Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Modeling Regularity to Improve Scalability of Model-Based Multiobjective Optimization Algorithms

Yaochu Jin, Aimin Zhou, Qingfu Zhang, Bernhard Sendhoff, Edward Tsang

2008

Preprint:

This is an accepted article published in Multiobjective Problem Solving from Nature. The final authenticated version is available online at: https://doi.org/[DOI not available]

To appear in: J. Knowles et al (Eds.), Multiobjective Problem Solving from Nature, Springer, 2008

Modeling Regularity to Improve Scalability of Model-based Multi-objective Optimization Algorithms

Yaochu Jin[1], Aimin Zhou[2], Qingfu Zhang[2], Bernhard Sendhoff[1], and Edward Tsang[2]

[1] Honda Research Institute Europe
Carl-Legien-Str. 30
63073 Offenbach, Germany
{yaochu.jin,bernhard.sendhoff}@honda-ri.de
[2] Department of Computer Science
University of Essex
Wivenhoe Park, Colchester, CO4 3QS, UK
{azhou,qzhang,edward}@essex.ac.uk

Summary. Model-based multi-objective optimization is one class of meta-heuristics for solving multi-objective optimization problems, where a probabilistic model is built from the current distribution of the solutions and new candidate solutions are generated from the model. One main difficulty in model-based optimization is to construct a probabilistic model that is able to effectively capture the structure of the problems to enable efficient search. This chapter advocates a new type of probabilistic models that takes the regularity in the distribution of Pareto-optimal solutions into account. We compare our model to two other model-based multi-objective algorithms on a number of test problems to demonstrate that our algorithm is scalable to high-dimensional optimization problems with or without linkage among the design variables.

1 Introduction

The last decade has witnessed a great success of evolutionary algorithms and other population-based meta-heuristic search methods in solving multi-objective optimization problems [8]. Nevertheless, several challenges still remain to be addressed for population-based search methods to deal with hard, real-world optimization problems. One of these challenges is algorithms' ability to efficiently solve optimization problems of a high search dimension, which is often known as the scalability of optimization algorithms.

For evolutionary multi-objective algorithms to be scalable to high search dimensions, they must be able to effectively take advantage of domain knowledge of the problem at hand during the search. Unfortunately, major search

operators of conventional evolutionary algorithms, such as crossover and mutation, are not efficient in taking problem-specific knowledge into account in search. To address this weakness, several approaches have been suggested for incorporating domain knowledge into evolutionary algorithms to guide the sampling process [13], among which model-based optimization methods, such as the estimation of distribution algorithms(EDAs) [6, 18], have widely been studied. It should be noticed that existing EDAs have mainly been developed to solve scalar optimization problems, which are not necessarily suited for solving multi-objective problems.

Another weakness of evolutionary algorithms that use crossover and mutation for generating new candidate solutions is that they do not explicitly exploit the correlation between design variables (also known as variable linkage) [10]. Model-based algorithms are believed to be able to learn the linkage among variables. However, the ability of learning linkage can be at the cost of scalability, if the probabilistic model is not chosen appropriately.

This chapter presents a methodology for incorporating additional knowledge into building probabilistic models for solving continuous multi-objective problems. The domain knowledge we use here is the regularity in the distribution of Pareto-optimal solutions, which has largely been overlooked in developing evolutionary multi-objective optimization algorithms. Since regularity is a general property for a large class of multi-objective problems, the proposed framework is applicable to a wide range of real-world problems.

The remainder of the chapter is organized as follows. A brief introduction to solving multi-objective optimization problems using a probabilistic model, together with a short discussion on the main difficulties of model-based algorithms is presented in Section 2. Three model-based multi-objective optimization algorithms, including the one that takes regularity into account, are described in details in Section 3. Section 4 provides the experimental setup, such as parameter settings of algorithms, the test functions, and the performance indicators for quantitatively evaluating the performance of the algorithms. Comparison results of the three models with respect to algorithms' scalability and ability to handle variable linkages are presented in Section 5. A summary and conclusions of the chapter are provided in Section 6.

2 Probabilistic Modeling for Multi-objective Optimization

The basic idea of population-based search using a probabilistic model is first to estimate the probability distribution of the solutions previously generated over and then generate new candidate solutions by sampling the probabilistic model, as show in Fig. 1.

A large family of probabilistic models can be used to estimate the distribution of continuous and discrete functions [18]. In this chapter, we limit our discussions to continuous optimization problems, where Gaussians or a

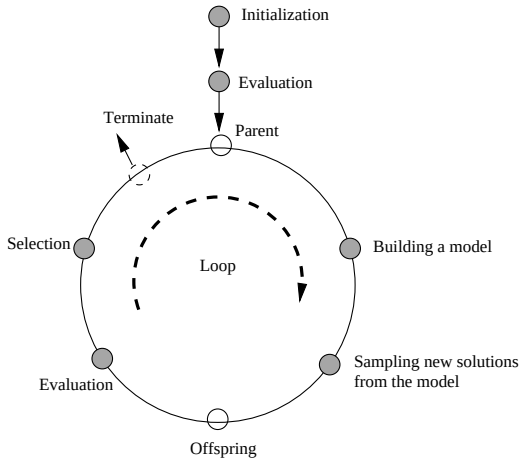

Fig. 1. A generic framework for model-based optimization algorithms using a population.

mixture of Gaussians are employed to model the distribution of the function to be optimized.

It would be ideal if we could use a full, joint probability distribution model, i.e., a probabilistic model that considers dependency between all variables. Unfortunately, accurate estimation of a full joint distribution model in a high-dimensional space remains an open problem. In order to estimate the distribution accurately, a huge number of data samples are needed, which is impractical in solving real-world problems due to the fact that calculation of the function value for a given design (often known as fitness evaluation in evolutionary optimization) is computationally very expensive. In this context, EDAs that require a huge population size are of very limited practical importance.

Several techniques have been adopted to address the curse of dimensionality. The simplest way to cope with high dimensionality is to neglect the linkage between variables and build a univariate distribution model for each variable [22, 30]. Unfortunately, such models are not able to capture the dependency between variables and they are not recommendable if there are strong correlations between the variables. One popular approach is to use factorized univariate or multivariate distributions, which are able to capture the independence between the variables. A multivariate factorized probability distribution is a probabilistic model in the form of a product of probability density functions. Both univariate factorization [3, 19] and multivariate factorization [3, 19, 4] have been employed for model-based optimization.

A natural extension to models consisting of a single factorized probability distribution is to use a weighted sum of single factorized distributions, which is usually known as mixture of Gaussians. Such models can often be obtained by dividing the search space into a number of subspaces and then a single

factorized distribution is constructed for each cluster [4]. This method is of particular interest for multi-modal scalar optimization and multi-objective optimization, where more than one solution needs to be achieved.

Although multivariate factorization is able to capture the dependency among at least two variables, it is not straightforward to select a model that is optimal for a given problem [5]. Another approach to factorization is to map the high-dimensional search space onto a latent space of a lower dimensionality and then a univariate or multivariate factorized distribution can be built. The mapping from the high-dimensional design space to the low-dimensional latent space can often be realized using dimension reduction techniques such as the principal component analysis [1]. Model-based optimization algorithm using a distribution model in latent space has been reported in [7, 27, 24]. One main difficulty is to determine the dimension of the latent space.

2.1 Modeling Regularity in Multi-objective Optimization

As previously discussed, incorporation of knowledge into search process helps to improve the search performance, especially the scalability of the search algorithms to high search dimensionality. In addition to domain knowledge that is specific to each particular problem, regularity in the distribution of the Pareto-optimal solutions is a nice property that holds for a large class of multi-objective optimization problems. So far, this nice property has largely been overlooked. The importance of taking advantage of regularity in evolutionary multi-objective optimization was first advocated in [14], where it is suggested that the success of local search in multi-objective optimization can most probably be attributed to the fact that local search is able to implicitly exploit the regular distribution of Pareto-optimal solutions. In that work, piece-wise linear models are constructed in the design space using the nondominated solutions achieved by an evolutionary algorithm. It has been demonstrated that the quality of the solutions generated from the linear models are better than the original solutions.

The regularity property can be induced from the Karush-Kuhn-Tucker condition [21, 26], which indicates that under certain smoothness conditions, the Pareto-optimal set in the design space of a continuous multi-objective optimization problem is an $(m-1)$-dimensional piecewise continuous manifold, where m is the number of the objectives.

The question now is how to efficiently exploit the regularity property using model-based multi-objective optimization. Although it is believed that model-based optimization is able to learn the problem structure, it must be pointed out that the model's ability to capture the problem structure heavily depends on the model in use. This is particularly true for multi-objective optimization, where the final solution is a Pareto front consisting of multiple solutions rather than a single optimum.

Most existing model-based multi-objective optimization algorithms for solving continuous problems employ Gaussian distributions with few excep-

tions, e.g., in [24], where a Voronoi mesh has been adopted. The most important *a priori* knowledge that can be derived from the regularity condition is that the Pareto front in the original n-dimensional search space can be modeled in an $(m-1)$-dimensional space without any information loss, where n is the dimensionality of the search space and in most cases, we have $m \ll n$. This knowledge removes exactly the main obstacle in latent variable based models, where the dimension of the latent space must be specified. Besides, knowing that the Pareto front is a principal curve or surface, we believe that first-order or second-order polynomials might be more efficient than Gaussian models in modeling the regular distribution of the Pareto-optimal solutions.

Take bi-objective optimization problems as an example. The regularity property has two implications. First, the Pareto front can be described by one or a few sections of one-dimensional model, regardless how large the design space is. Second, a linear curve is more efficient in leading the population to the final Pareto front, as illustrated in Fig. 2.

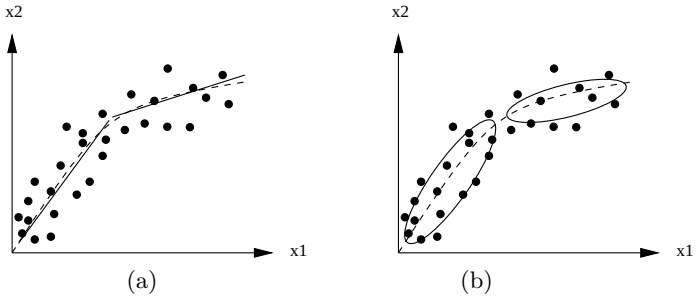

Fig. 2. Modeling Pareto set using (a) linear models; (b) Gaussian models.

The idea of modeling regularity in model-based multi-objective optimization has most recently been exploited by the authors and very competitive results have been achieved [31, 32, 33, 34] compared to some of the state-of-the-art evolutionary multi-objective optimization such as NSGA-II [9], GDE3 [17], and MIDEA [2]. In the following, we are going to compare one model-based multi-objective algorithm that exploits regularity to two other model-based multi-objective optimization algorithms with respect to scalability to search dimension, ability to handle variable linkage, and sensitivity to population size.

3 Three Model-based Algorithms

3.1 Regularity-based Latent Principal Curve Model (LPCM)

Modeling in a latent space is an attractive idea because the dimension of the latent space is usually much lower than that of the design space. According

to the regularity condition, the Pareto front of an m-objective optimization problem can be modeled in an $(m-1)$-dimensional space. For this purpose, the local principal curve analysis (LPCA) algorithm [15] has been employed. One elegant property of LPCA is that it simultaneously groups the population into a number of clusters while mapping it from the n-dimensional design space to the $(m-1)$-dimensional latent space.

The points in the k-th cluster (denoted by C^k) can be described by a uniform distribution on a $(m-1)$-dimensional manifold M^k:

$$P^k(S) = \begin{cases} \frac{1}{V^k}, \text{ if } S \in M^k, \\ 0, \quad \text{else} \end{cases} \tag{1}$$

where S is an $(m-1)$-dimensional random vector in the latent space, V^k is the volume of M^k bounded by:

$$a_i^k \leq s_i \leq b_i^k, i = 1, \cdots, (m-1), \tag{2}$$

and

$$a_i^k = \min_{X \in C^k} (X - \bar{X}^k)^T U_i^k, \tag{3}$$

$$b_i^k = \max_{X \in C^k} (X - \bar{X}^k)^T U_i^k, \tag{4}$$

where $\bar{X}^k$ is the mean of the points in C^k, U_i^k is the i-th principal component of the data in cluster C^k.

While the uniform distribution defined in Eqn.(1) is used to capture the regularity (centroid) in the distribution of the population, local dynamics of the population is described by an n-dimensional zero-mean Gaussian distribution in the design space:

$$N^k(X) = \frac{1}{(2\pi)^{n/2} |\Sigma^k|^{n/2}} \exp\left\{ -\frac{1}{2} X^T \Sigma^k X \right\}, \tag{5}$$

where $\Sigma^k = \delta^k I$, I is an $n \times n$ dimensional identity matrix, and δ^k is calculated by:

$$\delta^k = \frac{1}{n-m+1} \sum_{i=m}^{n} \lambda_i^k, \tag{6}$$

where λ_i^k are the i-th largest eigenvalue of the covariance matrix of the points in cluster k. Here, we assume that the inequality $n > m-1$ always holds. An illustration of the principal curve and the Gaussian models in a one dimensional latent space is provided in Fig. 3, where the population is divided into two clusters.

During the sampling process, the probability at which the model of cluster k is chosen is determined by

$$p(k) = \frac{V^k}{\sum_{k=1}^{K} V^k}, \tag{7}$$

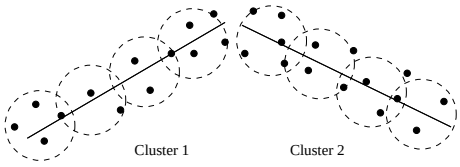

Fig. 3. Modeling the distribution of the population using a uniformly distributed principal curve and a Gaussian perturbation.

where V^k is the volume of the $(m-1)$-dimensional manifold. In case of a curve, it is the length of the curve.

The sampling process consists of three steps. In the first step, a point is generated on $(m-1)$-dimensional manifold M^k according to Eqn.(1), and is then mapped onto the n-dimensional design space. Assume S is an $(m-1)$-dimensional random vector generated in M^k for the k-th cluster, it is mapped onto the n-dimensional design space in the following way, if the manifold M^k is a first-order principal curve:

$$X_1 = \Theta_0^k + \Theta_1^k S \tag{8}$$

where X_1 is an n-dimensional random vector, Θ_0^k is the mean of data in cluster $C(X)^k$, and Θ_1^k is $n \times (m-1)$-dimensional matrix, which is composed of the eigenvectors corresponding to the $(m-1)$ largest eigenvalues.

In the second step, an n-dimensional random vector X_2 is generated from the Gaussian distribution defined in Eqn.(5). Finally, the following new candidate solution is generated:

$$X = X_1 + X_2. \tag{9}$$

This process continues until all offspring are generated.

3.2 Univariate Factorized Gaussian Model (UGM)

The basic idea on using a univariate factorized normal distribution for modeling the population has been considered in [2]. Before constructing the models, the population is divided into K clusters. For this purpose, the leader clustering algorithm [12] is employed, as suggested in [2]. In this clustering algorithm, it is not necessary to define the number of clusters, however, a threshold that defines the radius of the clusters must be given, which basically determines the number of clusters. One major drawback of the leader algorithm is that the clustering result is sensitive to the choice of the initial cluster center (leader). For cluster $k, k = 1, 2, ..., K$, a Gaussian model is then constructed for each search dimension:

$$p_i^k(x_i) = \frac{1}{\delta_i^k \sqrt{2\pi}} \exp\left\{ -\frac{(x_i - \mu_i^k)^2}{2(\delta_i^k)^2} \right\}, \tag{10}$$

where μ_i^k and δ_i^k are the mean and standard deviation of the Gaussian model for variable $i = 1, ..., n$. The mean and standard deviation of the univariate Gaussian distribution for cluster k can be calculated according to the individuals that are assigned to the cluster.

During the sampling process, one of the K clusters is chosen randomly at a probability of $1/K$. For the chosen cluster, one new candidate solution is generated using the n Gaussian models for each design variable. This procedure repeats until all the offspring solutions are generated.

3.3 Marginalized Multivariate Gaussian Model (MGM)

Univariate factorized Gaussian models neglect any correlation between the variables. As a result, the model cannot effectively learn the problem structure if there is dependency among the variables. To address problem, a joint Gaussian distribution model is considered in this model. However, building an accurate full joint distribution model in a high-dimensional space is almost intractable. For this reason, the population is first grouped into a number clusters and then a joint distribution model is built for each cluster. To cluster the population, the k-means clustering algorithm [12] is adopted. Therefore, the number (K) of clusters needs to be predefined by the user.

For k-th cluster, the following joint distribution model is constructed:

$$p^k(X) = \frac{1}{(2\pi)^{n/2}|\Sigma^k|^{n/2}} \exp\left\{-\frac{1}{2}(X - \Lambda^k)^T(\Sigma^k)^{-1}(X - \Lambda^k)\right\}, \qquad (11)$$

where X is an n-dimensional design vector, Λ^k is an n-dimensional vector of the mean value and Σ^k is an $n \times n$ covariance matrix estimated by the individuals in the k-th cluster.

Different to the univariate factorized model, the probability of sampling the model of the k-th cluster is calculated as follows:

$$p(k) = \frac{N^k}{\sum_{k=1}^{K} N^k}, \qquad (12)$$

where N^k is the number of individuals in the k-th cluster.

3.4 The General Algorithm Framework

For a fair comparison, all three algorithms use the same selection strategy, i.e., the MaxiMin sorting selection algorithm suggested in [28], which is a variant of the crowded non-dominated sorting selection proposed in [9]. The first steps in the MaxiMin sorting are the same as those in the crowded non-dominated sorting. First, the parent and offspring populations are combined. Second, the combined population is sorted according to the non-dominance ranks. During the ranking, non-dominated solutions in the combined population are assigned

with a rank 1, which belongs to the first non-dominated front. These individuals are removed temporarily from the population and the non-dominated individuals in the rest of the population are identified, which consists of the second non-dominated front of the population and are assigned with a rank 2. This procedure repeats until all individuals in the combined population are assigned with a rank from 1 to R, assuming that R non-dominated fronts can be identified in total. Instead of calculating the crowding distance as done in NSGA-II, selection starts directly after non-dominated sorting. During selection, solutions on the first non-dominated front are passed to the parent population of the next generation. If the number of solutions on the first non-dominated front is smaller than the population size, those on the second non-dominated front are moved to the parent population. However, it can happen that only part of the solutions on a non-dominated front can be selected. Let us assume there are L solutions on the j-th non-dominated front, and only M solutions are to be selected, where $M < L$. In NSGA-II, M solutions with the largest crowding distances are selected. In the MaxiMin selection method, the extreme solutions on the concerned non-dominated front is selected. Then, the solution that has the maximal distance to the selected solutions from the same non-dominated front are selected first. This process is continued until the parent population is filled up. An illustrative example is provided in Fig. 4. Assume that we need to select 10 solutions from 20 in the combined population. We first select the 6 solutions on the first non-dominated front. On the second non-dominated front, there are 6 solutions, from which 4 will be selected. According to the MaxiMin method, the two extreme solutions A and B are first selected. Then, solution C is selected because the minimal distance from solution C to those selected from the second non-dominated front (A and B) is the largest. Afterwards, solution D is selected because its minimal distance to the selected solutions (A, B, and C) is the maximal. It has been shown that the MaxiMin approach can lead to more diverse population with a lower computational complexity compared to the crowded non-dominated sorting selection method [28].

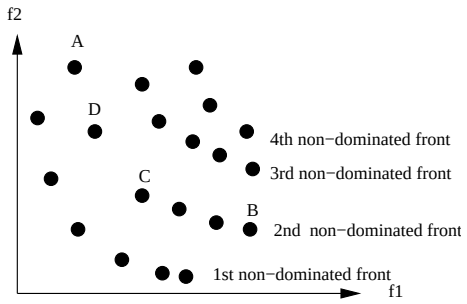

Fig. 4. MaxiMin non-dominated selection.

It should be noted that the reproduction strategy in our work is also different to that in MIDEA [2]. In this work, all solutions in the parent population is used to construct the probabilistic model, whereas in MIDEA, only a portion of individuals in the parent population is used. In addition, the number of new candidate solutions (offspring) generated from the model equals the number of parents, while in MIDEA, only those solutions that are not used in model building are replaced by newly generated offspring.

A generic diagram of the model-based multi-objective optimization algorithms studied in this work is presented in Fig. 5.

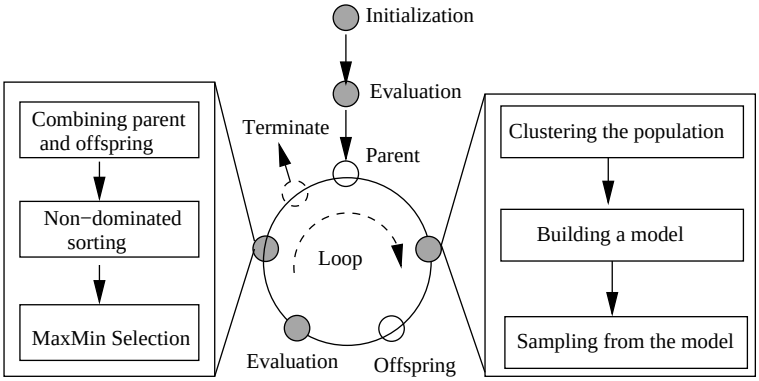

Fig. 5. A generic diagram of the model-based optimization algorithms.

4 Experimental Setup

4.1 Parameter Settings

To investigate the scalability of the algorithms' performance to the search dimension, we have performed simulations on the test problems with a dimension of $10, 20, 30, 40, 50, 60, 70, 80, 90, 100$. The sensitivity of the search performance to the size of the population is also studied. To this end, we have used a population size of 20,30,40,50,60,70,80,90,100,200,400, 600,800, and 1000. The baseline for comparison is the maximum number of fitness evaluations, which is listed in Table 1.

In UGM, the clustering of the population is conducted in the objective space using the leader algorithm, as recommended in [2]. The threshold used in clustering is set to 0.2, and a maximum of 10 clusters is allowed. For MGM, the population is clustered using the k-means algorithm, and the cluster number is set to 3 if the population size is smaller than or equal to 50, otherwise, the cluster number is set to 5. Note, however, that clustering for MGM is done in the design space. The reason why population clustering is carried out in

different spaces is that the leader algorithm produces better results in the objective space while the k-means algorithm shows more stable results in the parameter space.

In LPCM, the clustering of the population is conducted using the local principal component analysis algorithm, where the number of clusters is predefined to 3 for a population size smaller than or equal to 50. In case of a population size larger than 50, the number of clusters is defined to be 5.

Table 1. Maximum Evaluations

Pop Size	Max Evaluation	Max Gen
20	20000	1000
30	30000	1000
40	40000	1000
50	50000	1000
60	30000	500
70	35000	500
80	40000	500
90	45000	500
100	50000	500
200	60000	300
400	60000	150
600	60000	100
800	60000	75
1000	60000	60

4.2 Test Functions

The performance of the algorithms are studied on six test functions. Three of them are taken directly from the widely used ZDT test functions [35], namely, ZDT1, ZDT2, and ZDT3, whose Pareto front is convex, concave and discontinuous. Note that the ZDT test functions are slightly modified so that the Pareto front in the design space is shifted to:

$$x_2 = \cdots = x_n = 0.2,\ x_1 \in [0, 1].$$

In the ZDT test functions, there is no dependency among the design variables. To investigate how the algorithms can deal with variable linkage, three additional test functions derived from the ZDT test functions are also considered, which are termed ZDT1.2, ZDT2.2, and ZDT3.2. The Pareto front of these three test functions in the objective space is completely the same as the corresponding ZDT functions. However, there is a nonlinear dependency between the design variables. The mathematical description of the six test functions are presented in Table 2.

Table 2. Test Instances

Test function	Search space	Objectives
$ZDT1$	$[0,1]^n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{f_1(x)/g(x)}]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i-0.2)^2)/(n-1)$
$ZDT2$	$[0,1]^n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-(f_1(x)/g(x))^2]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i-0.2)^2)/(n-1)$
$ZDT3$	$[0,1]^n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{f_1(x)/g(x)}-\frac{x_1}{g(x)}sin(10\pi x_1)]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i-0.2)^2)/(n-1)$
$ZDT1.2$	$[0,1]^n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i^2-x_1)^2)/(n-1)$
$ZDT2.2$	$[0,1]^n$	$f_1(x) = \sqrt{x_1}$ $f_2(x) = g(x)[1-(f_1(x)/g(x))^2]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i^2-x_1)^2)/(n-1)$
$ZDT3.2$	$[0,1]^n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{x_1/g(x)}-\frac{x_1}{g(x)}sin(10\pi x_1)]$ $g(x) = 1+9(\sum_{i=2}^{n}(x_i^2-x_1)^2)/(n-1)$

4.3 Performance Indicators

To evaluate the performance of the algorithms, we adopted two performance indicators(PIs). The first PI is the inverted generational distance (IGD)[25], which is derived from the generational distance (GD) suggested in [29, 9]. IGD can be expressed as follows:

$$D(P,P^*) = \frac{1}{|P^*|}\sum_{x\in P^*}||x-x'||_2, \tag{13}$$

where P is the nondominated set achieved by the optimization algorithm, P^* is a reference Pareto-optimal set uniformly sampled from the true Pareto front, x is a solution in reference set P^*, and x' is a solution in set P that has the minimal distance to x. If the reference set represents the true Pareto

front adequately well, IGD can effectively measure the accuracy as well as the diversity of the achieved set P. The inverted generational distance is called D-metric hereafter.

The second PI we adopted in the comparison is the difference of the hypervolume (I-Metric for short) between the reference set P^* and the achieved set P [16]:

$$I_H^-(P) = I_H(P^*) - I_H(P), \tag{14}$$

where $I_H(P^*)$ and $I_H(P)$ are the hypervolume of P and P^*, respectively.

5 Simulation Results

5.1 Scalability to Search Dimension: Without Dependency

The first set of simulations has been performed to study the scalability of the three algorithms to search dimension for a given population size (100) on the three test functions without variable linkage. The simulation results on ZDT1, ZDT2, and ZDT3 are provided in Figs. 6, 7, and 8, respectively, where the best and worst Pareto fronts from 30 independent runs according to the D-metrics are plotted. It can be seen from the figures that the results from LPCM, UGM, and MGM, which are presented on the left, the middle, and the right panels of the figures, are quite similar when the dimension changes from 20 to 100, though degradation in the performance of the MGM is a little more serious than that of LPCM and UGM. This observation can be confirmed by the D-metric and the I-metric of the results listed in Table 3 and Table 4, respectively, in which the mean and standard deviation of 30 runs are listed.

The results indicate that both LPCM and UGM have very good scalability to search dimension for problems without variable linkage. It is worth noticing that the performance of MGM is also quite good, probably due to the fact that the distribution of the Pareto front in ZDT1, ZDT2, and ZDT3 is quite easy to model.

Table 3. Mean and Std. of the D-metric for test functions without variable linkage.

Instance	Method	Search Dimension				
		20	40	60	80	100
ZDT1	LPCM	0.0043±0.0001	0.0044±0.0001	0.0047±0.0001	0.0051±0.0001	0.0057±0.0002
	UGM	0.0043±0.0002	0.0044±0.0002	0.0046±0.0002	0.0049±0.0002	0.0053±0.0002
	MGM	0.0046±0.0002	0.0046±0.0002	0.0052±0.0004	0.0069±0.0011	0.0091±0.0013
ZDT2	LPCM	0.0040±0.0000	0.0042±0.0001	0.0045±0.0001	0.0048±0.0001	0.0054±0.0002
	UGM	0.0044±0.0001	0.0045±0.0001	0.0047±0.0002	0.0051±0.0002	0.0056±0.0002
	MGM	0.0045±0.0006	0.0043±0.0002	0.0048±0.0005	0.0061±0.0010	0.0096±0.0021
ZDT3	LPCM	0.0051±0.0000	0.0053±0.0001	0.0056±0.0001	0.0060±0.0001	0.0069±0.0003
	UGM	0.0054±0.0003	0.0058±0.0002	0.0069±0.0006	0.0080±0.0008	0.0098±0.0012
	MGM	0.0056±0.0003	0.0055±0.0003	0.0058±0.0004	0.0064±0.0005	0.0081±0.0011

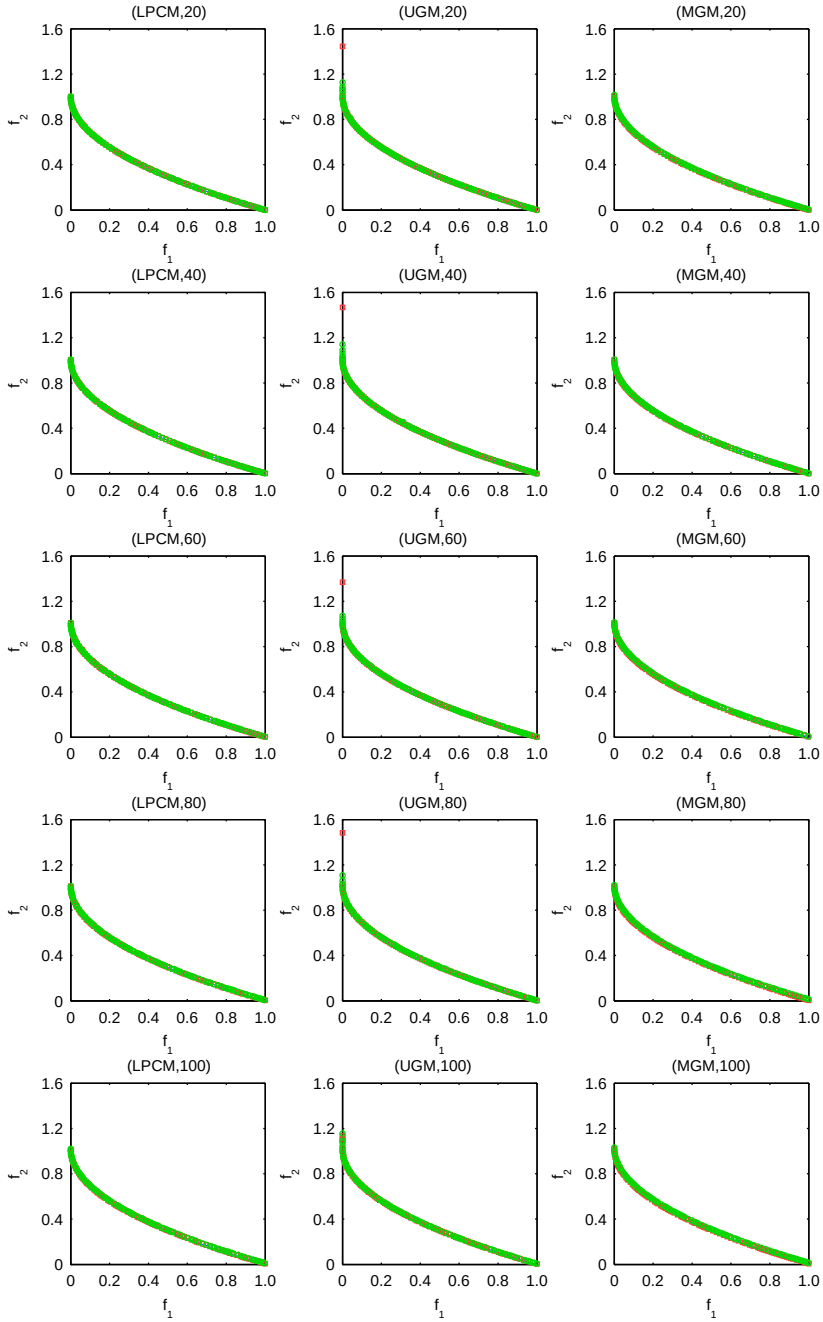

Fig. 6. Best and worst non-dominated set on ZDT1. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

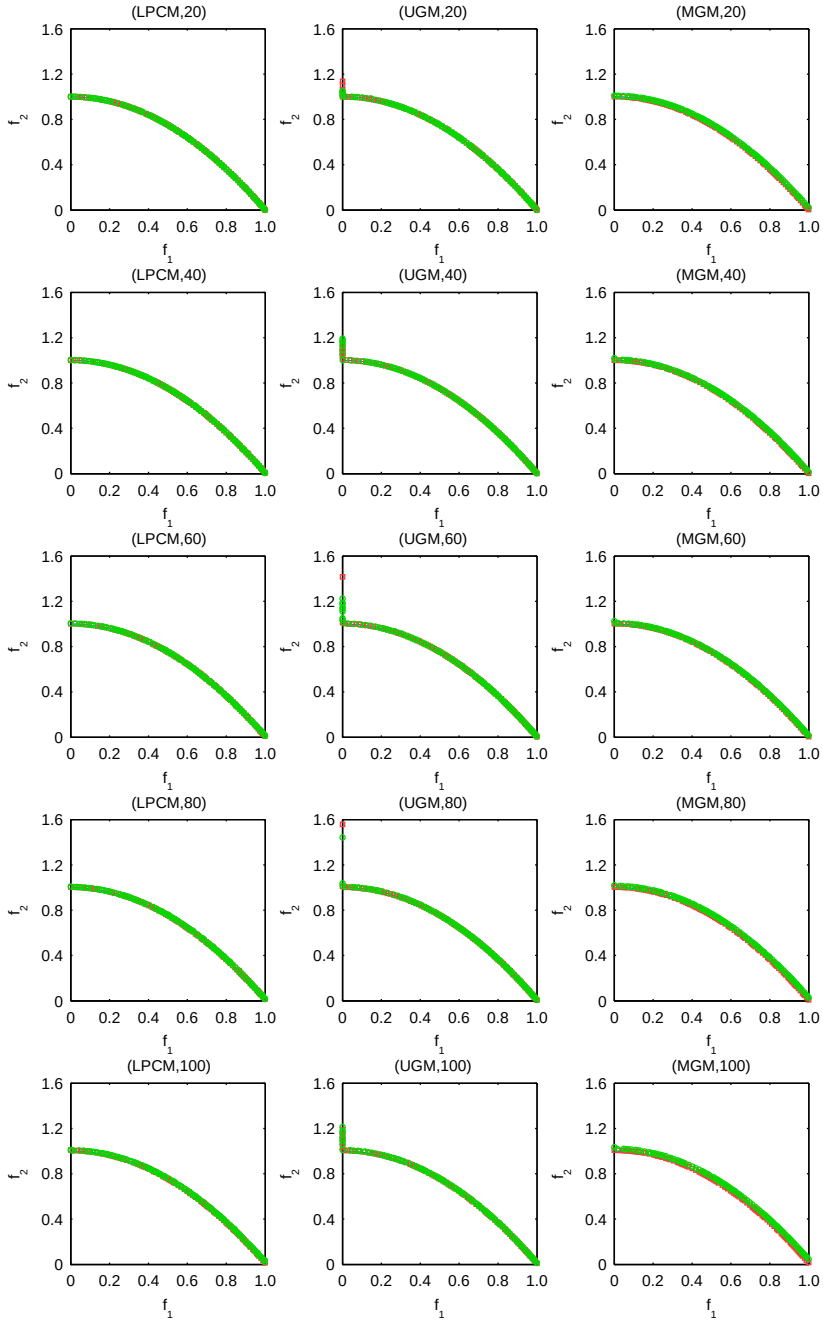

Fig. 7. Best and worst non-dominated set on ZDT2. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

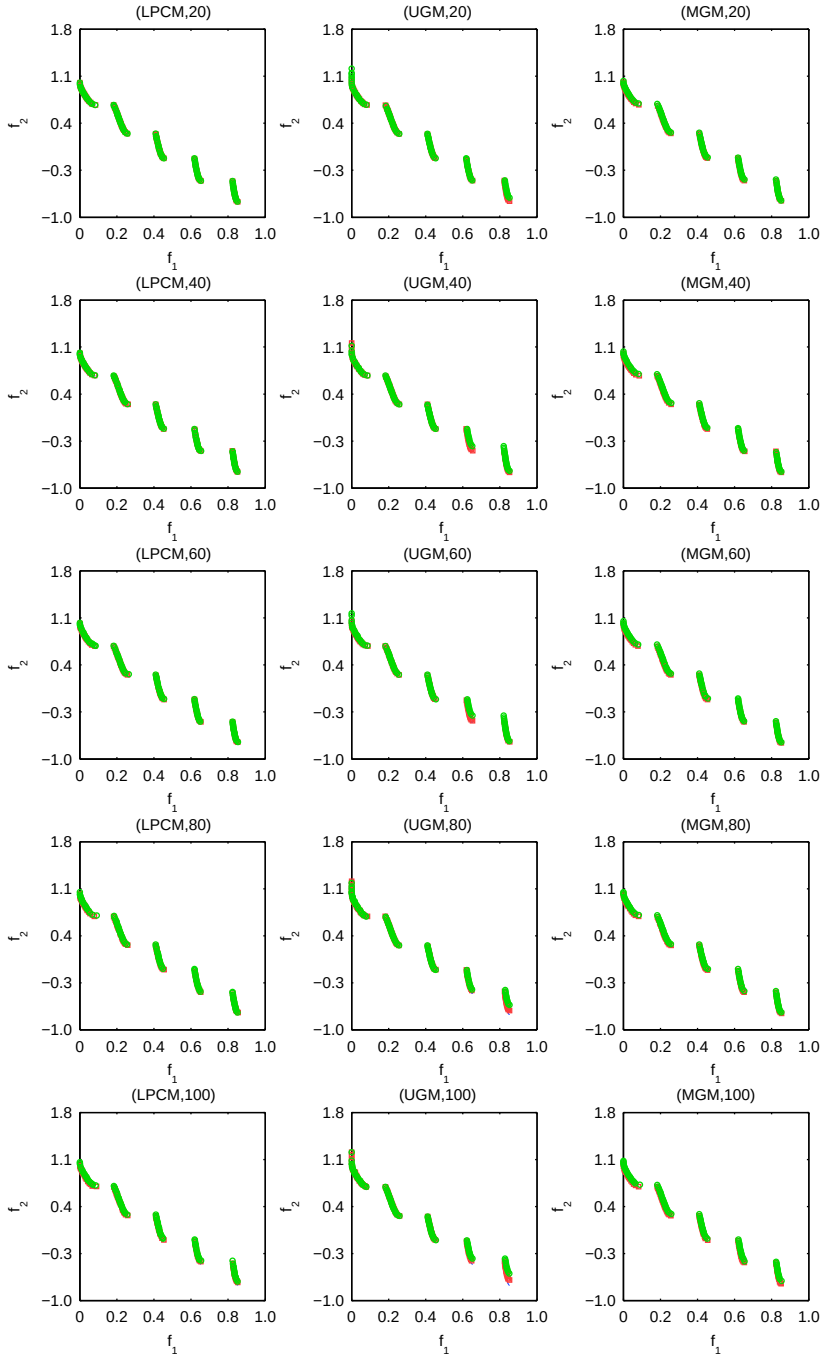

Fig. 8. Best and worst non-dominated set on ZDT3. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

Table 4. Mean and Std. of the I-metric for test functions without variable linkage.

Instance	Method	Search Dimension				
		20	40	60	80	100
ZDT1	LPCM	0.0048±0.0001	0.0058±0.0001	0.0069±0.0002	0.0083±0.0004	0.0098±0.0005
	UGM	0.0048±0.0002	0.0056±0.0003	0.0064±0.0003	0.0076±0.0004	0.0087±0.0004
	MGM	0.0061±0.0008	0.0061±0.0006	0.0081±0.0011	0.0120±0.0022	0.0163±0.0023
ZDT2	LPCM	0.0049±0.0001	0.0061±0.0002	0.0075±0.0003	0.0090±0.0004	0.0108±0.0006
	UGM	0.0050±0.0001	0.0059±0.0002	0.0071±0.0003	0.0083±0.0004	0.0099±0.0005
	MGM	0.0075±0.0021	0.0063±0.0010	0.0083±0.0014	0.0120±0.0025	0.0202±0.0046
ZDT3	LPCM	0.0043±0.0003	0.0080±0.0008	0.0129±0.0014	0.0185±0.0018	0.0265±0.0033
	UGM	0.0093±0.0061	0.0186±0.0040	0.0311±0.0058	0.0428±0.0079	0.0565±0.0096
	MGM	0.0151±0.0041	0.0107±0.0033	0.0147±0.0034	0.0210±0.0044	0.0328±0.0075

5.2 Scalability to Search Dimension: With Dependency

Simulations are also conducted on the three test functions with nonlinear linkage among the design variables. The best and worst Pareto fronts from 30 independent runs according to the D-metric for the three test functions, ZDT1.2, ZDT2.2, and ZDT3.2 are presented in Fig. 9, Fig. 10, and Fig. 11, respectively. From the figures (left panel), we find that there is a slight performance decrease of LPCM on ZDT1.2 and ZDT2.2, comparing its performance on the three ZDT functions without variable linkage. The performance decrease on ZDT3.2 seems more obvious, nevertheless, the best achieved Pareto front still approximates the true Pareto front very well. This indicates that the performance of LPCM scales well to the search dimension.

If we look at the results of UGM (middle panel), the performance becomes very poor. For all the three test functions, no single run is able to achieve a complete Pareto front, regardless of the search dimension. This strongly indicates that UGM is not suited for solving problems in which variable linkage exists.

We can see from the figures (right panel) that for test functions ZDT1.2 and ZDT2.2, MGM is able to achieve the entire Pareto front when the search dimension is low (20). However, the performance degrades seriously as the search dimension increases. This suggests that although MGM is able to capture the linkage between the design variables, the modeling accuracy decreases rapidly when the dimension becomes high.

The above observations made from the plot of the Pareto fronts can be confirmed the D-metric and the I-metric of the results form 30 independent runs, as listed in Table 5 and Table 6, respectively.

5.3 Sensitivity to Population Size

In solving hard real-world problems, the computational cost is often very high. One of the common approach is to parallelize the fitness evaluation using a computer cluster or even grid computing techniques [20]. Nevertheless, it is always desirable if the performance of an algorithm is not sensitive to the population size. To check the algorithms sensitivity to population size, we

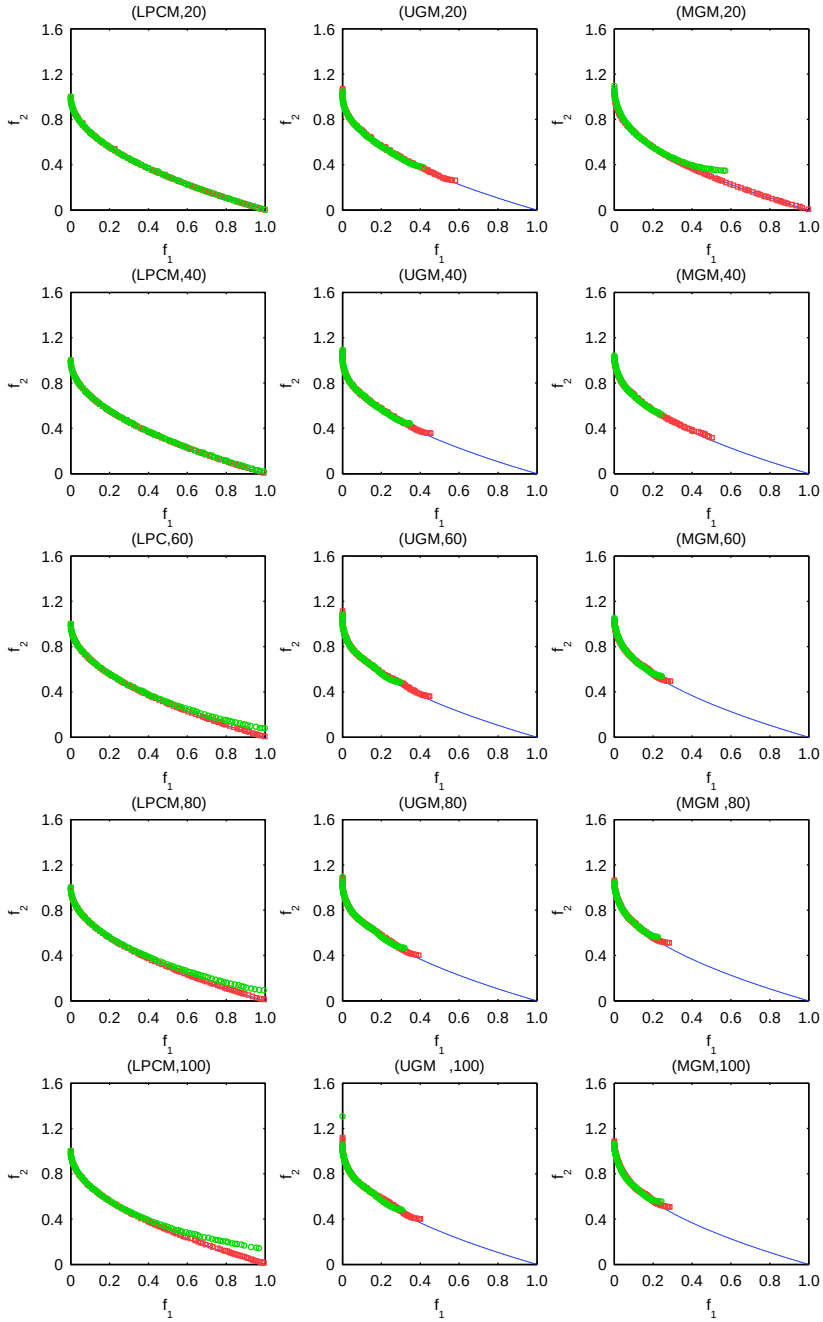

Fig. 9. Best and worst non-dominated set on ZDT1.2. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

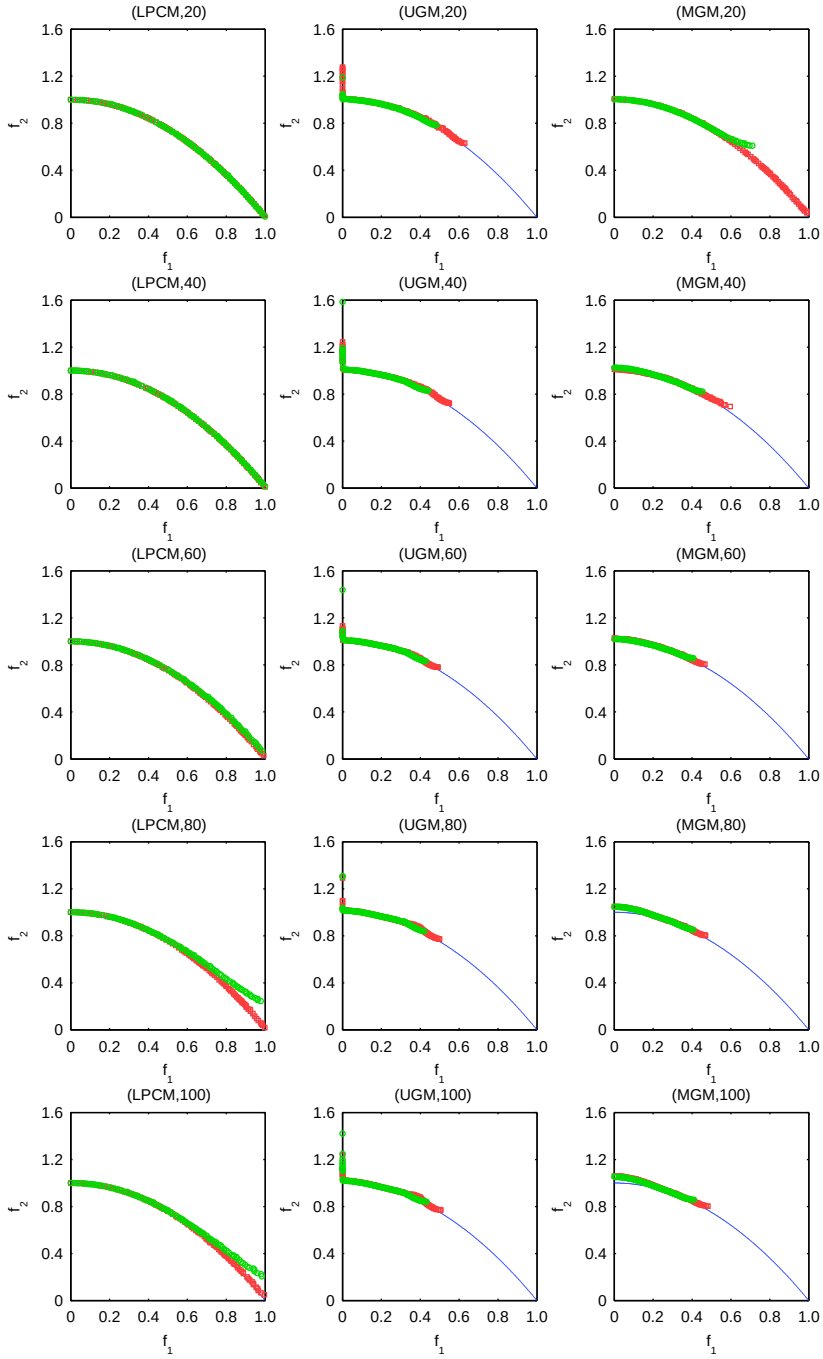

Fig. 10. Best and worst non-dominated set on ZDT2.2. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

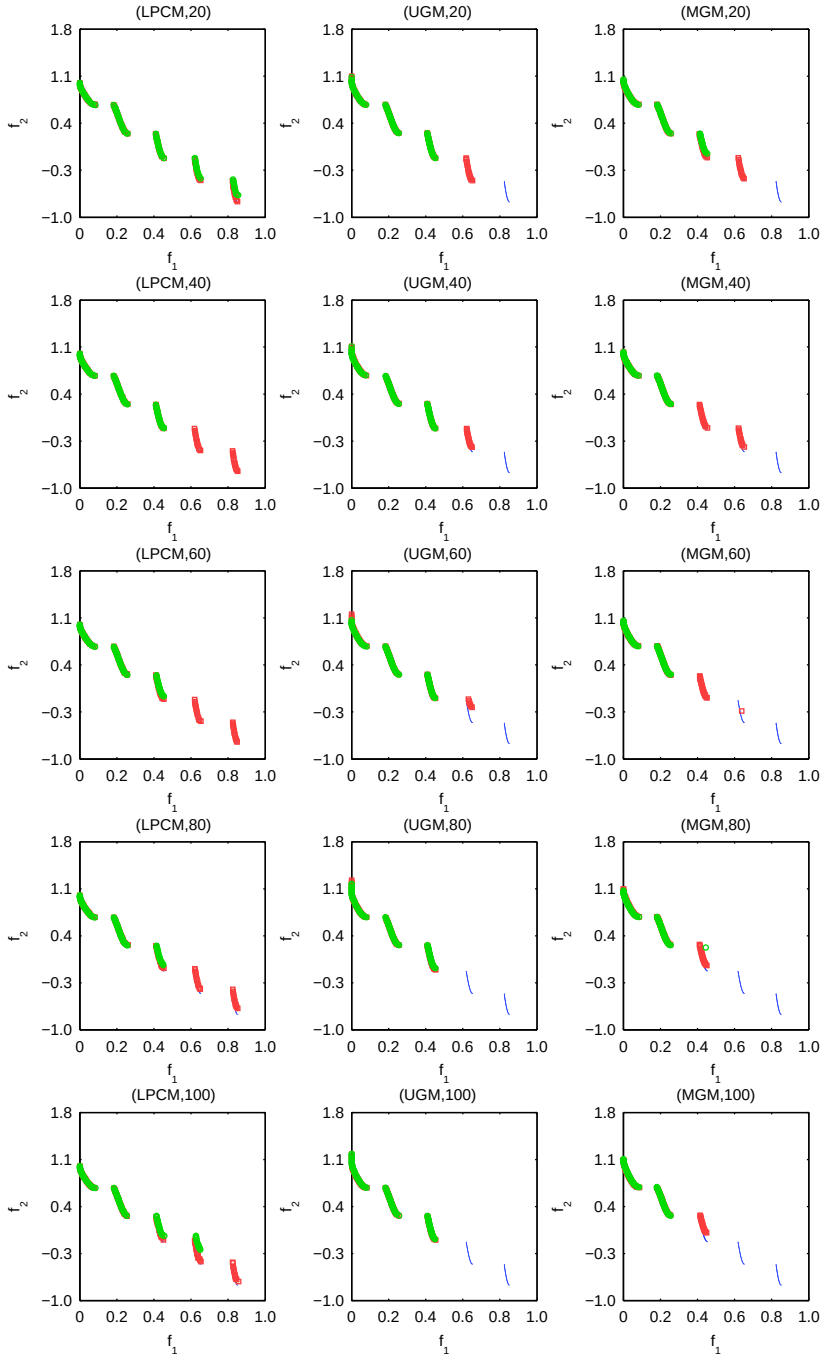

Fig. 11. Best and worst non-dominated set on ZDT3.2. Population size=100. The number of design variables ranges from 20 (top row) to 100 (bottom row).

Table 5. Mean and Std. of the D-metric for test functions with variable linkage.

Instance	Method	Search Dimension				
		20	40	60	80	100
ZDT1.2	LPCM	0.0045±0.0001	0.0049±0.0001	0.0058±0.0011	0.0091±0.0044	0.0120±0.0072
	UGM	0.1494±0.0266	0.2165±0.0213	0.2443±0.0224	0.2552±0.0134	0.2676±0.0151
	MGM	0.0138±0.0179	0.2032±0.0540	0.3526±0.0130	0.3624±0.0133	0.3637±0.0125
ZDT2.2	LPCM	0.0042±0.0001	0.0046±0.0001	0.0051±0.0002	0.0063±0.0033	0.0070±0.0020
	UGM	0.1845±0.0232	0.2222±0.0176	0.2408±0.0106	0.2512±0.0114	0.2508±0.0119
	MGM	0.0339±0.0269	0.1916±0.0236	0.2626±0.0125	0.2690±0.0100	0.2718±0.0119
ZDT3.2	LPCM	0.0051±0.0001	0.0109±0.0211	0.0077±0.0027	0.0103±0.0032	0.0126±0.0037
	UGM	0.0554±0.0296	0.0975±0.0284	0.1186±0.0090	0.1221±0.0014	0.1232±0.0008
	MGM	0.0868±0.0370	0.1059±0.0700	0.1428±0.0603	0.1690±0.0288	0.1993±0.0329

Table 6. Mean and Std. of the I-metric for test functions with variable linkage.

Instance	Method	Search Dimension				
		20	40	60	80	100
ZDT1.2	LPCM	0.0057±0.0001	0.0073±0.0005	0.0102±0.0029	0.0173±0.0090	0.0223±0.0134
	UGM	0.1494±0.0224	0.2072±0.0176	0.2323±0.0182	0.2434±0.0117	0.2551±0.0126
	MGM	0.0241±0.0291	0.2020±0.0422	0.3293±0.0117	0.3391±0.0121	0.3425±0.0114
ZDT2.2	LPCM	0.0066±0.0004	0.0091±0.0006	0.0115±0.0011	0.0160±0.0106	0.0195±0.0078
	UGM	0.2957±0.0249	0.3354±0.0173	0.3542±0.0101	0.3647±0.0107	0.3648±0.0114
	MGM	0.0869±0.0614	0.3053±0.0233	0.3751±0.0117	0.3822±0.0091	0.3854±0.0105
ZDT3.2	LPCM	0.0065±0.0029	0.0402±0.0562	0.0423±0.0221	0.0649±0.0212	0.0795±0.0239
	UGM	0.1778±0.0689	0.2832±0.0503	0.3188±0.0124	0.3282±0.0049	0.3319±0.0029
	MGM	0.2602±0.0834	0.3003±0.1485	0.3895±0.1072	0.4487±0.0613	0.5098±0.0552

compared the performance of LPCM, UGM and MGM using different population sizes, ranging from 20 to 1000, refer to Table 1. As we can see from Table 1, the allowed maximum number of fitness evaluations for large population sizes is larger than that allowed for small population sizes to improve the convergence. However, our simulation results still suggest that an overly large population size is not desirable when the number of fitness evaluations is limited.

The results in terms of the D-metric are presented in Figs. 12, 13, and 14 for ZDT1, ZDT2, and ZDT3, respectively.

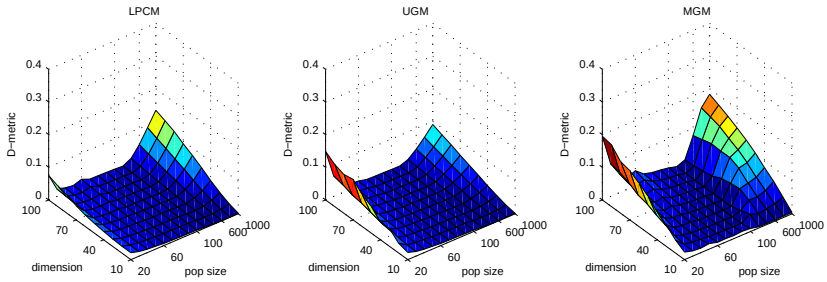

Fig. 12. Results on ZDT1 using different population sizes.

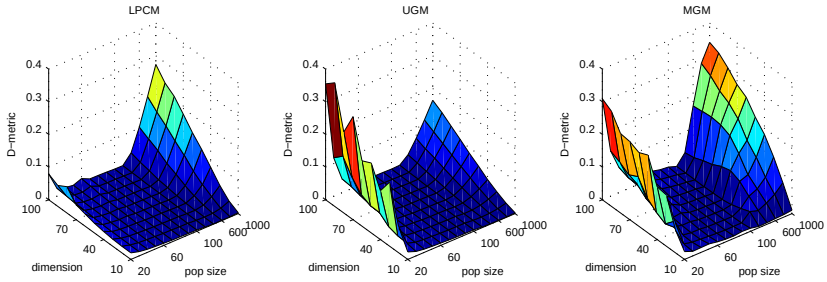

Fig. 13. Results on ZDT2 using different population sizes.

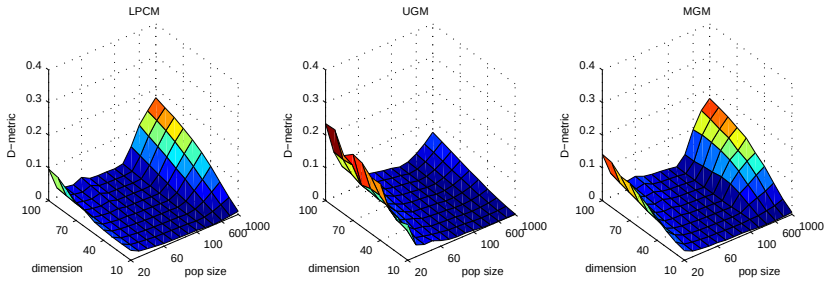

Fig. 14. Results on ZDT3 using different population sizes.

From the figures, we can observe that among the three algorithms, LPCM shows very robust performance for a wide range of population sizes on the ZDT functions without variable linkage, though a too large population size is not recommended for solving high-dimensional problems. UGM method, on the other hand, performs quite well with a medium to large population size. However, a population smaller than 60 turns out to be insufficient for the UGM. By contrast, the MGM is quite sensitive to the population size and a population size smaller than 60 or larger than 200 should not be used.

Simulation results on the test functions with variable linkage are plotted in Fig. 15, Fig. 16, and Fig. 17, respectively.

LPCM distinguishes itself with the other two algorithms more on the test functions with variable linkage. In solving problems with variable linkage, LPCM still performs very well with different population sizes for high-dimensional problems. On ZDT3.2, the performance is not very satisfactory for small population sizes. Contrary to that, UGM performs poorly in most cases, except for the case in which a very large population size is used for a low-dimensional problem. The bad performance of UGM can obviously be attributed to the fact that UGM is not able to efficiently solve problems with variable linkage. From the figures, we can see that MGM works well on low-dimensional problems, though it is quite clear that MGM is not suited for

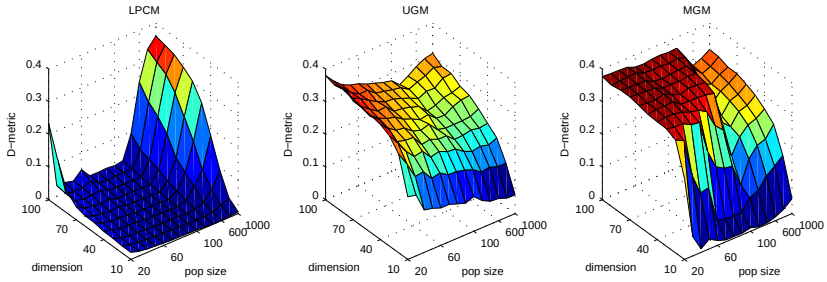

Fig. 15. Results on ZDT1.2 using different population sizes.

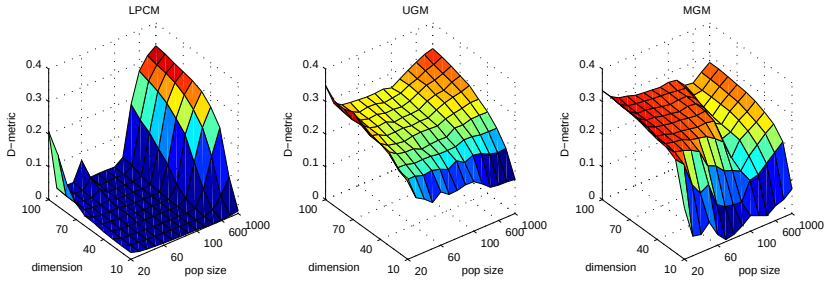

Fig. 16. Results on ZDT2.2 using different population sizes.

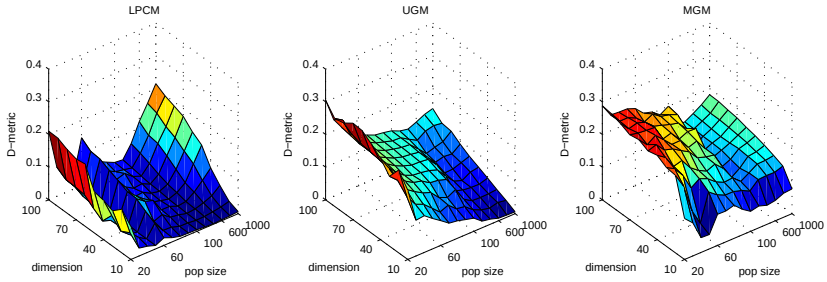

Fig. 17. Results on ZDT3.2 using different population sizes.

solving high-dimensional problems with variable linkage, despite that it is theoretically able to capture correlations between variables.

6 Conclusion

This chapter presents a model-based multi-objective optimization method that is able to explicitly take advantage of the regularity in the distribution of Pareto-optimal solutions. By using the regularity condition, the dimensionality of the latent space in which the model is constructed is greatly reduced. In

addition, a principal curve or surface model is used instead of a joint Gaussian distribution or a factorized Gaussian distribution model.

Simulation studies on comparing the scalability of the three multi-objective optimization algorithms, i.e., LPCM, UGM, and MGM, are conducted on six test problems with or without linkage among the design variables. From the simulation results, we demonstrate that LPCM exhibits excellent scalability to the increase in search dimension for problems with or without variable linkage. We also show that LPCM is in principle insensitive to population size ranging from 20 to 1000. We show that UGM is also scalable to the search dimension for problems without linkage among design variables. However, the performance of UGM deteriorates drastically when linkage exists among the design variables.

It is somehow surprising that MGM also shows quite good performance on high-dimensional test problems without variable linkage, though it is more sensitive to the population size than LPCM and UGM. While MGM shows better performance than UGM on low-dimensional problems with variable linkage, its performance is as poor as that of UGM for high-dimensional problems, regardless of the population size used.

From our comparative studies, we conclude that explicitly taking the regularity in the distribution of Pareto-optimal solutions into account is very helpful in improving the scalability of model-based multi-objective optimization algorithms.

Our future work is to compare the performance of the algorithms on more complex test problems where stronger correlations exist, such as those suggested in [23, 11]. In addition, the scalability of the algorithms to the number of objectives [**?**] is also an interesting issue to further investigate.

References

1. C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.
2. P. A. N. Bosman and D. Thierens. The naive MIDEA: A baseline multi-objective EA. In *Third International Conference on Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 428–442. Springer, 2005.
3. P.A.N Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Parallel Problem Solving from Nature*, pages 767–776, 2000.
4. P.A.N. Bosman and D. Thierens. Advancing continuous IDEAs with mixture of distributions and facorization selection metrics. In *Genetic and Evolutionary Computation Workshop on Optimization by Building and Using Probabilistic Models*, pages 208–212, 2001.
5. P.A.N. Bosman and D. Thierens. Learning probabilistic models for enhanced evolutionary computation. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 147–176. Springer, 2005.

6. P.A.N. Bosman and D. Thierens. Numerical optimization with real-valued estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling*. Springer, 2006.
7. D.-Y. Cho and B.-T. Zhang. Continuous estimation of distrubution algorithms with probabilistic component analysis. In *Congress on Evolutionary Computation*, pages 521–526. IEEE, 2001.
8. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, LTD, Chichester, 2001.
9. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
10. G.R. Harik and D.G. Goldberg. Learning linkage. In *Foundations of Genetic Algorithms*, pages 247–262, 1996.
11. S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In *Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 280–295. Springer, 2005.
12. A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
13. Y. Jin, editor. *Knowledge Incorporation in Evolutionary Computation*. Springer, Berlin, 2005.
14. Y. Jin and B. Sendhoff. Connectedness, regularity and the success of local search in evolutionary multi-objective optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*, pages 1910–1917, Canberra, Australia, 2003. IEEE.
15. N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.
16. J.D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, Computer Engineering and Networks Laboratory, ETH Zurich, Zurich, Switzerland, 2006.
17. S. Kukkonen and J. Lampinen. GDE3: The third evolution step of generalized differential evolution. In *Proceedings of the Congress on Evolutionary Computation (CEC 2005)*, pages 443–450, Edinburgh, September 2005. IEEE.
18. P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, 2001.
19. P. Larranaga, R. Etxeberria, J.A. Lozano, and J.M. Pena. Optimization in continuous domains by learning and simulation of Gaussian networks. In *Genetic and Evolutionary Computation Workshop on Optimization by Building and Using Probabilistic Models*, pages 201–204, 2000.
20. D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B.-S. Lee. Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems - The International Journal of Grid Computing: Theory, Methods, and Applications*, 2007. Accepted.
21. K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12 of *Kluwer's International Series in Operations Research & Management Science*. Kluwer Academic Publishers, 1999.
22. H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distribution I. Binary parameters. In *Parallel Problem Solving from Nature*, LNCS 1141, pages 178–187, 1996.

23. T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *Parallel Problem Solving from Nature*, LNCS 3242, pages 792–802. Springer, 2004.
24. T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *Congress on Evolutionary Computation*, pages 1594–1601, Portland, Oregon, 2004. IEEE.
25. M. Reyes Sierra and C. A. Coello Coello. A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In *Congress on Evolutionary Computation*, pages 65–72, Edinburgh, 2005. IEEE.
26. O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multilevel evolutionary subdivision techniques. In *Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 118–132, Faro, Portugal, 2003. Springer, LNCS 2632.
27. S.-Y. Shin, D.-Y. Cho, and B.-T. Zhang. Function optimization with latent variable models. In *The Third International Symposium on Adaptive Systems*, pages 145–152, 2001.
28. E.J. Solteiro Pires, P.B. de Moura Oliveira, and J.A. Tenreiro Machado. Multi-objective maxMin sorting scheme. In *The Third International Conference on Multi-Criterion Optimization*, LNCS 3410, pages 165–175. Springer, 2005.
29. D.A. van Veldhuizen and G. B. Lamont. Evolutionary computation and convergence to a Pareto front. In *Late Breaking Papers at the Genetic Programming Conference*, pages 221–228, Madison, Wisconsin, 1998. Stanford University Bookstore.
30. Q. Zhang. On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8(1):80–93, 2004.
31. Q. Zhang, A. Zhou, and Y. Jin. Modelling the regularity in estimation of distribution algorithm for continuous multi-objective evolutionary optimization with variable linkages. *IEEE Transactions on Evolutionary Computation*, 2007. Accepted.
32. A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *Congress on Evolutionary Computation*, pages 3234–3241, Vancouver, BC, July 2006. IEEE.
33. A. Zhou, Q. Zhang, Y. Jin, B. Sendhoff, and E. Tsang. Modelling the population distribution in multi-objective optimization by generative topographic mapping. In *Parallel Problem Solving From Nature - PPSN IX*, pages 443–452. Spinger, 2006.
34. A. Zhou, Q. Zhang, Y. Jin, E. Tsang, and T. Okabe. A model-based evolutionary algorithm for bi-objective optimization. In *Congress on Evolutionary Computation*, pages 2568–2575, Edinburgh, September 2005. IEEE.
35. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolution algorithms: empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.