An Orthogonal Multi-objective Evolutionary Algorithm with Lower-dimensional Crossover

Song Gao*, Sanyou Zeng*, Bo Xiao, Lei Zhang,Yulong Shi,
Xin Tian,Yang Yang, Haoqiu Long, Xianqiang Yang, Danping Yu, Zu Yan

***Abstract*— This paper proposes an multi-objective evolutionary algorithm. The algorithm is based on OMOEA-II[2]. A new linear breeding operator with lower-dimensional crossover and copy operation is used. By using the lower-dimensional crossover, the complexity of searching is decreased so the algorithm converges faster. The orthogonal crossover increase probability of producing potential superior solutions, which helps the algorithm get better results. Ten unconstrained problems in [1] are used to test the algorithm. For three problems, the obtained solutions are very close to the true Pareo Front, and for one problem, the obtained solutions distribute on part of the true Pareo Front.**

I. Introduction

Almost every real-world problem involves simultaneous optimization of several incommensurable and often competing objectives. Evolutionary algorithms have the ability to find multiple Pareto-optimal solutions in one single simulation run. They have often been used to solve multi-objective problems. Such as vector evaluated genetic algorithm (VEGA)[3], Hajela and Lins genetic algorithm(HLGA)[4], pareto-based ranking procedure(FFGA)[5], niched Pareto genetic algorithm (NPGA)[6], pareto archived evolution strategy (PAES)[7], nondominated sorting genetic algorithm (NSGA-II)[8], strength pareto evolutionary algorithm (SPEA2)[9], rMOGAxs[10], and generalized regression GA (GRGA)[11]. Algorithms mentioned above are based on pareto front. Some algorithms based on other selection mechanisms, such as Decomposition[12], are also proposed.

Orthogonal design method [13] is developed to sample a small, but representative set of combinations for experimentation to obtain good combination. Leung and Zhang incorporated orthogonal design in genetic algorithm for single objective problems [14][15], found such method was more robust and statistically sound. In [16], orthogonal design method is used in multi-objective evolutionary algorithm and developed algorithm was called OMOEA. It was showed that OMOEA could find good solutions. But OMOEA degraded its performance on both precision and distribution of the yielded solutions for problems with strong interaction between variables, and when the number of objectives increases, the solutions yielded by OMOEA increased exponentially. In [2], an improved version of OMOEA (OMOEA-II) is proposed. Orthogonal design method is nested in crossover operator to select better genes as offsprings, and consequently, enhances the performance of OMOEA. Both orthogonal crossover and linear crossover are used in OMOEA-II. By combining two crossover operators, faster convergence and better solutions are obtained.

In this paper, the algorithm is based on OMOEA-II. Linear crossover operator used in OMOEA-II takes 2 parents. In the new version of the algorithm, lower-dimensional crossover is used and it takes more than 2 parents(5 in this paper), this pass more and wider superior factors on to the offsprings. Copy operation is also introduced into the algorithm, which increase the probability of holding good factors unmodified during a cycle of evolution.

The performance assessment guidelines, and 10 unconstrained problem provided in [1], which is for the CEC 2009 Special Session and Competition, are used to test the algorithm.

The rest of this paper is organized as follows: Section II briefly describes orthogonal design method. Section III presents the algorithm. Section IV shows numerical experiment results. Finally, section V concludes with a summary of the paper.

II. Orthogonal Design Methods

An example was introduced in [17] to explain the basic concept of experimental design methods. Orthogonal design solutions may not really be optimal. However, orthogonal design has been proven to be optimal for additive and quadratic models.

A special class of orthogonal arrays $L_M(Q^P)$, which we shall use a simple permutation method to construct, will be used in this paper, where Q is prime and $M = Q^J$, where J is a positive integer satisfying

$$P = \frac{Q^J - 1}{Q - 1} \tag{1}$$

Denote the jth column of the orthogonal array $[a_{i,j}]_{M\times P}$ by a_j. Column a_j for $j = 1, 2, (Q^2-1)/(Q-1)+1, (Q^3-1)/(Q-1)+1, \ldots, (Q^{J-1}-1)/(Q-1)+1$ are called basic columns, and the others are called nonbasic columns. The algorithm first constructs the basic columns, and then generates the nonbasic columns. The details are as follows.

Algorithm 1: Construction of orthogonal array $L_M(Q^P)$
//Construct the basic columns as follows:
$FORk = 1TOJ$

Authors are with School of Computer Science & Technology, China University of Geosciences(Wuhan). Email:spectergs@gmail.com(Song Gao), sanyou-zeng@263.net(Sanyou Zeng); Address: School of Computer Science, Research Centre for Space Science & Technology and The State Key Laboratory of Geological Processes and Mineral Resources, China University of Geosciences, Wuhan, 430074, Hubei, China.

$j = \frac{Q^{k-1}-1}{Q-1} + 1$;
$FORi = 1TOQ^j$
$a_{i,j} = \lfloor \frac{i-1}{Q^{J-k}} \rfloor modQ$;
ENDFOR
ENDFOR
//Construct the nonbasic columns as follows:
FOR k=2 TO J
$j = \frac{Q^{k-1}-1}{Q-1} + 1$;
FOR s=1 TO j-1, t=1 TO Q-1
$a_{j+(s-1)(Q-1)+t} = (a_s \times t + a_j) modQ$;
ENDFOR
ENDFOR

The used orthogonal array in this paper is required to satisfy $P \geq N$ and M is as small as possible. That is the columns P of $L_M(Q^P)$ must be larger than the number of factors (or decision variables) in the hope of sampling small number of points (combinations) for obtaining better solution. It only needs to determine Q and J for determining $L_M(Q^P)$. $L_M(Q^P)$ is determined by solving the following minimization problem:

$$MinimizeM = Q^J$$
$$SubjecttoP = \frac{Q^J - 1}{Q - 1} > N \quad (2)$$

where Q is a prime and $Q \geq 3$, J is a positive integer.

$L_M(Q^P)$ is the full size of the orthogonal array, which has P columns. For a problem with N decision variables, we discard the last P . N columns of $L_M(Q^P)$ and get an orthogonal array $L_M(Q^N)$.

The proposed algorithm will require the mean value of the objective at each level of each factor. Denote the objective values of the orthogonal experiments by $[y_i]_{M\times 1}$ where the objective has the value y_i at the ith combination, the mean values by $[\triangle_{k,j}]_{Q\times N}$ where the objective has the mean value $\triangle_{k,j}$ at the kth level of the jth factor, and

$$\triangle_{k,j} = \frac{Q}{M} \sum_{a_{i,j}=k} y_i \quad (3)$$

where the orthogonal array $L_M(Q^N)$ has the value $a_{i,j}$ at ith row and jth column. That is, the jth factor has level $a_{i,j}$ in the ith combination(experiment). The objective has value y_i at the ith combination, and $\sum_{a_{i,j}=k} y_i$ implies the sum of y_i where $\forall i$ satisfy $a_{i,j} = k$. The details of the algorithm are as follows

Algorithm 2: Calculation of mean value $[\triangle_{k,j}]_{Q\times N}$
$[\triangle_{k,j}]_{Q\times N} = [0]_{Q\times N}$
//Add up objective result for each factor at each level
FOR i=1 TO M, j=1 TO N
$q = a_{i,j}; \triangle_{q,j} = \triangle_{q,j} + y_i$;
ENDFOR
//Average results for each factor at each level
$[\triangle_{k,j}]_{Q\times N} = [\triangle_{k,j}]_{Q\times N} \times Q/M$

Each factor has its best level according to the mean value matrix $[\triangle_{k,j}]_{Q\times N}$, the combination s of the best levels is potentially a good solution. For minimization problems, it is calculated by

$$\triangle_{k_j,j} = min\{\triangle_{1,j}, \triangle_{2,j}, \ldots, \triangle_{Q,j}\}, j = 1, 2, \ldots, N$$
$$s = (k_1, k_2, \ldots, k_N) \quad (4)$$

III. Algorithm Description

A. Framework of Algorithm

1) *Randomly create population P_0 with size $N_{PopSize}$. Let $t = 0$*
2) *Execute **Linear Breeding Operator** on P_t which yields offspring Q_t with size $N_{popSize}$, let P'_t=$P_t \cup Q_t$.*
3) *Execute **Orthogonal Breeding Operator** on P'_t with probability $p_{orthogonal}$ which yields offspring R_t, let P'_t =$P'_t \cup R_t$.*
4) *Execute **Selection Operator** on P'_t which yields next population. Let $P_{t+1} = P'_t, t = t + 1$.*
5) *If stopping criterion satisfied, goto Step 6*
Else goto Step 2.
6) *Terminate algorithm and output P_t.*

B. Linear Breeding Operator

The linear breeding operator execute *lower-dimensional crossover*, *mutation* and *copy* operation. i_{index} stands for the index of the individual to be copied, and $N_{ParentCount}$ stands for the number of parents for the crossover operation. The details are as follows.

1) *Let i_{index}=0.*
2) *Randomly select $N_{ParentCount}$ individuals:I_1, I_2, ..., $I_{N_{ParentCount}}$ from the current population, where index$\neq i_{index}$.*
3) *Randomly get $N_{ParentCount}$ variables r_1, r_2, ..., $r_{N_{ParentCount}}$ range from -1 to $N_{ParentCount}$, which satisfy $r_1 + r_2 + \ldots + r_{N_{ParentCount}} = 1$.*
4) *Produce a new individual I_{new} , for each decision variable $x_i (i = 1, 2, \ldots, N_{DNALength})$, execute following 3 sub-steps::*
   a) ***Lower-dimensional Crossover operation**:*
   $$Let \quad v = \sum_{j=1}^{N_{ParentCount}} r_j \times I_j.x_i.$$
   If v>$UpperBound_i$, let x_i=$UpperBound_i$
   Else If v<$LowerBound_i$, let x_i=$LowerBound_i$
   Else, let x_i=v
   b) ***Mutation operation**: Randomly create v ranges from $LowerBound_i$ to $UpperBound_i$, let $x_i = v$ with probability $p_{mutation}$.*
   c) ***Copy operation**: $x_i \Leftarrow Population[i_{index}].x_i$ with probability p_{copy}.*
5) *Add I_{new} into the current population.*
6) *If i_{index}=$N_{PopSize}$, terminate operation*
Else, let i_{index}=$i_{index} + 1$ and goto Step 2.

For lower-dimensional crossover operation, the offspring v of the crossover stays in the linear space defined by the $N_{ParentCount}$ parents I_1, I_2, ..., $I_{N_{ParentCount}}$. The dimension of the space is equal or smaller than $N_{ParentCount} - 1$.The constraints, $-1 \leq r_1, r_2, \ldots, r_{N_{ParentCount}} \leq N_{ParentCount}$, constrain v in a neighborhood of the $N_{ParentCount}$ parents. Therefore, v stays in a neighborhood of the $N_{ParentCount}$ parents with smaller than $N_{ParentCount} - 1$ dimensions no matter how many dimensions the decision space of the optimization problem has. We know the fast convergence of the gradient algorithm is due to its linear search along the gradient direction. Therefore, the new algorithm should converge fast with higher dimensional decision space especially for optimization problems.

C. Orthogonal Crossover Operator

1) Orthogonal Crossover: Orthogonal design method is used on the subspace extended by the randomly chosen parents.

$$\mathbf{H} = \{x_1, x_2, \ldots, x_N | l'_i \leq x_i \leq u'_i, i = 1, 2, \ldots, N\}$$
$$l'_i = min\{m_{1,i}, m_{2,i}\} u'_i = max\{m_{1,i}, m_{2,i}\} \quad (5)$$

For $(x_1, x_2, \ldots, x_n)$ in $\mathbf{H}$, x_i is regarded as the ith factor. Orthogonal array is selected by Equation (2). Each factor i is parted into Q-1 equal portions and yields Q levels $x_{1,i}, x_{2,i}, \ldots, x_{Q,i}$, where the design parameter Q must be prime and $x_{q,i}$ is given by

$$x_{q,i} = \begin{cases} l'_i & q = 1 \\ l'_i + (q-1)\delta_i & 2 \leq q \leq Q-1 \\ u'_i & q = Q \end{cases}$$
$$where \delta_i = \frac{u'_i - l'_i}{Q-1} \quad (6)$$

In other words, the difference between two successive levels is the same. For convenience, denote $x_j = \{x_{1,j}, x_{2,j}, \ldots, x_{Q,j}\}$, and call $x_{q,j}$ the qth level of the jth factor.

2) Orthogonal crossover operator:

1) *Randomly choose parents m_1 and m_2, Construct subspace **H** according to Equation (5)*
2) *Choose an objective k from the K objectives as optimizing objective*
3) *Employ orthogonal design method on **H**, where orthogonal array is determined by Equation (2)*
4) *Add the potentially good solutions from Equation (4) into the current population.*
5) *Terminate operation.*

D. Selection Operator

1) *Empty P_{t+1}*
2) *Find the non-dominated set B of R_t. If $|B| = N_P$ then $P_{t+1} \Leftarrow B$;if $|B| > N_P$ then execute cutoff operator followed which eliminate $|B| - N_P$ elements from B and assigned the reduced B to P_{t+1}:*
   a) *Initialize cluster set $\Psi : \Psi = \cup_{i \in B}\{\{i\}\}$ where each individual $i \in B$ constitute a distinct cluster.*
   b) *If $|\Psi| \leq N_p$, goto Step e), else goto Step c).*
   c) *Calculate the distance of all possible pairs of clusters. The distance d_c of two cluster $C_1, C_2 \in \Psi$ is given as the average distance between pairs of individuals across the two clusters*
   $$d_c = \frac{1}{|C_1| \cdot |C_2|} \sum_{i_1 \in C_1, i_2 \in C_2} d(i_1, i_2)$$
   where $d(i_1, i_2)$ is the distance between two individuals i_1 and i_2 (here the distance in objective space is used).
   d) *Determine two clusters C_1 and C_2 with minimal distance dc; the chosen clusters are amalgamated into a large cluster: $\Psi = \Psi\{C_1, C_2\} \cup \{C_1 \cup C_2\}$. Go to Step b)*
   e) *For each cluster, select a representative individual and remove all other individuals from the cluster. We consider the centroid (the point with minimal average distance to all other points in the cluster) as the representative individual. Compute the reduced non-dominated set by uniting the representative of the clusters: $P_{t+1} = \cup_{C \in \Psi} C$.*

   if $|B| < N_P$ then move B from R_t to P_{t+1}, i.e., $R_t \Leftarrow R_t \setminus B and P_{t+1} \Leftarrow P_{t+1} \cup B$, repeat the process of finding the non-dominated set of reduced R_t and moving the non-dominated set from R_t to P_{t+1} till $|P_{t+1}| = N_P$.
3) *Let current population be P_{t+1}.*

IV. Numerical Experiments and Discussion

A. Test problems

10 Unconstrained multi-objective optimization test instances for the *CEC 2009 Special Session and Competition* are taken to test the algorithm[1]. All the problems are required to take 30 decision variables. 7 of the test problems have 2 objectives to be minimized and the rest 3 problems have 3 objectives. All the test problems are treated as black-box problems.

B. Testing environment

1) Parameter setting: For all test instances, parameter settings are the same:

Population size	300
Number of parent in linear crossover	5
Mutation probability	0.05
Copy probability	0.05
Orthogonal crossover probability	0.1
Maximal number of function evaluations	300,000

2) PC Configuration:

CPU	Intel T7500 2.2 GHz,4MB
RAM	2 GB, 667MHz
Operating System	Windows Server 2003 with SP2
Middle-ware	.Net Framework 3.5
Computer Language	Visual C#.NET

3) Testing method: We use a performance metric[1] to assess the final solutions:

> Let P^* be a set of uniformly distributed points along the PF (in the objective space). Let A be an approximate set to the PF, the average distance from P^* to A is defined as:
>
> $$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|}$$
>
> where d(v,A) is the minimum Euclidean distance between v and the points in A. If $|P^*|$ is large enough to represent the PF very well, IGD(A, P^*) could measure both the diversity and convergence of A in a sense. To have a low value of IGD(A, P^*), the set A must be very close to the PF and cannot miss any part of the whole PF.

Due to the population size, 300 final solutions will be produced. We use the same algorithm as selection operator to reduce them. To meet the requirement of the contest, the numbers of solutions are reduced to 100 for 2-objective problems, and 150 for 3-objective problem. Then the metric above is used to assess the 100 solutions.

For each test problem, the algorithm is set to be run independently 30 times. The average IGD value of 30 test results is the assessed value for a test instance.

C. Results

1) Figures:

a) Distribution of obtained solutions (Figure 1,2,3,4,5): For each test problem, we draw a figure of obtained solutions from the test instance who has the best IGD value. The points for solutions are represented by $'+'$s. For contrast, the true Pareto Front represented by $'\cdot'$s are drawn in the same figure. The distribution figures show the diversity and how close the obtained solution set is close to the true Pareto Front.

b) Evolution of IGD values (Figure 6,7,8,9,10): For each test problem, we also draw a figure showing the evolution of the means(represented by blue lines)/standard deviations(represented by red bars) of IGD values of the approximate solution sets obtained with the number of function evaluations. For readability, the standard deviations are tenfold drawn.

2) Table: The Table I shows the means and standard deviations of IGD value of the 30 final test result for each test problem.

Fig. 1. Distribution of obtained solutions for problem 1 & 2

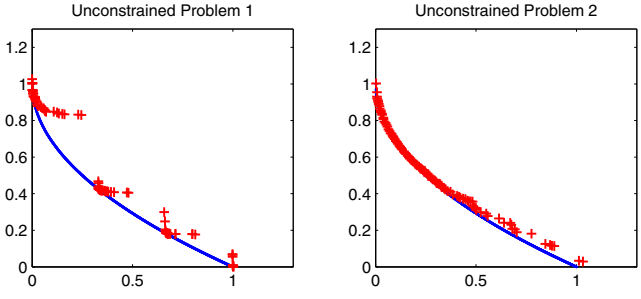


Fig. 2. Distribution of obtained solutions for problem 3 & 4

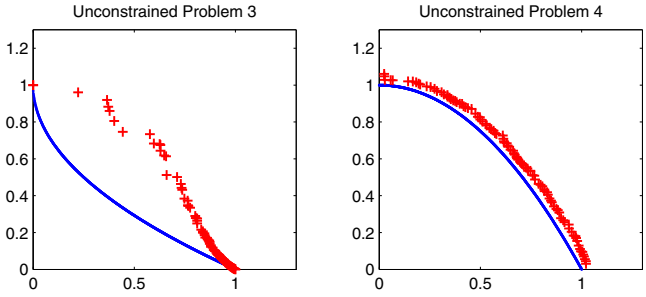


Fig. 3. Distribution of obtained solutions for problem 5 & 6

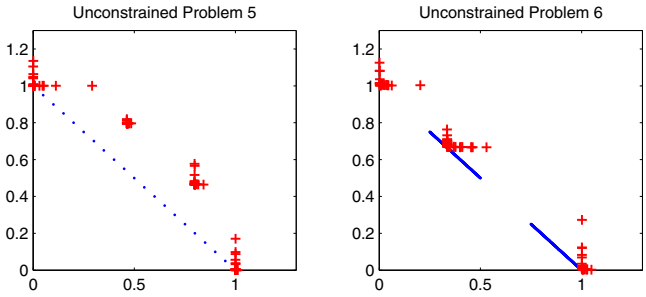


Fig. 4. Distribution of obtained solutions for problem 7 & 8

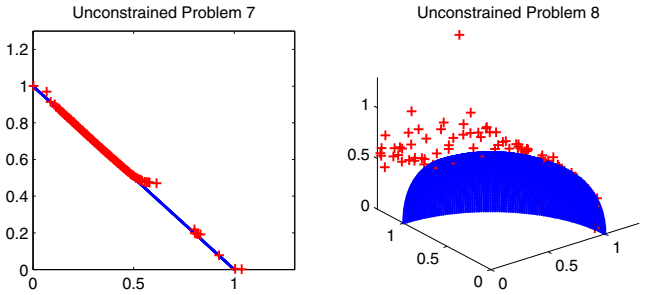


Fig. 5. Distribution of obtained solutions for problem 9 & 10

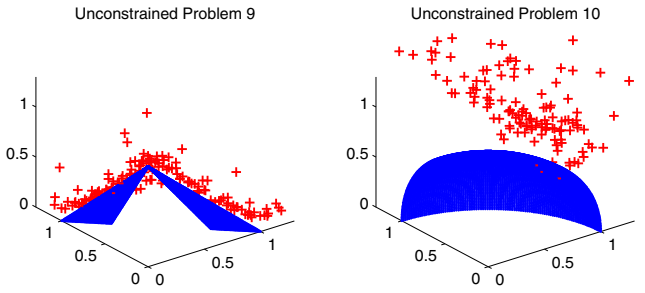


Fig. 6. Evolution of the means of IGD for problem 1 & 2

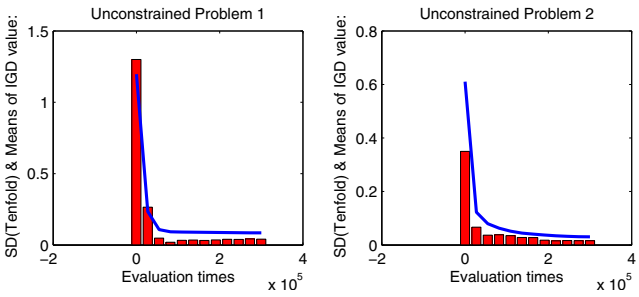


Fig. 7. Evolution of the means of IGD for problem 3 & 4

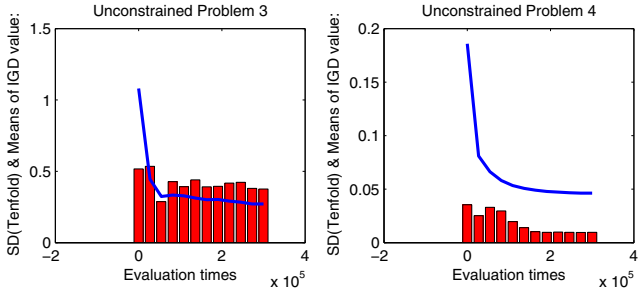


Fig. 8. Evolution of the means of IGD for problem 5 & 6

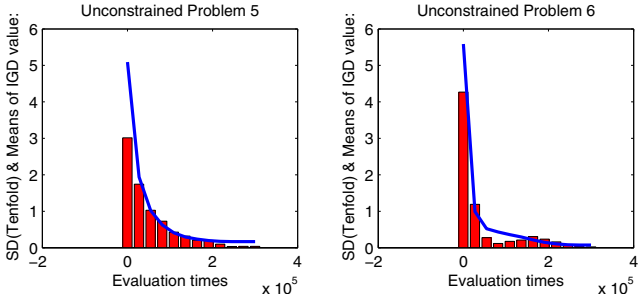


Fig. 9. Evolution of the means of IGD for problem 7 & 8

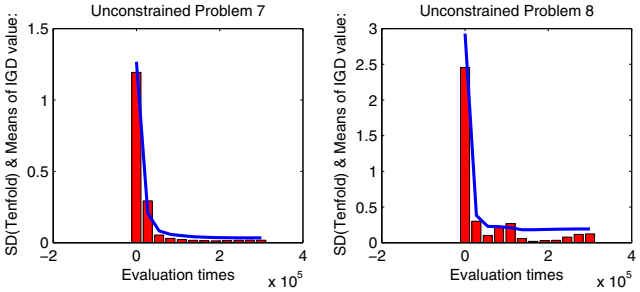
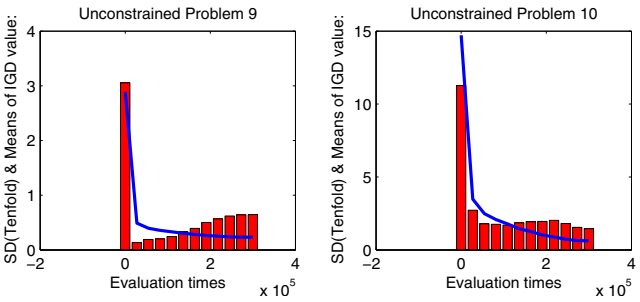


Fig. 10. Evolution of the means of IGD for problem 9 & 10



V. Conclusions

A. Use of Lower-dimensional Crossover

The lower-dimensional crossover, which searches a lower dimensional neighbor of the parents, decreases the complexity of searching. Therefore, the algorithm can converge fast. Because the parents are selected randomly, and it is the neighborhood but not convex space is searched, the ability of global search can be ensured.

B. Use of Orthogonal Crossover

The orthogonal design method used in the orthogonal crossover increase probability of producing potential superior solutions, which helps the algorithm get better results.

C. Performance

Among the 10 unconstrained problems in [1], the algorithm find solutions are very close to the true Pareo Front for three problems, and for one problem, the obtained solutions distribute on part of the true Pareo Front. The algorithm performs preferably, but still has limitations.

Acknowledgment This work was supported by the National Natural Science Foundation of China (No.s: 60871021, 60473037), and by the open research program of the Geological Processes and Mineral Resources (GPMR) ,China University of Geosciences(No. GPMR200618).

References

[1] Qingfu Zhang, Aimin Zhou, S. Zhao, P. N. Suganthan, Wudong Liu, Santosh Tiwari,*Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition*

[2] Sanyou Zeng, Shuzhen Yao, Lishan Kang, and Yong Liu, *An Efficient Multi-objective Evolutionary Algorithm: OMOEA-II*, Springer-Verlag Berlin Heidelberg 2005, pp.108-119,2005.

[3] Schaffer, J. D. (1984).*Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. Ph. D. thesis, Vanderbilt University. Unpublished.

[4] Hajela, P. and Lin, C. Y. (1992). *Genetic search strategies in multicriterion optimal design*. Structural Optimization 4, 99C107.

[5] Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, pp. 416C423. Morgan Kaufmann.

[6] Horn, J. and Nafpliotis, N. (1993). *Multiobjective optimization using the niched pareto genetic algorithm*. IlliGAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Champaign.

[7] Knowles, J.D., Corne, D.W.(2000): *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*. Evolutionary Computation 8(2), 149C172.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan (2002). *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation, 6(2):182-197.

[9] Eckart Zitzler, Marco Laumanns, and Lothar Thiele(2002). *SPEA2: Improving the Strength*. Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al., editor, Proceedings of the EUROGEN2001 Conference, pages 95-100, Barcelona, Spain, CIMNE

[10] Purshouse, R. C., Fleming, P. J. (2001). *The Multi-objective Genetic Algorithm Applied to Benchmark ProblemsCan Analysis*. Research Report No. 796. Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK.

[11] Tiwari, A., Roy, R.(2002). *Generalised Regression GA for Handling Inseparable Function Interaction: Algorithm and Applications*. Proceedings of the seventh international conference on parallel problem solving from nature. (PPSN VII). Granada, Spain

[12] Q. Zhang, H Li. *MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition*, IEEE Trans. on Evolutionary Computation, vol.11, no. 6, pp712-731, 2007.

[13] Wu, Q.(1978). *On the optimality of orthogonal experimental design*. Acta Math. Appl Sinica, 1(4), pp. 283-299.

[14] Q. Zhang and Y. W. Leung, *Orthogonal Genetic Algorithm for Multimedia Multicast Routing.* , IEEE Trans on Evolutionary Computation, Vol. 3, No. 1, 1999.

[15] Jinn-Tsong Tsai, Tung-Kuan Liu, and Jyh-Horng Chou (2004). *Hybrid*

TABLE I

MEANS AND STANDARD DEVIATIONS OF IGD VALUES

Problem	IGD Values			
	Best	Average	Worst	St. Deviation
Problem 1	0.07836255	0.08564624	0.09674832	0.00407076
Problem 2	**0.02757032**	**0.03057246**	0.03429509	0.00160912
Problem 3	0.20197888	0.27141506	0.35318674	0.03761221
Problem 4	**0.04444180**	**0.04624691**	0.04818113	0.00096680
Problem 5	0.16334994	0.16920102	0.17805277	0.00390177
Problem 6	0.06819333	0.07338196	0.07937132	0.00244898
Problem 7	**0.03117927**	**0.03354878**	0.03880302	0.00173570
Problem 8	0.13916320	0.19200591	0.20111423	0.01229657
Problem 9	0.10505533	0.23179599	0.34110383	0.06476747
Problem 10	0.43971699	0.62754414	1.08267135	0.14595430

Taguchi- Genetic Algorithm for Global Numerical Optimization, IEEE Transactions on Evolutionary Computation, 8(4),pp365-377.

[16] Sanyou Y.Zeng, Lishan S.Kang, Lixing X.Ding (2004). *An Orthogonal Multiobjective Evolutionary Algorithm for Multi-objective Optimization Problems with Constraints*. Evolutionary Computation, 12(1), pp77-98.

[17] Montgomery, D. C.(1991),*Design and Analysis of Experiments. 3rd ed.*, New York:Wiley.