International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.7 (2016), pp.391-406
http://dx.doi.org/10.14257/ijsip.2016.9.7.35

ML-MOEA/SOM: A Manifold-Learning-Based Multiobjective Evolutionary Algorithm Via Self-Organizing Maps

Wei Cao[1], Wei Zhan[2*] and ZhiQiang Chen[3]

[1]*School of computer and Information Engineering, Xiamen University of Technology, Xiamen, Fujian, China 361024;*
[2]*School of Computer Science, Yangtze University, JingZhou, HuBei, China 434023;*
[3]*School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, Missouri, 64110.*
**zhanwei814@yangtzeu.edu.cn*

Abstract

Under mild conditions, it can be induced from the Karush–Kuhn–Tucker condition that the Pareto set, in the decision space, of a continuous Multiobjective Optimization Problems(MOPs) is a piecewise continuous $(m-1)-D$ *manifold(where* m *is the number of objectives). One hand, the traditional Multiobjective Optimization Algorithms(EMOAs) cannot utilize this regularity property; on the other hand, the Regular Model-Based Multiobjective Estimation of Distribution Algorithm(RM-MEDA) only able to build the linear model of decision space using linear modelling algorithm, such as: the local principal component analysis algorithm(Local PCA).Aim at the shortcomings of EMOAs and RM-MEDA, the Manifold-Learning-Based Multiobjective Evolutionary Algorithm Via Self-Organizing Maps(ML-MOEA/SOM) is proposed for continuous multiobjective optimization problems. At each generation, first, via Self-Organizing Maps, the proposed algorithm learns such a nonlinear manifold in the decision space; then, new trial solutions is built through expanding the neurons of SOM with random noise; at the end, a nondominated sorting-based selection is used for choosing solutions for the next generation. Systematic experiments have shown that, overall, ML-MOEA/SOM outperforms NSGA-II, and is competitive with RM-MEDA in terms of convergence and diversity, on a set of test instances with variable linkages. We have demonstrated that, compared with NSGA-II and RM-MEDA, via self-Organizing maps, ML-MOEA/SOM can dig nonlinear manifold hidden in the decision space of multiobjective optimization problems.*

Keywords*: Manifold Learning; Multiobjective Optimizing; Evolutionary Algorithm; Self-organizing Feature Maps*

1. Introduction

In scientific research and engineering areas, many optimization problems belong to multiobjective optimization problems (MOPs). Very often, the objectives in a MOP conflict with each other and no single solution can optimize all the objectives at the same time. The solutions of a MOP are a set which called Pareto set/front in the decision/objective space.

During the past two decades, many evolutionary algorithms (EAs) have been successfully employed to tackle MOPs. The major advantage of these multiobjective evolutionary algorithms (MOEAs) over other methods is that they work with a population

*Corresponding Author

ISSN: 2005-4254 IJSIP

of candidate solutions and thus can produce a set to approximate the Pareto Set/Pareto Front (PS/PF) in a single run. Since the first multiobjective evolutionary algorithm VEGA [1]was proposed by Schaffer in 1985, more and more different MOEAs/MOGAs have been proposed, such as NSGA [2], NSGA-II [3], PAES [4], SPEA [5] and SPEA2 [6]. Among these MOEAs, most of them directly adopt traditional genetic recombination operators such as crossover and mutation and ignore the characteristics of MOPs. Very recently, Deb *et. al*. suggested that variable linkages of MOPs could cause difficulties for MOEAs and recombination operators are crucial to the performance of a MOEA [7].

Estimation of distribution algorithms (EDAs) are a new computing paradigm in evolutionary computation community. Unlike traditional MOEAs, there is no crossover or mutation in EDAs. Instead, they explicitly extract globally statistical information from the selected solutions, and build a probabilistic distribution model of promising solutions based on the extracted information. Then new solutions are generated by sampling from the probabilistic model thus built. Several EDAs have been developed for MOPs [9-12]. However, these EDAs do not take the distribution regularity of Pareto set into consideration in building probability models.

The Pareto optimal solutions to a MOP often distribute very regularly in both the decision space and the objective space. It has been observed that under mild smoothness conditions, the Pareto set, in the decision space, of a continuous MOP is a piecewise continuous (m-1)-dimensional manifold, where m is the number of the objectives.

In order to capture and utilize this regularity of the Pareto set explicitly, Qingfu Zhang and Aimin Zhou *et. al*. proposed a regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA) [13]. RM-MEDA uses local principal component analysis algorithm to build the probability model. Compared with three other state-of-the art algorithms on a set of biobjective or triobjective test instances with linear or nonlinear variable linkages, RM-MEDA performs well. Based on the research on RM-MEDA, Dongdong Yang *et. al*. develop a hybrid multiobjective estimation of distribution algorithm by local linear embedding and an immune inspired algorithm (HMEDA) [14]. Local linear embedding (LLE) is a manifold learning algorithm and is used to build the statistical model in the manifold space by global statistical information. Later, some of solutions are generated by the model, and the rest solutions are produced by an immune inspired sparse individual clone algorithm (SICA). Experiments show that hybridization of LLE and immune inspired algorithm is beneficial to the optimization process.

In this paper, the Manifold-Learning-Based Multiobjective Evolutionary Algorithm Via Self-Organizing Maps(ML-MOEA/SOM) is proposed for continuous multiobjective optimization problems.Self-organizing map (SOM) [15], based on a class of artificial neural networks , a valuable tool in analysis and visualization of high-dimensional data has been proposed [16]. Based on unsupervised learning the SOM performs a non-linear mapping from a high dimensional input space onto a low dimensional grid of neurons. This projection is both topology and distribution preserving. Topology preservation refers to the fact that similar data in the high dimensional space are mapped onto nearby neurons. The distribution preservation property makes that more neurons are allocated to patterns that appear more frequently in the input space. In our algorithm, SOM is used to capture and utilize the manifold structure of the Pareto set. After that, some of solutions are sampled from the SOM grid, and the rest of solutions are generated from crossover and mutation operators. In this way, our hybrid algorithm utilizes both the global statistical information of current population and the location information of the solutions.

The rest of this paper is organized as follows. Section II introduces the continuous MOPs and the regularity property of continuous MOPs. Section III describes the details of the proposed algorithm. Section IV compares the proposed algorithm with NSGA-II and RM-MEDA on a set of test problems with or without variable linkages. Section V outlines the conclusions and future work.

2. Problem Definition

The following continuous multiobjective optimization problem is considered in this paper:

$$\min_{x\in\Omega} F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \quad (1)$$

Where $x = (x_1, x_2, ..., x_n)^T \in \Omega$ is the decision variable vector, $\Omega \subseteq R^n$ is the decision space. $f_i(x), i = 1, 2, ..., m$ are the continuous objective functions to be minimized, R^m is the objective space. Let $u=(u_1,u_2,...,u_m)^T$, $v=(v_1,v_2,...,v_m)^T$ be two objective vectors, u is said to dominate v (denoted by $u \prec v$) if $u_i \le v_i$ for all $i=1,2,...,m$, and $u \ne v$. A solution $x^* \in \Omega$ is called Pareto optimal or nondominated solution if there is no $x \in \Omega$ such that $F(x) \prec F(x^*)$. All the Pareto optimal solutions in the decision space are made up of the Pareto optimal set, denoted by PS. The corresponding image of the Pareto optimal set in the objective space is called the Pareto optimal front, denoted by $PF=\{y \in F(x), x \in PS\}$.

The distribution of the Pareto optimal set of a continuous MOP often shows a high degree of regularity. It can be induced from the Karush-Kuhn-Tucker condition that the PS of a continuous MOP is a piecewise continuous $(m-1)-D$ manifold in the decision space, where m is the number of the objectives [13]. Therefore, the PS of a continuous biobjective optimization problem is a piecewise continuous curve in R^n [17], while the PS of a continuous triobjective MOP is a piecewise continuous surface, and so on.

3. The Alogorithm Framwork

3.1. Basic Idea

In order to capture and utilize the regularity of the PS, ML-MOEA/SOM uses SOM to learn the manifold structure. SOM is topology and distribution preserving. Compared with ISOMAP [18], LLE [19]and other manifold learning algorithms, it is much less sensitive to the noise of the data and is a robust unsupervised algorithm. Because SOM is a kind of artificial neural network algorithm, its learning effect depends on the input data set. Therefore, when the population is converged and submits to manifold distribution, the model which SOM built will be more accurate.

In this paper, we combine the SOM-based and genetics-based methods to generate offspring. That is to say, early in the algorithm, genetic operators are used to generate most solutions, while in the latter most solutions are sampled from the model SOM built.

3.2.The Algorithm Framework

ML-MOEA/SOM maintains a population of N solutions at each generation t, denoted by $Pop(t)=\{x^1, x^2, \dots, x^N\}$. The corresponding objective vector is defined as $\vec{F}(x)=(f_1(x), f_2(x), \dots, f_m(x))^T$, where m is the number of the objectives. The algorithm works as follows.

3.3. Framework of ML-MOEA/SOM:

*Step*1 **Initialization:** Set $t=0$, Randomly generate initial population $Pop(0)$ and compute the corresponding objective values of each solution. Randomly initialize the neuron's weights of SOM.

*Step*2 **Modelling via SOM:** Set the current population $Pop(t)$ as the training data of the SOM, and train the SOM to learn the manifold distribution of the solutions through iterations. As described in the following *Algorithm*1.

*Step*3 **Extend and Reproduction:** Set the number of generation is N, one hand, generate N_1 individual of offspring via uniformly extending neuron of SOM with noise vector by preforming *Algorithm*2; On the other hand, generate N_2 individual of offspring via genetic operation, obviously, $N=N_1+N_2$ and generation of offspring is denoted by Q ($|Q|=N_1+N_2$), then, compute fitness($\vec{F}$) of Q. Combine Q and $Pop(t)$, which is denote it by $Q \cup Pop(t)$.

*Step*4 **Elite Selection:** Select out N solutions from $Q \cup Pop(t)$ with elite selecting strategy which was proposed in NSGA-II.

*Step*5 **Stopping Condition:** If the stopping condition is met, stop and return the nodominated solutions in $Pop(t)$; otherwise, set $t=t+1$ and go to *Step*2

3.3Modeling

The SOM is employed in ML-MOEA/SOM to estimate the distribution of the solutions in the current population. The self-organizing map is a feed-forward network, and consists of an input and an output layer. Output layer consists of M units or neurons arranged on a regular grid, and each output neuron is connected to input vector. Fig.1 illustrates a typically two-dimensional SOM grid.

Each neuron in SOM gird has a specific topological position (an x, y coordinate in the lattice) and contains a vector of weights of the same dimension as the input vectors. That is to say, if the training data consists of vectors $v^i=(v_1^i, v_2^i, ..., v_n^i)^T$, $i=1,2,...,T$, n is the dimension and T is the number of vectors. Then each neuron will contain a corresponding weight vector $w^j=(w_1^j, w_2^j, ..., w_n^j)^T$, $j=1,2,...,M$.

The SOM training procedure can be summarized into the following framework and more details can be found in (T. Kohonen *et. al*., 2001).

*Algorithm*1 : *SOM Training*

*Step*1 **Initialization**: For each neuron j, initialize randomly its weight vector w^j. Set $t=0$.

*Step*2 **Input Vector**: A vector v^i, $i=1,2,...,T$ is chosen at random from the set of training data and presented to the lattice.

*Step*3 **Find the BMU**: For each neuron j, calculate the Euclid distance between its weight vector w^j and v^i. $dist(v^i, w^j)=\sqrt{\sum_{k=0}^{k=n}(v_k^i - w_k^j)^2}$, the best match unit (BMU) j^* can be defined as $BMU(w^{j^*})=\min dist(v^i, w^{j^*})$. That is to say, the neuron j^*'s weights are most like the input vector.

*Step*4 **Calculate The Radius of Neighborhood**: The radius of the neighborhood of the BMU is now calculated. This is a value that starts large, typically set to the radius of the lattice, but diminishes each training step. Any neurons found within this radius are deemed to be inside the BMU's neighborhood.

*Step*5 **Update Weight Vector**: Each neighboring neuron j 's weight vector w^j is adjusted to

make it more like the input vector. The weight vector is updated by $w^j(t+1) = w^j(t) + L(t)\Theta_{j^* j}(t)(v^i(t) - w^j(t))$, where $L(t)$ is learning rate, it diminishes as well as neighborhood radius. $\Theta_{j^* j}(t) = exp\left(-\frac{\left\|r^{j^*} - r^j\right\|^2}{2\sigma^2(t)}\right)$ is called as neighborhood function, where r^{j^*} and r^j denote the topological position vectors of the neuron j^* and j, respectively. $\left\|r^{j^*} - r^j\right\|$ is the Euclid distance between r^{j^*} and r^j. From the definition, we can see that the closer a neuron is to the BMU, the more its weights g*et. al*tered; the training step t is increasing, the alteration is decreasing.

*Step*5 **Stopping Condition**: If the stopping condition is met, stop and return; otherwise, set $t = t+1$ and go to *Step*2 .

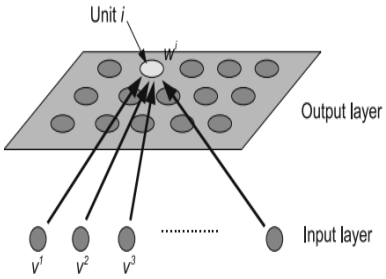

Figure 1. The Architecture of a Self-Organizing Maps. It is Consists of an Input and an Output Layer

3.4. Extend and Reproduction

In our algorithm, the SOM training algorithm is used to capture the distribution regularity of the solutions. After training, N_1 new solutions will be sampled from the SOM neurons. The reproduction procedure can be described as follows.

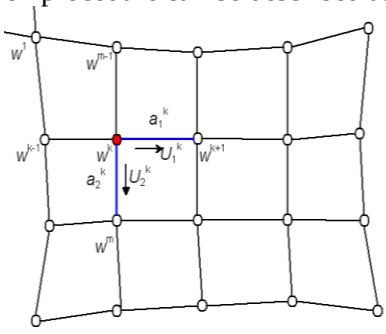

Figure 2. Sample New Solutions by Adjacent Neurons. Along the Two Directions (Blue Lines) a Uniformly Distributed Random Point is Generated

*Algorithm*2 : *Generate* N_1 *Solutions From SOM*

*Step*1 **Performing SOM Training**: Set the current population *Pop*(*t*) as the set of training data of the SOM. The number of neurons of SOM is *M* and the number of

iterations is N. The initial learning rate is denoted by L_0 and learning rate is defined by $L(t) = L_0 exp\left(-\frac{t}{N}\right), \quad t = 1, 2, ..., N.$ The radius of neighborhood is defined by $\sigma(t) = \sigma_0 exp\left(-\frac{t}{\lambda}\right), \quad t = 1, 2, ..., N$, where σ_0 is the initial neighboring radius which typically set to the radius of the lattice, $\lambda = N / lg(\sigma_0)$ is a constant. Then, perform *Algorithm*1 to train the SOM.

*Step*2 **Modeling**: After the training, the neurons of SOM will distribute in an ordered fashion on the manifold. Figure2 shows a 2-D SOM lattice for a triobjective MOP, and each neuron could be considered as a center point on the manifold. We generate new solutions in the quadrangle of adjacent neurons in the lattice. Taking the red neuron w^k as a center point, w^{k+1}、w^m are neighborhoods of w^k .we compute the two unit vector U_1^k、U_2^k and the Euclid distances a_1^k and a_2^k between neighboring neurons in the lattice. Here,

$$a_1^k = \left\| w^{k+1} - w^k \right\|, a_2^k = \left\| w^m - w^k \right\|$$

$$U_1^k = (w^{k+1} - w^k) / \left\| w^{k+1} - w^k \right\| \; U_2^k = (w^m - w^k) / \left\| w^m - w^k \right\|.$$

Notice that the w^k、w^{k+1}、w^m are all weight vectors of its corresponding neurons.

*Step*3 **Reproduction**: The set of the points covered by SOM lattice can be defined as

$$\begin{aligned} &\psi = \{x \in R^n \mid x = w^k \\ &\quad + \sum_{i=1}^{m-1} \alpha_i U_i^k + N(0, \sigma I),\ 0 \le \alpha_i \le a_i^k\}, \\ &k \in \{1, 2, ..., M\}, \\ &\sigma = \frac{1}{\sqrt{n}} e^{-t/T} \end{aligned} \tag{2}$$

where $N(0, \sigma I)$ is a n-dimensional zero-mean Gaussian vector, I is the $n \times n$ identity matrix, n is the dimension of decision variable vector x and m is the number of the objectives. t is the current generation of algorithm, T is the max generation of algorithm. Therefore, the noise vector is adaptive and decreasing over the optimization. In this paper, in order to generate points outside the area covered by the SOM, we also extend the boundary of the SOM lattice as Figure2 shows.

*Step*4: N_1 new solutions are sampled from the model built in *Step*3. As described above, we employ genetic operators to generate most solutions early in the algorithm, and use the model (2) to sample most solutions when the population is converged. The numbers of solutions generated by these two methods are controlled by an adaptive strategy. It works as follows.

*Algorithm*3: *Adaptive GA Operation*

*Step*1 **Adaptive Strategy**: N_2 new solutions are generated by genetic operators. Here, N_2 is defined as $N_2 = 0.7Ne^{-t/T}$, where N is the population size, t is the current generation of algorithm, T is the max generation of algorithm. Obviously, $N_2 \approx 0.7N$ at

the beginning of algorithm and N_2 is decreasing as t is increasing and $N_2 = 0.7Ne^{-1}$ finally.

*Step*2 **Crossover and Mutation**: Perform crossover and mutation on $Pop(t)$ to create N_2 new solutions. The crossover and mutation operators used in our algorithm are multi-parent crossover and polynomial mutation [3], respectively.

3.5 Selection

The selection operator is the same as in NSGA-II. The details of the selection operator can be found in [3].

4. Experimental Studies

4.1. Test Instances

In this paper, we compare the performances of ML-MOEA/SOM, RM-MEDA and NSGA-II experimentally on a set of test problems. These test instances are ZDT1, ZDT1.1, ZDT1.2, ZDT2, ZDT2.1, ZDT2.2, DTLZ2, DTLZ2.1, DTLZ2.2, DTLZ7.

4.2. Performance Metric

Three performance metrics are used to compare the performance of the different algorithms in our experimental studies.

(1) Υ metric[3] measures the convergence (closeness of the non-dominated solutions to the Pareto front) of a population. It is defined as follows.

$$\begin{aligned} &\Upsilon(S, S^*) = \frac{1}{|S|}\sum_{x\in S} d(x, S^*) \\ &d(x, S^*) = \min_{y\in S^*} \|F(x) - F(y)\|^2 \end{aligned} \tag{3}$$

Where S is the set of the non-dominated solutions and S^* is a set uniformly sampled from the true Pareto front. The smaller $\Upsilon(S, S^*)$ is, the closer S to S^* and $\Upsilon(S, S^*) = 0$ once $S \subseteq S^*$.

(2) Δ metric[3] measures the diversity of a population. The original metric in works only for 2-objectives problems. In this paper, we adopt the extension version in [20]. It calculates the distance from a point to its nearest neighbor:

$$\begin{aligned} &\Delta(S, S^*) = \frac{\sum_{i=1}^{m} d(e_i, S) + \sum_{x\in S} \left| d(x, S) - \bar{d} \right|}{\sum_{i=1}^{m} d(e_i, S) + |S|\bar{d}} \\ &d(x, S) = \min_{y\in S, y\neq x} \|F(x) - F(y)\|^2 \\ &\bar{d} = \frac{1}{|S|}\sum_{y\in S} d(y, S) \end{aligned} \tag{4}$$

Where S and S^* are the same as in Υ metric. $e_1, e_2, ..., e_m$ are m extreme solutions in S^*. If the achieved solutions are well distributed and include those extreme solutions, $\Delta(S, S^*) = 0$.

(3) IGD (Inverted Generational Distance) metric [13] could measure both the diversity and convergence of a population. Let P^* be a set of uniformly distributed points along the PF. Let P be an approximation to the PF, the IGD from P^* to P is defined as

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{\left|P^*\right|} \tag{5}$$

Where $d(v, P)$ is the minimum Euclid distance between v and the points in P .To have a low value of $IGD(P^*, P)$, P must be very close to the PF and can't miss any part of the PF.

4.3. Experimental Setting

The experimental settings are as follows:

(1)Public parameters: real code, randomly initialization, the population size in 2 and 3 objective problems is 100 and 200 respectively.The number of decision variables is 30 for all the test problems. The generation of algorithm is 200.

(2) NSGA-II: The distribution indexes for the simulated binary crossover and polynomial mutation are 15 and 20, and the crossover and mutation probabilities are 1.0 and 0.5 respectively. In ML-MOEA/SOM, the number of parents of the multi-parent crossover is set to be 4, and the crossover and mutation probabilities are the same as in NSGA-II. For all 2-objectives problems, the SOM lattice is set to be 1×10, and 4×5 for all 3-objectives problems. The training steps are 1000 and initial learning rate is 0.1 for SOM. The number of clusters in RM-MEDA is set to be 5.

(3)Number of the algorithm runs and stopping condition: 10 independent runs are performed on each test problems. The algorithms stop after a given number of function evaluations. The maximal number of function evaluations in each algorithm is 20000 for 2-objective test problems and 40000 for 3-objective test problems.

4.4. Experimental Results

The convergence metric and IGD metric results are shown in Table 1-10. From the tables, we can see that: (1) ML-MOEA/SOM shows the best performance on ZDT1, ZDT1.1, ZDT2, ZDT2.1, DTLZ2, DTLZ2.1, while on ZDT1.2, ZDT2.2 and DTLZ2.2, RM-MEDA performs better. ZDT1, ZDT2 and DTLZ2 are MOPs without variable linkage. Therefore, the Local PCA model of RM-MEDA is hard to discover the variable relations since there are no variable linkages at all; (2) NSGA-II performs best on DTLZ7. DTLZ7 has discontinuous PS and could make some difficulties for ML-MOEA/SOM and RM-MEDA. In contrast, conventional genetic operators are good at solving discontinuous problems; (3) especially on ZDT1.1, ZDT1.2, ZDT2.1, ZDT2.2, DTLZ2.1 and DTLZ2.2, which are introduced by the linear or nonlinear variable linkages, ML- MOEA/SOM and RM-MEDA are much better than NSGA-II on IGD metric, since NSGA-II has no efficient mechanism for using the regularity of the PS.

Table 1. The Mean and Variance of Performance Metric on ZDT1

Algorithm	ZDT1		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.040709±0.008069	0.376196±0.068360	0.038647±0.006809
RM-MEDA	0.042218±0.008282	**0.210959±0.022507**	0.041582±0.000116
MOEA/SOM	**0.035758±0.006963**	0.255311±0.031633	**0.034987±0.006821**

Table 2. The Mean and Variance of Performance Metric on ZDT2

Algorithm	ZDT2		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.072239±0.007518	0.557177±0.071762	0.065155±0.006842
RM-MEDA	0.050285±0.022078	**0.300842±0.096472**	0.050220±0.022120
MOEA/SOM	**0.043578±0.000597**	0.415434±0.062008	**0.042545±0.005954**

Table 3. The Mean and Variance of Performance Metric on dtlz2

Algorithm	DTLZ2		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.082040±0.013245	0.451950±0.035201	0.091724±0.009653
RM-MEDA	0.084612±0.012321	**0.346967±0.068888**	0.082729±0.008819
OM-MOEA	**0.038201±0.002244**	0.383479±0.026420	**0.062213±0.002229**

Table 4. The Mean and Variance of Performance Metric on dtlz7

Algorithm	DTLZ7		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	**0.023332±0.001186**	0.536266±0.039985	**0.048576±0.002192**
RM-MEDA	0.028198±0.001456	0.496164±0.025794	0.054868±0.002482
ML_MOEA/SOM	0.026114±0.000949	**0.485186±0.025878**	0.060510±0.003920

Table 5.The Mean and Variance of Performance Metric on ZDT1.1

Algorithm	ZDT1.1		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.021230±0.003292	0.724661±0.050664	0.111683±0.036066
RM-MEDA	**0.001429±0.000161**	**0.141164±0.013741**	**0.003938±0.000076**
MOEA/SOM	0.001694±0.000153	0.147793±0.010920	0.004150±0.000096

Table 6. The Mean and Variance of Performance Metric on ZDT2.1

Algorithm	ZDT2.1		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.028407±0.013604	1.002302±0.063827	0.277415±0.077414
RM-MEDA	**0.001429±0.000161**	0.160962±0.018670	**0.004058±0.000080**
MOEA/S	0.001442±0.000	**0.125449±0.015**	0.004195±0.000

Table 7. The Mean and Variance of Performance Metric on DTLZ2.1

Algorithm	DTLZ2.1		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.279189±0.117000	1.215399±0.285118	0.284848±0.026751
RM-MEDA	0.019445±0.008185	0.391852±0.060346	0.047088±0.001748
MOEA/S	**0.012411±0.000**	**0.390347±0.017**	**0.046295±0.000**

Table 8. The Mean and Variance of Performance Metric on ZDT1.2

Algorithm	ZDT1.2		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.010132±0.001184	0.786300±0.026410	0.311743±0.047102
RM-MEDA	**0.002592±0.000248**	0.152935±0.019939	**0.004624±0.000180**
MOEA/S	0.004829±0.000	**0.137854±0.018**	0.006849±0.000

Table 9. The Mean and Variance of Performance Metric on ZDT2.2

Algorithm	ZDT2.2		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.049310±0.007405	0.908554±0.033490	0.554455±0.004525
RM-MEDA	**0.002924±0.000578**	0.146815±0.010378	**0.005073±0.000383**
MOEA/S	0.007336±0.000	**0.135108±0.016**	0.008296±0.000

Table 10.The Mean and Variance of Performance Metric on DTLZ2.2

Algorithm	DTLZ2.2		
	Convergence ϒ	Diversity Δ	IGD
NSGA-II	0.306508±0.073715	1.448311±0.027336	0.319979±0.024331
RM-MEDA	0.0427097±0.007627	**0.356518±0.035539**	0.061335±0.003039
MOEA/S	**0.036424±0.0026**	0.430564±0.040	**0.057755±0.002**

Figure 3-12 show the evolution of the average IGD metric of the non-dominated solutions in the current populations among 10 independent runs with the number of function evaluations in three algorithms. From these figures, we can compare the convergence rapidity of the algorithms intuitively.

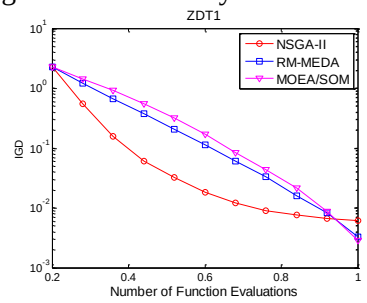

Figure 3.The Evolution of the Average IGD metric with the Number of 20000 Function Evaluations in Three Algorithms for ZDT1

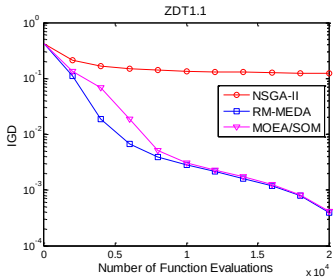

Figure 4. The Evolution of the Average IGD Metric with the Number of 20000 Function Evaluations in Three Algorithms for ZDT1.1

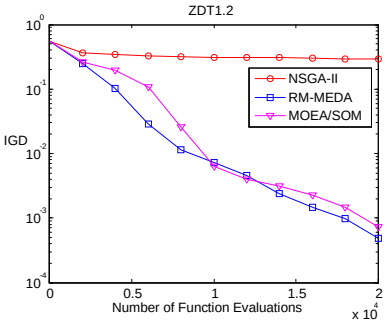

Figure 5. The Evolution of the Average IGD Metric with the Number of 20000 Function Evaluations in Three Algorithms for ZDT1.2

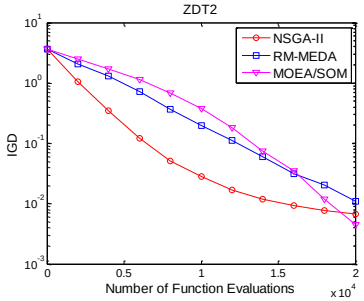

Figure 6. The Evolution of the Average IGD Metric with the Number of 20000 Function Evaluations in Three Algorithms for ZDT2

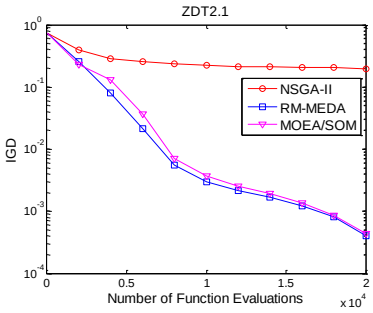

Figure 7.The Evolution of the Average IGD Metric with the Number of 40000 Function Evaluations in Three Algorithms for ZDT2.1

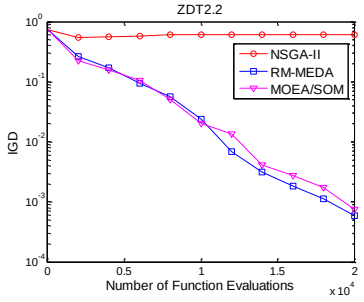

Figure 8. The Evolution of the Average IGD Metric with the Number of 20000 Function Evaluations in Three Algorithms for ZDT2.2

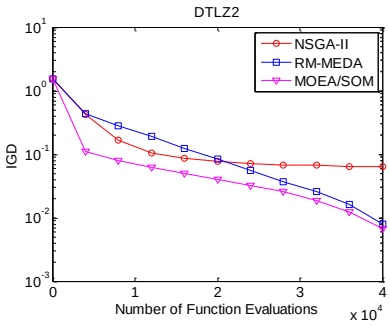

Figure 9.The Evolution of the Average IGD Metric with the Number of 40000 Function Evaluations in Three Algorithms for DTLZ2.

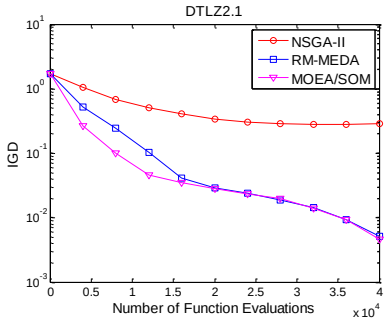

Figure 10. The Evolution of the Average IGD Metric with the Number of 40000 Function Evaluations in Three Algorithms for DTLZ2.1

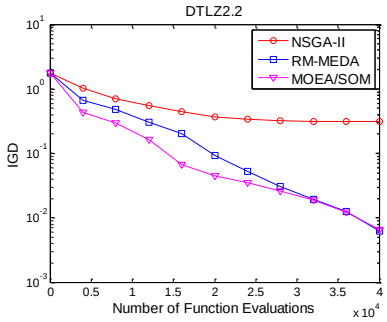

Figure 11. The Evolution of the Average IGD Metric with the Number of 40000 Function Evaluations in Three Algorithms for DTLZ2.2

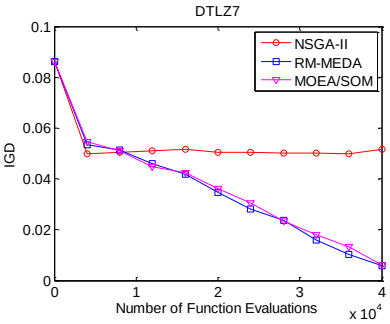

Figure 12. The Evolution of the Average IGD Metric with the Number of 40000 Function Evaluations in Three Algorithms for DTLZ7

From Figure3 to Figure12, for ZDT1 and ZDT2, the convergence speed of NSGA-II is fastest, RM-MEDA and ML-MOEA/SOM is both almost. For ZDT1.1, ZDT1.2, ZDT2.1,

ZDT2.2, the convergence speed of ML-MOEA/SOM and RM-MEDA is both almost, NSGA-II is lowliest. For DTLZ2, DTLZ2.1, DTLZ2.2, DTLZ7, on the earlier stage of algorithm running, the convergence speed of ML-MOEA/SOM is obviously superior to the other two, but, on the later stage of algorithm running, the convergence speed of RM-MEDA and ML-MOEA/SOM is both almost, but the performance of NSGA-II is worst.

5. Conclusion

Traditional reproduction operators such as crossover and mutation usually did not perform well on MOPs with variable linkages. In this paper, based on the regularity of continuous MOPs, we propose a hybrid multiobjective evolutionary algorithm based on self-organizing map and genetic operation. The proposed algorithm employs self-organizing map to capture the manifold distribution of the Pareto set. After that, some of new trail solutions are sampled from the SOM grid, and the rest of solutions are generated from adaptive genetic operators. In this way, our hybrid algorithm utilizes both the global statistical information of current population and the location information of the solutions.

We compared the proposed algorithm with NSGA-II and RM-MEDA on a set of test problems with or without variable linkages. The experimental results show that the proposed algorithm performs better than NSGA-II, and is competitive with RM-MEDA. The future research may investigate other simple and efficient modeling methods to exploit the regularity of the solutions. The better strategy for offspring generation is also a research direction.

Acknowledgement

The authors would like to acknowledge the help of Jiankai Zhu, I am grateful for the anonymous reviewers, and the anonymous associate editor for their insightful comments and suggestions.

References

[1] J. D. Schaffer, “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”, In Proceedings of the 1st International Conference on Genetic Algorithms, Lawrence Erlbaum, **(1985)**, pp. 93-100.
[2] N. Srinivas and K. Deb, “Multiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation, vol. 2, no. 3, **(1994)**, pp. 221-248.
[3] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, **(2002)**, pp. 182-197.
[4] J. Knowles and D. W. Corne, “Local search multiobjective optimization and the Pareto achived evolutionary strategy”, In Proceedings of the third Australia-Japan joint workshop on intelligent and evolutionary systems, **(1999)**, pp. 209-216.
[5] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach”, IEEE Transactions on Evolutionary Computation, vol. 3, **(1999)**, pp. 257-271.
[6] E. Zitzler, M. Laumanns and L. Thiele, “SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK)”, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, **(2001)**.
[7] K. Deb, A. Sinha and S. Kukkonen, “Multiobjective test problems, linkages, and evolutionary methodologies”, In Proc. Genetic Evol. Comput. Conf. (GECCO 2006), Seattle, Washington, **(2006)**, pp.1141-1148.
[8] D. Thierens and P. A. N. Bosman, “Multiobjective 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.
[9] M. Costa and E. Minisci, “MOPED: A Multiobjective Parzen-based Estimation of Distribution Algorithm for Continuous Problems”, In Fonseca C. M. editors, Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization. **(2003)**, pp. 282-294.
[10] K.Sastry, M. Pelikan and D. E. Goldberg, “Decomposable Problems, Niching, and Scalability of Multiobjective Estimation of Distribution Algorithms”, IlliGAL Report 2005004, Illinois Genetic Algorithms Laboratory, **(2005)**.

[11] N. Khan, D. E. Goldberg and M. Pelikan, "Multiobjective Bayesian optimization algorithm", In Proceedings of the Genetic and Evolutionary Computation Conference, **(2005)**, pp. 684-689.
[12] T. Okabe, Y. Jin and B. Sendhoff, "Voronoi-based Estimation of Distribution Algorithm for Multiobjective Optimization", Evolutionary Computation, **(2004)**, pp.1594-1601.
[13] 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, **(2008)**, pp. 41-63.
[14] D. Yang, L. Jiao, M. Gong and X. Feng, "Hybrid Multiobjective Estimation of Distribution Algorithm by Local Linear Embedding and an Immune Inspired Algorithm", In Proceedings of The 2009 IEEE Congress on Evolutionary Computation, **(2009)**, pp. 463-470.
[15] T. Kohonen, M. R. Schroeder and T. S. Huang, "Self-Organizing Maps", Springer-Verlag New York, **(2001)**.
[16] H. B. Amor and A. Rettinger, "Intelligent Exploration for Genetic Algorithms Using Self-Organizing Maps in Evolutionary Computation", Proceedings of the 2005 conference on Genetic and evolutionary computation, **(2005)**, pp. 1531-1538.
[17] A. Zhou, Q. Zhang and Y. Jin, "A Model-Based Evolutionary Algorithm for Bi-objective Optimization",Congress on Evolutionary Computation, **(2005)**, pp. 2568-2575.
[18] J. B. Tenenbaum, V. Silva and J. C. Langford, "A Global Geometirc Framework for Nonlinear Dimensionality Reduction", Science, vol. 290, no. 22, **(2000)**, pp. 2319-2323.
[19] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding", Science, vol. 290, **(2000)**, pp. 2323-2326.
[20] A. Zhou, Y. Jin and Q. Zhang, "Combining Model-based and Genetics-based Offspring Generation for Multiobjective Optimization Using a Convergence Criterion", IEEE Congress on Evolutionary Computation, **(2006)**, pp. 3234-3241.

