Open access • Proceedings Article • DOI:10.1109/CEC.2005.1555016

A model-based evolutionary algorithm for bi-objective optimization

Aimin Zhou, Qingfu Zhang, Yaochu Jin, Edward Tsang ...+1 more authors

Institutions: University of Essex

Published on: 12 Dec 2005 - Congress on Evolutionary Computation

Topics: Estimation of distribution algorithm, Genetic algorithm, Evolutionary algorithm, Optimization problem and Crossover

A Model-Based Evolutionary Algorithm for Bi-objective Optimization

Aimin Zhou[1], Qingfu Zhang[1], Yaochu Jin[2] and Edward Tsang[1]
[1]Department of Computer Science, University of Essex, Colchester, Wivenhoe Park, CO4 3SQ, UK
[2]Honda Research Institute Europe, 63073, Offenbach/Main, Germany

Abstract- The Pareto optimal solutions to a multi-objective optimization problem often distribute very regularly in both the decision space and the objective space. Most existing evolutionary algorithms do not explicitly take advantage of such a regularity. This paper proposed a model-based evolutionary algorithm (M-MOEA) for bi-objective optimization problems. Inspired by the ideas from estimation of distribution algorithms, M-MOEA uses a probability model to capture the regularity of the distribution of the Pareto optimal solutions. The Local PCA and the least-squares method are employed for building the model. New solutions are sampled from the model thus built. At alternate generations, M-MOEA uses crossover and mutation to produce new solutions. The selection in M-MOEA is the same as in NSGA-II. Therefore, MOEA can be regarded as a combination of EDA and NSGA-II. The preliminary experimental results show that M-MOEA performs better than NSGA-II.

1 Introduction

Many engineering areas involve the following multi-objective optimization problem:

$$\min_{x\in\Omega} F(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} \quad (1)$$

where $x = (x_1, \ldots, x_n)^T \in R^n$ is the decision variable vector. $\Omega \subseteq R^n$ is the decision space, $f_i(x)(i = 1, 2, \ldots, m)$ are the objective functions to be minimized. Generally there is no solution which can minimize every f_i simultaneously. In multi-objective optimization, the componentwise order is used in ranking the solutions. Let $a = (a_1, \ldots, a_m)$, $b = (b_1, \ldots, b_m) \in R^m$ be two vectors, a is said to dominate b, denoted by $a \prec b$, if $a_i \leq b_i$ for all $i = 1, \ldots, n$, but $a \neq b$. A solution $x^\star \in \Omega$ is called (globally) Pareto optimal if there is no $x \in \Omega$ such that $F(x) \prec F(x^\star)$. The set of all the Pareto optimal solutions, denoted by $\Omega^\star$, is called the Pareto optimal set. The set of all the Pareto optimal objective vectors, $PF = \{y \in R^m | y = F(x), x \in \Omega^\star\}$, is called the Pareto front. Multi-objective optimization algorithms aim to find an approximation of the Pareto optimal set and/or the Pareto front.

Since the publication of Schaffer's seminal work [1], a number of multi-objective evolutionary algorithms(MOEA) have been developed [2, 3, 4, 5, 6]. In MOEAs, multiple individuals search for multiple solutions in a collaborative way and can produce a set of nearly Pareto optimal solutions in a single run. Most MOEAs focus on the Pareto front and try to find a set of solutions that are as close to the Pareto front as possible and as diverse as possible. The approximation of the Pareto optimal set has not been explicitly addressed in these algorithms. Their solutions are often poor in terms of closeness to the Pareto optimal set and uniformity in the decision space. As argued in [7, 8], the Pareto optimal solutions often distribute so regularly in the decision that they can be described by (piecewise) continuous surfaces (curves in the case of bi-objective optimization). In fact, it has been found that the Pareto optimal sets can be defined as linear or piecewise functions for most widely-used test problems of multi-objective optimization in the evolutionary computation community [8]. Most MOEAs ignore such a regularity.

Estimation of distribution algorithms (EDAs) are a new computing paradigm in evolutionary computation. There is no crossover or mutation in EDAs. Instead, they explicitly extract global statistical information from the selected solutions and build a posterior probability distribution model of promising solutions, based on the extracted information. New solutions are sampled from the model thus built and fully or in part replace the old population. Several EDAs have been developed for multi-objective optimization problems [9, 10, 11]. However, these EDAs do not take the regularity into consideration in building probability models. Note that probability modelling techniques under regularity have been widely investigated in the area of statistical learning, it is very suitable to take the advantage of the regularity in the design of EDAs for MOP. Compared with traditional evolutionary algorithms, EDAs mainly rely on global statistical information collected from the previous search for guiding their further search. The information about the locations of the solutions found so far is not directly used in the search. Recently, combinations of traditional GAs and EDAs have been proposed for solving single objective optimization problems [12, 13, 14].

As one of the first attempts to capture and utilize the regularity of the distribution of Pareto solutions in the decision space, Voronoi-based Estimation of Distribution Algorithm (VEDA) for MOP has been proposed in [15] very

recently. In VEDA, the distribution of promising solutions in the decision space is learned by clustering and principal component analysis (PCA) algorithms. Then Voronoi meshes are constructed to model the distribution under the assumption of the regularity. Their preliminary experimental results are very encouraging. However, combination of clustering and PCA are not very suitable to learn nonlinear continuous curves or surfaces and in addition, building Voronio meshes can be very time-consuming, particularly, for the problems with a large number of decision variables.

Inspired by the idea behind VEDA for MOP and successful combinations of GAs and EDAs for single optimization problems [12, 13, 14], this paper proposes a model-based evolutionary algorithm (M-MOEA) for bi-objective optimization problems. M-MOEA has the following features:

- A probability model is built to model the promising area in the decision space. New solutions are sampled from the model thus built. Taking into consideration the regularity in the distribution, the model assumes that the Pareto optimal solutions for a bi-objective optimization problem is a piecewise continuous curve.
- At alternate generations, crossover and mutation are employed to generate new solutions. In this way, the location information of the non-dominated solutions found so far is used to guide the further search, which compensates for the ignorance of the location information in the EDA method.
- The selection method is the same as in NSGA-II[6].

The rest of the paper is organized as follows. M-MOEA is described in detail in the next section. The experimental results are shown in Section 3 to compare the performance of M-MOEA and NSGA-II. The final section outlines the conclusions and further research topics.

2 Algorithm

2.1 The Framework

The proposed M-MOEA is for solving the bi-objective optimization problem (Problem (1) with $m = 2$). It maintains a population of candidate solutions $P(t)$ at generation t. GA offspring generators (i.e., crossover and mutation) and an EDA method are used for generating new solutions at alternate generations. The next generation are selected from the new solutions and the current population. The structure of M-MOEA is shown in Figure 1.

The details of the main ingredients of M-MOEA are explained in the following.

Algorithm(M-MOEA)

Step 0 Initialization: Set $t = 0$ and initialize $P(t)$.

Step 1 Reproduction:

If $t\%2 == 0$, perform crossover and mutation on $P(t)$ to generate a set of new solutions, $P_s(t)$.

Else use the EDA method to generate $P_s(t)$.

Step 2 Selection: Select $P(t+1)$ from $P_s(t) \cup P(t)$.

Step 3 Stopping Condition: If the stopping condition is met, stop; otherwise, set $t = t + 1$ and go to **Step 1**.

Figure 1: The framework of M-MOEA

2.2 Initialization

N solutions are sampled from the decision space randomly and uniformly to constitute $P(0)$.

2.3 Crossover and Mutation

Recent studies show that EDAs alone cannot solve hard problems very well since EDAs do not utilize the location information of the best solutions found so far, the solutions generated by EDAs can be far away from the best solutions found so far, particulary in the early stage of the search. The combination of EDAs and traditional GAs has proved an efficient way for solving hard problems [12, 13, 14].

In Step 2 of M-EOEA, if $t\%1 == 0$, we perform crossover and mutation on $P(t)$ to create N new solutions to form $P_s(t)$. The crossover and mutation operators used are the same as in NSSA-II [6].

2.4 EDA Method for Generating Offspring

The proposed EDA method for generating offspring estimates the shape of Pareto optimal set from the current population and then uses this estimation to guide the further search. It first partitions $P(t)$ into several subsets. Then a probability model is built for each subset for estimating the distribution of its solutions. These models are repeatedly sampled to generate new solutions.

2.4.1 Probability Model

If f_1 and f_2 are continuous, Pareto optimal set of a bi-objective optimization problem is very likely to be a continuous curve or several continuous curves. After a few generations, the population in an EA for such a problem should be distributed around these curves in the decision space. For this reason, M-MOEA assumes that the population $P(t)$ can

be partitioned into several clusters. The points in each cluster distribute around a bounded continuous curve. More precisely, the points in a cluster can be regarded as independent observations of the following n-D random vector:

$$\xi = \xi_1 + \xi_2 \tag{2}$$

where ξ_1 is uniformly distributed along a continuous curve and ξ_2 is a random noise vector. The underlying curve for ξ_1 is called the centroid curve of ξ in this paper. For the sake of simplicity, we assume that ξ_1 and ξ_2 are independent of each other, and ξ_2 obeys a normal distribution.

2.4.2 Partition

The Local PCA algorithm [16] is used in the proposed EDA method for partitioning $P(t)$ into K disjoint clusters $S_1, \ldots, S_K$ (where K is a user-specified algorithmic parameter in the local PCA algorithm).

Suppose S_k contain N_k points $x^{k,1}, \ldots, x^{k,N_k}$, the mean of S_k is:

$$\bar{x}^k = \frac{1}{N_k} \sum_{i=1}^{N_k} x^{k,i},$$

and the covariance matrix of the points in S_k is

$$V_k = \frac{1}{|N_k| - 1} \sum_{i=1}^{N_k} (x^{k,i} - \bar{x}^k)(x^{k,i} - \bar{x}^k)^T.$$

The i-th principal component $v_{k,i}$ of S_k is a unity eigenvector associated with the i-th largest eigenvalue $\lambda_{k,i}$ of V_k. Let L_k be the line passing through the point $\bar{x}^k$ in the direction of $v_{k,1}$, the partition of $P(t)$ by the Local PCA algorithm minimizes the squared reconstruction distance:

$$\sum_{k=1}^{K} \sum_{i=1}^{N_k} [d(x^{k,i}, L_k)]^2,$$

where $d(x^{k,i}, L_k)$ is the shortest Euclidean distance from $x^{k,i}$ to the curve L_k.

Compared with the widely-used K-means clustering, the Local PCA algorithm is advantageous for dealing with the data whose distribution can be approximately modelled by (2). The details of the local PCA algorithm can be found in [16].

2.4.3 Modelling

As discussed in 2.4.1, the underlying random vector for each cluster can be modelled by (2).

To model the distribution of the underlying random vector ξ_k for S_k, the scalar projection of $x^{k,i} - \bar{x}^k$ along the first principal component $v_{k,1}$ is computed:

$$\theta_{k,i} = (x^{k,i} - \bar{x}^k)^T v_{k,1}.$$

Therefore, $\bar{x}^k + \theta_{k,i} v_{k,1}$ is the projection point of $x^{k,i}$ on the line L_k.

Let A_k and B_k denote the minimum and maximum of $\theta_{k,i}, i = 1, 2, \ldots, N_k$ respectively. Set

$$\rho_k = \frac{\lambda_{k,2}}{B_k - A_k}.$$

If ρ_k is small, then the points in S_k are close to its centroid curve. Therefore, it is very likely to obtain a good approximation of its centroid curve from the points in S_k. In this case, we model the centroid curve as a cubic curve. On the other hand, if ρ_k is not small, then it is impractical to use a fine model for approximating its centroid curve. We use a linear model for large ρ_k. In the following, the details of modelling of the distribution of ξ_k is given.

Case A when $\rho_k \geq \rho$, where ρ is an algorithmic parameter, the underlying random vector ξ_k is modelled as

$$\xi_k = \xi_{k,1} + \xi_{k,2},$$

where

- $\xi_{k,1} = \bar{x}^k + \eta v_{k,1}$, η is a uniform random variable in $[A_k - 0.1(B_k - A_k), B_k + 0.1(B_k - A_k)]$,
- $\xi_{k,2} = \Sigma_{i=2}^{n} \varepsilon_i v_{k,i}$, each ε_i obeys the normal distribution $N(0, \lambda_{k,i})$, $i = 2, \ldots, n$,
- the random variables $\eta, \varepsilon_2, \ldots, \varepsilon_n$ are independent of each other.

Case B when $\rho_k < \rho$, the underlying random variable ξ_k is modelled as

$$\xi_k = \xi_{k,1} + \xi_{k,2},$$

where

- $\xi_{k,1} = g(\eta)$, η is a uniform random variable in $[A_k - 0.1(B_k - A_k), B_k + 0.1(B_k - A_k)]$. $x = (g_1(s), \ldots, g_n(s))$ is a curve parameterized by s, its component $x_j = g_j(s)$ is the least-squares cubic which fits:

$$x_j^{k,i} \approx g(\theta_{k,i}).$$

- $\xi_{k,2} = \Sigma_{i=2}^{n} \varepsilon_i v_{k,i}$, each ε_i obeys the normal distribution $N(0, \lambda_{k,i})$, $i = 2, \ldots, n$,
- the random variables $\eta, \varepsilon_2, \ldots, \varepsilon_n$ are independent of each other.

2.4.4 Sampling

After establishing the probability models for all the clusters $S_1, \ldots, S_K$, we sample from these models for generating new solutions to form $P_s(t)$. For each cluster S_k, we sample $[\frac{N_k}{3}]$ points from its model.

Note that the range of the random variable $\xi_{k,1}$ is larger than that of the scalar projections $\theta_{k,i}, i = 1, 2, \ldots, N_k$ in the models. Therefore, the algorithm is able to explore new areas in the decision space. This exploration is guided by the models. On the other hand, the new solutions sampled are distributed around the centroid curves (or lines). In such a way, the algorithm intensifies its search in the promising areas.

2.5 Selection

The selection operator is the same as in NSGA-II. The details of the selection operator can be found in [6].

3 Experimental Results

We compare the performances of M-MOEA and NSGA-II experimentally on a set of test problems.

3.1 Test Problems

The following test problems are used in our experimental studies.

Test Problem 1 (ZDT1.1):

$$\begin{array}{l} f_1 = x_1 \\ f_2 = g \times (1.0 - \sqrt{\frac{f_1}{g}}) \\ g(x_2, \ldots, x_n) = 1.0 + \frac{9}{n-1}\Sigma_{i=2}^{n}(x_i - x_1)^2 \\ 0 \leq x_i \leq 1, i = 1, \ldots, n \end{array} \quad (3)$$

This problem is a modified version of ZDT1 [17]. The Pareto optimal set of ZDT1 is

$$\Omega^{\star}_{ZDT1} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = 0\},$$

which all the decision variables but x_1 are constant. Therefore, ZDT1 is not hard for any model-based EAs. This is the reason why we do not use ZDT1 in our experiments. The Pareto optimal set of ZDT1.1 is

$$\Omega^{\star}_{ZDT1.1} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = x_1\},$$

which is a line segment in R^n.

Test Problem 2 (ZDT1.2):

$$\begin{array}{l} f_1 = x_1 \\ f_2 = g \times (1.0 - \sqrt{\frac{f_1}{g}}) \\ g(x_2, \ldots, x_n) = 1.0 + \frac{9}{n-1}\Sigma_{i=2}^{n}(x_i^2 - x_1)^2 \\ 0.0 \leq x_i \leq 1.0, i = 1, \ldots, n \end{array} \quad (4)$$

This problem is another modified version of ZDT1. Its Pareto optimal set is

$$\Omega^{\star}_{ZDT1.2} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = \sqrt{x_1}\},$$

which is a bounded nonlinear curve.

Test Problem 3 (ZDT2.1):

$$\begin{array}{l} f_1 = x_1 \\ f_2 = g \times (1.0 - (\frac{f_1}{g})^2) \\ g(x_2, \ldots, x_n) = 1.0 + \frac{9}{n-1}\Sigma_{i=2}^{n}(x_i - x_1)^2 \\ 0.0 \leq x_i \leq 1.0, i = 1, \ldots, n. \end{array} \quad (5)$$

This problem is a modified version of ZDT2 [17]. The Pareto optimal set of original ZDT2 is

$$\Omega^{\star}_{ZDT2.1} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = 0\},$$

which is a line parallel to the x_1- axis. In contract, the Pareto optimal set of ZDT2.1 is

$$\Omega^{\star}_{ZDT2.1} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = x_1\},$$

which is a line with nonzero slope at any axis.

Test Problem 4 (ZDT2.2):

$$\begin{array}{l} f_1 = \sqrt{x_1} \\ f_2 = g \times (1.0 - \frac{f_1}{g}) \\ g(x_2, \ldots, x_n) = 1.0 + \frac{9}{n-1}\Sigma_{i=2}^{n}(x_i^2 - x_1)^2 \\ 0.0 \leq x_i \leq 1.0, i = 1, \ldots, n \end{array} \quad (6)$$

Its Pareto optimal set is

$$\Omega^{\star}_{ZDT2.2} = \{0 \leq x_1 \leq 1, x_2 = \ldots = x_n = \sqrt{x_1}\},$$

which is a bounded nonlinear curve.

Test Problem 5 (FON [17]):

$$\begin{array}{l} f_1 = 1 - exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2) \\ f_2 = 1 - exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2) \\ -4.0 \leq x_i \leq 4.0, i = 1, \ldots, n \end{array} \quad (7)$$

Its Pareto optimal set is

$$\Omega^{\star}_{FON} = \{-\frac{1.0}{\sqrt{n}} \leq x_1 \leq \frac{1.0}{\sqrt{n}}, x_2 = \ldots = x_n = x_1\},$$

which is a line segment.

3.2 Performance Measures

Two performance indices are used to compare the performance of different performances in our experimental studies.

Let S_1 and S_2 be two finite subsets of the solution space, the first index *Coverage of Two Sets* [18] is defined as:

$$C(S_1, S_2) = \frac{|\{x | x \in S_1, \exists y \in S_2 : F(x) \prec F(y)\}|}{|S_1|}, \quad (8)$$

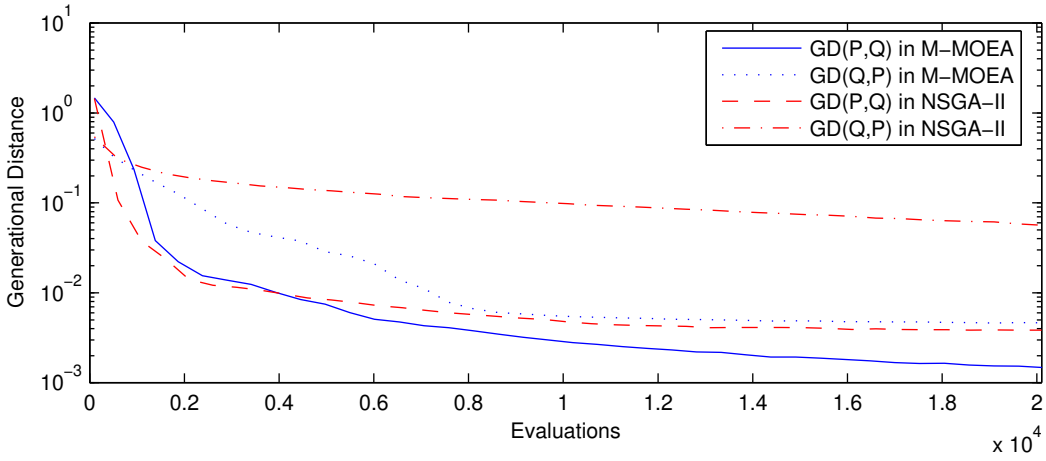

Figure 2: The evolution of the average $GD(P,Q)$ and $GD(Q,P)$ of 20 runs with the number of F-function evaluations in $ZDT2.2$

where $C(S_1, S_2)$ represents the percentage of the solutions in S_1 which are dominated by at least one solution from S_2. $C(S_1, S_2)$ is not necessarily equal to $1 - C(S_2, S_1)$. If $C(S_1, S_2)$ is small and $C(S_2, S_1)$ is large, then S_1 are better than S_2 in a sense.

The second index is *Generational Distance(GD)* [19], which is defined as follows[1]

$$GD(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} d(x, S_2), \tag{9}$$

where

$$d(x, S_2) = \min_{y \in S_2} \|F(x) - F(y)\|_2.$$

$GD(S_1, S_2)$ measures the distance from S_1 to S_2. Generally, $GD(S_1, S_2) \neq GD(S_2, S_1)$. If $Q = \{\tilde{x}_1, \ldots, \tilde{x}_J\} \subset \Omega^\star$ and its corresponding $F-$vectors $F(\tilde{x}_1), \ldots, F(\tilde{x}_J)$ are uniformly distributed along the Pareto front PF, then $GD(S_1, Q)$ can measure the closeness of S_1 to $\Omega^\star$ while $GD(Q, S_1)$ measures the spread of S_1 to a certain degree.

3.3 Experimental Setup

The number of decision variables in all the test problem is 10. The setting of the algorithmic parameter values in M-MOEA and NSGA-II is given in Table 1.

The stopping condition for both algorithms: The algorithms stop after a given number of evaluations of the objective function $F(x)$.

[1]The definition is little different from the original one:$GD(S_1, S_2) = \frac{1}{|S_1|}\sqrt{\sum_{x \in S_1} d(x, S_2)^2}$.

Table 1: Parameter Setting in M-MOEA and NSGA-II

Parameters	M-MOEA	NSGA-II
Population Size (N)	100	100
The Number of Clusters (K) in Local PCA	3	
ρ	0.01	
Crossover Rate	1.0	1.0
Mutation Rate	0.1	0.1

3.4 Results

In order to compare the average behaviors of M-MOEA and NSGA-II, 20 independent comparisons have been performed between these two algorithms for each test problem. In each comparison, two algorithms start with the same initial population that are randomly generated, $GD(P,Q)$ and $GD(Q,P)$ at each generation are computed and recorded for each algorithm, where Q is set as in Section 3.2 and its size is 1000, P is the current population.

Figure 2 shows the evolution of the average $GD(P,Q)$ and $GD(Q,P)$ of 20 runs with the number of F-function evaluations in both M-MOEA and NSGA-II for ZDT2.2. Clearly, $GD(P,Q)$ and $GD(Q,P)$ are smaller in M-MOEA than in NSGA-II after $1,000$ F-function evaluations. Therefore, we can conclude that M-MOEA performs better than NSGA-II in terms of the quality of the solutions in the long term, although NSGA-II outperforms M-MOEA in the early generations.

Let P_M and P_N be the final solution sets resulted from M-MOEA and NSGA-II, respectively, the average value, the maximum and the minimum of $C(\cdot)$'s and $GD(\cdot)$'s in 20 runs in both M-MOEA and NSGA-II for all the test problems are presented in Table 2. The algorithms stop af-

Table 2: Comparison between M-MOEA and NSGA-II

		Performance Index					
		$C(P_N, P_M)$	$C(P_M, P_N)$	$GD(P_M, Q)$	$GD(Q, P_M)$	$GD(P_N, Q)$	$GD(Q, P_N)$
$ZDT1.1$	Mean	0.027000	0.403550	0.002015	0.004929	0.004919	0.014848
	Min	0.000000	0.190000	0.001303	0.004484	0.004013	0.007026
	Max	0.090000	0.610000	0.002549	0.005393	0.005551	0.042239
$ZDT1.2$	Mean	0.051900	0.273650	0.002610	0.005318	0.004445	0.055772
	Min	0.000000	0.100000	0.002127	0.004864	0.003177	0.008089
	Max	0.180000	0.520000	0.003757	0.006260	0.006028	0.159617
$ZDT2.1$	Mean	0.039100	0.470800	0.002646	0.005825	0.006388	0.015197
	Min	0.000000	0.260000	0.001545	0.004765	0.005472	0.008084
	Max	0.150000	0.710000	0.003440	0.007755	0.007907	0.037905
$ZDT2.2$	Mean	0.057300	0.301075	0.002801	0.005436	0.004749	0.097906
	Min	0.000000	0.030000	0.002106	0.004907	0.002248	0.013462
	Max	0.180000	0.530000	0.003924	0.006216	0.006338	0.365518
FON	Mean	0.057600	0.593250	0.004104	0.007574	0.008304	0.013236
	Min	0.000000	0.150000	0.002364	0.005084	0.007123	0.009542
	Max	0.310000	0.880000	0.008041	0.013749	0.010078	0.016432

ter $14,000$ $F-$function evaluations for FON and $10,000$ for the other functions. We can conclude from these results that the quality of the solutions produced in M-MOEA are higher than in NSGA-II for the test problems.

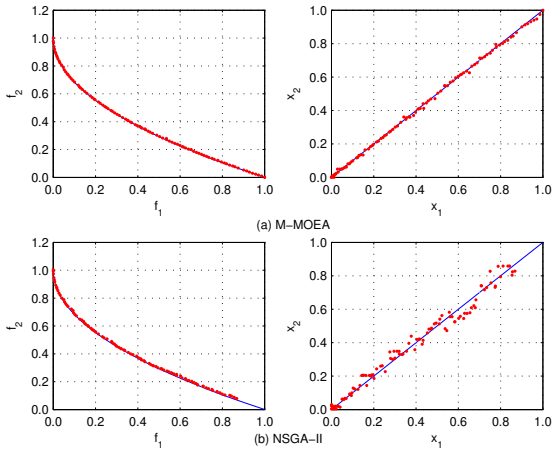

Figure 3: The results on $ZDT1.1$

Figure 3-7 show the distributions of the final solutions found by each algorithm after 10,000 $F-$function evaluations in a single run in both $x_1 - x_2$ space (there are 10 decision variables in the test problems) and the objective space. As we can see from these figures, M-MOEA performs better than NSGA-II, particularly in the aspect of spread of solutions in both decision space and objective space. These results imply that the proposed EDA mechanism can improve the algorithm.

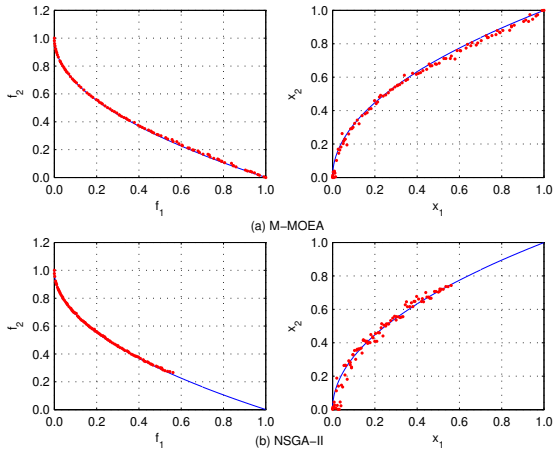

Figure 4: The results on $ZDT1.2$

4 Conclusions

M-MOEA, a combination of EDA-like methods and NSGA-II for bi-objective optimization problems, has been proposed in this paper. M-MOEA exploits the regularity in the distribution of Pareto optimal solutions of a MOP and builds a probability model to estimate the promising area in the search space. The Local PCA algorithm and least-squares fitting are employed in modelling. New solutions are sampled from the model thus built. In order to take the advantage of the location information collected in the previous search, M-MOEA also uses crossover and mutation to generate new solutions at alternate generations.

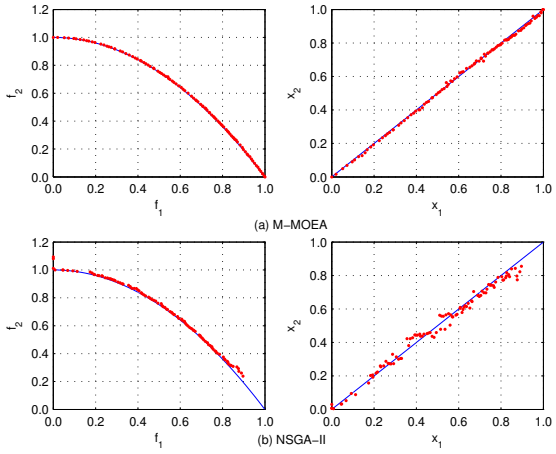

Figure 5: The results on $ZDT2.1$

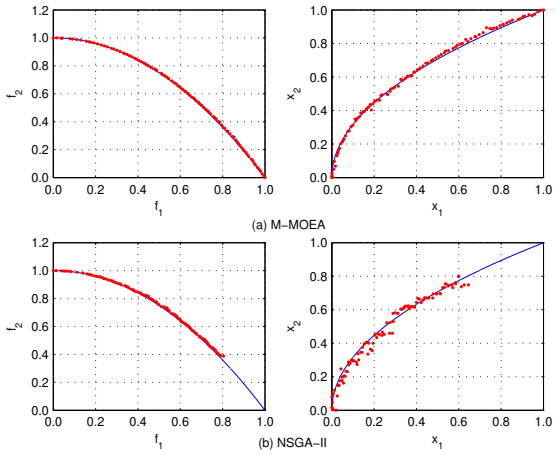

Figure 6: The results on $ZDT2.2$

We compared the proposed M-MOEA and NSGA-II on five test problems. Our preliminary experimental results showed that M-MOEA performed better than NSGA-II.

The future work may investigate other simple and efficient modelling methods to exploit the regularity of the Pareto optimal solutions in both the decision space and the objective space. The incorporation of traditional optimization methods with M-MOEA is also a further avenue for investigation. How to generalize M-MOEA to MOP with more than two objectives and MOPs with a disconnected Pareto front is also a challenging research topic.

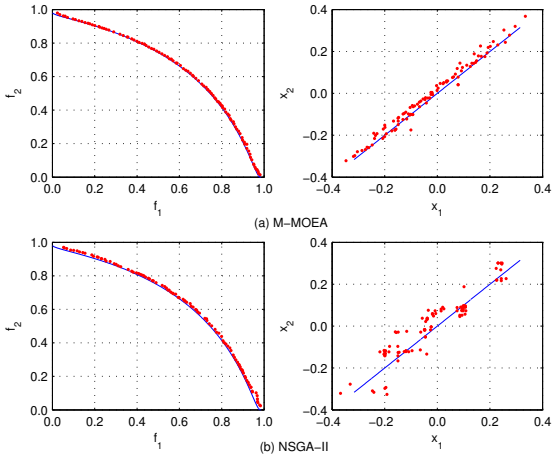

Figure 7: The results on FON

Acknowledgments

The authors would like to thank Bernhard Sendhoff and Tatsuya Okabe for their insightful discussions and kind support.

Bibliography

[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Pittsburgh, PA, USA: Lawrence Erlbaum Associates, July 1985, pp. 93–100.

[2] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Genetic Algorithms: Proceedings of the Fifth International Conference*. Urbana-Champaign, IL, USA: Morgan Kaufmann, June 1993, pp. 416–423.

[3] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *International Conference on Evolutionary Computation*, 1994, pp. 82–87.

[4] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1. Washington D.C., USA: IEEE, 1999, pp. 98–105.

[5] 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*. Barcelona, Spain: CIMNE, 2002, pp. 95–100.

[6] K. Deb, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] Y. Jin and B. Sendhoff, "Connectedness, regularity and the success of local search in evolutionary multiobjective optimization," in *Proceedings of the 2003 Congress on Evolutionary Computation*. Canberra, Australia: IEEE, December 2003, pp. 1910–1917.

[8] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff, "On test functions for evolutionary multi-objective optimization," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. Lecture Notes in Computer Science, vol. 3242. Birmingham, UK: Springer, September 2004, pp. 792–802.

[9] M. Laumanns and J. Ocenasek, "Bayesian optimization algorithms for multi-objective optimization," in *Parallel Problem Solving From Nature - PPSN VII*, ser. Lecture Notes in Computer Science, vol. 2439. Granada,Spain: Springer, September 2002, pp. 298–307.

[10] D. Thierens and P. A. N. Bosman, "Multi-objective mixture-based iterated density estimation evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco,California: Morgan Kaufmann, 2001, pp. 663–670.

[11] N. Khan, D. E. Goldberg, and M. Pelikan, "Multi-objective bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: Morgan Kaufmann, July 2002, p. 684.

[12] J. M. P. Sánchez, V. Robles, P. Larrañaga, V. Herves, F. Rosales, and M. S. Pérez, "GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms," in *The 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, 2004, pp. 361–371.

[13] J. Sun, Q. Zhang, and E. P. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3, pp. 249–262, 2005.

[14] Q. Zhang, J. Sun, and E. Tsang, "Evolutionary algorithm with the guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, 2005.

[15] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer, "Voronoi-based estimation of distribution algorithm for multi-objective optimization," in *Proceedings of the Congress on Evolutionary Computation*. Portland, Oregon, USA: IEEE, June 2004, pp. 1594–1601.

[16] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1493–1516, October 1997.

[17] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Baffins Lane,Chichester: John Wiley & Sons,LTD, 2001.

[18] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study," in *Parallel Problem Solving from Nature - PPSN V*, ser. Lecture Notes in Computer Science, vol. 1498. Amsterdam, The Netherlands: Springer, September 1998, pp. 292–304.

[19] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a pareto front," in *Late Breaking Papers at the Genetic Programming 1998 Conference*, July, pp. 221–228.