Computers & Operations Research 40 (2013) 1616–1624

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Multiobjective cuckoo search for design optimization

Xin-She Yang [a],*, Suash Deb [b]

[a] Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
[b] Department of Computer Science & Engineering, C.V. Raman College of Engineering, Bidyanagar, Mahura, Janla, Bhubaneswar 752054, India

ARTICLE INFO

Available online 10 October 2011

Keywords:
Cuckoo search
Metaheuristic
Multiobjective
Optimization

ABSTRACT

Many design problems in engineering are typically multiobjective, under complex nonlinear constraints. The algorithms needed to solve multiobjective problems can be significantly different from the methods for single objective optimization. Computing effort and the number of function evaluations may often increase significantly for multiobjective problems. Metaheuristic algorithms start to show their advantages in dealing with multiobjective optimization. In this paper, we formulate a new cuckoo search for multiobjective optimization. We validate it against a set of multiobjective test functions, and then apply it to solve structural design problems such as beam design and disc brake design. In addition, we also analyze the main characteristics of the algorithm and their implications.

1. Introduction

Engineering design often concerns multiple design objectives under complex, highly nonlinear constraints. In reality, different objectives often conflict each other, and sometimes, truly optimal solutions may not exist at all, and some compromise and approximations are often needed [8,11,23]. In addition to these challenges and complexity, a design problem is subjected to various design constraints, limited by design codes or standards, material properties and choice of available resources and costs [11,15]. Even for global optimization problems with a single objective, if the design functions are highly nonlinear, global optimality is not easy to reach. Furthermore, many real-world problems are often NP-hard, which means there is no known efficient algorithm which can be used for a given problem. Therefore, for a given problem, heuristic choices of algorithms and techniques are usually used in practice, in combination with the problem-specific knowledge as some guidance.

On the other hand, metaheuristic algorithms are very powerful in dealing with this kind of optimization, and there are many review articles and excellent textbooks [6,9,11,36,38,39,42]. In contrast with single objective optimization, multiobjective problems are typically much difficult and complex [9,17,18]. In a single objective, we have to find the optimal solution which is often a single point in the solution space, except the case where there are multiple, equally optimal points. For a multiobjective optimization problem, there are multiple optimal solutions which form the so-called Pareto front. In order to get the sense of the unknown Pareto front, we have to generate many different solution points, and therefore computational effort will increase depending on the number of approximate points, complexity of the problem and the way of handling solution diversity. Ideally, the solutions obtained on the Pareto front should distribute relatively uniformly and un-biased. However, there is no technique to ensure that this can be achieved in practice.

From the implementation point of view, algorithms work well for single objective optimization usually cannot directly work for multiobjective problems, unless under special circumstances such as combining multiobjectives into a single objective using some weighted sum method. Substantial modifications are needed to make an algorithm work. In addition to these difficulties, a further challenge is how to generate solutions with enough diversity so that new solutions can sample the search space efficiently. Furthermore, real-world optimization problems always involve some degree of uncertainty or noise. For example, materials properties for a design product may vary significantly, an optimal design should be robust enough to allow such inhomogeneity and also provides good choice for decision-makers or designers.

Despite these challenges, multiobjective optimization has many powerful algorithms with many successful applications [1,4,10,15,22,39,31,40,26]. In addition, metaheuristic algorithms start to emerge as a major player for multiobjective global optimization, they often mimic the successful characteristics in nature, especially biological systems [2,42]. Many new algorithms are emerging with many important applications [1,20,27,33,39,49].

Recently, a new metaheuristic search algorithm, called Cuckoo Search (CS), has been developed by Yang and Deb (2009) [44,45]. Preliminary studies show that it is very promising and could outperform existing algorithms such as PSO. In this paper, we will

* Corresponding author. Present address: Mathematics and Scientific Computing, National Physical Laboratory, Teddington TW11 0LW, UK.
E-mail addresses: xin-she.yang@npl.co.uk, xy227@cam.ac.uk (X.-S. Yang).

0305-0548/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.09.026

extend CS to solve multiobjective problems and formulate a multiobjective cuckoo search (MOCS) algorithm. We will first validate it against a subset of multiobjective test functions. Then, we will apply it to solve design optimization problems in engineering, including bi-objective beam design and a design of a disc brake. Meanwhile, we will also discuss the unique features of the proposed algorithm as well as topics for further studies.

2. Multiobjective cuckoo search

In order to extend the Cuckoo Search for single optimization to solve multiobjective problems, let us briefly review the interesting breed behavior of certain cuckoo species. Then, we will outline the basic ideas and steps of the proposed algorithm.

2.1. Cuckoo behavior

Cuckoo are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the *ani* and *Guira* cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs [28]. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species). There are three basic types of brood parasitism: intraspecific brood parasitism, cooperative breeding, and nest takeover. Some host birds can engage direct conflict with the intruding cuckoos. If a host bird discovers the eggs are not its owns, it will either throw these alien eggs away or simply abandons its nest and builds a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic *Tapera* have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in color and pattern of the eggs of a few chosen host species [28]. This reduces the probability of their eggs being abandoned and thus increases their reproductivity.

2.2. Efficiency of Lévy flights

In nature, animals search for food in a random or quasi-random manner. In general, the foraging path of an animal is effectively a random walk because the next move is based on the current location/state and the transition probability to the next location. Which direction it chooses depends implicitly on a probability which can be modeled mathematically. For example, various studies have shown that the flight behavior of many animals and insects has demonstrated the typical characteristics of Lévy flights [7,34,29].

A recent study by Reynolds and Frye shows that fruit flies or *Drosophila melanogaster*, explore their landscape using a series of straight flight paths punctuated by a sudden 90° turn, leading to a Lévy-flight-style intermittent scale-free search pattern [34]. Even light can be related to Lévy flights [5]. Subsequently, such behavior has been applied to optimization and optimal search, and preliminary results show its promising capability [29].

Broadly speaking, Lévy flights are a random walk whose step length is drawn from the Lévy distribution, often in terms of a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \le 2$ is an index. Mathematically speaking, a simple version of Lévy distribution can be defined as [46]

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\dfrac{\gamma}{2\pi}} \exp\left[-\dfrac{\gamma}{2(s-\mu)}\right] \dfrac{1}{(s-\mu)^{3/2}} & \text{if } 0<\mu<s<\infty, \\ 0 & \text{if } s \le 0. \end{cases} \quad (1)$$

Here $\mu > 0$ is a minimum step and γ is a scale parameter. Clearly, as $s \to \infty$, we have

$$L(s,\gamma,\mu) \approx \sqrt{\frac{\gamma}{2\pi}} \frac{1}{s^{3/2}}. \quad (2)$$

This is a special case of the generalized Lévy distribution.

In general, Lévy distribution should be defined in terms of Fourier transform

$$F(k) = \exp[-\alpha|k|^{\beta}], \quad 0 < \beta \le 2, \quad (3)$$

where α is a scale parameter. The inverse of this integral is not easy, as it does not have analytical form, except for a few special cases.

For the case of $\beta = 2$, we have

$$F(k) = \exp[-\alpha k^2], \quad (4)$$

whose inverse Fourier transform corresponds to a Gaussian distribution [43]. Another special case is $\beta = 1$, and we have

$$F(k) = \exp[-\alpha|k|], \quad (5)$$

which corresponds to a Cauchy distribution

$$p(x,\gamma,\mu) = \frac{1}{\pi} \frac{\gamma}{\gamma^2 + (x-\mu)^2}, \quad (6)$$

where μ is the location parameter, while γ controls the scale of this distribution.

For the general case, the inverse integral

$$L(s) = \frac{1}{\pi} \int_0^{\infty} \cos(ks) \exp[-\alpha|k|^{\beta}]\, dk, \quad (7)$$

can be estimated only when s is large. We have

$$L(s) \to \frac{\alpha\beta\Gamma(\beta)\sin(\pi\beta/2)}{\pi|s|^{1+\beta}}, \quad s \to \infty. \quad (8)$$

Here $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t}\, dt. \quad (9)$$

In the case when $z=n$ is an integer, we have $\Gamma(n) = (n-1)!$

Fig. 1 shows an example of drawing 100 steps which obey a Lévy distribution, while Fig. 2 shows their path during a Lévy flight. It is worth pointing out that Lévy flights are more efficient than Brownian random walks in exploring unknown, large-scale search space. There are many reasons to explain this efficiency, and one of them is due to the fact that the variance of Lévy flights

$$\sigma^2(t) \sim t^{3-\beta}, \quad 1 \le \beta \le 2, \quad (10)$$

increases much faster than the linear relationship (i.e., $\sigma^2(t) \sim t$) of Brownian random walks.

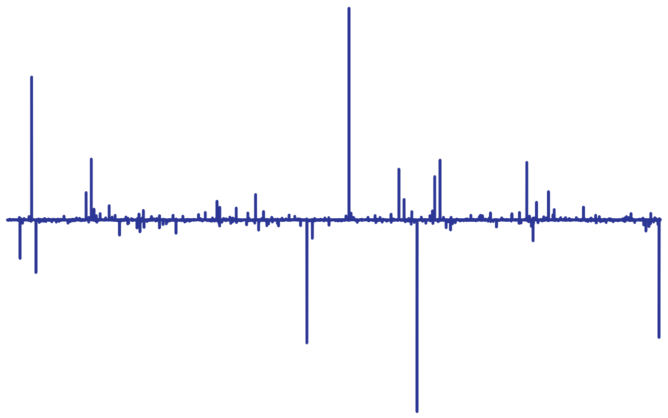
Fig. 1. Distribution of 100 consecutive steps.

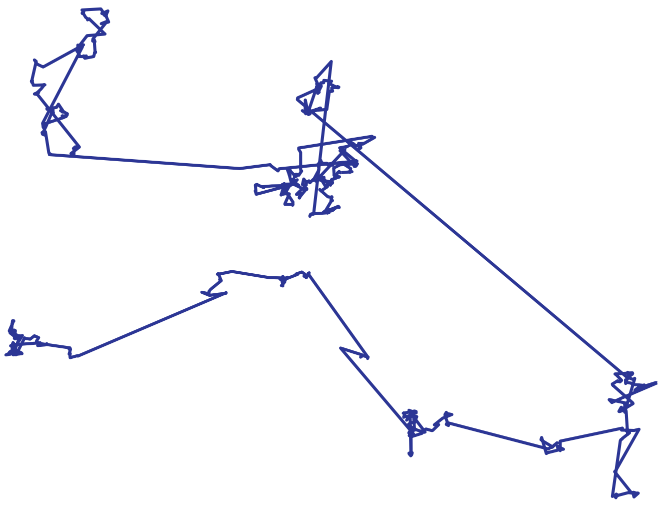

Fig. 2. 2D Lévy flights in 100 steps.

2.3. Multiobjective cuckoo search algorithm

In the original Cuckoo Search for single objective optimization by Yang and Deb [44], the following three idealized rules are used:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0,1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

For multiobjective optimization problems with K different objectives, then we modify the first and last rules to incorporate multiobjective needs:

- Each cuckoo lays K eggs at a time, and dumps them in a randomly chosen nest. Egg k corresponds to the solution to the kth objective.
- Each nest will be abandoned with a probability p_a and a new nest with K eggs will be built, according to the similarities/differences of the eggs. Some random mixing can be used to generate diversity.

For simplicity, this last assumption can be approximated by a fraction p_a of the n nests being replaced by new nests (with new random solutions at new locations). For the maximization of objectives, the quality or fitness of a solution can simply be proportional to each objective function, and a non-dominated solution should be sought.

Mathematically speaking, the first rule can be converted into a randomization process, so that a new solution can be randomly generated either by a random walk or by Lévy flight. At the same time, a localized random permutation is carried out over solutions, which can be considered as a form of crossover. For each nest, there can be K solutions which are generated in the same way as (11). In essence, the second rule corresponds to elitism so that the best solutions are passed onto the next generation, and such selection of the best helps to ensure the algorithm converge properly. In addition, the third rule can be considered as the mutation so that the worst solutions are discarded with a probability and new solutions are generated, according to the similarity of solutions to the other solution. This implies that the mutation is a vectorized operator via the combination of Lévy flights and differential quality of the solutions. These unique features work in a combination can ensure the efficiency of the proposed algorithm.

Based on these three rules, the basic steps of the Multi-objective Cuckoo Search (MOCS) can be summarized as the pseudo code shown in Fig. 3.

When generating new solutions $\boldsymbol{x}^{(t+1)}$ for, say cuckoo i, a Lévy flight is performed

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \alpha \oplus \text{Lévy}(\beta), \tag{11}$$

where $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. In most cases, we can use $\alpha = O(1)$. In order to accommodate the difference between solution quality, we can also use

$$\alpha = \alpha_0 (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}), \tag{12}$$

where α_0 is a constant, while the term in the bracket corresponds to the difference of two randomly solutions. This mimics that fact that similar eggs are less likely to be discovered and thus new solutions are generated by the proportionality of their difference.

The product $\oplus$ means entry-wise multiplications. Lévy flights essentially provide a random walk while their random steps are drawn from a Lévy distribution for large steps

$$\text{Lévy} \sim u = t^{-1-\beta}, \quad (0 < \beta \le 2), \tag{13}$$

which has an infinite variance with an infinite mean. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail.

In addition, a fraction p_a of the worst nests can be abandoned so that new nests can be built at new locations by random walks and mixing. The mixing of the eggs/solutions can be performed by random permutation according to the similarity/difference to the host eggs.

Obviously, the generation of step size s samples is not trivial using Lévy flights. A simple scheme discussed in detail by Yang [46,45] can be summarized as

$$s = \alpha_0 (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}) \oplus \text{Lévy}(\beta) \sim 0.01 \frac{u}{|v|^{1/\beta}} (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}), \tag{14}$$

where u and v are drawn from normal distributions. That is

$$u \sim N(0, \sigma_u^2) \quad v \sim N(0, \sigma_v^2), \tag{15}$$

```
Initialize objective functions $f_1(\boldsymbol{x}), ..., f_K(\boldsymbol{x})$ $\boldsymbol{x} = (x_1, ..., x_d)^T$
Generate an initial population of $n$ host nests $\boldsymbol{x}_i$ and each with $K$ eggs
while ($t$ <MaxGeneration) or (stop criterion)
    Get a cuckoo (say $i$) randomly by Lévy flights
    Evaluate and check if it is Pareto optimal
    Choose a nest among $n$ (say $j$) randomly
    Evaluate $K$ solutions for nest $j$
    if new solutions of nest $j$ dominate those of nest $i$,
        Replace nest $i$ by the new solution set of nest $j$
    end
    Abandon a fraction ($p_a$) of worse nests
    Keep the best solutions (or nests with non-dominated sets)
    Sort and find the current Pareto optimal solutions
end
Postprocess results and visualization.
```

Fig. 3. Multiobjective cuckoo search (MOCS).

with

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \tag{16}$$

Here Γ is the standard Gamma function.

It is worth pointing out that the unique features of the MOCS algorithm are: exploration by Lévy flight, mutation by a combination of Lévy flights and vectorized solution difference, crossover by selective random permutation, and elitism. Cuckoo search essentially uses a good combination of all these basic components, and thus it is potentially more powerful than algorithms using one or some of these components. For example, particle swarm optimization use an update of velocity vector which consists a term of $\varepsilon(\boldsymbol{x}_i - \boldsymbol{g}^*)$, which is the difference of the current solution $\boldsymbol{x}_i$ and the current global best $\boldsymbol{g}^*$. This is the main way for randomization, which limits the steps that are proportional to the solution difference. In cuckoo search, however, the randomization can be more efficient as the steps obey a Lévy distribution which can be approximated by a power law; therefore, the steps consist of many small steps and occasionally large-step, long-distance jumps. Comparing with PSO, this long jumps may increase the search efficiency of cuckoo search significantly in some cases, especial for multimodal, nonlinear problems.

2.4. Pareto front

A solution vector, $\boldsymbol{u} = (u_1, .., u_n)^T \in \mathcal{F}$, is said to dominate another vector $\boldsymbol{v} = (v_1, \ldots, v_n)^T$ if and only if $u_i \le v_i$ for $\forall i \in \{1, \ldots, n\}$ and $\exists i \in \{1, \ldots, n\} : u_i < v_i$. In other words, no component of $\boldsymbol{u}$ is larger than the corresponding component of $\boldsymbol{v}$, and at least one component is smaller. Similarly, we can define another dominance relationship $\preceq$ by

$$\boldsymbol{u} \preceq \boldsymbol{v} \iff \boldsymbol{u} \prec \boldsymbol{v} \vee \boldsymbol{u} = \boldsymbol{v}. \tag{17}$$

It is worth pointing out that for maximization problems, the dominance can be defined by replacing $\prec$ with $\succ$. Therefore, a point $\boldsymbol{x}_* \in \mathcal{F}$ is called a non-dominated solution if no solution can be found that dominates it [9].

The Pareto front PF of a multiobjective can be defined as the set of non-dominated solutions so that

$$PF = \{\boldsymbol{s} \in \mathcal{S} \,|\, \nexists \boldsymbol{s}' \in \mathcal{S} : \boldsymbol{s}' \prec \boldsymbol{s}\}, \tag{18}$$

or in term of the Pareto optimal set in the search space

$$PF^* = \{\boldsymbol{x} \in \mathcal{F} \,|\, \nexists \boldsymbol{x}' \in \mathcal{F} : \boldsymbol{f}(\boldsymbol{x}') \prec \boldsymbol{f}(\boldsymbol{x})\}, \tag{19}$$

where $\boldsymbol{f} = (f_1, \ldots f_K)^T$.

To obtain a good approximation to Pareto front, a diverse range of solutions should be generated using efficient techniques [19,16,22]. In the present approach, Lévy flights ensure the good diversity of the solutions, as we can see from later simulations.

3. Numerical results

3.1. Parametric studies

The proposed multiobjective cuckoo search is implemented in Matlab, and computing time is within a few seconds to less than a minute, depending on the problem of interest. We have tested it using a different range of parameters such as population size (n), Lévy exponent β, and discovery probability p_a. By varying $n = 5, 10, 15, 20$, to $50, 100, 150, 200, 250, 300, 400, 500$, $\beta = 0.5, 1, 1.5, 2$ and $p_a = 0.1, 0.2, \ldots, 0.5, \ldots, 0.9$, we found that the best parameters for most applications are: $n=25$ to 50, $p_a=0.25$ to 0.5 and $\beta = 1$ or 1.5. In addition, the parameter α_0 can be in the range of 0.01–0.5, and the value $\alpha_0 = 0.1$ works well for most applications.

The stopping criterion can be defined in many ways. We can either use a given tolerance or a fixed number of iterations. From the implementation point of view, a fixed number of iterations is not only easy to implement, but also suitable to compare the closeness of Pareto front of different functions. So we have set the fixed number iterations as 5000, which is sufficient for most problems. If necessary, we can also increase it to a larger number.

In order to generate more optimal points on the Pareto front, we can do it in two ways: increase the population size n or run the program a few more times. Through simulations, we found that to increase of n typically leads to a longer computing time than to re-run the program a few times. This may be due to the fact that manipulations of large matrices or longer vectors usually take longer. So to generate 200 points using a population size $n=50$ requires to run the program four times, which is easily done within a few minutes. Therefore, in all our simulations, we will use the fixed parameters: $n=50$, $\beta = 1.5$ and $p_a=0.5$.

3.2. Multiobjective test functions

There are many different test functions for multiobjective optimization [13,48,50,51], but a subset of a few widely used functions provides a wide range of diverse properties in terms Pareto front and Pareto optimal set. To validate the proposed MOCS, we have selected a subset of these functions with convex, non-convex and discontinuous Pareto fronts. We also include functions with more complex Pareto sets. To be more specific in this paper, we have tested the following five functions:

- Schaffer's Min–Min (SCH) test function with convex Pareto front [37,47]

$$f_1(x) = x^2, \quad f_2(x) = (x-2)^2, \quad -10^3 \le x \le 10^3. \tag{20}$$

- ZDT1 function with a convex front [50,51]

$$f_1(x) = x_1, \quad f_2(x) = g(1 - \sqrt{f_1/g}),$$

$$g = 1 + \frac{9\sum_{i=2}^{d} x_i}{d-1}, \quad x_1 \in [0,1], \; i = 1, \ldots, 30, \tag{21}$$

where d is the number of dimensions. The Pareto-optimality is reached when $g=1$.

- ZDT2 function with a non-convex front

$$f_1(x) = x_1, \quad f_2(x) = g\left(1 - \frac{f_1}{g}\right)^2,$$

- ZDT3 function with a discontinuous front

$$f_1(x) = x_1, \quad f_2(x) = g\left[1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1)\right],$$

where g in functions ZDT2 and ZDT3 is the same as in function ZDT1. In the ZDT3 function, f_1 varies from 0 to 0.852 and f_2 from -0.773 to 1.

- LZ function [24,49]

$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j \in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right]^2,$$

$$f_2 = 1 - \sqrt{x_1} + + \frac{2}{|J_2|}\sum_{j \in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right]^2, \tag{22}$$

where $J_1 = \{j | j$ is odd and $2 \le j \le d\}$ and $J_2 = \{j | j$ is even and $2 \le j \le d\}$. This function has a Pareto front $f_2 = 1 - \sqrt{f_1}$ with a

Pareto set

$$x_j = \sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2, 3, \ldots, d, \quad x_1 \in [0, 1]. \tag{23}$$

After generating 200 Pareto points by MOCS, the Pareto front generated by MOCS is compared with the true front $f_2 = 1 - \sqrt{f_1}$ of ZDT1 (see Fig. 4).

Let us define the distance or error between the estimate Pareto front PF^e to its correspond true front PF^t as

$$E_f = \|PF^e - PF^t\|^2 = \sum_{j=1}^{N} (PF_j^e - PF_t)^2, \tag{24}$$

where N is the number of points. The convergence property can be viewed by following the iterations. Fig. 5 shows the exponential-like decrease of E_f as the iterations proceed (left) and the logarithmic scale (right). We can see clearly that our MOCS algorithm indeed converges almost exponentially. The results for all the functions are summarized in Table 1, and the estimated Pareto fronts and true fronts of other functions are shown in Figs. 6 and 7.

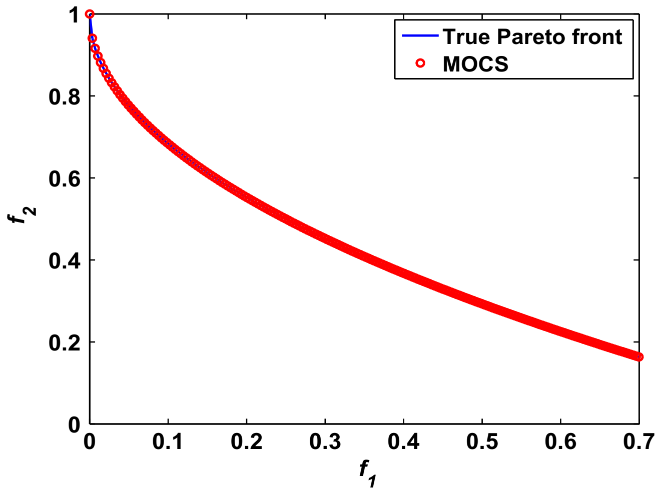

Fig. 4. Pareto front of ZDT1: a comparison of the front found by MOCS and the true front.

In order to compare the performance of the proposed MOCS with other established multiobjective algorithms, we have carefully selected a few algorithms with available results from the literature. In case of the results are not available, we have tried to implement the algorithms using well-documented studies and then generated new results using these algorithms. In particular, we have used other methods for comparison, including vector evaluated genetic algorithm (VEGA) [37], NSGA-II [12], multiobjective differential evolution (MODE) [41,3], differential evolution for multiobjective optimization (DEMO) [35], multiobjective bees algorithms (Bees) [30], and strength Pareto evolutionary algorithm (SPEA) [12,25]. The performance measures in terms of generalized distance D_g are summarized in Table 2 for all the above major methods.

4. Design optimization

Design optimization, especially design of structures, has many applications in engineering and industry. As a result, there are many different benchmarks with detailed studies in the literature [21,30–32]. Among the well-known benchmarks are the welded beam design, and disc brake design. In the rest of this paper, we will solve these two design benchmarks using MOCS.

4.1. Design of a welded beam

Multiobjective design of a welded beam is a classical benchmark which has been solved by many researchers [10,18,32]. The problem has four design variables: the width w and length L of the welded area, the depth d and thickness h of the main beam. The objective is minimize both the overall fabrication cost and the end deflection δ.

Table 1
Summary of results.

Functions	Errors (1000 iterations)	Errors (5000 iterations)
SCH	4.9E−09	2.1E−40
ZDT1	1.2E−6	3.3E−32
ZDT2	7.3E−6	4.5E−25
ZDT3	2.2E−5	1.7E−19
LZ	9.1E−7	2.3E−21

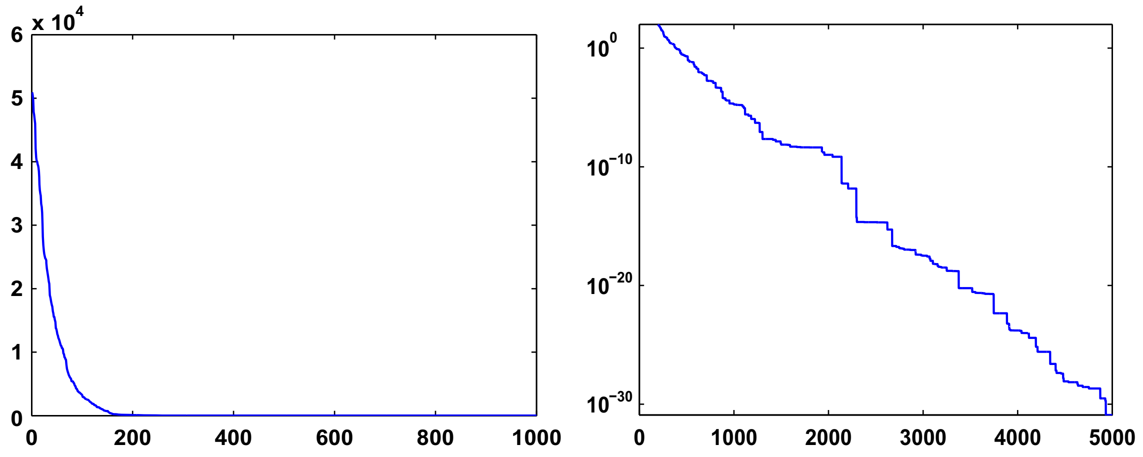

Fig. 5. Convergence of the proposed MOCS. The least-square distance from the estimated front to the true front of ZDT1 for the first 1000 iterations (left) and the logarithmic scale for 5000 iterations (right).

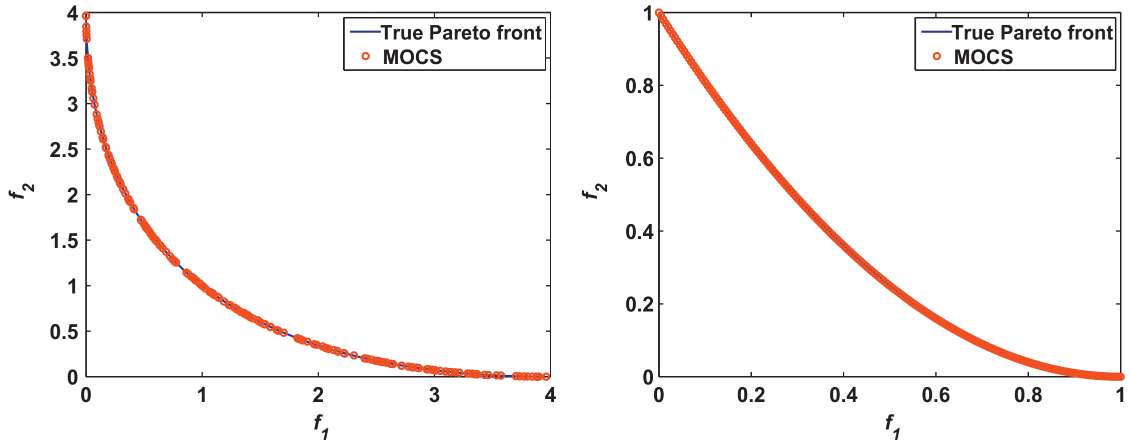

Fig. 6. Pareto fronts of test functions SCH and ZDT2.

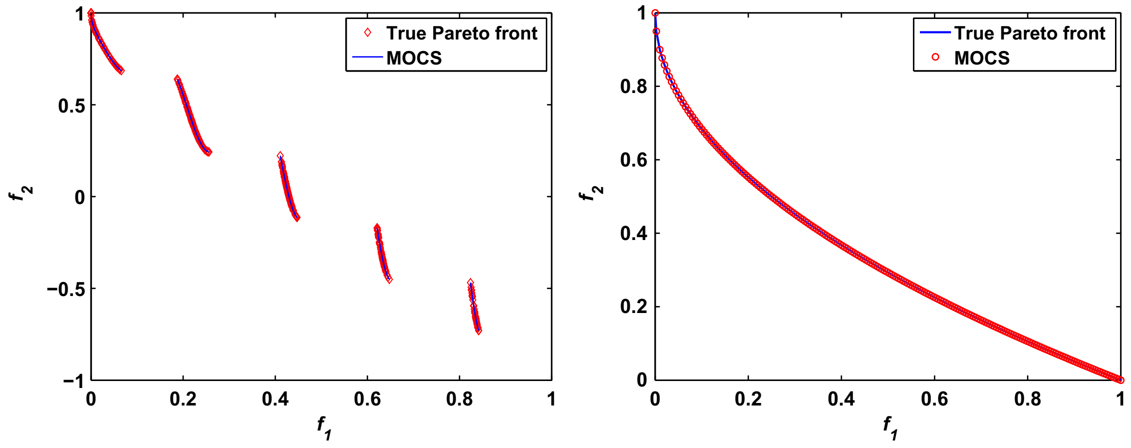

Fig. 7. Pareto fronts of test functions ZDT2 and LZ.

Table 2
Comparison of D_g for $n=50$ and $t=500$ iterations.

Methods	ZDT1	ZDT2	ZDT3	SCH	LZ
VEGA	3.79E−02	2.37E−03	3.29E−01	6.98E−02	1.47E−03
NSGA-II	3.33E−02	7.24E−02	1.14E−01	5.73E−03	2.77E−02
MODE	5.80E−03	5.50E−03	2.15E−02	9.32E−04	3.19E−03
DEMO	1.08E−03	7.55E−04	1.18E−03	1.79E−04	1.40E−03
Bees	2.40E−02	1.69E−02	1.91E−01	1.25E−02	1.88E−02
SPEA	1.78E−03	1.34E−03	4.75E−02	5.17E−03	1.92E−03
MOCS	1.17E−07	2.23E−05	2.88E−05	1.25E−07	4.19E−05

The problem can be written as

$$\text{minimise } f_1(\boldsymbol{x}) = 1.10471w^2L + 0.04811dh(14.0+L), \text{ minimize } f_2 = \delta, \tag{25}$$

subject to

$$g_1(\boldsymbol{x}) = w - h \le 0,$$

$$g_2(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \le 0,$$

$$g_3(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13,600 \le 0,$$

$$g_4(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30,000 \le 0,$$

$$g_5(\boldsymbol{x}) = 0.10471w^2 + 0.04811hd(14+L) - 5.0 \le 0,$$

$$g_6(\boldsymbol{x}) = 0.125 - w \le 0,$$

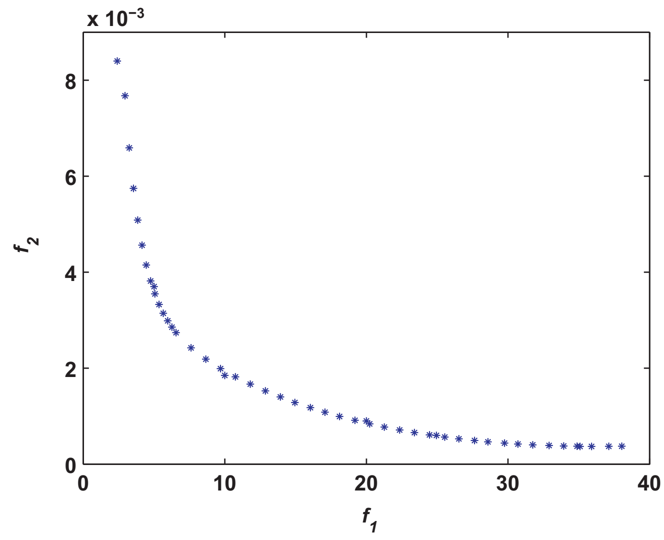

Fig. 8. Pareto front for the bi-objective beam design.

$$g_7(\boldsymbol{x}) = 6000 - P(\boldsymbol{x}) \le 0, \tag{26}$$

where

$$\sigma(\boldsymbol{x}) = \frac{504,000}{hd^2}, \quad Q = 6000\left(14 + \frac{L}{2}\right),$$

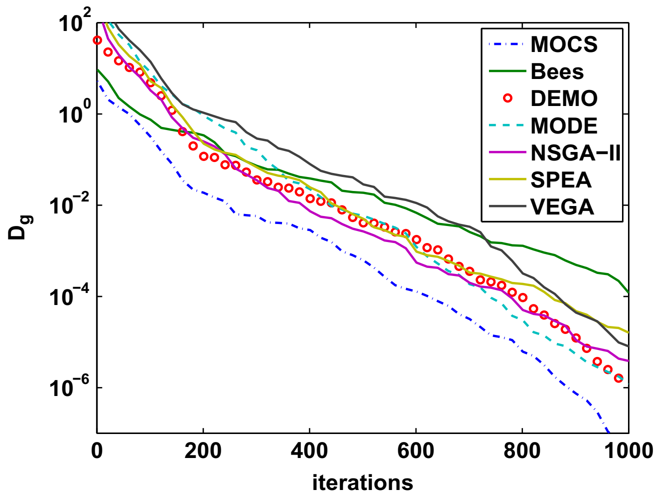

Fig. 9. Convergence comparison for the beam design.

$$D=\frac{1}{2}\sqrt{L^2+(w+d)^2},\quad J=\sqrt{2}wL\left[\frac{L^2}{6}+\frac{(w+d)^2}{2}\right],$$

$$\delta=\frac{65,856}{30,000hd^3},\quad \beta=\frac{QD}{J},$$

$$\alpha=\frac{6000}{\sqrt{2}wL},\quad \tau(\boldsymbol{x})=\sqrt{\alpha^2+\frac{\alpha\beta L}{D}+\beta^2},$$

$$P=0.61423\times10^6\frac{dh^3}{6}\left(1-\frac{d\sqrt{30/48}}{28}\right). \tag{27}$$

The simple limits or bounds are $0.1\le L,d\le 10$ and $0.125\le w,h\le 2.0$.

By using the MOCS, we have solved this design problem. The approximate Pareto front generated by the 50 non-dominated solutions after 1000 iterations is shown in Fig. 8. This is consistent with the results obtained by others [32,30]. In addition, the results are more smooth with fewer iterations.

In order to see how the proposed MOCS performance for the real-world design problems, we also solved the same problems using other available multiobjective algorithms. The comparison of the convergence rates is plotted in the logarithmic scales in Fig. 9. By comparison, the convergence rates of MOCS are of the highest in an exponentially decreasing way.

4.2. Design of a disc brake

Design of a multiple disc brake is another benchmark for multiobjective optimization [18,27,32]. The objectives are to minimize the overall mass and the braking time by choosing optimal design variables: the inner radius r, outer radius R of the discs, the engaging force F and the number of the friction surface S. This is under the design constraints such as the torque, pressure, temperature, and length of the brake. This bi-objective design problem can be written as

$$\text{Minimize } f_1(\boldsymbol{x})=4.9\times10^{-5}(R^2-r^2)(s-1),$$
$$f_2(\boldsymbol{x})=\frac{9.82\times10^6(R^2-r^2)}{Fs(R^3-r^3)}, \tag{28}$$

subject to

$$g_1(\boldsymbol{x})=20-(R-r)\le 0,$$

$$g_2(\boldsymbol{x})=2.5(s+1)-30\le 0,$$

$$g_3(\boldsymbol{x})=\frac{F}{3.14(R^2-r^2)}-0.4\le 0,$$

$$g_4(\boldsymbol{x})=\frac{2.22\times10^{-3}F(R^3-r^3)}{(R^2-r^2)^2}-1\le 0,$$

$$g_5(\boldsymbol{x})=900-\frac{0.0266Fs(R^3-r^3)}{(R^2-r^2)}\le 0. \tag{29}$$

The simple limits are

$$55\le r\le 80,\quad 75\le R\le 110,\quad 1000\le F\le 3000,\quad 2\le s\le 20. \tag{30}$$

The Pareto front of 50 solution points after 1000 iterations obtained by MOCS is shown in Fig. 10, where we can see that the results are smooth and are the same or better than the results obtained in [32].

The comparison of the convergence rates is plotted in the logarithmic scales in Fig. 11. We can see from these two figures that the convergence of MOCS are of the highest and exponential in both cases. This again suggests that MOCS provides better solutions in a more efficient way.

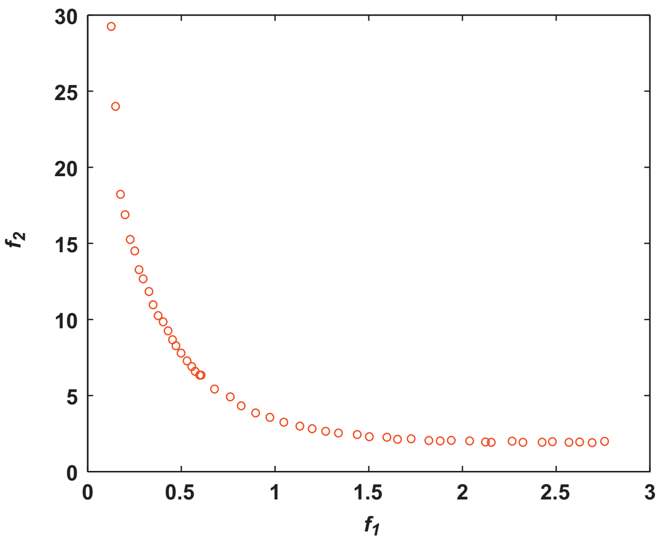

Fig. 10. Pareto front for the disc brake design.

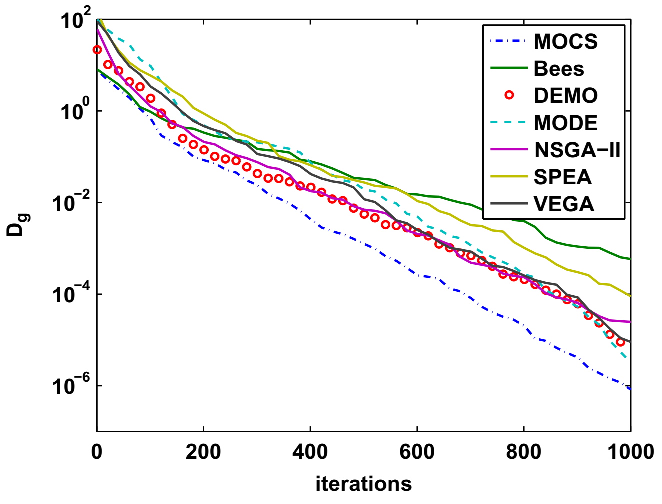

Fig. 11. Convergence comparison for the disc brake design.

The simulations for these benchmarks and test functions suggest that MOCS is a very efficient algorithm for multiobjective optimization. It can deal with highly nonlinear problems with complex constraints and diverse Pareto optimal sets.

5. Conclusions

Multiobjective optimization problems are typically very difficult to solve. In this paper, we have successfully formulated a new algorithm for multiobjective optimization, namely, multiobjective cuckoo search, based on the recently developed cuckoo search algorithm. The proposed MOCS has been tested against a subset of well-chosen test functions, and then been applied to solve design optimization benchmarks in structural engineering. Results suggest that MOCS is an efficient multiobjective optimizer.

In comparison with other algorithms, cuckoo search performs well for almost all these test problems. This superiority can be attributed to the fact that cuckoo search uses a combination of vectorized mutation, crossover by permutation and Lévy flights and selective elitism among the best solutions. In addition, the not-so-good solutions can be replaced systematically by new solutions, and new solutions are often generated by preferring quality solutions in the solution sets. Thus, the mechanism of the overall search moves is more subtle and balanced, compared with the simple mechanism used in particle swarm optimization. Obviously, a more detailed validation study over a wide range of test functions will be carried out in the follow-up work in the near future.

Additional test and comparison of the proposed are highly needed. In the future work, we will focus on the parametric studies for a wider range of test problems, including discrete and mixed type of optimization problems. We will try to test the diversity of the Pareto front it can generate so as to identify the way to improve this algorithm to suit for a diverse range of problems. There are a few efficient techniques to generate diverse Pareto fronts [14,16,19], and some combination with these techniques may improve MOCS even further.

Further research can also emphasize the performance comparison of this algorithm with other popular methods for multiobjective optimization [9,10,38]. In addition, hybridization with other algorithms may also prove to be fruitful.

References

[1] Abbass HA, Sarker R. The Pareto differential evolution algorithm. International Journal on Artificial Intelligence Tools 2002;11(4):531–52.
[2] Adamatzky A, Bull L, De Lacy Costello B, Stepney S, Teuscher C. Unconventional computing 2007. UK: Luniver Press; 2007.
[3] Babu BV, Gujarathi AM. Multi-objective differential evolution (MODE) for optimization of supply chain planning and management. In: IEEE Congress on Evolutionary Computation (CEC 2007), pp. 2732–9.
[4] Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization. Part II: hydridisation combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing 2008;20:109–24.
[5] Barthelemy P, Bertolotti J, Wiersma DS. A Lévy flight for light. Nature 2008; 453:495–8.
[6] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptural comparison. ACM Computing Surveys 2003;35:268–308.
[7] Brown C, Liebovitch LS, Glendon R. Lévy flights in Dobe Ju/'hoansi foraging patterns. Human Ecology 2007;35:129–38.
[8] Cagnina LC, Esquivel SC, Coello CA. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 2008;32:319–26.
[9] Coello CAC. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In: Proceedings of 1999 congress on evolutionary computation, CEC99, 1999. doi: 10.1109/CEC.1999.781901.
[10] Deb K. Evolutionary algorithms for multi-criterion optimization in engineering design. In: Evolutionary algorithms in engineering and computer science. Wiley; 1999. p. 135–61.
[11] Deb K. Multi-objective optimization using evolutionary algorithms. New York: John Wiley & Sons; 2001.
[12] Deb K, Pratap A, Agarwal S, Mayarivan T. A fast and elitist multiobjective algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6: 182–97.
[13] Deb EZK, Thiele L. Comparison of multi objective evolutionary algorithms: empirical results, Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, February 1999.
[14] Erfani T, Utyuzhnikov S. Directed search domain: a method for even generation of Pareto frontier in multiobjective optimization. Engineering Optimization 2011;43(5):467–84.
[15] Farina M, Deb K, Amota P. Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Transactions on Evolutionary Computation 2004;8:425–42.
[16] Figueira JR, Tavares G, Wiecek MM. Labeling algorithms for multiple objective integer knapsack problems. Computers & OR 2010;37(4):700–11.
[17] Floudas CA, Pardalos PM, Adjiman CS, Esposito WR, Gumus ZH, Harding ST, Klepeis JL, Meyer CA, Scheiger CA. Handbook of test problems in local and global optimization. Springer; 1999.
[18] Gong WY, Cai ZH, Zhu L. An effective multiobjective differential evolution algorithm for engineering design. Structural and Multidisciplinary Optimization 2009;38:137–57.
[19] Gujarathi AM, Babu BV. Improved strategies of multi-objective differential evolution (MODE) for multi-objective optimization. In: Proceedings of fourth Indian international conference on artificial intelligence (IICAI-09), December 16–18, 2009.
[20] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Piscataway, NJ, 1995, pp. 1942–8.
[21] Kim JT, Oh JW, Lee IW. Multiobjective optimization of steel box girder bridge. In: Proceedings of seventh KAIST-NTU-KU trilateral seminar/workshop on civil engineering, Kyoto, December 1997.
[22] Konak A, Coit DW, Smith AE. Multiobjective optimization using genetic algorithms: a tutorial. Reliability Engineering and System Safety 2006;91: 992–1007.
[23] Leifsson L, Koziel S. Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction. Journal of Computer Science 2010;1:98–106.
[24] Li H, Zhang QF. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation 2009;13:284–302.
[25] Madavan NK. Multiobjective optimization using a Pareto differential evolution approach. In: Congress on evolutionary computation (CEC'2002), vol. 2. New Jersey: IEEE Service Center; 2002. p. 1145–50.
[26] Marler RT, Arora JS. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 2004;26: 369–95.
[27] Osyczka A, Kundu S. A genetic algorithm-based multicriteria optimization method. In: Proceedings of first world congress on structural and multidisciplinary optimization. Elsevier Science; 1995. p. 909–14.
[28] Payne RB, Sorenson MD, Klitz K. The cuckoos. Oxford University Press; 2005.
[29] Pavlyukevich I. Lévy flights, non-local search and simulated annealing. Journal of Computational Physics 2007;226:1830–44.
[30] Pham DT, Ghanbarzadeh A. multi-objective optimisation using the bees algorithm. In: Third international virtual conference on intelligent production machines and systems (IPROMS 2007): Whittles, Dunbeath, Scotland, 2007.
[31] Rangaiah G. Multi-objective optimization: techniques and applications in chemical engineering. World Scientific Publishing; 2008.
[32] Ray L, Liew KM. A swarm metaphor for multiobjective design optimization. Engineering Optimization 2002;34(2):141–53.
[33] Reyes-Sierra M, Coello CAC. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. International Journal of Computational Intelligence Research 2006;2(3):287–308.
[34] Reynolds AM, Frye MA. Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. PLoS One 2007;2:e354.
[35] Robič T, Filipič B. DEMO: differential evolution for multiobjective optimization. In: Coello CA, editor. EMO 2005, Lecture Notes in Computer Sciences, vol. 3410; 2005. p. 520–33.
[36] Sayadi MK, Ramezanian R, Ghaffari-Nasab N. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations 2010;1:1–10.
[37] Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithms; 1985. p. 93–100.
[38] Talbi E-G. Parallel combinatorial optimization. John Wiley and Sons; 2006. p. 330.
[39] Talbi E-G. Metaheuristics: from design to implementation. John Wiley and Sons; 2009. p. 624.
[40] Tantar E, Dhaenens C, Figueira JR, Talbi E-G. A priori landscape analysis in guiding interactive multi-objective metaheuristics. In: IEEE congress on evolutionary computation 2008; 2008. p. 4104–11.
[41] Xue F. Multi-objective differential evolution: theory and applications. PhD thesis. Rensselaer Polytechnic Institute; 2004.

[42] Yang XS. Nature-Inspired Metaheuristic Algorithms. Luniver Press; 2008 (first edition, 2008; second edition, 2010).
[43] Yang XS. Introduction to computational mathematics. World Scientific Publishing; 2008.
[44] Yang XS, Deb S. Cuckoo search via Lévy flights. In: Proceedings of world congress on nature & biologically inspired computing (NaBIC 2009 India). USA: IEEE Publications; 2009. p. 210–4.
[45] Yang XS, Deb S. Engineering optimization by cuckoo search. Int J Math Modelling Num Opt 2010;1(4):330–43.
[46] Yang XS. Engineering optimisation: an introduction with metaheuristic applications. John Wiley and Sons; 2010.
[47] Zhang LB, Zhou CG, Liu XH, Ma ZQ, Liang YC. Solving multi objective optimization problems using particle swarm optimization. In: Proceedings of the 2003 congress on evolutionary computation CEC2003, vol. 4. Canberra Australia: IEEE Press; December 2003. p. 2400–5.
[48] Zhang QF, Zhou AM, Zhao SZ, Suganthan PN, Liu W, Tiwari S. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, University of Essex, Nanyang Technological University, and Clemson University; April 2009.
[49] Zhang QF, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 2007;11:712–31.
[50] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 1999;3:257–71.
[51] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. Evolution Computing 2000;8:173–95.