The Pareto-Following Variation Operator as An Alternative Approximation Model

A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya

Abstract— **This paper presents a critical analysis of the Pareto-Following Variation Operator (PFVO) when used as an approximation method for Multiobjective Evolutionary Algorithms (MOEA). In previous work, we have described the development and implementation of the PFVO. The simulation results reported indicated that when the PFVO was integrated with NSGA-II there was a significant increase in the convergence speed of the algorithm. In this study, we extend this work. We claim that when the PFVO is combined with any MOEA that uses a non-dominated sorting routine before selection, it will lead to faster convergence and high quality solutions. Numerical results are presented for two base algorithms: SPEA-II and RM-MEDA to support are claim. We also describe enhancements to the approximation method that were introduced so that the enhanced algorithm was able to track the Pareto-optimal front in the right direction.**

I. Introduction

Most numerical optimization problems require experiments and/or simulations to evaluate design objectives and constraints. For many real world problems, however, a single simulation can take a long time. As a result, routine tasks such as design optimization, design space exploration, sensitivity analysis and "what-if" analysis become impossible since they require thousands or even millions of simulation evaluations [1]. One way of alleviating this burden is to construct approximation models, known as *Surrogate Models*, *Response Surface Models (RSM)*, *Meta Models* or *Emulators* that mimic the behavior of the simulation model as closely as possible, while being computationally cheap(er) to evaluate.

The main idea of Surrogate/RSM is to use a set of designed experiments to obtain an optimal response. In this case, the exact, inner working of the simulation code is not assumed to be known (or even understood). The input-output behavior is the important thing. A model is constructed based on modeling the response of the simulator to a limited number of intelligently chosen data points. This approach is also known as *Behavioral Modeling* or *Black-box Modeling* [2].

Given the development of approximation/surrogate model for single-objective optimization problems, it is not surprising that there has been increased interest in applying these techniques in the multi-objective domain. However, in many cases the implementation strategies have been very similar to that of single-objective approaches. This particular approach has limitations. In Multiobjective Optimization Problems (MOP's), the goal is to find a set of compromising solutions that satisfy two or more objectives. Consequently, the implementation of surrogate models is not as straight forward as in single-objective EAs. In fact, this area is one of the most challenging research topics in MOEA.

A number of different approaches have been described in the relevant literature to address this challenge. The most trivial approach is to design the approximation functions using a direct implementation of the surrogate models as in single-objective EAs. In this case, we have to construct a number of surrogate models, one for each of the objective functions [3]. Another approach is to decompose the multiple objectives into several scalar optimization problems [4] and construct the surrogate model. For a detailed survey on surrogate models in MOEA, readers are referred to [5]. Here, we present a brief discussion of the limitations that we have identified when using conventional surrogate models for MOEAs:

- Since MOP's deal with more than one objective, we have to design different approximation models for different objective functions. If we are going to use a computationally expensive method such as Kriging [6], RBF network [7] or Artificial Neural Network [8], [9], [10], [11], the learning cost increases proportionally with the number of objective functions.
- The main goal of a MOEA is to find all possible solutions on the Pareto-front. However, the shape of the Pareto-front may impact on the performance of the surrogate. Surrogate models are designed for continuous functions, so when a broken (not connected) Pareto-front is encountered, the model may not work properly.
- Surrogate models are also error-prone. Deceptive phenomenon and falling into the trap of "false optima" are common occurences in single objective optimization [12] [13]. The same problem exists for MOP's.

The Pareto-following Variation Operator (PFVO) is similar to the work presented in [8], [9], [14]. However the main difference is that in [8], the model was built using an *Inverse* Artificial Neural Network (ANN). The term *Inverse* actually defines the technique where we "reverse map" the design variables from the objective functions to the design space. This idea has a special significance in the case of MOP's. If we could some how extrapolate the next (future) Pareto-front from the existing (present) one, this reverse mapping technique will enable us to approximate those design variables that would make up the future Pareto-front. To implement this reverse mapping idea, we have to consider the optimizer as a *Dynamic System* that takes *design variables* as inputs and *objective functions* as outputs. Under this assumption,

Authors are with the Department of Computer Science and Software Engineering, The University of Melbourne, Carlton, Melbourne 3053, Victoria, Australia (phone: 61-03-83441426; email: {akmkat, mkirley}@csse.unimelb.edu.au).

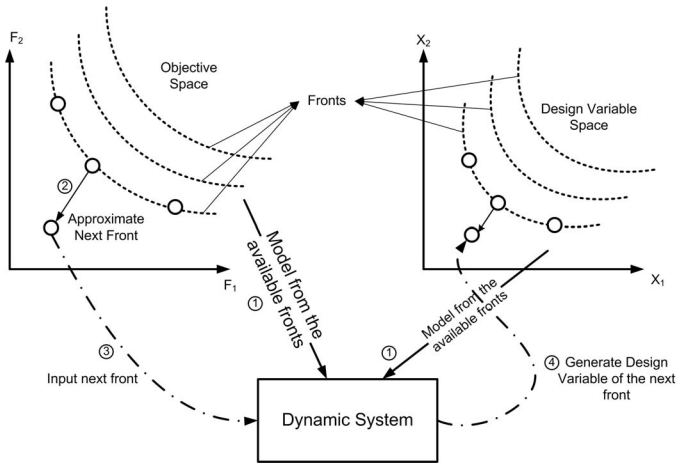


Fig. 1. The reverse mapping procedure

the ANN mimics this *Dynamic System* or *Optimizer*. Figure 1 illustrates the reverse mapping procedure at an abstract level.

Although, ANN based models have been successful in some circumstance, we have identified significant implementation limitations:

- It is very difficult to design an accurate ANN structure (both in terms of accurate weights and neuron connections) that will exactly mimic the behavior of the original expensive fitness function/evolutionary optimizer (in the case of "inverse mapping").
- ANN have a corresponding huge computational cost, attributed to learning costs. A good ANN may help us to design an efficient approximator, however, the computational cost degrades most of the benefits.
- In the case of the "reverse mapping" procedure, if the total number of objective is smaller than the number of design variables (which is always the case), it is very difficult to design an efficient ANN model – under these circumstance, the number of input neurons is smaller than the number of output neurons.

ANN based models are widely used [15], primarily due to their high accuracy levels. However, for MOP's, high levels of accuracy is not as critical. The approximation of the next Pareto-front from the existing one needs to be just "good enough" to bypass the extra computation for the evolutionary optimizer, rather than being accurate. Therefore, in our PFVO model, we carry out a similar modelling procedure using Least Square Approximation (QR-Factorization) in place of the ANN. Moreover, our model does not replace the original objective functions with an approximation model. Instead, we will approximate the search trajectory of the hosting optimizer[1] and skip the function evaluations for non-promising individuals so that the hosting optimizer can reach to the true Pareto-front within small amount of computational cost.

[1]Here "hosting optimizer" refers to any kind of MOEA that uses non-domination based sorting on the population before selection process. It is considered as the "base MOEA" for our experiment.

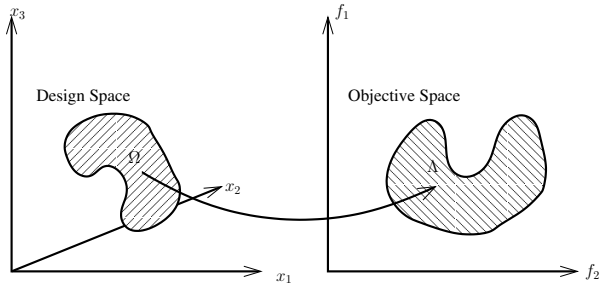


Fig. 2. Representation of the decision variable space and the corresponding objective space

The remainder of the paper is organized as follows: In section II, we present an overview of MOPs. In section III we discuss the detailed design and implementation of our model. We then describe the integration mechanism with two popular baseline MOEA's – the "Strength Pareto Evolutionary Algorithm - II (SPEA-II)" [16], and the recently proposed "Regularity Model Based Estimation of Distribution Algorithm (RM-MEDA)" [17]. In the next section, we describe the detailed numerical simulations and present the results and analysis. In the final section, we will conclude the paper and suggests directions for future research.

II. Multiobjective Optimization Problem(MOP)

The general *Multi-objective Optimization Problem* can be defined as follows - Find the vector, $\vec{x}^* = [{x_1}^*, {x_2}^* \dots {x_n}^*]^T$ 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. Please refer to Figure 2 for the concept of objective

space and design variable space. For a given MOP $\vec{f}(x)$, the *Pareto-optimal Set* ($\mathcal{PS}^*$) is defined as

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

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,

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

For more details on basic MOP, readers are referred to [18], [19], [20].

III. The Pareto Following Variation Operator(PFVO)

The PFVO is an alternate surrogate model. A key design feature is the fact that it does not approximate the original objective function, rather it approximates the behavior (input/output or design-variable/objective-value relation) of the underlying hosting optimizer. In the next subsection we start the model description (formulation) by introducing useful notations. This is followed by a detailed description and complexity analysis.

A. Notations

In any typical non-dominated sorting MOEA, a number of individuals are generated randomly, evaluated and then sorted according to non-domination criteria. Figure 3 provides a schematic view of the the individuals in objective space and design variable space after sorting.

Here, we consider M individuals with n design variables and k objectives. Suppose, $x_i^p(\phi)$ and $f_j^p(\phi)$ denote the design variable i and the objective value j of an individual p in front $\mathcal{F}_\phi$. If we sort the individuals in every front with respect to one objective, we can also assume that a specific individual p of each front is the same individual moving towards front $\mathcal{F}_\phi$ from front $\mathcal{F}_{\phi-2}$ (Refer to Figure 3). Here, decreasing values of ϕ represent a worse front[2]. Now if we could somehow extrapolate the trajectory of p, we can infer that this individual will eventually reach the next front $\mathcal{F}_{\phi+1}$ (Refer to Figure 3). 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" [21], [22] (with transfer function H), which takes the series

$$\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2) \dots\} \quad (8)$$

as input and generates

$$\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2) \dots\} \quad (9)$$

[2]Here, the fronts $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \dots, \mathcal{F}_1$ are generated in a single generation, $\mathcal{F}_\phi$ and $\mathcal{F}_{\phi-i}$ are not from two consecutive generations.

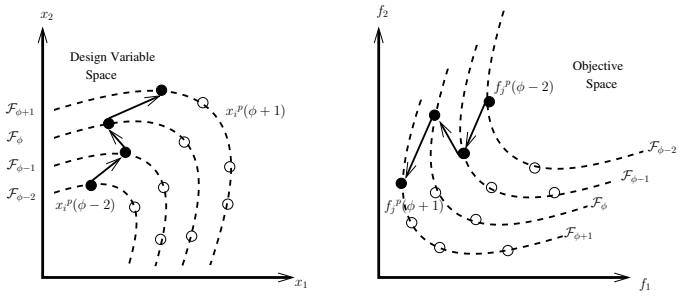


Fig. 3. After non-dominated sorting, individuals in each front are sorted with respect to one objective. For modeling purposes, (•) is considered as the same individual moving from front $\mathcal{F}_{\phi-2}$ to front $\mathcal{F}_\phi$

as output. Again, if we consider an inverse system (with transfer function H^{-1}), then it will generate

$$\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2) \dots\} \quad (10)$$

as output when

$$\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2) \dots\} \quad (11)$$

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"[3] $f_j^p(\phi+1)$. Approximation of $f_j^p(\phi+1)$ is quite straight-forward 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 $\mathcal{F}_{\phi+1}$ will be smaller than that of the current front $\mathcal{F}_\phi$ by some amount Δf. So for a minimization problem,

$$f_j^p(\phi+1) = f_j^p(\phi) - \Delta f \quad (12)$$

The preceding discussions dictate that PFVO 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 value
- from $f_j^p(\phi+1)$, approximate the design variable $x_i^p(\phi+1)$ of the next front $\mathcal{F}_{\phi+1}$

B. Model Formulation

Before describing the model formulation in detail, first we construct a simple linear system for the design variable i and objective j:

$$x_i(\phi) + a_0 x_i(\phi-1) = b_0 f_j(\phi) + b_1 f_j(\phi-1) + \epsilon(\phi) \quad (13)$$

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

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \begin{bmatrix} a_0 \\ b_0 \\ b_1 \end{bmatrix} + \epsilon(\phi) \quad (14)$$

[3]"Mirage Solutions" or "Projected Solutions" means the objective values of the next (future) front that are approximated from current design variables, for more details please see [9] and [8]

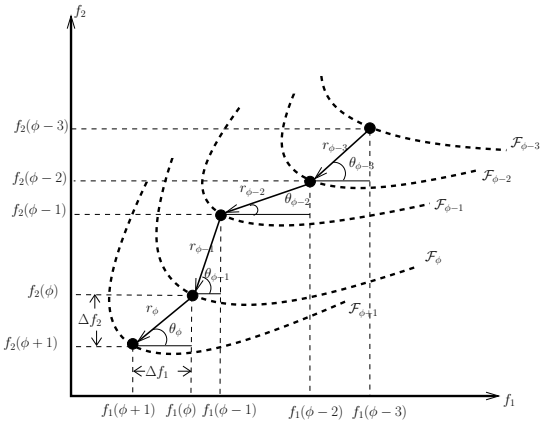


Fig. 4. Calculation of Δf from existing fronts

Now, we can adopt a matrix formation:

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

$$+ \underbrace{\begin{bmatrix} \epsilon(\phi) \\ \epsilon(\phi-1) \\ \epsilon(\phi-2) \\ \vdots \\ \epsilon(2) \end{bmatrix}}_{\boldsymbol{\epsilon}} \tag{15}$$

Or we can rewrite

$$\mathbf{y} = \mathbf{\Phi} \cdot \boldsymbol{\beta_{ij}} + \boldsymbol{\epsilon} \tag{16}$$

Here, matrix $\boldsymbol{\beta_{ij}}$ denotes parameter of the dynamic system for the design variable i and objective j. $\boldsymbol{\beta_{ij}}$ can be approximated using Least Squares. Let us denote $\hat{\boldsymbol{\beta}}_{\boldsymbol{ij}}$ as approximated $\boldsymbol{\beta_{ij}}$:

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{ij}} = \underbrace{(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T}_{\text{pseudo-inverse}}\ \mathbf{y} \tag{17}$$

So,

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{ij}} \tag{18}$$

Now we have $\hat{\boldsymbol{\beta}}_{\boldsymbol{ij}}$. From equation 12 and 18, now we can easily approximate the i^{th} design variable of the next front $\mathcal{F}_{\phi+1}$ -

$$x_i(\phi+1) = \begin{bmatrix} -x_i(\phi) & f_j(\phi+1) & f_j(\phi) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{ij}} \tag{19}$$

$$= \begin{bmatrix} -x_i(\phi) & f_j(\phi)-\Delta f & f_j(\phi) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{ij}} \tag{20}$$

For more details on relevant concepts and formulation of PFVO, readers are referred to [23], [24] and [25].

C. Calculation of Δf

The critical portion of the algorithm is to calculate the approximate objective values of the next front. In the initial implementation of PFVO we have assigned the value of Δf from empirical trials. In most cases, the value of Δf was calculated based on the difference in nadir and utopia objective values [19]. However, the performance was not identical for all problems/host algorithms. So, in the later implementation, we have adopted an adaptive calculation of Δf based on the average Δf of the previous points. Since, PFVO does not necessarily need the exact predicted values of the next front, we have found that a "rough guess" is feasible when approximating the design variables for the next front[4]. Moreover, it is also necessary to limit the computational complexity of the whole procedure. The calculation is conducted as follows for two objectives problem (see Figure 4) -

$$\text{For first objective} \quad f_1, \quad \Delta f_1 = r_\phi \cos\theta_\phi \tag{21}$$

$$\text{For second objective} \quad f_2, \quad \Delta f_2 = r_\phi \sin\theta_\phi \tag{22}$$

$$where, \quad r_\phi = \sum_{i=\phi-1}^{1} \frac{r_i}{N} \quad and \quad \theta_\phi = \sum_{i=\phi-1}^{1} \frac{\theta_i}{N} \tag{23}$$

D. Algorithm

The algorithm for PFVO is listed in Algorithm 1. Readers can review the schematics illustrated in Figure 1 and 5 to clarify understanding of the functionality.

A basic flowchart for the mathematical operations are illustrated in Figure 5. This figure represents the schematic of PFVO for a scenario where the problem has three design variables and two objectives. After non-dominated sorting, three fronts are created. The approximated population has km_ϕ individuals.

E. Complexity Analysis

We now provide a description of 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 [26]. The complexity of the algorithm largely depends on the size of the matrix $\mathbf{\Phi}$ in equation 16. 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 14. We are doing QR factorization on matrix $\mathbf{\Phi}$ 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 $\mathbf{R}$ has a dimension of $t \times t$ and the orthogonal matrix $\mathbf{Q}$ has a dimension of $(|\phi|-1) \times t$. After doing this, we are solving the systems parameter $\boldsymbol{\beta_{ij}}$ in equation 17. This requires one inversion on the upper triangular matrix $\mathbf{R}$, one multiplication on $\mathbf{Q}^T\mathbf{y}$ and another multiplication on

[4]Please note that PFVO is not for exact prediction, it will just generate a "best guess" objectives value for the next front so that the host optimizer can explore the search space in right way.

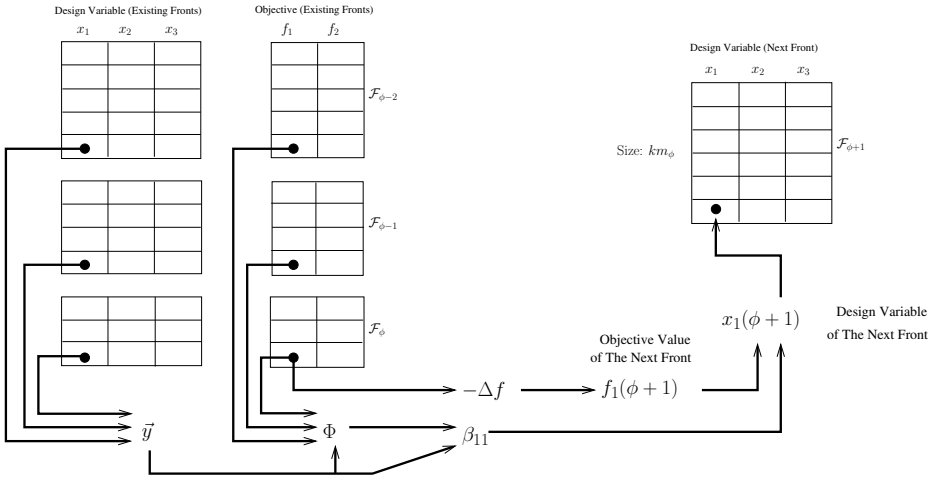


Fig. 5. Mathematical steps in the Pareto-following Variation Operator. Here the operation is illustrated for objective function f_1 and design variable x_1

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

$\mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y}$. So equation 17 and 20 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, $\boldsymbol{\beta_{ij}}$ 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*, [27] for matrix operations.

IV. Integration Mechanism

A. Integration with SPEA-II

In this case, we have applied PFVO on the external archive before the original variation in SPEA-II occurs (see line 15 of Algorithm 2). The overall procedure of this integration scheme is illustrated in Algorithm 2. For the original pseudo code of SPEA-II, please refer to [16]. We have used the original SPEA-II code available at `http://www.tik.ee.ethz.ch/sop/pisa/`.

B. Integration with RM-MEDA

RM-MEDA [17] provides a special interest to our experiment since it does not fall into the general MOEA categories. This algorithm is an instance of so called *Estimation of Distributed Algorithm (EDA)* [28]. EDAs are those type of techniques where normal genetic operators, such as mutation/crossover are not applied, rather probabilistic modeling based techniques are adopted. We have used the original RM-MEDA source code available at `http://privatewww.essex.ac.uk/~azhou/publication.htm`. The integration mechanism is described in Algorithm 3.

V. Simulation

To test the performance, we have executed each pair of algorithms (PFVO enhanced hosting optimizer and the

Algorithm 2 SPEA-II with Pareto Following Variation Operator

Require: Randomly generated parent population P_t at generation t with population size N and an archive of external population $\overline{P}_t$ with size $\overline{N}$
Ensure: After t_{max} number of iteration, population $\overline{P}_{t_{max}}$ will represent solution of the problem.
Start with initial population, $P_0 := \emptyset$ and an empty archive (external set) $\overline{P}_0 := \emptyset$
set $t = 0$
while $t \leq t_{max}$ **do**
  Calculate fitness values of P_t and $\overline{P}_t$
  $\mathcal{F}$:=Apply Non-dominated Sort on P_t, create ϕ number of fronts.
  i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
  $\overline{\mathcal{F}}$:=Apply Non-dominated Sort on $\overline{P}_t$, create ϕ number of fronts.
  i.e. $\overline{\mathcal{F}} := \{\overline{\mathcal{F}}_\phi, \overline{\mathcal{F}}_{\phi-1} \ldots \overline{\mathcal{F}}_1\}$
  Apply environmental selection. Copy all non-dominated individuals from P_t and $\overline{P}_t$ to $\overline{P}_{t+1}$
  if $|\overline{P}_{t+1}| > N$ **then**
    Truncate individuals from $\overline{P}_{t+1}$ using clustering algorithm
  else
    Fill $\overline{P}_{t+1}$ with dominated individuals from P_t and $\overline{P}_t$
  end if
  Again apply non-dominated sort on $\overline{P}_{t+1}$ and create ϕ number of fronts.
  i.e. $\overline{\mathcal{F}} := \{\overline{\mathcal{F}}_\phi, \overline{\mathcal{F}}_{\phi-1} \ldots \overline{\mathcal{F}}_1\}$
  if $\phi > 1$ **then**
    $\overline{\mathcal{F}}_{\phi+1}$:=Pareto Following Variation($\overline{P}_{t+1}$,Δf)
    $|\overline{\mathcal{F}}_{\phi+1}| = km_\phi$
    Insert newly approximated population to $\overline{P}_{t+1}$, $\overline{P}_{t+1} := \overline{P}_{t+1} \cup \overline{\mathcal{F}}_{\phi+1}$
  end if
  Perform binary tournament selection with replacement on $\overline{P}_{t+1}$ to fill the mating pool.
  Apply crossover and mutation on individuals in $\overline{P}_{t+1}$ and copy them to P_{t+1}
  set $t := t + 1$
end while

Algorithm 3 RM-MEDA with Pareto Following Variation Operator

Require: Randomly generated parent population P_t at generation t with population size N
Ensure: After t_{max} number of iteration, population $P_{t_{max}}$ will represent solution of the problem.
Start with initial population, $P_0 := \emptyset$
set $t = 0$
while $t \leq t_{max}$ **do**
  Calculate fitness values of P_t
  set $t := t + 1$
  Build the probability model for the distribution of the solutions in P_t using Local Principal Component Analysis (Local-PCA).
  Generate new solution set Q from the built probability model.
  Evaluate fitness values of individuals in Q
  Select N individuals from P_t and Q and create new population P_{t+1}
  i.e. $P_{t+1} = P_t \cup Q$
  $\mathcal{F}$:=Apply Non-dominated Sort on P_{t+1}, create ϕ number of fronts.
  i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
  if $\phi > 1$ **then**
    $\mathcal{F}_{\phi+1}$:=Pareto Following Variation(P_{t+1},Δf)
    $|\mathcal{F}_{\phi+1}| = km_\phi$
    Insert newly approximated population to P_{t+1}, $P_{t+1} := P_{t+1} \cup \mathcal{F}_{\phi+1}$
    Retain the size of population in P_{t+1} to N. If it exceeds this limit, just remove the individuals from the worst front.
  end if
  set $t := t + 1$
end while

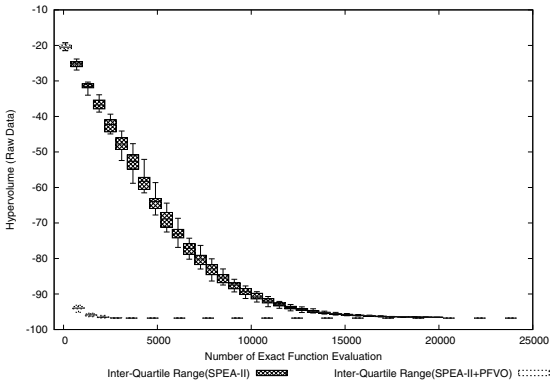
hosting optimizer without PFVO) on two benchmark problem sets: the ZDT [29] and DTLZ [30] test suites. We have executed each scenario (base algorithms, PFVO enhanced algorithm, problem set) 30 times with different random seeds (random seeds for each pair of the algorithms were same). From the collected solutions sets ($\mathcal{PF}^*$ values), we have calculated the *Hypervolume*(I_H) and *Epsilon*($I_{\epsilon+}$) indicator values. These indicators are two of the most widely used performance metrics in MOEA research. The space constraints does not permit us to present the detailed formulation of these indicators. For more details, readers are referred to [31] and [32].

The indicator values are presented with respect to the number of exact function evaluations (FE) so that we can draw clear conclusions about any difference in convergence speeds (see Figure 6 and 7). Please note that lower values of I_H indicates a better front. Therefore, the algorithm which shows lower values for I_H in a smaller number of FE is considered to be better. On the other hand, for *Epsilon* indicator, $I_{\epsilon+}(PFVO, HostOptimizer) \leq 0$, $I_{\epsilon+}(HostOptimizer, PFVO) > 0$ indicates "Host Optimizer + PFVO" is better than "Host Optimizer" (and vice-versa). If $I_{\epsilon+}(PFVO, HostOptimizer) > 0$, $I_{\epsilon+}(HostOptimizer, PFVO) > 0$; they are incomparable. Unfortunately, due to the space constraints, we are unable to provide the results of each pair of algorithms for all test problems, so we have included some of them in the next subsections. For interested readers, detailed results and analysis are available online at `http://www.csse.unimelb.edu.au/~akmkat`.

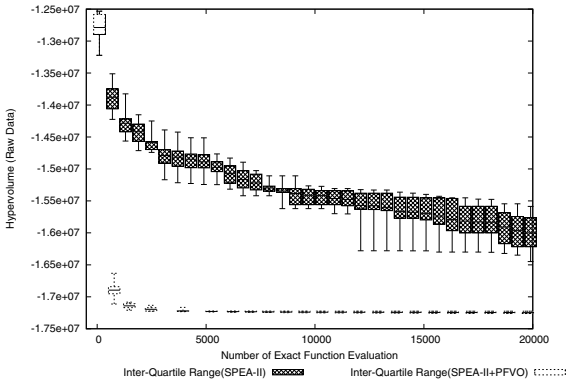
A. Analysis

From Figure 6(a), we can see that the PFVO enhanced SPEA-II can reach the $\mathcal{PF}^*$ within approximately 5000 function evaluations for ZDT6 and if we execute SPEA-II alone, it takes 20,000 function evaluations. For DTLZ3, we see convergence rate is also promising. For RM-MEDA on ZDT4, PFVO is capable of improving the speed of convergence. On the other hand, in terms of the $I_{\epsilon+}$ indicator, the performance of both of the base algorithms was improved when we integrated the PFVO with them. These result clearly indicate that our approximation method can track the search path in right direction so that the hosting optimizer can converge to the $\mathcal{PF}^*$ faster by avoiding the evaluation and/or creation of non-promising solutions. In our previous work [24], we have integrated PFVO into NSGA-II [33] and found similar improvement.

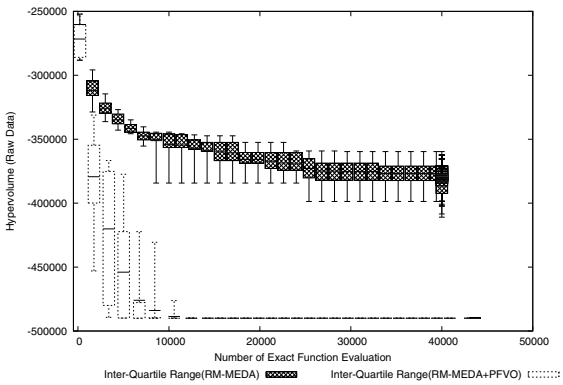
For statistical analysis, we have executed each pair of algorithms on every problem for 30 trials. In each trial, the I_H and $I_{\epsilon+}$ values were collected at every generation. From these values, we have conducted a Kruskal-Wallis [34] test, with $\alpha = 0.01$ (i.e. 99% significance). The detailed p-value results are provided in `http:\\www.csse.unimelb.edu.au\~akmkat`. Please note that, for all experiments, we have used the same parameters as suggested in both host algorithm [16], [17] implementations.



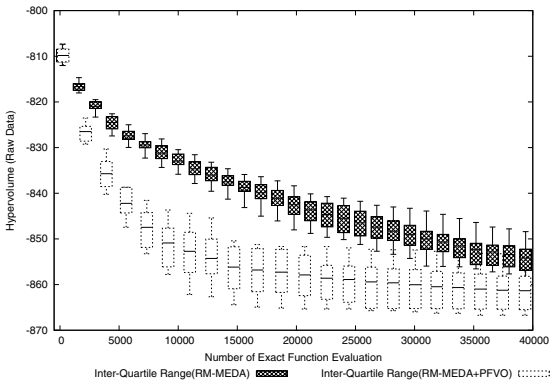
(a) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT6



(b) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ3

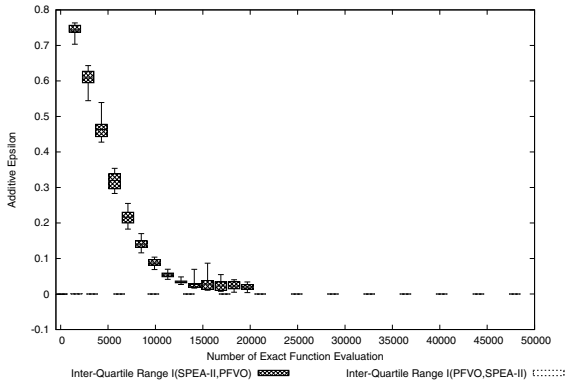


(c) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT4

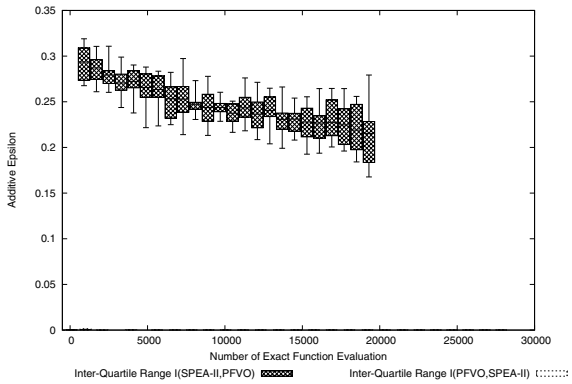


(d) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ6

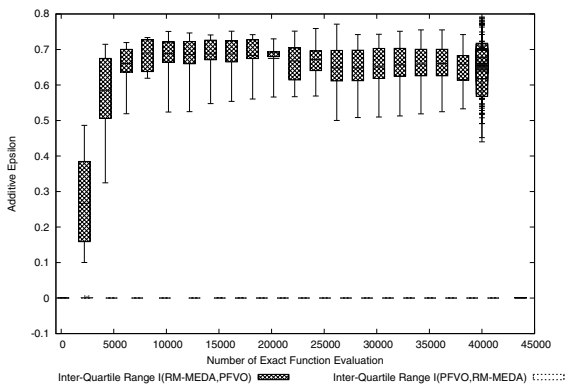
Fig. 6. Boxplots of I_H vs. Exact Number of Function Evaluations (FE)



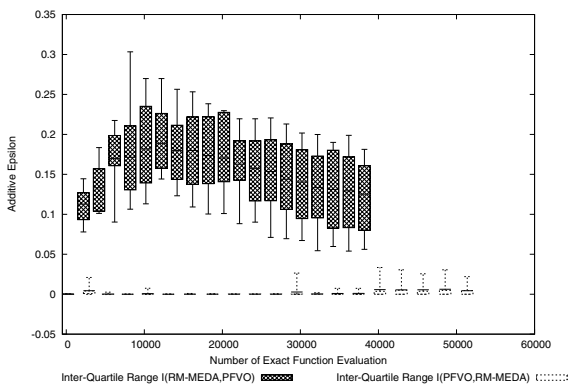
(a) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT6



(b) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ3



(c) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT4



(d) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ6

Fig. 7. Boxplots of $I_{\epsilon+}$ vs. Exact Number of Function Evaluations (FE)

VI. Conclusion and Future Work

In our previous work [23], [24], the value of Δf used in the PFVO was estimated from empirical trials. This value had to be tuned manually for different problems and different host algorithms. In the PFVO implementation described in this paper, we have adopted an "adaptive" calculation for Δf. We have also shown that PFVO can be easily "plugged into" a wide range of baseline MOEA's, thereby improving the convergence speed of the algorithm. One limitation of the PFVO, is the fact that is does not scale to more than two objectives. However, in another version of PFVO (implemented from the dynamic system identification in Fourier domain [23]), we did report results for three objective problems. In the next version of PFVO, we will explore similar ideas to solve the scalability issue.

References

[1] N. Queipo, R. Haftka, W. S. and. T. Goel, and R. Vaidyanathan, "Surrogate-based analysis and optimization," *Journal of Progress in Aerospace Sciences*, vol. 41, pp. 1–28, 2005.

[2] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[3] I. Kampolis, A. Zymaris, V. Asouti, and K. Giannakoglou, "Multilevel optimization strategies based on metamodel-assisted evolutionary algorithms, for computationally expensive problems," in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC '07)*, September 2007, pp. 4116–4123.

[4] W. Liu, Q. Zhang, E. P. K. Tsang, C. Liu, and B. Virginas, "On the Performance of Metamodel Assisted MOEA/D." in *ISICA-2007*, ser. LNCS, L. Kang, Y. Liu, and S. Y. Zeng, Eds., vol. 4683. Wuhan, China: Springer, August 2007, pp. 547–557.

[5] I. Voutchkov and A. J. Keane, "Multiobjective Optimization using Surrogates," in *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, I. C. Parmee, Ed. Bristol, UK: The Institute for People-centred Computation, April 2006, pp. 167–175.

[6] J. Knowles, "ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, February 2006.

[7] L. Santana-Quintero, V. Serrano-Hernandez, C. C. Coello, A. Hernandez-Diaz, and J. Molina, "Use of Radial Basis Functions and Rough Sets for Evolutionary Multi-Objective Optimization," in *Proceedings of The IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, April 2007, pp. 107–114.

[8] 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)*. Honolulu, HI: IEEE Press, April 2007, pp. 325–332.

[9] 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)*. London, England: ACM Press, NY, USA, 2007, pp. 734–740.

[10] M. Farina, "A Neural Network Based Generalized Response Surface Multiobjective Evolutionary Algorithm," in *IEEE Congress on Evolutionary Computation (CEC-2002)*, vol. 1. Honolulu, HI: IEEE Press, May 2002, pp. 956–961.

[11] A. Gaspar-Cunha, A. Vieira, and C. M. Fonseca, "Multi-Objective Optimization: Hybridization of an Evolutionary Algorithm with Artificial Neural Networks for fast Convergence," in *Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*, Nottingham, UK, November 2004.

[12] 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)*. London, England: ACM Press, 2007, pp. 1288–1295.

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

[14] A. Gaspar-Cunha and A. Vieira, "A hybrid multi-objective evolutionary algorithm using an inverse neural network," in *Hybrid Metaheuristics*, C. Blum, A. Roli, and M. Sampels, Eds., 2004, pp. 25–30.

[15] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, Mar 1990.

[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[17] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, February 2008.

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

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

[20] C. A. C. Coello and G. B. Lamont, *Applications of Multi-Ojective Evolutionary Algorithms*, ser. Advances In Natural Computation. World Scientific Publishing, Singapore, 2004, vol. 1.

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

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

[23] A. K. A. Talukder and M. Kirley, "A pareto following variation operator for evolutionary dynamic multi-objective optimization," *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2270–2277, June 2008.

[24] A. K. A. Talukder, M. Kirley, and R. Buyya, "A pareto following variation operator for fast-converging multiobjective evolutionary algorithms," in *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2008, pp. 721–728.

[25] A. K. A. Talukder, "Towards high speed multiobjective evolutionary optimizers," in *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2008, pp. 1791–1794.

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

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

[28] P. Larrañga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[29] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[30] K. Deb, L. Thiele, M. Laumanns, and E. Ziztler, "Scalable Test Problems for Evolutionary Multi-objective Optimization," India, Tech. Rep. KanGAL Report No 2001001, August 2001.

[31] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, Tech. Rep. 214, February 2006, revised Version.

[32] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, April 2003.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[34] W. Connover, *Practical Nonparametric Statistics*, 3rd ed. New York, NY: John Wiley and Sons, 1999.