Solving Multiobjective Optimization Problem by Constraint Optimization

He Jiang[1,2], Shuyan Zhang[1], and Zhilei Ren[3]

[1] School of Software, Dalian University of Technology
jianghe@dlut.edu.cn, shyzhang@mail.dlut.edu.cn
[2] Department of Computer Science, University of Vermont
[3] School of Mathematical Sciences, Dalian University of Technology
ren@mail.dlut.edu.cn

Abstract. Multiobjective optimization problems (MOPs) have attracted intensive efforts from AI community and many multiobjective evolutionary algorithms (MOEAs) were proposed to tackle MOPs. In addition, a few researchers exploited MOEAs to solve constraint optimization problems (COPs). In this paper, we investigate how to tackle a MOP by iteratively solving a series of COPs and propose the algorithm named multiobjective evolutionary algorithm based on constraint optimization (*MEACO*). In contrast to existing MOEAs, *MEACO* requires no complex selection mechanism or elitism strategy in solving MOPs. Given a MOP, *MEACO* firstly constructs a new COP by transforming all but one of objective functions into constraints. Then, the optimal solution of this COP is computed by a subroutine evolutionary algorithm so as to determine some Pareto-optimal solutions. After that, a new COP with dramatically reduced search space can be constructed using existing Pareto-optimal solutions. This new generated COP will be further solved to find more Pareto-optimal solutions. This process is repeated until the stopping criterion is met. Experimental results on 9 well-known MOP test problems show that our new algorithm outperforms existing MOEAs in terms of convergence and spacing metrics.

Keywords: Multiobjective Optimization, Constraint Optimization, Evo- lutionary Algorithm.

1 Introduction

Multiobjective optimization problems (MOPs) arising from real-world applications have attracted great efforts from AI community. Since those objective functions in MOPs may conflict with each other, researchers usually seek for a set of trade-off solutions (Pareto-optimal solution set) rather than a unique solution. Since evolutionary algorithms (EAs) are capable of handling a set of solutions for complex problems, many multiobjective evolutionary algorithms (MOEAs) have been proposed for MOPs in recent years. Those MOEAs fall into two categories [1]. Algorithms of the first category use Pareto-ranking based selection mechanisms and employ fitness sharing to retain diversity. Some representative algorithms include *VEGA* [2], *MOGA* [3], *NSGA* [4], *NPGA* [5]. Algorithms of the second category employ elitism strategy and some other mechanisms (e.g., the clustering procedure, the nearest neighbor density estimation technique, and the adaptive grid algorithm) to maintain diversity. Some representative algorithms include *SPEA*2 [6], *PAES* [7], *NSGA*-II [8], *PESA*-II [9], *RM-MEDA* [10], and *NNIA* [11].

Constraint optimization problems (COPs) aim to achieve the optimal solution of the objective function under certain constraints, including inequality and equality constraints. Due to the great success of MOEAs, many researchers have exploited MOEAs to solve COPs by transforming a fraction of those constraints in COPs into objective functions. Some representative algorithms include *IS-PAES* [12], *COMOGA* [13], and *HCOEA* [14].

In this paper, we investigate how to solve MOPs in a reverse way by transforming them into a series of COPs. Motivated by this idea, the multiobjective evolutionary algorithm based on constraint optimization (*MEACO*) is proposed. Given a MOP, *MEACO* retains only one objective function and transforms other ones into constraints. In this way, a new COP is constructed. Then, the optimal solution of this COP can be achieved to initialize the Pareto-optimal solution set. With existing Pareto-optimal solutions, another new COP with dramatically reduced search space can be constructed. This new COP can be solved to find more Pareto-optimal solutions. This process is iteratively repeated until no Pareto-optimal solution can be found. To evaluate the performance of *MEACO*, experiments are conducted on 9 widely used MOP test problems. Experiments indicate that *MEACO* can achieve better convergence and spacing metrics than *NSGA*-II, *SPEA*2, and *NNIA*.

2 Notations

Without loss of generality, we consider the minimization problem forms for both MOPs and COPs in this paper. A MOP can be defined as follows.

$$\min F(x) = \left(f_1(x), f_2(x), \ldots, f_m(x)\right)^T \tag{1}$$

s.t. $x = (x_1, x_2, \ldots, x_n) \in \Omega$, where x is the decision vector and Ω is the feasible region in decision space.

Let $x^1, x^2 \in \Omega$, x^1 is said to *d*ominate x^2 (denoted as $x^1 \succ x^2$) iff $\forall i \in \{1, 2, \ldots, m\}$, $f_i(x^1) \leq f_i(x^2)$ and $\exists i^* \in \{1, 2, \ldots, m\}$, $f_{i^*}(x^1) < f_{i^*}(x^2)$. A decision vector x^* is a *Pareto-optimal solution* in (1) iff there is no $x \in \Omega$ such that $x \succ x^*$. The *Pareto-optimal set* is defined as $P^* = \{x^* \in \Omega | \neg\exists x \in \Omega, x \succ x^*\}$. The *Pareto-optimal front* is the image of the Pareto-optimal set in the objective space, which is defined as $PF^* = \{F(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_m(x^*))^T | x^* \in P^*\}$. Under those definitions, the goal is to achieve a set of Pareto-optimal solutions equidistantly approximating the true Pareto-optimal front.

In contrast to MOP, a COP can be defined as follows.

$$\min f(x) \tag{2}$$

s.t. $g_i(x) \leq 0$ ($i \in \{1, 2, \ldots, q\}$) and $h_j(x) = 0$ ($j \in \{1, 2, \ldots, r\}$), where $x = (x_1, x_2, \ldots, x_n) \in R^n$ is a n-dimensional decision vector and $f(x)$ is the objective function. If a decision vector satisfies all the inequality and equality constraints, we say it is a feasible solution. Otherwise, it is an infeasible solution.

There are many metrics for evaluating the performance of MOEAs in the literature. In this paper, we adopt two widely used metrics, i.e., convergence metric [15] and spacing metric [16].

Convergence metric: let $P^* = \{p^1, p^2, p^3, \ldots, p^{|P^*|}\}$ be the target set of points on the true Pareto-optimal front and $P = \{x^1, x^2, x^3, \ldots, x^{|P|}\}$ be the solution set by an algorithm. For every solution $x^i \in P$, the smallest normalized Euclidean distance to P^* is defined as

$$d_i = \min_{j=1}^{|P^*|} \sqrt{\sum_{k=1}^{m} \left(\frac{f_k(x^i) - f_k(p^j)}{f_k^{max} - f_k^{min}} \right)^2} \tag{3}$$

wheref_k^{max} and f_k^{min} are the maximum and minimum values of the k^{th} objective function in P^*, respectively. Then convergence metric is defined as the average value of those normalized distance for all solutions in P, i.e., $Con(P) = \sum_{i=1}^{|P|} d_i/|P|$. It indicates the extent to which P approximates the true Pareto-optimal front. The smaller $Con(P)$ is, the better an algorithm is.

Spacing metric: let $P = \{x^1, x^2, x^3, \ldots, x^{|P|}\}$ be the solution set by an algorithm. Let $\hat{d}_i = \min_{j=1}^{|P|} \sum_{k=1}^{m} |f_k(x^i) - f_k(x^j)|$ for every $x^i \in P$. Let $\bar{d}$ be the average value of $\hat{d}_i$. Spacing metric is defined as follows.

$$S(P) = \sqrt{\frac{1}{|P| - 1} \sum_{i=1}^{|P|} \left(\bar{d} - \hat{d}_i \right)^2} \tag{4}$$

Spacing metric presents the spread measure of the solution set obtained by an algorithm. The smaller this value is, the better distribution an algorithm provides. This value is zero when all the solutions in P are equidistantly spaced.

3 MEACO

Due to the paper length limit, we will explain our algorithm for bi-objective optimization problem in this paper.

3.1 An Example

Fig.1 presents an example for a bi-objective optimization problem. We transform the second objective function f_2 into a constraint. Given a value f^*, there may exists several solutions $x^1, x^2, x^3, \ldots, x^w$ such that $f_2(x^i) = f^* (i \in \{1, 2, \ldots, w\})$ and $f_1(x^1) \leq f_1(x^2) \leq f_1(x^3) \leq \ldots \leq f_1(x^w)$ (see Fig.1 (a)). For brevity, we only plot the pixel $(f^*, f_1(x^1))$ in Fig.1.

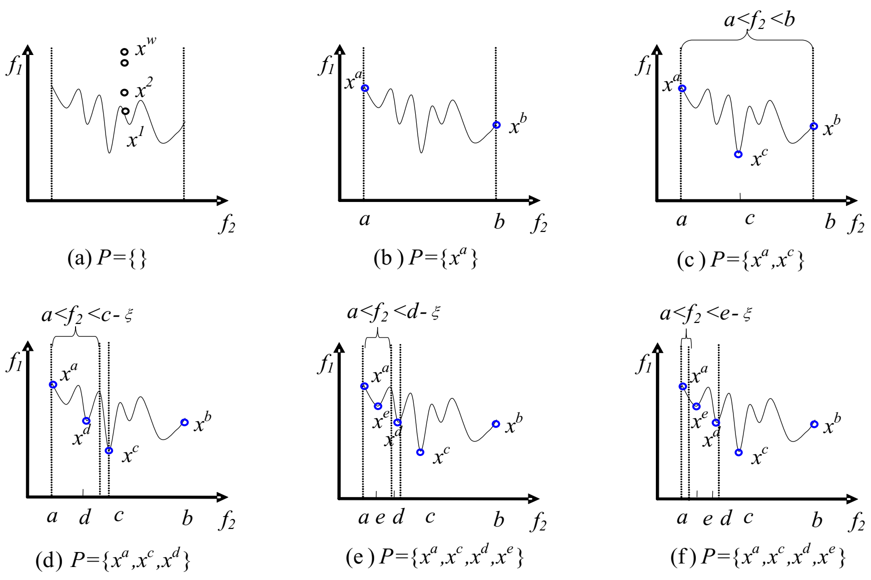

Fig. 1. Illustration of *MEACO*

For this example, our algorithm works as follows. Firstly, the minimum and maximum values (denoted as a and b, respectively) of f_2 are calculated (see Fig.1 (b)). Obviously, this computation for a can be done by solving a single-objective optimization problem (SOP) $\min f_2(x)$ s.t. $x \in \Omega$. Similarly, the computation for b is equivalent to solving a SOP $\max f_2(x)$ s.t. $x \in \Omega$. According to the definition of Pareto-optimal solution, the solution x^a must be Pareto-optimal and be added to the solution set P. A COP is then constructed as $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$. Let x^c be the optimal solution to this new COP, let $c = f_2(x^c)$ (see Fig.1 (c)). Obviously, x^b is dominated by x^c. It can be easily verified that x^c is Pareto-optimal, and no Pareto-optimal solution exists when $f_2(x) > c$. Therefore, the solution set P is updated by adding x^c to it. Then, we continue to construct a new COP as $\min f_1(x)$ s.t. $a < f_2(x) < c - \xi$, where $x \in \Omega$ and ξ is a predefined parameter aiming to keep the spacing of P. This latest constructed COP is solved to achieve the optimal solution x^d (see Fig.1 (d)). Similar to x^c, we can verify that x^d is Pareto-optimal and P is further updated. Since no other Pareto-optimal solution exists when $d < f_2(x) < c - \xi$, we further solve the COP $\min f_1(x)$ s.t. $a < f_2(x) < d - \xi$, where $x \in \Omega$(see Fig.1 (e)). This process is repeated until the stopping criterion is met. Usually, the stopping criterion is determined by spacing metric.

3.2 Framework of MEACO

Table 1 presents the framework of our *MEACO* algorithm. Firstly, the minimum value and maximum values of $f_2(x)$ are achieved by solving SOPs $\min f_2(x)$ or

$\mathbf{max} f_2(x)$ s.t. $x \in \Omega$ (see Step (1)). When multiple solutions $x^1, x^2, x^3, \ldots, x^w$ can be returned, the only solution x^i is retained such that $f_1(x^i)$ is minimal among $f_1(x^1), f_1(x^2), f_1(x^3), \ldots, f_1(x^w)$. By this way, we have two solutions x^a and x^b, where $a = f_2(x^a), b = f_2(x^b)$ are the minimum, maximum values for $f_2(x)$, respectively. Obviously, x^a is Pareto-optimal to be added to the Pareto-optimal solution set P (see Step (2)). It's still uncertain at this time whether x^b is Pareto-optimal or not. Then, a COP is constructed as $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$. After the optimal solution x^c is obtained (see Step (3)), we can decide whether x^b is Pareto-optimal or not. There're two cases as follows.

Case 1: x^b is Pareto-optimal (see Step (4.1)–(4.5))

After x^b is added to P (see Step (4.1)), we compare x^a and x^c to check whether x^c is dominated by x^a. If so, no Pareto-optimal solution exists under

Table 1. *MEACO* algorithm

```
Algorithm: MEACO
Input: $f_1, f_2$
Output: solution set $P$
Begin
(1) obtain the minimum value $a$, maximum value $b$ of $f_2$, let $x^a$, $x^b$ be the solution
    related to $a, b$, respectively
(2) let $P = \{x^a\}$ // $x^a$ must be Pareto-optimal
(3) obtain the optimal solution $x^c$ to $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$, let
    $c = f_2(x^c)$
(4) if $f_1(x^b) < f_1(x^c)$ and $f_1(x^b) < f_1(x^a)$ then //$x^b$ is Pareto-optimal
    (4.1) $P = P \cup \{x^b\}, \delta = (|f_1(x^a) - f_1(x^b)| + |b - a|)/200$
    (4.2) if $f_1(x^a) < f_1(x^c)$ then return $P$ //$x^c$ isn't Pareto-optimal
    (4.3) if $|f_1(x^a) - f_1(x^c)| + |c - a| > \delta$ and $|f_1(x^c) - f_1(x^b)| + |b - c| > \delta$ then
        (4.3.1) $P = P \cup \{x^c\}$
        (4.3.2) $P' = IntervalOpt(a, c - \xi, x^a, x^c, \delta, \xi)$
        (4.3.3) $P = P \cup P'$
        (4.3.4) return $P$
    (4.4) if $|f_1(x^a) - f_1(x^c)| + |c - a| \leq \delta$ then return $P$;
    (4.5) if $|f_1(x^c) - f_1(x^b)| + |b - c| \leq \delta$ then
        (4.5.1) $P' = IntervalOpt(a, c - \xi, x^a, x^b, \delta, \xi)$
        (4.5.2) $P = P \cup P'$
        (4.5.3) return $P$
    else // $x^b$ isn't Pareto-optimal
    (4.6) if $f_1(x^a) < f_1(x^c)$ then return $P$ //$x^c$ isn't Pareto-optimal
    (4.7) if $|f_1(x^a) - f_1(x^c)| + |c - a| \leq \delta$ then return $P$
        else
        (4.7.1) $P = P \cup \{x^c\}, \delta = (|f_1(x^a) - f_1(x^c)| + |c - a|)/200$
        (4.7.2) $P' = IntervalOpt(a, c - \xi, x^a, x^c, \delta, \xi)$
        (4.7.3) $P = P \cup P'$
        (4.7.4) return $P$
End
```

the constraint $a < f_2(x) < b$ and the current solution set P is then returned (see Step (4.2)). Otherwise, we need to further check if x^c satisfies the spacing requirement that the distance (in the objective space) between any two solutions in P must be greater than a threshold δ. If the spacing requirement is satisfied, x^c can be added to P and we continue to obtain Pareto-optimal solutions under the constraint $a < f_2(x) < c-\xi$ by calling a subroutine function *IntervalOpt* (see Step (4.3)), where ξ(ξ is set to be δ /10) is introduced to avoid endless loops when the Pareto-optimal front is smoothly continuous. If the spacing requirement is not met, there may be two possible reasons.

For the first reason that x^c is too close to x^a (see Step (4.4)), it's unnecessary to further investigate those solutions under either the constraint $c+\xi < f_2(x) < b$ or the constraint $a < f_2(x) < c - \xi$. On the one hand, every solution under the constraint $c + \xi < f_2(x) < b$ is dominated by x^c. On the other hand, every solution under the constraint $a < f_2(x) < c - \xi$ is either dominated by x^a or closer to x^a than x^c.

For the second reason that x^c is too close to x^b (see Step (4.5)), we need to further achieve those solutions under the constraint $a < f_2(x) < c - \xi$ by calling the subroutine function *IntervalOpt* (see Step (4.5.1)–(4.5.3)). It's now unnecessary to consider those solutions under the constraint $c + \xi < f_2(x) < b$, since all such solutions are dominated by x^c.

Case 2: x^b isn't Pareto-optimal (see Step (4.6)–(4.7))

In this case, x^b will be dominated by x^c or x^a. We compare x^a and x^c to check whether x^c is dominated by x^a. If so, no Pareto-optimal solution exists under the constraint $a < f_2(x) < b$ and P is then returned (see Step (4.6)). Otherwise, we continue to check whether x^c satisfies the spacing requirement or not (see Step (4.7)). The following work (Step (4.7.1)–(4.7.4)) can be explained in a similar way as Step (4.3.1)–(4.3.4).

3.3 IntervalOpt

In *MEACO*, a subroutine named *IntervalOpt* is called to achieve the Pareto-optimal solution set under certain constraints. As shown in Tab.2, *IntervalOpt* takes in several parameters, including a, b, x^1, and x^2, where a and b impose the constraint $a < f_2(x) < b$, x^1 and x^2 provide the spacing requirements. *IntervalOpt* consists of 5 steps.

In Step (1), we achieve the optimal solution x^c to the COP $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$.

Then we check whether x^c is dominated by x^1 in Step (2). If so, it can be verified that no other Pareto-optimal solution exists under the constraint $a < f_2(x) < b$. Therefore, an empty solution set will be returned. It should be noted that we needn't to compare x^c with x^2, since $f_2(x^c) < b \leq f_2(x^2)$ implies that x^c will never be dominated by x^2.

When x^c is Pareto-optimal, *InterverOpt* continues to check if x^c is too close to x^1 in Step (3). If so, no other Pareto-optimal solution can be found under the constraint $a < f_2(x) < b$, due to the same reason as Step (4.4) in *MEACO*. Otherwise, *IntervalOpt* continues to check whether it's too close to x^2 (see Step

Table 2. *IntervalOpt* algorithm

```
Algorithm: IntervalOpt
Input: $a, b, x^1, x^2, \delta, \xi$
Output: solution set $P$
Begin

(1) obtain the optimal solution $x^c$ to $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$, let
    $c = f_2(x^c)$
(2) if $x^c$ is dominated by $x^1$ then return $\phi$
(3) if $|f_1(x^c) - f_1(x^1)| + |c - f_2(x^1)| \leq \delta$ then return $\phi$
(4) if $|f_1(x^c) - f_1(x^2)| + |c - f_2(x^2)| \leq \delta$ then
    (4.1) $P = IntervalOpt(a, c - \xi, x^1, x^2, \delta, \xi)$
    (4.2) return $P$
(5) if $|f_1(x^c) - f_1(x^1)| + |c - f_2(x^1)| > \delta$ and $|f_1(x^c) - f_1(x^2)| + |c - f_2(x^2)| > \delta$ then
    (5.1) $P = \{x^c\}$
    (5.2) $P' = IntervalOpt(a, c - \xi, x^1, x^c, \delta, \xi)$
    (5.3) $P = P \cup P'$
    (5.4) return $P$

End
```

(4)). If so, the Pareto-optimal solutions will be further achieved by recursively calling *IntervalOpt* with more restrictive constraints (see Step (4.1)–(4.2)).

Finally, when x^c satisfies all spacing requirements, x^c will be added to P and *IntervalOpt* recursively obtains new Pareto-optimal solutions with more restrictive constraints (see Step (5)).

3.4 Solving COPs with EAs

As shown in both *MEACO* and *IntervalOpt*, we need to solve some COPs transformed from the original MOPs. (1) In Step (1) of *MEACO*, we need to compute the minimum and maximum values of f_2. (2) In Step (3) of *MEACO*, we need to compute the optimal solution x^c to $\min f_1(x)$, s.t. $a < f_2(x) < b$. (3) In Step (1) of *IntervalOpt*, we also need to compute the the optimal solution x^c to $\min f_1(x)$, s.t. $a < f_2(x) < b$.

Although any existing algorithm for COPs can be incorporated into *MEACO*, we employ EAs to tackle those COPs in this paper. In contrast to other algorithms, EAs are more widespread with no restriction on COPs. Due to the paper length limit, all the details of those EAs are presented in our technical report http://www.cems.uvm.edu/∼hejiang/MEACO-tech.pdf.

4 Experimental Results

To evaluate our algorithm, experiments are conducted on 9 well-known bi-objective optimization problems, including SCH, FON, POL, KUR, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. In the experiment, *MEACO* is coded in Microsoft

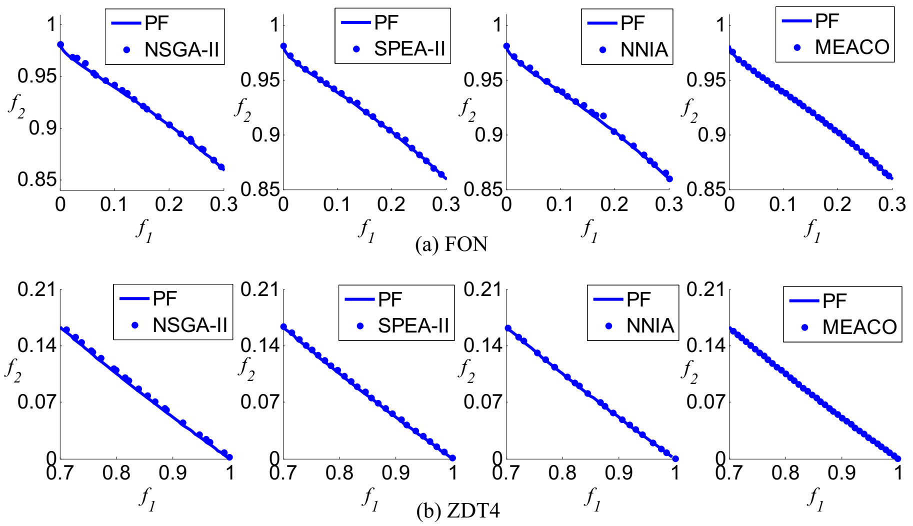

Fig. 2. Pareto-optimal front of algorithms

VC++ 6.0 and run on a Pentium Dual Core 2.8G with 4G memory running Win XP. For comparison, we also run *NSGA*-II, *SPEA*2, and *NNIA* on the same test problems. Those source codes of *NSGA*-II and *SPEA*2 are downloaded from (http://www.lania.mx/∼ccoello/EMOO/). The source code of *NNIA* is obtained from (http://see.xidian.edu.cn/iiip/mggong/Projects/NNIA.htm). All the parameters for *NSGA*-II, *SPEA*2, and *NNIA* are also from [11].

In this conference paper, we only plot part of the Pareto-optimal front of algorithms on FON (see Fig.2(a)) and ZDT4 (see Fig.2(b)). Similar results can be concluded for other MOPs. It can be observed that all the algorithms can well approximate the true Pareto-optimal fronts on both FON and ZDT4. Out of all 4 algorithms, our new algorithm achieves solutions which are the most equidistantly distributed on the true Pareto-optimal fronts on both FON and ZDT4.

Table 3 present the numerical results of convergence, spacing and time metrics on all MOPs. For every MOP, the averaged values over 30 runs are given and the best ones are marked in bold font. It can be observed from Table 3 that *MEACO* outperforms other algorithms on all MOPs except POL and ZDT6 ni terms of convergence metric, while SPEA2 achieves the best convergence metric on POL and *NSGA*-II achieves the best on ZDT6. Table 3 illustrates that *MEACO* could achieve better spacing metrics on 6 MOPs out of all MOPs.From Table 3, we can observe that NSGA-II uses least time for all test problems among those 4 algorithms.The running of our algorithm is similar to that of *SPEA*2.

Table 3. Convergence spacing and time results of algorithms on MOPs

	NSGA			*SPEA2*			*NNIA*			*MEACO*		
	Conv.	Spac.	Time	Conv.	Spac.	Time	Conv.	Spac.	Time	Conv.	Spac.	Time
SCH	0.0020	0.0193	**0.0373**	0.0020	**0.0147**	0.1564	0.0021	0.0272	0.0646	**0.0019**	0.0209	0.0380
FON	0.0034	0.0068	**0.0414**	0.0025	0.0029	0.1826	0.0031	0.0060	0.1033	**0.0020**	**0.0019**	0.1568
POL	0.0017	0.1002	**0.0418**	**0.0015**	**0.0394**	0.1715	0.0016	0.0916	0.0929	0.0022	0.0719	0.2017
KUR	0.0022	0.1016	**0.0416**	**0.0018**	0.0823	0.1744	0.0020	0.0878	0.0978	**0.0018**	**0.0814**	0.1347
ZDT1	0.0024	0.0068	**0.0401**	0.0022	0.0034	0.1240	0.0020	0.0072	0.0648	**0.0019**	**0.0018**	0.1475
ZDT2	0.0020	0.0072	**0.0382**	0.0019	0.0032	0.1445	**0.0018**	0.0073	0.0648	**0.0018**	**0.0016**	0.1600
ZDT3	0.0019	0.0077	**0.0360**	0.0019	**0.0039**	0.1589	0.0018	0.0079	0.0634	**0.0016**	0.0101	0.0878
ZDT4	0.0031	0.0075	**0.0361**	0.0032	0.0028	0.2729	0.0027	0.0069	0.0643	**0.0020**	**0.0021**	0.5304
ZDT6	**0.0019**	0.0075	**0.0450**	0.0020	0.0023	0.0805	0.0020	0.0056	0.0686	0.0030	**0.0013**	0.1015

5 Conclusion and Future Work

In this paper, a new algorithm *MEACO* is proposed to solve MOPs by transforming them into a series of COPs. Several EAs are also presented to tackle those transformed COPs. Experimental results indicate that our algorithm is a promising way to tackle MOPs. In contrast to solving COPs with MOEAs, work of this paper throws a light on how to solve MOPs by methods for COPs. In future work, we will extend *MEACO* for MOPs with more objective functions. It's also interesting to incorporate some other efficient algorithms (for COPs) into *MEACO*.

Acknowledgement

Our work is partially supported by the Natural Science Foundation of China under Grant No. 60805024, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20070141020.

References

1. Coello Coello, C.A.: Evolutionary Multi-Objective Optimization: A Historical View of the Field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006)
2. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proc. of the 1st Int'l Conf. on Genetic Algorithms, pp. 93–100. L. Erlbaum Associates, Inc., Hillsdale (1985)
3. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Proc. of the 5th Int'l Conf. on Genetic Algorithms, pp. 416–423. Morgan Kauffman, San Mateo (1993)
4. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
5. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proc. of the 1st IEEE Conference on Evolutionary Computation, pp. 82–87. IEEE, Piscataway (1994)

6. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, pp. 95–100. Springer, Berlin (2001)
7. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (2002)
9. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization. In: Proc. of the Genetic and Evolutionary Computation Conf., pp. 283–290. Morgan Kaufmann Publishers, San Francisco (2001)
10. Zhang, Q.F., Zhou, A.M., Jin, Y.: RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm. IEEE Trans. on Evolutionary Computation 11(1), 41–63 (2007)
11. Gong, M.G., Jiao, L.C., Du, H.F., Bo, L.F.: Multiobjective Immune Algorithm with Nondominated Neighbor-Based Selection. Evolutionary Computation 16(2), 225–255 (2008)
12. Aguirre, A.H., Rionda, S.B., Coello Coello, C.A., Lizrraga, G.L., Montes, E.M.: Handling Constraints Using Multiobjective Optimization Concepts. Int'l Journal for Numerical Methods in Engineering 59(15), 1989–2017 (2004)
13. Surry, P.D., Radcliffe, N.J.: The COMOGA Method: Constrained Optimization by Multi-Objective Genetic Algorithms. Control and Cybernetics 26(3), 391–412 (1997)
14. Wang, Y., Cai, Z.X., Guo, G.Q., Zhou, Y.R.: Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems. IEEE Trans. on System, Man And Cybernetics 37(3), 560–575 (2007)
15. Deb, K., Jain, S.: Running Performance Metrics for Evolutionary Multi-Objective Optimization, Technical Report. Kanpur: Indian Institute of Technology Kanpur. NO. 2002004 (2002)
16. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA (1995)