Multiobjective Bacterial Foraging Optimization using Archive Strategy

Cuicui Yang and Junzhong Ji
College of Computer, Beijing University of Technology, Beijing Municipal Key Laboratory of Multimedia and Intelligent Software, Pingleyuan 100, Chaoyang District, Beijing, China

Keywords: Bacterial Foraging Optimization, Archive Strategy, Conjugation, Multiobjective Optimization.

Abstract: Multiobjective optimization problems widely exist in engineering application and science research. This paper presents an archive bacterial foraging optimizer to deal with multiobjective optimization problems. Under the concept of Pareto dominance, the proposed algorithm uses chemotaxis, conjugation, reproduction and elimination-and-dispersal mechanisms to approximate to the true Pareto fronts in multiobjective optimization problems. In the optimization process, the proposed algorithm incorporates an external archive to save the nondominated solutions previously found and utilizes the crowding distance to maintain the diversity of the obtained nondominated solutions. The proposed algorithm is compared with two state-of-the-art algorithms on four standard test problems. The experimental results indicate that our approach is a promising algorithm to deal with multiobjective optimization problems.

1 INTRODUCTION

The multiobjective optimization problem (MOP) usually involves more than one conflicting objectives that need to be optimized simultaneously and is an important class of scientific and engineering problems in real-world (Deb, 2001). The solution to a MOP is a set of trade-off solutions known as Pareto optimal solutions or non-dominated solutions which cannot improve all the objectives simultaneously. Evolutionary computation methods based on Darwin's biological evolution theory deal with a group of candidate solutions, which makes that they are natural to be used to handle multiobjective optimization problems (MOPs). In the past two decades, researchers have proposed a variety of evolutionary computation-based approaches for solving MOPs (Deb, 2001; Coello Coello CA, Van Veldhuizen DA, Lamont GB, 2007), such as the well-known algorithms PESA-II (Corne D W, Jerram N R, Knowles J D, et al, 2001), NSGA-II (Deb K, Pratap A, Agarwal S, et al, 2002) and SPEA2 (E Zitzler, M Laumanns, L Thiele, 2002). Evolutionary multiobjective optimization (EMO) that uses evolutionary computation methods to solve MOPs has become a relatively hot research area.

In recent years, some new bio-inspired optimization technologies have been successfully introduced to deal with MOPs, where Particle swarm optimization (PSO) is a prominent example. So far, there have been a number of methods based on PSO for solving MOPs (X Li, 2003; Coello C A C, Pulido G T, Lechuga M S, 2004; Tripathi P K, Bandyopadhyay S, Pal S K, 2007). Bacterial foraging optimization (BFO) is another popular bio-inspired optimization technology which simulates the foraging behavior of E. coli bacteria (K.M. Passino, 2002). BFO has been proved to be an efficient optimization method for single objective optimization problems (Agrawal V, Sharma H, Bansal J C, 2011), and more recently researchers have also shown promising results for MOPs (Panigrahi B K, Pandi V R, Das S, et al, 2010; Guzmán M A, Delgado A, De Carvalho J, 2010; Niu B, Wang H, Tan L, et al, 2012; Niu B, Wang H, Wang J, et al, 2013). However, to the best of our knowledge, none of these algorithms incorporate an external archive to preserve the elitism solutions, namely, there is no way to keep the non-dominated solutions found previously in the optimization process. Elitism-preservation is an importance strategy in multiobjective search, which has been recognized and supported experimentally (Parks G T, Miller I, 1998; Zitzler E, Deb K, Thiele L, 2000). To further explore the potential of BFO algorithm in finding Pareto optimal solutions for MOPs, this paper presents an archive bacterial foraging optimizer for multiobjective optimization, called as MABFO. MABFO mainly includes four optimization mechanisms: chemotaxis, conjugation, reproduction and elimination-and-dispersal. The implements of these

Yang, C. and Ji, J.
Multiobjective Bacterial Foraging Optimization using Archive Strategy.
DOI: 10.5220/0005668601850192
In *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2016)*, pages 185-192
ISBN: 978-989-758-173-1

mechanisms are different from other current methods based on BFO. More improtant, MABFO incorporates an external archive to save the nondominated solutions previously found and uses the crowding distance to maintain the diversity of the nondominated solutions. These strategies are conducive to produce well-distributed and high-quality solutions.

MABFO is validated on four standard test problems and compared against two start-of-the-art EMO approaches NSGA-II (Deb K, Pratap A, Agarwal S, et al, 2002) and SPEA2 (E Zitzler, M Laumanns, L Thiele, 2002). The experimental results indicate that MABFO is a promising algorithm and can be considered an alternative to deal with MOPs.

The remainder of the paper is structured as follows. Section II briefly introduces basic concepts involving MOPs. Section III presents the details of the MABFO algorithm. We then present experiments in section IV, and finally, section V concludes the paper.

2 BASIC CONCEPTS

A MOP can be described as the problem of finding a vector of decision variables that optimizes a vector function and satisfies some restrictions. Without loss of generality, a MOP is formulated as follows (Deb, 2001):

$$\begin{cases} min & y = F(x) = (f_1(x), f_2(x), \cdots, f_m(x))^T \\ s.t. & g_i(x) \leq 0, i = 1,2,\cdots,p \\ & h_j(x) = 0, j = 1,2,\cdots,q \\ & x_i^L \leq x_i \leq x_i^U \end{cases}, \quad (1)$$

where $x = (x_1, x_2, \cdots, x_n) \in X \subset R^n$ is a n-dimensional decision vector, X represents a n-dimensional decision space, x_i^L and x_i^U are the upper and lower boundary values of x_i, respectively. $y = (y_1, y_2, \cdots, x_m) \in Y \subset R^m$ is a m-dimensional objective vector, Y represents a m-dimensional objective space. $F(x)$ is a mapping function from n-dimensional decision space to m-dimensional objective space. $g_i(x) \leq 0$ $(i = 1,2,\cdots,p)$ and $h_j(x) = 0$ $(j = 1,2,\cdots,q)$ defines p inequalities and q equalities, respectively.

In the following, we will list four definitions involving MOPs.

Pareto Dominant: x_α, x_β are two feasible solutions for problem (1), x_α is Pareto dominant compared with x_β if and only if:

$$\begin{array}{l} \forall i = 1,2,\cdots,m, f_i(x_\alpha) \leq f_i(x_\beta) \wedge \\ \exists j = 1,2,\cdots,m, f_i(x_\alpha) < f_i(x_\beta) \end{array}. \quad (2)$$

We call this relationship $x_\alpha \succ x_\beta$, x_α dominate x_β, or x_β is dominated by x_α.

Pareto Optimal Solution: Ω is the feasible solution set of problem (1), $x^* \in \Omega$, x^* is a Pareto optimal solution if and only if:

$$\neg\exists x \in \Omega : x \succ x^*. \quad (3)$$

Pareto Optimal Set: The Pareto optimal set of problem (1) includes all the Pareto optimal solutions and is given as follows:

$$X^* = \{x^* | \neg\exists x \in \Omega : x \succ x^*\}. \quad (4)$$

Pareto Front: The Pareto front of problem (1) includes all the objective vectors corresponding to X^* and is given as follows:

$$PF = \{F(x_*) = (f_1(x_*), f_2(x_*), \cdots, f_m(x_*))^T | x_* \in X^*\}. \quad (5)$$

3 THE MABFO ALGORITHM

In this section, we will introduce MABFO algorithm. Algorithm 1 is the framework of MABFO. In the following, we will describe its five important operators chemotaxis, archive updating, conjugation, reproduction and elimination-and-dispersal as Algorithm 1 shown.

Algorithm 1: MABFO algorithm (main loop).

Input: Different parameters:
N_c: maximum number of chemotaxis,
N_{re}: maximum number of reproduction,
N_{ed}: maximum number of elimination-and-dispersal,
$N1$: the size of population P,
$N2$: the maximum size of external archive A.
Output: A (Pareto optimal set)
1: **Initialization:** Generate an initial population P and an empty archive A.
2: **for** $l = 1$ to N_{ed} **do**
3: **for** $k = 1$ to N_{re} **do**
4: **for** $j = 1$ to N_c **do**
a). Each bacterium in P takes a **chemotaxis** process.
b). **Archive updating:** copy all nondominated individuals in joint population of A and P to **A**. If the size of A exceeds $N2$, then reduce it based on the **crowding distance**.
c). Each bacterium takes a **conjugation** process.
5: **end for**
6: The population P perform a **reproduction** process which selects $N1$ superior individuals in the joint population P and A and then copy them to P.
7: **end for**
8: The population P perform an **elimination-and-dispersal** process and update the archive A as step 4 b).
9: **end for**
10: Return A.

3.1 Chemotaxis

Chemotaxis simulates the foraging movement of E.coli bacteria through tumbling and swimming. To absorb more nutrients, each bacterium tries to find food in two ways: tumbling and swimming. A bacterium tumbles in a random direction to exploratively search for food. If the food is rich in the selected direction, the bacterium will swim along this direction, till the food gets bad or the bacterium has swum the fixed steps.

A random direction is a random unit vector in n-dimensional space. In the original BFO algorithm, it is very cumbersome to calculate a random unit vector for each bacterium in every chemotaxis. In the proposed MABFO algorithm, we use a simple unit vector e_m with n dimensions as the chemotaxis direction, in which only one component randomly selected (i.e. the mth component) is either -1 or 1 and all the others are 0. The method of updating a solution is given as follows:

$$x^i(j+1,k,l) = x^i(j,k,l) + c(i)\ e_m, \tag{6}$$

where $x^i(j,k,l)$ represents the ith bacterium at jth chemotaxis, kth reproduction, lth elimination-and-dispersal step. $c(i)$ is the step size along the direction e_m. The original BFO algorithm uses a fixed step size which would lead to bad convergence performances (S Dasgipta, S Das, A Abraham and A Biswas, 2009). In the proposed MABFO algorithm, a dynamic step size is adopted and is given as follows:

$$c(i) = r\ (x^{i'}_m(j,k,l) - x^i_m(j,k,l)), \tag{7}$$

where r is a random number uniformly distributed between -1 and 1, $x^{i'}_m(j,k,l)$ denotes the mth component of another different bacterium i' which is selected randomly in P.

When bacterium i carries out a chemotaxis operator, it first generates a direction e_m and a step size $c(i)$ as described above, and then swims a step along the direction e_m according to Eq.(6). If the new solution $x^i(j+1,k,l)$ dominates the old solution $x^i(j,k,l)$, bacterium i will swim another step in the direction e_m according to Eq.(6). This process is continued until bacterium i has swum the maximum steps N_s or the obtained new solution $x^i(j+1,k,l)$ is dominated by the old solution $x^i(j,k,l)$. Such chemotaxis mechanism is an important driving force for locally optimizing each candidate solution, where each bacterium ties its best to search for non-dominated solutions.

3.2 Archive Updating

The main goal of the external archive is to keep a historical record of the nondominated solutions obtained in the search process. In MABFO, we use an external archive A with a fixed number $N2$. Whenever the population P carries out a chemotaxis process, we will reselect all the nondominated solutions from the joint population of P and A, and then update archive A by copying them to A. If the size of A exceeds $N2$, an archive truncation procedure is invoked, which iteratively remove individuals from A based on the crowding distance until the size of A is $N2$. At each iteration, the individual which has the minimum distance to another individual is removed according to the formula:

$$\begin{aligned} &dis(x_i) < dis(x_j) \ \Leftrightarrow \\ &\forall\ 0 < k < |Q| : \sigma^k_i = \sigma^k_j \ \vee \\ &\exists\ 0 < k < |Q| : [(\forall 0 < l < k : \sigma^l_i = \sigma^l_j) \wedge \sigma^k_i < \sigma^k_j] \end{aligned} \tag{8}$$

where Q is a group of individuals, $|Q|$ denotes the size of Q and here $Q = A$. σ^k_i is the distance of ith solution x_i to its kth nearest neighbor in Q.

The external archive A promises not to miss the nondominated solutions found in the search process. The crowding distance eliminates the individuals in dense area, which can ensure the obtained solutions distributed evenly. The using of external archive and crowding distance is beneficial to find well-distributed Pareto solutions.

3.3 Conjugation

Conjugation, as well as chemotaxis, is an important biological behavior of bacteria. Bacterial conjugation is the transfer of part of plasmid (genetic material) from donor bacteria to recipient bacteria by directly physical contact and is often regarded as the sexual reproduction or mating between bacteria. Some researchers have taken the bacterial conjugation as a message passaging mechanism in their work (C Perales-Graván, R Lahoz-Beltra, 2008; A Balassubramaniam, Memeber, IEEE, P Lio, 2013).

In this paper, we also simulate the bacterial conjugation behavior as an information exchange mechanism between bacteria in population P and archive A. To model this biological behavior, we first define conjugation length L $(L < n)$ measured by the number of decision variables. Then each bacterium i in population P will randomly select another bacterium i' in A and a conjugation point Bt $(Bt < n - L)$ to take a conjugation step. In this step, the way of updating the solution is given as follows:

$$x^i_{new}(j,k,l) = x^i(j,k,l) + \omega \circ (x^{i'}(j,k,l) - x^i(j,k,l)), \tag{9}$$

where $x^i_{new}(j,k,l)$ denotes the new solution after bacterium i performing the conjugation operator. ω is

Table 1: Test problems used in this paper.

Problem	n	Variable range	Objective functions	Pareto-optimal solutions
ZDT1	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0, i = 2,\cdots,n$
ZDT2	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-(x_1/g(x))^2]$ $g(x) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0, i = 2,\cdots,n$
ZDT3	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{x_1/g(x)} - \frac{x_1}{g(x)} sin(10\pi x_1)]$ $g(x) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0, i = 2,\cdots,n$
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$, $i = 2,\cdots,n$	$f_1(x) = x_1$ $f_2(x) = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1+10(n-1)+\sum_{i=2}^{n}[x_i^2 - 10cos(4\pi x_i)]$	$x_1 \in [0,1]$ $x_i = 0, i = 2,\cdots,n$

a n-dimensional random vector in which the values of components Bt to $Bt+L-1$ are uniformly distributed random numbers between -1 and 1, and those of other components are 0. "$\circ$" is an operator, called Hadamard product, which represents multiplying the corresponding elements of two vectors. Take 3-dimensional vectors for an example, if $a = (a_1, a_2, a_3)$, $b = (b_1, b_2, b_3)$, then $D = A \circ B = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3)$. If the new solution $x^i_{new}(j,k,l)$ is dominated by the old solution $x^i(j,k,l)$, $x^i(j,k,l)$ is kept; otherwise, $x^i_{new}(j,k,l)$ replaces $x^i(j,k,l)$. With this communication mechanism, each bacterium in the population P searches for the nondominated solutions under the guidance of the superior individuals in the archive A. Thus, the bacterial population are likely to quickly converge to the global Pareto front.

3.4 Reproduction

The bacteria grow longer with the increasing of the nutrients absorbed. The more nutrients a bacterium gets, the healthier it is. Under appropriate conditions, some of bacteria in a population who are healthy enough will asexually split into two bacteria, and the other ones will die. Essentially, the reproduction mechanism is to generate new population based on the superior individuals in current population. In the proposed MABFO algorithm, the reproduction operator selects $N1$ superior individuals from the joint population of the current population P ($|P| = N1$) and the external archive A and produces a new population P by copying these selected $N1$ superior individuals to P. The way to select $N1$ superior individuals is as follows: first sort the joint population of P and A into different nondominated levels as reference (**?**) did. The first nondominated level F_1 contains all the current nondominated solutions. Then we pick out solutions in ascending order of the nondominated level to form a new group R until the size of R would be equal or larger than $N1$ if we incorporate the tth nondominated level F_t. In the first case, the size of R is exactly equal to $N1$ if putting F_t into R, just put it into R and this select step is completed. In the second case, the size of R is larger than $N1$ if F_t is loaded into R. In this case, we need to first pick out $N1 - |R|$ solutions from F_r based on the crowding distance according to (8) and then put them into R. In this way, $|R|$ is also exactly the same with $N1$. Next, R is used to update the old population P by copying it to P. Thus, the new population P which contains the best individuals is obtained. Such reproduction mechanism follows the rule of survival of the fittest, which plays a role of transmitting good information among the whole population and speeding up the convergence.

3.5 Elimination-and-dispersal

With changes to the local environment in which a population of bacteria lives, all of the bacteria in this region may be killed, or a group of bacteria may be dispersed into a new environment to find better food sources. To simulate this phenomenon, an elimination-and-dispersal step is taken in BFO after N_{re} reproduction steps. Each bacterium in the population may be eliminated or dispersed to a new location with a given probability P_{ed}. The rule is shown in the following:

$$x = \begin{cases} x', & if \quad r < P_{ed} \\ x, & otherwise \end{cases}, \tag{10}$$

where r is a random number uniformly distributed in $[0,1]$, x is the current solution associated with a bacterium, x' is a new solution generated at random in n-dimension search space. That is, for each bacterium, if the number generated randomly is smaller than P_{ed}, it will move to a new random solution, otherwise, it will keep the original solution unchanged. This mechanism is helpful to escape from local Pareto optima

Table 2: Performance comparisons of NSGA-II, SPEA2 and MABFO on four test problems.

Problem	Algorithm	GD		SP	
		Average	Std. Dev.	Average	Std. Dev.
ZDT1	NSGA-II	5.10e-04	1.24e-04	9.14e-03	1.13e-03
	SPEA2	3.76e-04	6.36e-05	3.28e-03	3.80e-04
	MABFO	**1.96e-04**	4.55e-05	**3.24e-03**	5.48e-04
ZDT2	NSGA-II	3.66e-04	2.80e-04	1.02e-02	3.09e-03
	SPEA2	1.74e-04	2.83e-05	3.15e-03	3.62e-04
	MABFO	**9.26e-05**	4.42e-06	**3.04e-03**	2.98e-04
ZDT3	NSGA-II	5.22e-04	2.46e-04	1.02e-02	1.50e-03
	SPEA2	3.73e-04	1.10e-04	**4.42e-03**	5.79e-04
	MABFO	**1.59e-04**	1.24e-05	4.85e-03	1.11e-03
ZDT4	NSGA-II	2.45e-03	9.01e-04	1.33e-02	5.07e-03
	SPEA2	1.88e-03	7.50e-04	4.17e-03	1.25e-03
	MABFO	**2.34e-04**	2.20e-04	**2.40e-03**	1.52e-03

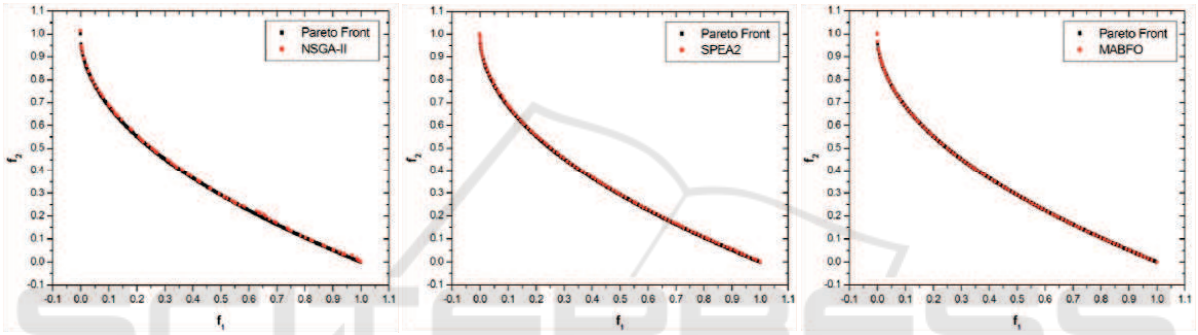

Figure 1: Pareto fronts produced by NSGA-II (left), SPEA2 (middle) and MABFO (right) for the ZDT1 test problem.

and to explore the global Pareto optima in the search space.

4 EXPERIMENTS

In this section, we will compare the proposed MABFO algorithm with two state-of-the-art algorithms NSGA-II and SPEA2. The experimental platform is a PC with Inter(R) Core(TM) i5- 3470 CPU 3.20GHz, 4GB RAM and Windows 7, and all the algorithms are implemented using C++ language.

4.1 Test Functions and Evaluation Metrics

We choose four test problems ZDT1, ZDT2, ZDT3 and ZDT4 which are suggested by Zizler and commonly used in a number of significant past studies. All the test problems have two objective functions and have not any constraint. Table 1 lists these test problems and also provides the number of variables, their ranges, the Pareto-optimal solutions for each problem.

In general, there are two issues to consider for assessing a method in multiobjective optimization area: (1) the convergence of the obtained Pareto front toward the true Pareto front and (2) spread of the obtained solutions. Based on this notion, we adopt two common metrics: Generational distance (GD) (Van Veldhuizen D A, Lamont G B, 1998) and Space (SP) (Schott J R, 1995).

GD measures the distance between the Pareto front found so far and the true Pareto front. It is defined as:

$$GD = \frac{\sum_{i=1}^{n} d_i^2}{n}, \tag{11}$$

where n is the number of members in the Pareto front found so far, d_i is the Euclidean distance between the ith member of the Pareto front found and the nearest member of the true Pareto front. The smaller the value of this metric, the nearer the Pareto front found so far to the true Pareto front.

SP judges how well the Pareto front found so far distributed and is formulated as follows:

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2}, \tag{12}$$

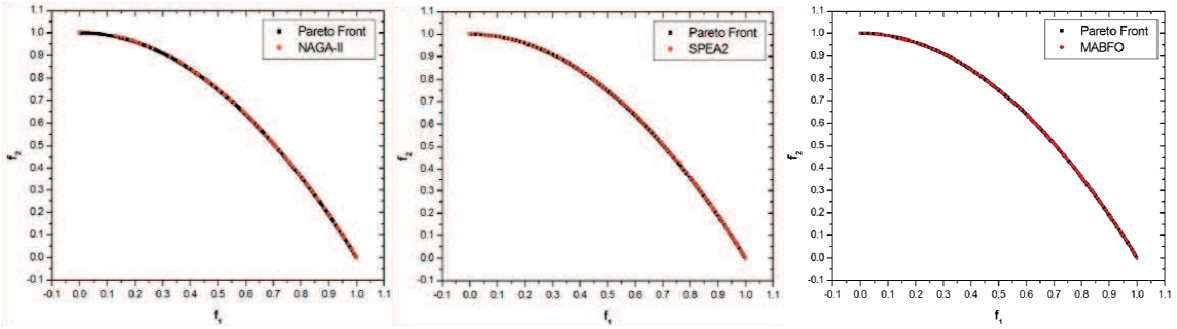

Figure 2: Pareto fronts produced by NSGA-II (left), SPEA2 (middle) and MABFO (right) for the ZDT2 test problem.

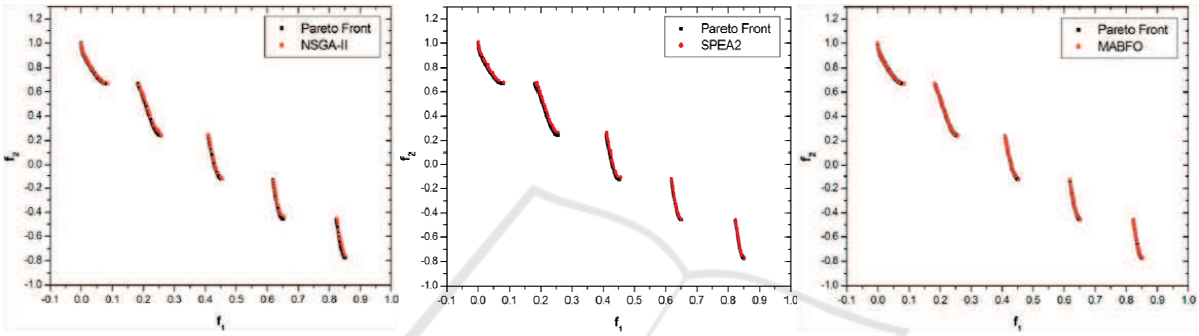

Figure 3: Pareto fronts produced by NSGA-II (left), SPEA2 (middle) and MABFO (right) for the ZDT3 test problem.

where n is the number of members in the Pareto front found so far, d^i is the minimum Manhatton distance between the ith member and other members in the Pareto front found and $d_i = min_j(|f_1(x^i) - f_1(x^j)| + |f_2(x^i) - f_2(x^j)| + \cdots + |f_m(x^i) - f_m(x^j)|)$, $j = 1, 2, \cdots, n$. m is the number of the objectives. $\overline{d}$ is the average value of all d^i. The smaller the value of this metric, the more uniform the Pareto front found is distributed.

4.2 Results and Analysis

In the experiments, each algorithm was tested 30 independent runs on each test problem. To be fair, the population sizes and the elite archive sizes of the three algorithms were set to 100. The generations of NSGA-II and SPEA2 were set to 500. For the other parameters in NSGA-II and SPEA2 algorithms, we tried to use identical settings as suggested in the original studies. For the left parameters in the proposed MABFO algorithm, we didn't make any serious attempt to find the best settings and only chose a reasonable set of values: $N_s = 4$, $N_c = 10$, $N_{re} = 25$, $N_{ed} = 2$, $P_{ed} = 0.2$ and $L = 0.4 \times n$.

Table 2 provides the average results (Average) and standard deviations (Std. Dev.) with respect to the two metrics GD and SP. The best average results with respect to each metric are shown in bold. It can be seen from the table, the average performance of MABFO is the best with respect to the GD and SP metrics on ZDT1, ZDT3 and ZDT4. As for ZDT2, MABFO achieves the best result in term of GD and is only slightly worse than SPEA2 in term of SP, but it has the smallest deviation with respect to SP.

Figures 1-4 show the Pareto fronts obtained by NSGA-II, SPEA2 and our MABFO algorithm on the four test problems–ZDT1, ZDT2, ZDT3 and ZDT4, respectively. The Pareto fronts displayed correspond to the median results over 30 runs with respect to the GD metric. From these figures, we can see that the three algorithms are able to cover the entire Pareto fronts on ZDT1, ZDT2 and ZDT3. But our MABFO algorithm produces better-distributed and higher-quality Pareto front on these three test problems, especially on the ZDT3 problem. For the ZDT4 problem, both NSGA-II and SPEA2 fail to cover the true Pareto front, whereas our MABFO algorithm successfully does it. Through the comparison with two best EMO algorithms NSGA-II and SPEA2, our MABFO algorithm is a viable alternative to solve MOPs.

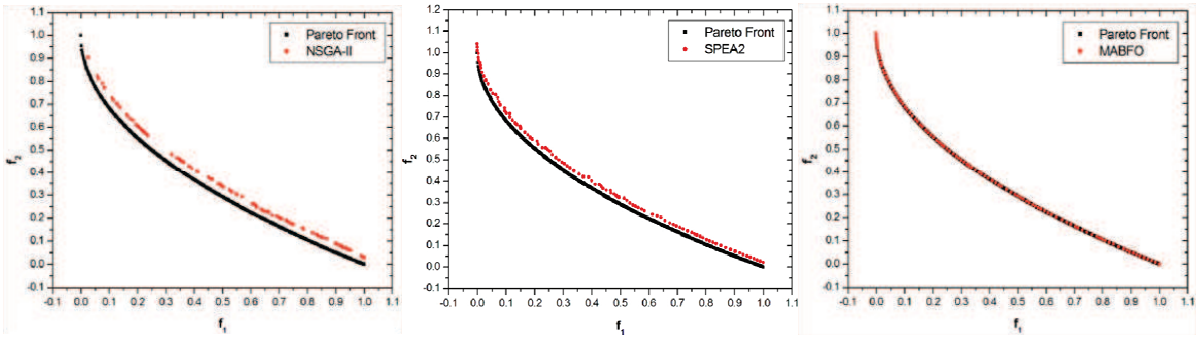

Figure 4: Pareto fronts produced by NSGA-II (left), SPEA2 (middle) and MABFO (right) for the ZDT4 test problem

5 CONCLUSIONS

The MOP is a very important research topic in both science and engineering communities. In recent years, researchers are interested in using some new bio-inspired optimization models to solve MOPs. Although several methods based on BFO have been proposed for handling MOPs. But to the best of our knowledge, the current methods based on BFO have not incorporated the elitist strategy which always plays an important role in getting global Pareto front. In this paper, we present a new algorithm based on archive bacterial foraging optimization for multiobjective optimization (called MABFO). MABFO simulates four biological mechanisms: chemotaxis, conjugation, reproduction and elimination-and-dispersal. The implements of these mechanisms are different from other methods based on BFO. More important, MABFO incorporates an external archive to save the nondominated solutions previously found and maintains the diversity of the nondominated solutions found based on the crowding distance. To demonstrate the performance of MABFO algorithm, we compared it with two best methods known to date (NSGA-II and SPEA2) on four test problems, the results indicate that MABFO is a promising alternative since it has the best average performance with respect to two metrics in most cases.

In the future, we will give an detailed analysis of the algorithm, make a comprehensive testing on more test problems and continue to explore more effective diversity preservation strategies to better cover the global Pareto front. We also hope to extend this algorithm so that it can handle dynamic functions.

ACKNOWLEDGEMENTS

This work is partly supported by the NSFC Research Program (61375059, 61332016), the National 973 Key Basic Research Program of China (2014CB744601), Specialized Research Fund for the Doctoral Program of Higher Education (20121103110031), and the Beijing Municipal Education Research Plan key project (Beijing Municipal Fund Class B) (KZ201410005004).

REFERENCES

A Balassubramaniam, Memeber, IEEE, P Lio (2013). Multi-hop conjugation based bacteria nanonetworks. In *IEEE Trans. on Nanobiosci.*, volume 12, pages 47–59.

Agrawal V, Sharma H, Bansal J C (2011). Bacterial foraging optimization: A survey. In *SocProS'2011, Proceedings of the International Conference on Soft Computing for Problem Solving*, pages 227–242, India.

C Perales-Graván, R Lahoz-Beltra (2008). An am radio receiver designed with a genetic algorithm algorithm based on a bacterial conjugation genetic operator. In *IEEE Trans. Evol. Comput.*, volume 12, pages 129–142.

Coello C A C, Pulido G T, Lechuga M S (2004). Handling multiple objectives with particle swarm optimization. In *IEEE Transactions on Evolutionary Computation*, volume 8, pages 256–279.

Coello Coello CA, Van Veldhuizen DA, Lamont GB (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, springer-verlag edition.

Corne D W, Jerram N R, Knowles J D, et al (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In *GECCO'2001, Proceedings of the Genetic and Evolutionary Computation Conference*, pages 283–290.

Deb (2001). *Multi-objective Optimization using Evoulutionary Algorithms*. Chichester, john wiley & sons edition.

Deb K, Pratap A, Agarwal S, et al (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. In *IEEE Transactions on Evolutionary Computation*, volume 6, pages 182–197.

E Zitzler, M Laumanns, L Thiele (2002). SPEA2: Improving the strength pareto evolutionary algorithm. In *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Berlin.

Guzmán M A, Delgado A, De Carvalho J (2010). A novel multiobjective optimization algorithm based on bacterial chemotaxis. In *Engineering Applications of Artificial Intelligence*, volume 23.

K.M. Passino (2002). Biomimicry of bacterial foraging for distributed optimization and control. In *Control Systems*, volume 22, pages 52–67.

Niu B, Wang H, Tan L, et al (2012). Multi-objective optimization using bfo algorithm. In *Bio-Inspired Computing and Applications*, pages 582–587.

Niu B, Wang H, Wang J, et al (2013). Multi-objective bacterial foraging optimization. In *Neurocomputing*, volume 116, pages 336–345.

Panigrahi B K, Pandi V R, Das S, et al (2010). Multiobjective fuzzy dominance based bacterial foraging algorithm to solve economic emission dispatch problem. In *Energy*, volume 35, pages 227–242.

Parks G T, Miller I (1998). Selective breeding in a multiobjective genetic algorithm. In *Parallel Problem Solving From NaturełPPSN V*, pages 250–259.

S Dasgipta, S Das, A Abraham and A Biswas (2009). Adaptive computational chemotaxis in bacterial foraging optimiztion: An analysis. In *IEEE Transactions on Evolutionary Computation*, volume 13, pages 919–941.

Schott J R (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. In *MS Thesis, Massachusetts institute of Technology*, Cambridge.

Tripathi P K, Bandyopadhyay S, Pal S K (2007). Multiobjective particle swarm optimization with time variant inertia and acceleration coefficients. In *Information Sciences*, volume 177, pages 5033–5049.

Van Veldhuizen D A, Lamont G B (1998). Multiobjective evolutionary algorithm research: A history and analysis. In *Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology*, Ohio.

X Li (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *GECCO'2003, Proceedings of Genetic and Evolutionary Computation*, pages 37–48.

Zitzler E, Deb K, Thiele L (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. In *Evolutionary computation*, volume 8, pages 173–195.