in front ϕ is represented as $I^p(\vec{f}^p(\phi), \vec{x}^p(\phi))$.

Algorithm 1 Pareto Following Variation Operator(P_t,Δf)

Require: Parent population P_t is sorted with respect to non domination.

$$P_t := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1\}$$

and individuals in $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1$ are sorted again with respect to one objective. Each front $\mathcal{F}_i$ has size m_i and $\mathcal{F}_\phi$ is the best front. Δf is the objective distance from current best front to next approximating front.

Ensure: Creates km_ϕ number of approximated individuals of the front $\mathcal{F}_{\phi+1}$

1: **for all** all objectives j such that $1 \le j \le k$ **do**
2: **for all** all individuals p such that $1 \le p \le m_\phi$ **do**
3: **for all** all design variables i such that $1 \le i \le n$ **do**
4: Construct matrix y from p^{th} individuals from every front $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1$
5: Construct matrix Φ from p^{th} individuals from every front $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1$
6: Calculate $\hat{\beta}_{ij} := (\Phi^T\Phi)^{-1}\Phi^T y$
7: Approximate $f_j^p(\phi+1) := f_j^p(\phi) - \Delta f$
8: $x_i(\phi+1) := \begin{bmatrix} -x_i(\phi) & f_j(\phi) - \Delta f & f_j(\phi) \end{bmatrix} \cdot \hat{\beta}_{ij}$
9: $x_i(\phi+1)$ is the i^{th} design variable of the new $(p + (j-1)m_\phi)^{th}$ approximated individual $I^{p+(j-1)m_\phi}$ So, $x_i^{p+(j-1)m_\phi}(\phi+1) := x_i(\phi+1)$
10: **end for**
11: **end for**
12: **end for**
13: Returns new approximated population $\mathcal{F}_{\phi+1}$ with size km_ϕ

Algorithm 2 NSGA-II with Pareto Following Variation Operator

Require: Randomly generated parent population P_t at generation t with size M.

Ensure: After t_{max} number of iteration, population $P_{t_{max}}$ will represent solution of the problem.

1: **while** $t \le t_{max}$ **do**
2: Start with child population, $Q_t := \emptyset$
3: Create mixed population, $R_t := P_t \cup Q_t$
4: $\mathcal{F}$:=Apply Nondominated Sort on R_t, create ϕ number of fronts. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
5: **if** $\phi > 1$ **then**
6: $\mathcal{F}_{\phi+1}$:=Pareto Following Variation(R_t,Δf) $|\mathcal{F}_{\phi+1}| = km_\phi$
7: Insert newly approximated population to R_t, $R_t := R_t \cup \mathcal{F}_{\phi+1}$
8: $\mathcal{F}$:=Apply Nondominated Sort on R_t, create ϕ number of fronts. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
9: **end if**
10: $P_t := \emptyset$ and $i := 1$
11: **repeat**
12: Assign crowding distance on $\mathcal{F}_i$
13: $P_{t+1} := P_{t+1} \cup \mathcal{F}_i$
14: $i := i + 1$
15: **until** $|P_{t+1}| + |\mathcal{F}_i| \le M$
16: Apply crowding distance sorting on $\mathcal{F}_i$
17: Choose the first $(M - |P_{t+1}|)$ individuals of $\mathcal{F}_i$
18: Use selection, crossover and mutation to create child population Q_{t+1} from P_{t+1}
19: **end while**

Where

$$\begin{aligned} \vec{f}^p(\phi) &= \{f_1^p(\phi), f_2^p(\phi), \ldots f_k^p(\phi)\} \\ \vec{x}^p(\phi) &= \{x_1^p(\phi), x_2^p(\phi), \ldots x_n^p(\phi)\} \end{aligned}$$

If we consider the subpopulation at the i^{th} front, $\mathcal{F}_i$, has m_i individuals, then obviously,

$$\begin{aligned} \mathcal{F}_i := \{I^1(\vec{f}^1(i), \vec{x}^1(i)), I^2(\vec{f}^2(i), \vec{x}^2(i)), \ldots \\ \ldots, I^{m_i}(\vec{f}^{m_i}(i), \vec{x}^{m_i}(i))\} \end{aligned}$$

Since decreasing values of ϕ represent a worse front, let us suppose ϕ is the best front. Here, m_ϕ is the size of front ϕ.

After all these introductory notations we can now proceed to the design of the variation operator, which is described in Algorithm 1. Figure 4 also illustrates the basic functionality of the operator.

The procedure presented in Algorithm 1 can be easily "plugged in" to any kind of multiobjective optimizer. We have selected NSGA-II as a preferable candidate. After applying the nondominated sorting procedure on the mixed population R_t in NSGA-II, we apply the Pareto following variation operator to create the individuals from the next approximated front $\mathcal{F}_{\phi+1}$. These new individuals are then combined with the newly created individuals in the parent population P_t (refer to line 6 and 7 of the Algorithm 2).

3.4 Algorithm Complexity

We now focus on the complexity of our algorithm. To find the pseudo inverse in equation 13, we have used QR factorization with the aid of the Householder transformation [30]. The complexity of the algorithm largely depends on the size of the matrix Φ in equation 11. Here we denote this time (or front) steps as t and the initial "guess" of the dynamic model starts with 2 time (or front) steps in equation 9. We are doing QR factorization on matrix Φ whose dimension is $(|\phi| - 1) \times t$. Here $|\phi|$ is the size of the best front ϕ. This operation requires $2(|\phi| - 1)t^2 - 2/3t^3$ computations. So it has a computational complexity of $\mathcal{O}(2(|\phi| - 1)t^2 - 2/3t^3)$.

After the QR factorization, the upper triangular matrix R has a dimension of $t \times t$ and the orthogonal matrix Q has a dimension of $(|\phi| - 1) \times t$. After doing this, we are solving the systems parameter β_{ij} in equation 13. This requires one inversion on the upper triangular matrix R, one multiplication on $Q^T y$ and another multiplication on $R^{-1}Q^T y$. So equation 13 and 16 have an overall complexity of:

$$\begin{aligned} \mathcal{O}(2(|\phi| - 1)t^2 - 2/3t^3) + \mathcal{O}((|\phi| - 1)t) \\ +\mathcal{O}(t^3) + \mathcal{O}((|\phi| - 1)^2 t) + \mathcal{O}(t^2) \\ \approx \mathcal{O}(2|\phi|^2 t^2) \\ \text{Here, } |\phi| \gg t \end{aligned}$$

From line 1 to 12 in Algorithm 1, β_{ij} is evaluated for $nk|\phi|$ times. So the overall complexity of the variation operator will be $\mathcal{O}(2nk|\phi|^3 t^2)$. Where n is the number of design variables, k is the number of objectives.

If we consider the complexity with N individuals in the worst case, there will be only two fronts, where the best front has $N-1$ individuals and worst one has only 1. In that case, $|\phi| = N - 1$. So the overall worst case complexity will be $\mathcal{O}(2nk(N-1)^3 t^2) \approx \mathcal{O}(nkN^3 t^2)$. Moreover, in our experiment $t = 3$, and obviously in the worst case $|\phi| \gg t$, so the variation operator has the complexity of $\mathcal{O}(nkN^3)$, which largely depends on the population size. Actually, this added complexity will not reduce the overall running time of the host optimizer, since this operator can save extra objective evaluation of the hosting optimizer by approximate mapping of the future Pareto front to future design variables.

In this experiment we have used the Linear Algebra package **Meschach 1.2b**, [33] for matrix operations. In the next section, we provided a comparison of the average running time complexity (our algorithm with NSGA-II) in Table 1.

Table 1: Runtime Comparison (With Population Size: 256)

Algorithm	Criteria	Problems				
		ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	Hypervolume	3.01	1.50	14.42	7.52	1.59
	Av. Time(sec)	39.78	39.44	39.70	42.20	44.47
	Generation	200	200	200	200	400
With Pareto Following Operator	Hypervolume	3.01	1.50	14.57	7.52	1.59
	Av. Time(sec)	24.86	22.37	20.02	32.03	39.20
	Generation	89	74	44	112	300

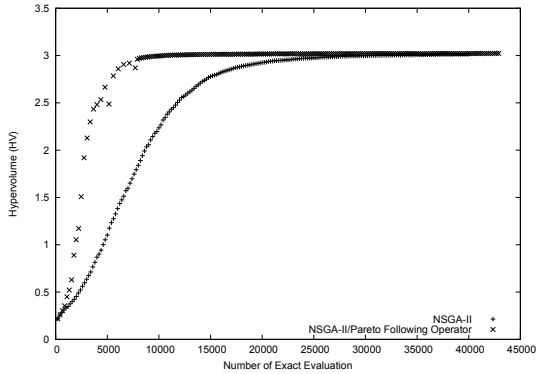

Figure 5: Hypervolume (HV) vs Number of Exact Function Evaluation for ZDT1

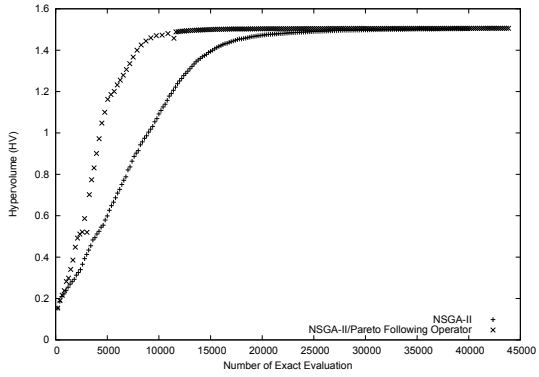

Figure 6: Hypervolume (HV) vs Number of Exact Function Evaluation for ZDT2

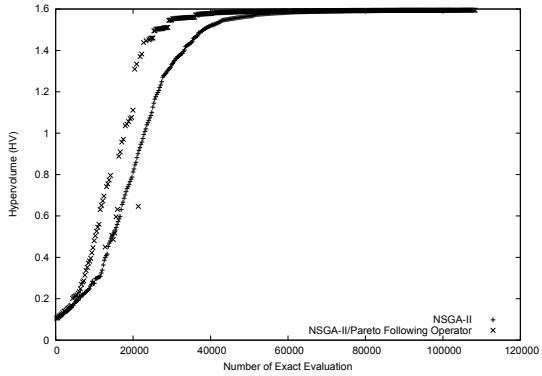

Figure 9: Hypervolume (HV) vs Number of Exact Function Evaluation for ZDT6

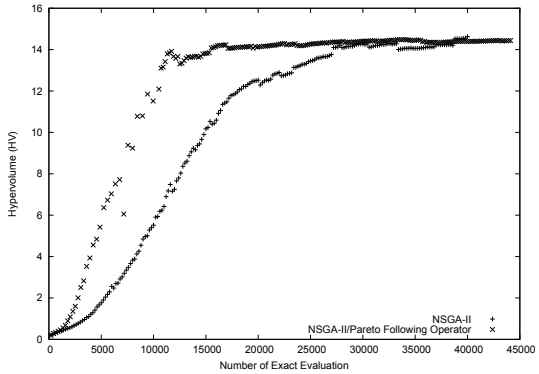

Figure 7: Hypervolume (HV) vs Number of Exact Function Evaluation for ZDT3

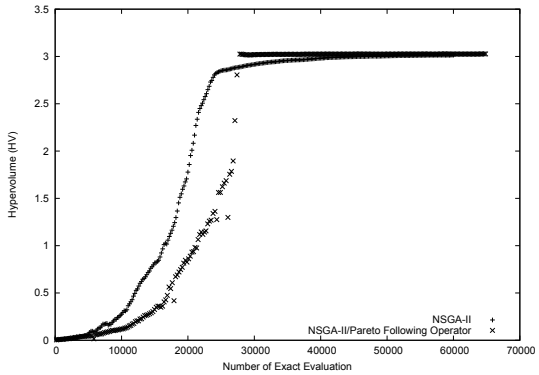

Figure 8: Hypervolume (HV) vs Number of Exact Function Evaluation for ZDT4

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of our Pareto following variation operator, a number of simulation experiments were conducted using several well-known benchmark problems from [35]. These problems are the unconstrained bi-objective problem set ZDT1, ZDT2, ZDT3, ZDT4, ZDT6. All the problem parameters were same as original NSGA-II implementation available at:
`http://www.iitk.ac.in/kangal/codes.shtml`.

The key hypothesis tested in this paper was that the inclusion of the Pareto following variation operator with a base optimizer (NSGA-II in this case) would increase the speed of the hosting optimizer algorithm. Therefore, we have compared the performance of the algorithm with respect to total number of function evaluation and Hypervolume (HV) mea-

sure [11] [16]. The Hypervolume calculates the volume (in the objective space) covered by the members of the known Pareto front for problems where all objectives are to be minimized. This metric can give us both the convergence and spread of solutions on the Pareto front. Obviously, an algorithm with a large Hypervolume is desirable.

To measure the time complexity of our algorithm, we have executed NSGA-II on each problem for the same number of generations as indicated in its original implementation. After convergence of the algorithm, we have collected the Hypervolume measure for every problem. We then executed our algorithm to measure the time taken to reach the same Hypervolume. For the convenience of the time measurement, the initial population size was 256 individuals for each problem. We have repeated this process 30 times. The mean running times are recorded. Details on running time comparison are given in Table 1. For all problems, the inclusion of the Pareto-following variation operator significantly reduces the total number of exact function evaluations.

Figure 5 to 7 are provided to present the comparison of our algorithm with NSGA-II for the ZDT benchmark problem set. We have conducted all our experiments on Windows XP machine with Pentium 4, 2.79GHz processor. From the test results given in Figure 5 to 7 (averaged over 30 runs), it becomes apparent that the Pareto following operator is capable of saving extra function evaluations to reach the same Hypervolume achieved by NSGA-II. In terms of runtime analysis, our method is also capable of speeding up the hosting optimizer up to average of 32.64%. Specially, for problems ZDT1, ZDT2 and ZDT3, our model can speed up the convergence time up to more than 10 seconds (Refer to Table 1). Moreover, the choice of Δf is problem dependant. We have found different Δf for different problems that can generate a good approximated solution. Detailed enumerated results have been omitted for clarity.

5. CONCLUSION AND FUTURE WORK

In this paper, we have designed and evaluated the efficacy of a novel approximation model for MOEA, which has a comparatively smaller computational cost than other surrogate models. An important contribution of this work was that our Pareto following variation operator can be used in conjunction with any nondominated sorting MOEA.

The Pareto following variation operator, was developed using a dynamic systems approach by taking only two time steps (front steps) in equation 9. Here, we assumed that the intra-front trajectory of an individual was piece-wise linear, subsequently a Linear Time Invariant (LTI) model was sufficient for approximation. There is obviously scope to investigate the relative worth of a non-linear dynamic system model such as the nonlinear ARX and Hammerstein-Wiener models [23]. This particular line of research is currently underway. Recursive System Identification may also be a useful approach to investigate. When solving equation 13, we have used the Householder Transformation for QR factorization, however this approach suffers from mathematical instability. In the case of QR factorization, Singular Value Decomposition can also be used to assure more mathematically stable solution to find β_{ij}.

Moreover, in this experiment, we have chosen the value of Δf from empirical experimentations and its value is different for different problems. Adaptive change in Δf (using upper and lower bound of objective values in current generation) can be implemented to obtain more robust solutions. In the future, we are going to focus on these issues.

6. REFERENCES

[1] S. Adra, A. Hamody, I. Griffin, and P. J. Fleming. A Hybrid Multi-objective Evolutionary Algorithm Using an Inverse Neural Network for Aircraft Control System Design. In *IEEE Congress on Evolutionary Computation (CEC-2005)*, volume 1, pages 1–8, Edinburgh, UK, September 2005. IEEE Pres.

[2] S. F. Adra, I. Griffin, and P. J. Fleming. An Informed Convergence Accelerator for Evolutionary Multiobjective Optimiser. In *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*, pages 734–740, London, England, 2007. ACM Press, NY, USA.

[3] P. A. N. Bosman and E. D. de Jong. Exploiting Gradient Information in Numerical Multi-Objective Evolutionary Optimization. In *The 7th Annual Conference on Genetic and Evolutionary Computation (GECCO-2005)*, pages 755–762, Washington DC, USA, 2005. ACM Press.

[4] P. A. N. Bosman and E. D. de Jong. Combining Gradient Techniques for Numerical Multi-Objective Evolutionary Optimization. In *The 8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*, pages 627–634, Seattle, Washington, USA, 2006. ACM Press.

[5] J. Branke, T. Kaußler, and H. Schmeck. Guiding Multi-Objective Evolutionary Algorithms Towards Interesting Regions. In I. C. Parmee, editor, *Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000), Poster Proceedings*, pages 1–4, Plymouth, UK, 2000. Plymouth Engineering Design Centre, University of Plymouth.

[6] M. Brown and R. E. Smith. Effective Use of Directional Information in Multi-objective Evolutionary Computation. In *The 5th Annual Conference on Genetic and Evolutionary Computation (GECCO-2003)*, volume 2723/2003 of *LNCS*, pages 778–789, Chicago, Illinois, USA, 2003. Springer.

[7] M. Brown and R. E. Smith. Directed Multiobjective Optimization. *International Journal of Computers, Systems and Signals*, 6(1):3–17, 2005.

[8] D. Buche, N. Schraudolph, and P. Koumoutsakos. Accelerating Evolutionary Algorithms with Gaussian Process Fitness Function Models. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2):183–194, May 2005.

[9] D. Chafekar, L. Shi, K. Rasheed, and J. Xuan. Multiobjective GA Optimization Using Reduced Models. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2):261–265, May 2005.

[10] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum Publishers, NY, USA, 2002.

[11] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, Ltd, West Sussex, England, 2002.

[12] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, August 2006.

[13] M. Farina. A Neural Network Based Generalized Response Surface Multiobjective Evolutionary Algorithm. In *IEEE Congress on Evolutionary Computation (CEC-2002)*, volume 1, pages 956–961, Honolulu, HI, May 2002. IEEE Press.

[14] A. Gaspar-Cunha and A. Vieira. A Multiobjective Evolutionary Algorithm Using Neural Networks to Approximate Fitness Evaluations. *International Journal of Computers, Systems and Signals*, 6(1):18–36, 2005.

[15] K. Harada, J. Sakuma, and S. Kobayashi. Local Search for Multiobjective Function Optimization: Pareto Descent Method. In *The 8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*, pages 659–666, Seattle, Washington, USA, 2006. ACM Press.

[16] S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transaction on Evolutionary Computation*, 10(5):477–506, October 2006.

[17] Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9:3–12, January 2005.

[18] Y. Jin, M. Olhofer, and B. Sendhoff. A Framework for Evolutionary Optimization with Approximate Fitness Functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, October 2002.

[19] M. Kirley and R. Stewart. An Analysis of The Effects of Population Structure on Scalable Multiobjective Optimization Problems. In *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*, pages 845–852, London, England, 2007. ACM Press.

[20] J. Knowles. ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, February 2006.

[21] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff. A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. In *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*, pages 1288–1295, London, England, 2007. ACM Press.

[22] W. Liu, Q. Zhang, E. P. K. Tsang, C. Liu, and B. Virginas. On the Performance of Metamodel Assisted MOEA/D. In L. Kang, Y. Liu, and S. Y. Zeng, editors, *ISICA-2007*, volume 4683 of *LNCS*, pages 547–557, Wuhan, China, August 2007. Springer.

[23] L. Ljung. *System Identification: Theory for The User*. Prentice-Hall Inc., NJ, USA, 1999.

[24] L. Ljung and T. Glad. *Modelling of Dynamic Systems*. Prentice-Hall Inc., NJ, USA, 1994.

[25] P. K. Nain and K. Deb. Computationally Effective Search and Optimization Procedure Using Coarse to Fine Approximations. In *IEEE Congress on Evolutionary Computation (CEC-2003)*, volume 3, pages 2081–2088, Canberra, NSW, Australia, December 2003. IEEE Press.

[26] A. J. Nebro, E. Alba, and F. Luna. Multi-Objective Optimization using Grid Computing. *Soft Computing*, 11(6):531–540, 2007.

[27] Y. S. Ong, K. Y. Lum, and P. B. Nair. Hybrid Evolutionary Algorithm with Hermite Radial Basis Function Interpolants for Computationally Expensive Adjoint Solvers. *Computational Optimization and Applications*, 39:97–119, January 2008.

[28] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):689–696, 2003.

[29] Y.-S. Ong, P. B. Nair, and K. Lum. Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, August 2006.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[31] T. Ray and W. Smith. A Surrogate Assisted Parallel Multiobjective Evolutionary Algorithm for Robust Engineering Design. *Engineering Optimization*, 38(8):997–1011, 2006.

[32] H. Soh, Y. S. Ong, M. Salahuddin, T. Hung, and B. S. Lee. Playing in the Objective Space: Coupled Approximators for Multi-Objective Optimization. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (ICDM-2007)*, pages 325–332. IEEE Press, April 2007.

[33] D. E. Stewart and Z. Leyk. Meschach: Matrix Computations in C. In *Centre for Mathematics and Its Applications*, volume 32, Australian National University, Canberra, Australia, 1994. Available at: `http://www.netlib.org/c/meschach/` and `http://www.math.uiowa.edu/~dstewart/meschach/`.

[34] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(1):66–76, January 2007.

[35] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.