A Clustering Multi-objective Evolutionary Algorithm Based on Orthogonal and Uniform Design

Yuping Wang, Chuangyin Dang, Hecheng Li, Lixia Han and Jingxuan Wei

***Abstract*— Designing efficient algorithms for difficult multi-objective optimization problems is a very challenging problem. In this paper a new clustering multi-objective evolutionary algorithm based on orthogonal and uniform design is proposed. First, the orthogonal design is used to generate initial population of points that are scattered uniformly over the feasible solution space, so that the algorithm can evenly scan the feasible solution space once to locate good points for further exploration in subsequent iterations. Second, to explore the search space efficiently and get uniformly distributed and widely spread solutions in objective space, a new crossover operator is designed. Its exploration focus is mainly put on the sparse part and the boundary part of the obtained non-dominated solutions in objective space. Third, to get desired number of well distributed solutions in objective space, a new clustering method is proposed to select the non-dominated solutions. Finally, experiments on thirteen very difficult benchmark problems were made, and the results indicate the proposed algorithm is efficient.**

I. Introduction

Many real-world problems involve simultaneous optimization of several incommensurable and often competing objectives. They usually can be written as

$$\min_{x\in\Omega}\{f_1(X), f_2(X), ..., f_s(X)\}, \tag{1}$$

where $X = (x_1, x_2, ..., x_n) \in R^n$ and Ω is the feasible solution space.

Evolutionary algorithms (EAs) have become well-known methods for exploring the Pareto front in multi-objective optimization problems that are too complex to be solved by exact methods such as nonlinear programming approaches. After the first pioneering studies on evolutionary multi-objective optimization appeared in the middle of 1980's, many EAs were proposed in 1990's and the beginning of 2000's (e.g., [1]∼[8]). These approaches and their variants have been successfully applied to various multi-objective optimization problems. However, they still can not successfully solve very difficult problems ([9]∼ [11]). Recently, some new algorithms have been proposed for these difficult problems and the results seem very encouraging([9]∼ [11]).

In this paper, a new evolutionary algorithm is proposed for these very difficult problems. First, the orthogonal design is used to generate initial population of points. Second, a new crossover operator is designed based on uniform design. Its exploration focus is mainly put on the sparse part and the boundary part of the obtained non-dominated solutions in objective space. Third, a new clustering method is proposed to select the non-dominated solutions. The experiments on thirteen very difficult benchmark problems were made, and the results indicate the proposed algorithm is efficient.

Yuping Wang is with the School of Computer Science and Technology, Xidian University, Xi'an, China (email: ywang@xidian.edu.cn). Chuangyin Dang is with Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon Tong, Hong Kong, China (mecdang@cityu.edu.hk). Hecheng Li, Lixia Han and Jingxuan Wei are Ph.D students with the School of Computer Science and Technology, Xidian University, Xi'an, China

II. Experimental Design Methods

In this section, we briefly introduce two experimental design methods: uniform design method and orthogonal design method. The main objective of uniform design and orthogonal design is to sample a small set of points from a given closed and bounded set $G \subset R^M$ such that the sampled points are uniformly scattered on G. In the following, we consider only two specific cases of uniform design and orthogonal design, and describe the main features of uniform and orthogonal designs. For more details, please refer to ([12])- [16]).

A. A Uniform Design Method

We first explain the meaning of uniformly scattered points on an M-dimensional unit hypercube C ([12]):

$$C = \{(\theta_1, \theta_2, ..., \theta_M) \mid 0 \le \theta_i \le 1,\ i = 1, ..., M\}. \tag{2}$$

For any given point in C, say $r = (r_1, r_2, ..., r_M)$, a hyper-rectangle between 0 and r, denoted by $C(r)$, is defined by

$$C(r) = \{(\theta_1, \theta_2, ..., \theta_M) \mid 0 \le \theta_i \le r_i,\ i = 1 \sim M\}. \tag{3}$$

For a given set of q points in C, suppose that $q(r)$ of these points fall in the hyper-rectangle $C(r)$. Then the fraction of the points falling in the hyper-rectangle $C(r)$ is given by $q(r)/q$. Since the volume of the unit hypercube is 1, hence, the fraction of the volume of this hyper-rectangle is given by $r_1 r_2 \cdots r_M$. The uniform design on C is then defined as determining q points in C such that the following discrepancy is minimized:

$$\sup_{r\in C}\left|\frac{q(r)}{q} - r_1 r_2 \cdots r_M\right|. \tag{4}$$

Let $l = (l_1, ..., l_n)$ and $u = (u_1, ..., u_n)$ be two points of R^n, where $l_i \le u_i$, $i = 1 \sim n$. Denote

$$[l, u] = \{(\theta_1, \theta_2, ..., \theta_n) \mid l_i \le \theta_i \le u_i,\ i = 1 \sim n\}. \tag{5}$$

Finding a set of exactly uniformly scattered points on $[l, u]$ and C is, in general, very difficult, but there are some efficient methods to look for a set of well approximately uniformly scattered points on $[l, u]$ and C. One of such simple and efficient methods is the Good-Lattice-Point method

(GLP) ([12]), which generates a set of q approximately uniformly scattered points on $[l, u]$, denoted by $O(n, q)$, as follows:

- Given q and M, determine a number σ (In this paper, we take $q = 5$, $M = q - 1$ and $\sigma = 2$).
- Generate a $q \times M$ integer matrix called uniform array denoted by

$$\begin{array}{ll} G(M,q) & = [G_{ij}]_{q\times M}, \\ where & G_{ij} = (i\sigma^{j-1} \ mod \ q) + 1, \\ & i = 1 \sim q, \ j = 1 \sim M. \end{array} \quad (6)$$

- Each row of matrix $G(M, q)$ defines a point $O^i = (o_{i1}, o_{i2}, \cdots, o_{in}) \in O(n, q)$ in the following way: Randomly divide l and u into M blocks of sub-vectors, respectively, :

$$l = (A^1, A^2, \cdots, A^{q-1}) \text{ and } u = (B^1, B^2, \cdots, B^{q-1}), \quad (7)$$

where A^j and B^j are sub-vectors of l and u in the same dimension. Then the k-th point

$$O^k = (O_{k1}, O_{k2}, \cdots, O_{kq-1}) \quad (8)$$

is generated by

$$O_{kj} = A^j + \frac{2G_{kj}+1}{2q}(B^j - A^j), \ j = 1 \sim q-1 \quad (9)$$

for $k = 1 \sim q$.

B. An Orthogonal Design Method

We use an example to introduce the basic concept of experimental design methods. For more details, see [13]-[16]. The yield of a vegetable depends on: 1) the temperature, 2) the amount of fertilizer, and 3) the pH value of the soil. These three quantities are called the factors of the experiment. Each factor has three possible values, and we say that each factor has three *levels* ([15], [16]).

To find the best combination of levels for a maximum yield, we can do one experiment for each combination, and then select the best one. In the above example, there are $3 \times 3 \times 3 = 27$ combinations, and hence there are 27 experiments. In general, when there are N factors and Q levels per factor, there are Q^N combinations. When N and Q are large, it may not be possible to do all Q^N experiments. Therefore, it is desirable to sample a small, but representative set of combinations for experimentation.

The *orthogonal design* was developed for this purpose. It provides a series of orthogonal arrays for different N and Q. We let $L_M(Q^N)$ be an orthogonal array for N factors and Q levels, where "L" denotes a Latin square and M is the number of combinations of levels. It has M rows, where every row represents a combination of levels. For convenience, we denote $L_M(Q^N) = [a_{i,j}]_{M\times N}$ where the jth factor in the ith combination has level $a_{i,j}$ and $a_{i,j} \in \{1, 2, \cdots, Q\}$.

In general, the orthogonal array $L_M(Q^N)$ has the following properties ([15], [16]).

1) For the factor in any column, every level occurs M/Q times.
2) For the two factors in any two columns, every combination of two levels occurs M/Q^2 times.
3) For the two factors in any two columns, the M combinations contain the following combinations of levels: $(1,1), (1,2), \cdots, (1,Q), (2,1), (2,2), \cdots, (2,Q), \cdots, (Q,1), (Q,2), \cdots, (Q,Q)$.
4) If any two columns of an orthogonal array are swapped, the resulting array is still an orthogonal array.
5) If some columns are taken away from an orthogonal array, the resulting array is still an orthogonal array with a smaller number of factors.

Consequently, the selected combinations are scattered uniformly over the space of all possible combinations and their good representatives.

C. Construction of Orthogonal Array

As we will explain, the proposed algorithm may require different orthogonal arrays for different optimization problems. Although many orthogonal arrays have been tabulated in the literature (e.g., [12]$\sim$ [16])), it is impossible to store all of them for the proposed algorithm. We will only need a special class of orthogonal arrays $L_M(Q^N)$, where Q is odd and $M = Q^J$, where J is a positive integer fulfilling

$$n = \frac{Q^J - 1}{Q - 1}. \quad (10)$$

In this subsection, we design a simple permutation method to construct orthogonal arrays of this class.

We denote the jth column of the orthogonal array $[a_{i,j}]_{M\times N}$ by $\boldsymbol{a}_j$. Columns $\boldsymbol{a}_j$ for $j = 1, 2, (Q^2-1)/(Q-1)+1, (Q^3-1)/(Q-1)+1, \cdots, (Q^{J-1}-1)/(Q-1)+1$ are called the *basic columns*. and the others are called the *nonbasic columns*. We first construct the basic columns, and then construct the nonbasic columns. The details are as follows.

Algorithm 1 ([16])): *Construction of Orthogonal Array*

1. Construct the basic columns as follows: FOR $k = 1$ TO J DO BEGIN

$$j = \frac{Q^{k-1} - 1}{Q - 1} + 1;$$

FOR $i = 1$ TO Q^J DO

$$a_{i,j} = \left\lfloor \frac{i-1}{Q^{J-k}} \right\rfloor \ mod \ Q;$$

END.

2. Construct the nonbasic columns as follows: FOR $k = 2$ TO J DO BEGIN

$$j = \frac{Q^{k-1} - 1}{Q - 1} + 1;$$

FOR $s = 1$ TO $j - 1$ DO FOR $t = 1$ TO $Q - 1$ DO

$$a_{j+(s-1)(Q-1)+t} = (a_s \times t + a_j) \ mod \ Q;$$

3. Increment $a_{i,j}$ by one for all $1 \le i \le M$ and $1 \le j \le N$.

III. A Clustering Multi-objective Evolutionary Algorithm

Before an optimization is solved, we may have no information about the Pareto optimal solutions. It is reasonable that the individuals of initial population should be uniformly scattered over whole search space, so that the algorithm can explore the whole search space evenly.

A. Generation of Initial Population by Using Orthogonal Design

Note that an orthogonal array specifies a small number of combinations that are scattered uniformly over the space of all the possible combinations. Therefore, it is potential and possible for orthogonal design methods to generate a good initial population.

We define x_i to be the i-th factor, so that each chromosome has n factors. These factors are continuous, but the orthogonal design is applicable to discrete factors only. To overcome this issue, we use the similar idea as that in [16] to quantize each factor into a finite number of values. In particular, we quantize the domain $[l_i, u_i]$ of x_i into Q levels $\alpha_{i,1}, \cdots, \alpha_{i,Q}$, where the design parameter Q is odd and $\alpha_{i,j}$ is given by

$$\alpha_{i,j} = \begin{cases} l_i & j = 1 \\ l_i + (j-1)\left(\frac{u_i - l_i}{Q-1}\right) & 2 \le j \le Q-1 \\ u_i & j = Q. \end{cases} \quad (11)$$

For convenience, we call $\alpha_{i,j}$ the j-th level of the i-th factor, and we denote $\alpha_i = (\alpha_{i,1}, \cdots, \alpha_{i,Q})$.

Note that Algorithm 1 can only construct $L_M(Q^n)$, where $M = Q^J$ and J is a positive integer fulfilling $(Q^J - 1)/(Q - 1) \ge n$. Since the problem dimension n is given, we can choose proper Q and the smallest $J = 2$ such that

$$\frac{Q^J - 1}{Q - 1} \ge n. \quad (12)$$

We execute Algorithm 1 to construct an orthogonal array with $N' = (Q^J - 1)/(Q - 1) = (Q + 1)$ factors, and then delete the last $N' - n$ columns to get an orthogonal array with N factors. The details are given in the following algorithm.

Algorithm 2: *Construction of $L_M(Q^n)$:*

1. Select the proper Q fulfilling $(Q + 1) \ge n$.
2. Let $N' = Q + 1$.
3. Execute Algorithm 1 to construct the orthogonal array $L_{Q^2}(Q^{N'})$.
4. Delete the last $N' - n$ columns of $L_{Q^2}(Q^{N'})$ to get $L_M(Q^n)$ where $M = Q^2$.

After constructing $L_M(Q^n) = [a_{i,j}]_{M \times n}$ we set $Q = Q_1,\ Q_2,\ Q_3$, respectively. For each $Q = Q_i$ $(i = 1,\ 2,\ 3)$, we get a sample of $M_i = Q_i^2$ combinations out of Q_i^n combinations for $i = 1 \sim 3$. We apply these M_i combinations to generate the following M_i chromosomes as a part of initial population

$$\begin{cases} (\alpha_{1,a_{1,1}}, \alpha_{2,a_{1,2}}, \cdots, \alpha_{n,a_{1,n}}) \\ (\alpha_{1,a_{2,1}}, \alpha_{2,a_{2,2}}, \cdots, \alpha_{n,a_{2,n}}) \\ \cdots \\ (\alpha_{1,a_{M_i,1}}, \alpha_{2,a_{M_i,2}}, \cdots, \alpha_{n,a_{M_i,n}}) \end{cases} \quad (13)$$

Then the initial population size is $(M_1 + M_2 + M_3)$ and we select the non-dominated solutions of the initial population as the first generation population. Note that the non-dominated solutions at the beginning of each generation will be as the current population of this generation in the proposed algorithm.

B. Crossover Operator

To explore the search space effectively and efficiently, we shall have two phases to do crossover. In the first phase, we find the non-dominated solutions in the sparse parts of the non-dominated front (i.e., the non-dominated solutions in the objective space) and do the crossover among them. So that some new solutions will be generated in these sparse parts. In the second phase, we find the boundary non-dominated solutions along the non-dominated front and do the crossover among them. So that some new boundary solutions can be generated and these new boundary solutions can make the non-dominated front wider spread. The detail is as follows:

Algorithm 3 (*Crossover*)

1. Select the non-dominated solutions in the sparse parts of the non-dominated front from the current population.
2. Match each selected solution with its neighbor solution as a pair and use formulas (8) to (9) to generate offspring of each pair of matched solutions.
3. Select the boundary non-dominated solutions. For any boundary solution x, suppose that its neighbor solution is y. Find the intersect point z of the boundary of search space and the line starting at x along the direction $x - y$.
4. Match boundary solution x and z as a pair and use formulas (8) to (9) to generate offspring of each pair of matched solutions.
5. The set of all offspring is denoted as O.

C. Mutation Operator

Suppose that the probability of mutation is $p_m > 0$ and current population is denoted as $POP(t) = \{x_1,\ x_2, \cdots, x_\theta\}$, where θ is the population size at current generation t.

Algorithm 4 (*Mutation*)

1. Select the solutions to take part in the mutation in the the current population based on the mutation probability $p_m > 0$: randomly generate θ real numbers $r_1,\ r_2, \cdots, r_\theta \in [0, 1]$. If $r_i \le p_m$, then select x_i to undergo mutation.
2. Suppose that $x_{k_1}, x_{k_2}, \cdots, x_{k_s}$ are selected to undergo the mutation. For each selected point x_{k_i}, randomly choose one of its component, say, the j-th component, and randomly generate a number $r \in [0, 1]$.

Then the offspring of x_{k_i} is generated by replacing its j-th component by $L(j) + r[U(j) - L(j)]$.

3. The set of all offspring is denoted as $O1$.

D. Select the next generation population by a new clustering method

Suppose that the required number of solutions is nT. To make the non-dominated solutions found in objective space uniformly distributed, we shall use a new clustering method to delete the additional solutions if the population size is great than nT. The detail is as follows;

Algorithm 5 (*Selection Based on A New Clustering Method*)

1. Find the set of non-dominated solutions of set $POP(t) \cup O \cup O1$ and denoted as $T1$. Find the two boundary solutions along the direction of each objective function. Select these boundary solutions and suppose that the number of these boundary solutions is $n1$.
2. Let T denote the set of solutions obtained by delete these boundary solutions from $T1$, and suppose that T contains $|T|$ solutions.
3. If T contains more than $nT - n1$ solutions, go to step 4, otherwise, let $POP(t+1) = T$, $t = t+1$.
4. Use new clustering method to delete $|T| - (nT - n1)$ solutions from T by following clustering method.

Algorithm 6 (*A New Clustering Method*)

1. Calculate the distance between any two solutions of T in objective space. We can get a $|T| \times |T|$ matrix $D = [d_{ij}]$ of distances, where d_{ij} is the distance between the i-th solution and the j-th solution of T.
2. Calculate the mean distance of all these distances D_{mean}.
3. For each solution $x \in T$, define its neighborhood with radius D_{mean} called the cluster with respect to x, and find the number of solutions belonging to this cluster. Find the solution whose neighborhood contains the largest number of solutions. Delete this solution from T. Repeat Step 3 until T contains nT solutions.

E. A Clustering Multi-objective Evolutionary Algorithm

Algorithm 7 (*A Clustering Multi-objective Evolutionary Algorithm*)

1. Given mutation probability $p_M > 0$, the number of required solutions nT and a relative large integer $NT > nT$. Choose odd integers $Q_1 \sim Q_5$ such that $Q_i > n$ for $i = 1 \sim 5$. Generate a part of initial population by taking $Q = Q_i$ in method of Section 3.1 for each $i = 1 \sim 5$. The union of these parts is the initial population $POP(0)$. Let $t = 0$.
2. Find the set of non-dominated solutions of $POP(t)$ as $POP(t+1)$.
3. Crossover: Use Algorithm 3 to do the crossover and the set of offspring is denoted as O.
4. Mutation: Use Algorithm 4 to do the mutation and the set of offspring is denoted as $O1$.
5. Selection of next generation population: Find the set of non-dominated solutions of set $POP(t) \cup O \cup O1$ and denoted as $T1$. If $|T1| \leq NT$, let $POP(t+1) = T1$, $t = t+1$, go to Step 3; otherwise, use Algorithm 5 to select the next generation population $POP(t+1)$. Let $t = t+1$, go to Step 3.
6. If stop criterion is satisfied, stop.

IV. Experimental Results

A. Test Problems

Thirteen unconstrained test problems UF1 to UF10, R2-DTLZ2, R3-DTLZ3 and WFG1 are chosen from [17], where $UF1$ to $UF7$ are two objective function problems, $UF8$ to $UF10$ are three objective function problems, and R2-DTLZ2, R3-DTLZ3 and WFG1 are five objective function problems. These test problems are much more difficult than the existing well-known test problems such as the test problems proposed in [18]. Thus they are challenging enough to test the performance of the proposed algorithm. For more details please refer to [9] and [17].

B. Parameter Values

In the experiments, we adopt the following parameter values:

- Parameters used for generating initial population: $Q_1 = 32,\ Q_2 = 33,\ Q_3 = 34$.
- Mutation probability: 0.02
- Number of offspring of each pair of parents: $q = 5$
- Maximum number of the expected non-dominated solutions found: $nT = 100,\ 150,\ 800$ for two, three and five objectives, respectively.
- Stop criterion: Maximum number of function evaluations: 300000.

C. Experimental Results

For each test problem the proposed algorithm has been executed 30 independent times. All problems were run using matlab on a Pentium(R) 4 PC computer with CPU 3.00GHz and 480MB. On each run, the outcome of the simulation is a non-dominated front (i.e., the non-dominated solutions in the objective space). These outcomes have been used to compute the values of performance measure, where the performance metric adopted in this paper is IGD [17] and IGD is the mean distance from a set of uniformly distributed solutions taken from the true Pareto front to the set of solutions obtained by an algorithm in objective space. The smaller the IGD value for the set of obtained solutions is, the better of the performance of the corresponding algorithm will be. The results are given in Tables 1 to 2, and Figures 1 to 10, respectively.

For all two objective function problems UF1 to UF7 except for problem UF5, it can be seen from Table 1 that the best, mean and worst IGD values of obtained solution sets found by the proposed algorithm are all small. This indicates that the proposed algorithm can find the high quality and well distributed solution sets in objective space for almost all two objective function problems. Although the best, mean

TABLE I

THE BEST, MEAN, WORST IGD VALUES AND MEAN CPU TIME FOR ALL PROBLEMS.

	Best	Mean	Worst	CPU(Sec.)
UF1	**0.0207**	**0.0299**	**0.0347**	**347.90**
UF2	**0.0191**	**0.0228**	**0.0300**	**352.54**
UF3	**0.0331**	**0.0549**	**0.0927**	**255.88**
UF4	**0.0513**	**0.0585**	**0.0648**	**259.92**
UF5	**0.1971**	**0.2473**	**0.3517**	**208.68**
UF6	**0.0809**	**0.0871**	**0.0965**	**331.78**
UF7	**0.0184**	**0.0223**	**0.0260**	**398.15**
UF8	**0.2000**	**0.2383**	**0.2624**	**1536.54**
UF9	**0.2000**	**0.2934**	**0.3789**	**1360.97**
UF10	**0.3383**	**0.4111**	**0.4786**	**538.67**
R2-DTLZ2	**1.1305**	**1.2401**	**1.3053**	**5685.34**
R3-DTLZ3	**705.16**	**1039.36**	**1225.29**	**2104.38**
WFG1	**3.2497**	**3.4043**	**3.5071**	**5666.60**

TABLE II

STD VALUE OF IGD FOR ALL PROBLEMS.

Std for UF1-UF5.				
0.0033	0.0023	0.0147	0.0027	0.0384
Std for UF6-UF10.				
0.0057	0.0020	0.0230	0.0781	0.0501
Std for R2-DTLZ2, R3-DTLZ3 and WFG1.				
0.0954	290.03	0.1363		

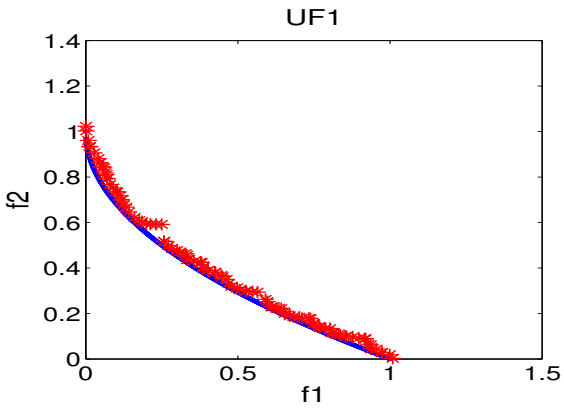

Fig. 1. Nondominated Front obtained for test problem UF1

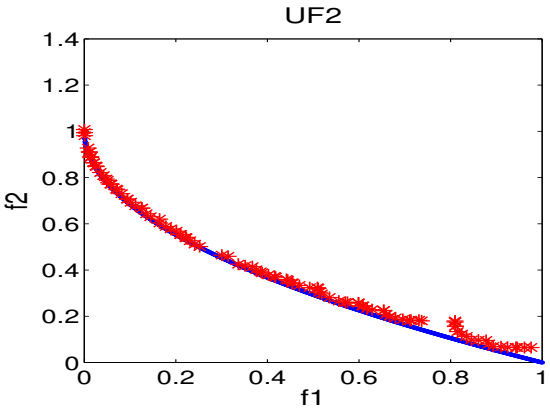

Fig. 2. Nondominated Front obtained for test problem UF2

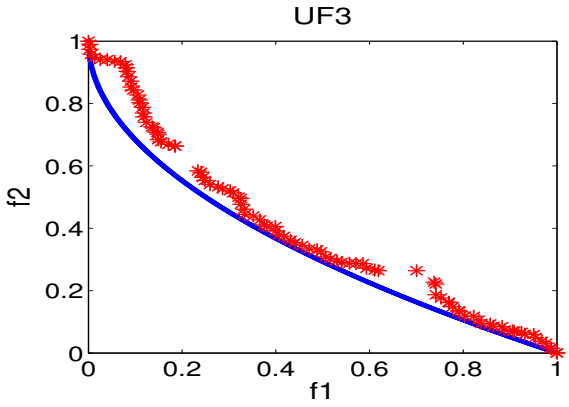

Fig. 3. Nondominated Front obtained for test problem UF3

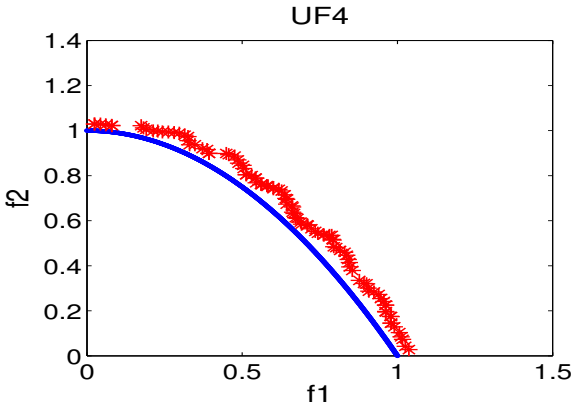

Fig. 4. Nondominated Front obtained for test problem UF4

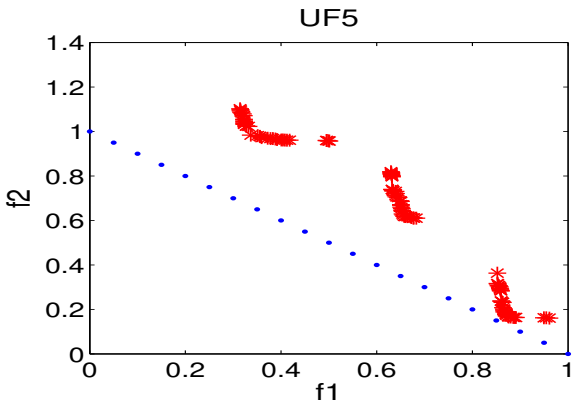

Fig. 5. Nondominated Front obtained for test problem UF5

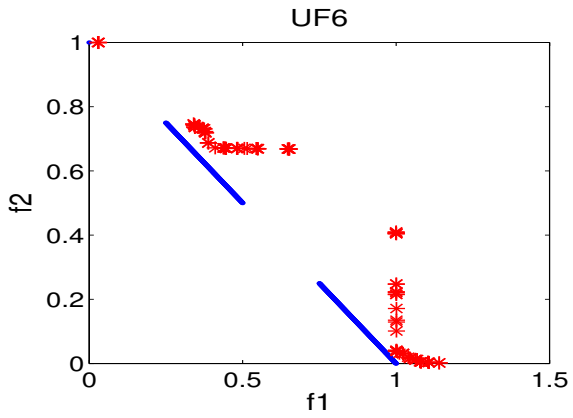

Fig. 6. Nondominated Front obtained for test problem UF6

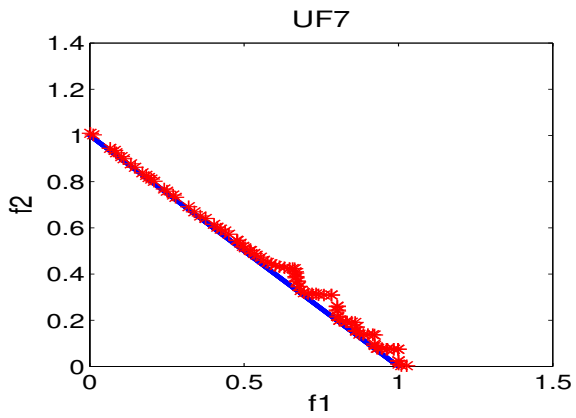

Fig. 7. Nondominated Front obtained for test problem UF7

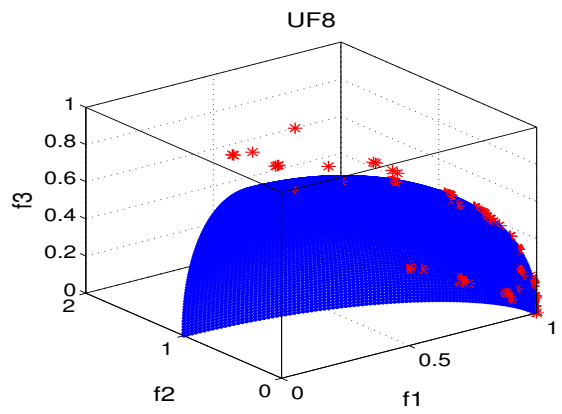

Fig. 8. Nondominated Front obtained for test problem UF8

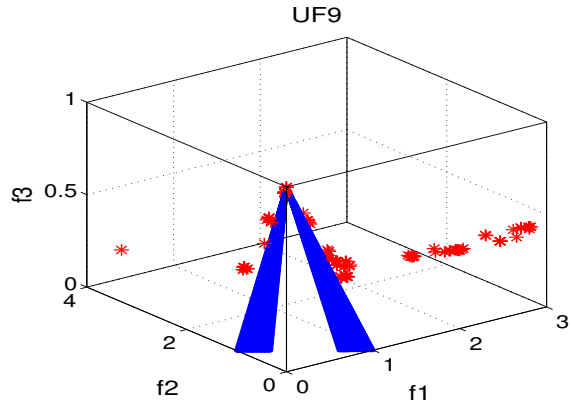

Fig. 9. Nondominated Front obtained for test problem UF9

and worst IGD values of obtained solution set found by the proposed algorithm for UF5 are relatively larger than those found for other two objective function problems, they are still not big because UF5 has a discrete and uniformly distributed Pareto front and it is hard for an algorithm to deal with. Thus the proposed algorithm can find the high quality and well distributed solution sets for two objective function problems. Moreover, it can be seen from Table 2 that the values of standard deviation for all two objective function problems are all very small. This illustrates the proposed algorithm is robust.

For three objective function problems UF8 to UF10, it can be seen from Table 1 that the best, mean and worst IGD values of obtained solution sets found by the proposed algorithm are also relatively small. This indicates that the proposed algorithm can find the high quality and well distributed solution sets in objective space for three objective function problems. Also the values of standard deviation for these problems are very small. Thus the proposed algorithm is robust.

For five objective function problems R2-DTLZ2 and WFG1, it can be seen from Tables 1 and 2 that the best, mean, worst IGD values and standard deviation values of obtained solution sets found by the proposed algorithm are small. Thus the proposed algorithm can find high quality and well distributed solution sets for these two problems. However, it also can be seen from Tables 1 and 2 that the proposed algorithm is difficult to find high quality and well spread solution set for problem R3-DTLZ3.

Figures 1 to 10 demonstrate the non-dominated fronts which were represented by star in objective space and found by the proposed algorithm for two and three objective problems UF1 to UF10, respectively. The true Pareto fronts are represented by dot. It can be seen from these Figures that the proposed algorithm can find well distributed and high quality non-dominated fronts for UF1-UF4, UF7, UF8 and UF10, but can not find well distributed non-dominated fronts only for UF5, UF6 and UF9. Thus the proposed algorithm can find well distributed non-dominated fronts for most test problems.

By summary, the proposed algorithm can find high quality and well distributed solution sets in objective space for almost test problems, and is difficult to find high quality solution sets for two problems UF5 and R3-DTLZ3.

V. Conclusions

In this paper a new clustering multi-objective evolutionary algorithm based on orthogonal and uniform design is proposed. In each generation, the exploration focus is mainly

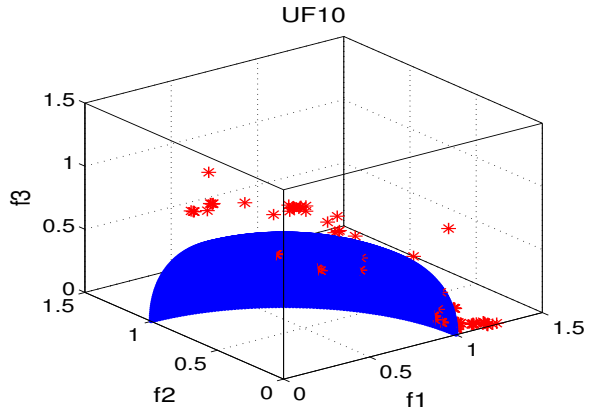

Fig. 10. Nondominated Front obtained for test problem UF10

put on the sparse part and the boundary part of the obtained non-dominated solutions in objective space, and to get desired number of well distributed solutions in objective space, a new clustering method is proposed to select the non-dominated solutions. The experiments on thirteen very difficult benchmark problems were made, and the results indicate the proposed algorithm is efficient.

ACKNOWLEDGMENT

The authors would like to thanks Mr. Hecheng Li, Ms. Lixia Han and Ms. Jingxuan Wei for their help in doing the experiments. This work are supported by the National Natural Science Foundation of China (No.60873099) and SRG: 7002289 of City University of Hong Kong .

REFERENCES

[1] E. Zitzler, K. Deb, and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strenth Pareto approach," *IEEE Transactions on Evolutionary Computation*, Vol.3, No.4, pp.257-271, 1999.

[2] Y.W. Leung and Y.P. Wang, "Multiobjective programming using uniform design and genetic algorithm," *IEEE Trans. Systems, Man, and Cybernetics-Part C: Applications and Reviews*, Vol.30, No. 3, pp. 293-304, 2000.

[3] Y.W. Leung and Y.P. Wang, "U-Measure: A quality measure for multiobjective programming," *IEEE Trans. Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.33, no.3, pp. 337-343, 2003.

[4] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for multi-objective optimization problems," *IEEE Trans. Evolutionary Computation*, vol.10, no. 1, pp. 50-66, Feb. 2006.

[5] K. Deb, M. Mohan and S. Mishra, "Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions," *Evolutionary Computation*, Vol. 13, No. 4, pp.501-525, 2005.

[6] K. C. Tan, C. K. Goh, Y. J. Yang and T. H. Lee, "Evolving better population distribution and exploration in evolutionary multi-objective optimization," *European Journal of Operational Research*, vol.171, no.2, pp.463-495,2006.

[7] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, vol.6,no.2, pp.182-197, 2002.

[8] E. Zitzler, "SPEA-II: improving the Strength Pareto Evolutionary Algorithm," Swiss Federal Institute of Technology, Lausanne, Switzerland. Tech. Rep. TIK-Rep. 103, 2001.

[9] H. Li and Q. Zhang, "Multi-objective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Trans. Evolutionary Computation*, vol.12, no. 2, 2008.

[10] A. Zhou, Q. Zhang and Y. Jin, "Approximating the Set of Pareto Optimal Solutions in Both the Decision and Objective Spaces by an Estimation of Distribution Algorithm," Working Report CES-485, Dept of CES, University of Essex, 06/2008.

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

[12] K.T. Fang and Y. Wang, *Number-Theoretic Method in Statistics*, Chapman and Hall, London, 1994.

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

[14] C. R. Hicks, Fundamental Concepts in the Design of Experiments, 4th ed. TX: Saunders College Publishing, 1993.

[15] Q. Zhang and Y.W. Leung, An orthogonal genetic algorithm for multimedia multicast routing, *IEEE Trans. Evol. Comput.*, vol. 3, pp. 53C62, Apr. 1999.

[16] Y.W. Leung and Y. Wang, An orthogonal genetic algorithm with quantization for numerical global optimization, *IEEE Trans. Evol. Comput.*, vol. 15, no.1, pp. 41-53, Feb. 2001.

[17] Q. Zhang, A. Zhou, S. Zhao, P. Nagaratnam Suganthan, W. Liu and S. Tiwari, Mmulti-objective optimization test Instances for the CEC 2009, Working Report, Department of Computing and Electronic Systems, University of Essex, 2008.

[18] K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, *Evol. Comput.*, vol. 7, no. 3, pp. 205-230, 1999.