Engineering with Computers
DOI 10.1007/s00366-012-0254-1

ORIGINAL ARTICLE

Multiobjective firefly algorithm for continuous optimization

Xin-She Yang

Received: 25 September 2011 / Accepted: 5 January 2012

Abstract Design problems in industrial engineering often involve a large number of design variables with multiple objectives, under complex nonlinear constraints. The algorithms for multiobjective problems can be significantly different from the methods for single objective optimization. To find the Pareto front and non-dominated set for a nonlinear multiobjective optimization problem may require significant computing effort, even for seemingly simple problems. Metaheuristic algorithms start to show their advantages in dealing with multiobjective optimization. In this paper, we extend the recently developed firefly algorithm to solve multiobjective optimization problems. We validate the proposed approach using a selected subset of test functions and then apply it to solve design optimization benchmarks. We will discuss our results and provide topics for further research.

Keywords Algorithm · Firefly algorithm · Metaheuristic · Multiobjective · Engineering design · Global optimization

1 Introduction

Design optimization in engineering and industry often concerns multiple design objectives under complex, highly nonlinear constraints. Different objectives often conflict each other, and sometimes, truly optimal solutions do not exist, and some compromises and approximations are often needed [1–3]. Further to this complexity, a design problem is subjected to various design constraints, limited by design codes or standards, material properties and the optimal utility of available resources and costs [2, 4]. Even for global optimization problems with a single objective, if the design functions are highly nonlinear, global optimality is not easy to reach. Metaheuristic algorithms are very powerful in dealing with this kind of optimization, and there are many review articles and textbooks [5–11].

In contrast with single objective optimization, multiobjective problems are much more difficult and complex [5, 12]. First, no single unique solution is the best; instead, a set of non-dominated solutions should be found in order to get a good approximation to the true Pareto front. Second, even if an algorithm can find solution points on the Pareto front, there is no guarantee that multiple Pareto points will distribute along the front uniformly; often they do not. Third, algorithms which 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 algorithms for single objective optimization 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, which provides a set of multiple feasible solution sets. Consequently, optimal solutions among the robust Pareto set can provide good options so that decision-makers or designers can choose to suit their needs. Despite these challenges, multiobjective optimization has many powerful algorithms with many successful applications [6, 13–15, 43]. In addition, metaheuristic

X.-S. Yang (✉)
Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
e-mail: xy227@cam.ac.uk

Published online: 24 January 2012

algorithms start to emerge as a major player for multiobjective global optimization; they often mimic the successful characteristics in Nature, especially biological systems [9, 10], while some algorithms are inspired by the beauty of music [16]. Many new algorithms are emerging with many important applications [8, 10, 13, 17–20].

For example, multiobjective genetic algorithms are widely known [15, 21], while multiobjective differential evolution algorithms are also very powerful [22, 23]. In addition, multiobjective particle swarm optimizers are becoming increasingly popular [19]. As there are many algorithms, one of our motivations in the present study is to compare the performance of these algorithms for real-world application.

Most metaheuristic algorithms are based on the so-called swarm intelligence. PSO is a good example; it mimics some characteristics of birds and fish swarms. Recently, a new metaheuristic search algorithm, called firefly algorithm (FA), has been developed by Yang [9, 11]. FA mimics some characteristics of tropic firefly swarms and their flashing behaviour [10, 11]. A firefly tends to be attracted towards other fireflies with higher flash intensity. This algorithm is thus different from PSO and can have two advantages: local attractions and automatic regrouping. As light intensity decreases with distance, the attraction among fireflies can be local or global, depending on the absorbing coefficient, and thus all local modes as well as global modes will be visited. In addition, fireflies can also subdivide and thus regroup into a few subgroups because neighboring attraction is stronger than long-distance attraction; thus it can be expected each subgroup will swarm around a local mode. This latter advantage makes it particularly suitable for multimodal global optimization problems [10, 11].

Preliminary studies show that it is very promising and could outperform existing algorithms such as particle swarm optimization (PSO). For example, a firefly-LGB algorithm, based on firefly algorithm and Linde–Buzo–Gray (LGB) algorithms for vector quantization of digital image compression, was developed by Horng and Jiang [24], and their results suggested that firefly-LGB is faster than other algorithms such as particle swarm optimization LBG (PSO-LBG) and honey-bee mating optimization LBG (HBMO-LBG). Apostolopoulos and Vlachos [25] provided a detailed background and analysis over a wide range of test problems, and they also solved multiobjective load dispatch problem using a weighted sum method by combining multiobjectives into a single objective, and their results are very promising. The preliminary successful results of firefly-based algorithms provide another motivation for this paper. That is to see how this algorithm can be extended to solve multiobjective optimization problems.

In this paper, we will extend FA to solve multiobjective problems and formulate a multiobjective firefly algorithm (MOFA). We will first validate it against a subset of multiobjective test functions. Then, we will apply it to solve design optimisation problems in engineering, including bi-objective beam design and a design of a disc brake. Finally, we will discuss the unique features of the proposed algorithm as well as topics for further studies.

2 Multiobjective firefly algorithm

In order to extend the firefly algorithm for single objective optimization to solve multiobjective problems, let us briefly review its basic version.

2.1 The basic firefly algorithm

Firefly Algorithm was developed by Yang for continuous optimization [9, 10, 27], which was subsequently applied to structural optimization [26] and image processing [24]. FA was based on the flashing patterns and behaviour of fireflies. In essence, FA uses the following three idealized rules: (1) fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex; (2) the attractiveness of a firefly is proportional to its brightness and they both decrease with distance. Thus for any two flashing fireflies, the less brighter one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly; (3) the brightness of a firefly is determined by the landscape of the objective function.

For a maximization problem, the brightness can simply be proportional to the value of the objective function. As both light intensity and attractiveness affect the movement of fireflies in the firefly algorithm, we have to define their variations. For simplicity, we can always assume that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function. In the simplest case for maximum optimization problems, the brightness I of a firefly at a particular location $\boldsymbol{x}$ can be chosen as $I(\boldsymbol{x}) \propto f(\boldsymbol{x})$.

However, the attractiveness β is relative, it should be seen in the eyes of the beholder or judged by the other fireflies. Thus, it will vary with the distance r_{ij} between firefly i and firefly j. Therefore, we can now define the attractiveness β of a firefly by

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{1}$$

where β_0 is the attractiveness at $r = 0$. In fact, Eq. (1) defines a characteristic distance $\Gamma = 1/\sqrt{\gamma}$ over which the attractiveness changes significantly from β_0 to $\beta_0\, e^{-1}$. The distance between any two fireflies i and j at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$,

respectively, is the Cartesian distance $r_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$. It is worth pointing out that the distance r defined above is *not* limited to the Euclidean distance. In fact, any measure that can effectively characterize the quantities of interest in the optimization problem can be used as the 'distance' r. We can define other distance r in the n-dimensional hyperspace, depending on the type of problem of our interest.

For any given two fireflies $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the movement of firefly i is attracted to another more attractive (brighter) firefly j is determined by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha_t\, \boldsymbol{\epsilon}_i^t, \tag{2}$$

where the second term is due to the attraction. The third term is randomization with α_t being the randomization parameter, and $\boldsymbol{\epsilon}_i^t$ is a vector of random numbers drawn from a Gaussian distribution or uniform distribution. The location of fireflies can be updated sequentially, by comparing and updating each pair of them in every iteration cycle.

For most implementations, we can take $\beta_0 = 1$ and $\alpha_t = O(1)$, though we found that it is better to use a time-dependent α_t so that randomness can be reduced gradually as iterations proceed. It is worth pointing out that (2) is a random walk biased towards the brighter fireflies. If $\beta_0 = 0$, it becomes a simple random walk. Furthermore, the randomization term can easily be extended to other distributions such as Lévy flights [28].

The parameter γ now characterizes the variation of the attractiveness, and its value is crucially important in determining the speed of the convergence and how the FA algorithm behaves. In theory, $\gamma \in [0, \infty)$, but in practice, $\gamma = O(1)$ is determined by the characteristic distance Γ of the system to be optimized. Thus, for most applications, it typically varies from 10^{-5} to 10^5.

To consider the scale variations of each problem, we now use rescaled, vectorized parameters

$$\alpha = 0.01L, \quad \gamma = 0.5/L^2, \tag{3}$$

with $L = (U_b - L_b)$ where U_b and L_b are the upper and lower bounds of $\boldsymbol{x}$, respectively. Here the factor 0.01 is to make sure the random walks is not too aggressive, and this value has been obtained by a parametric study.

2.2 Multiobjective firefly algorithm

For multiobjective optimization, one way is to combine all objectives into a single objective so that algorithms for single objective optimization can be used without much modifications. For example, FA can be used directly to solve multiobjective problems in this manner, and a detailed study was carried out by Apostolopoulos and Vlachos [25].

Another way is to extend the firefly algorithm to produce Pareto optimal front directly. By extending the basic ideas of FA, we can develop the following Multiobjective firefly algorithm (MOFA), which can be summarized as the pseudo code listed in Table 1.

The procedure starts with an appropriate definition of objective functions with associated nonlinear constraints. We first initialize a population of n fireflies so that they should distribute among the search space as uniformly as possible. This can be achieved by using sampling techniques via uniform distributions. Once the tolerance or a fixed number of iterations is defined, the iterations start with the evaluation of brightness or objective values of all the fireflies and compare each pair of fireflies. Then, a random weight vector is generated (with the sum equal to 1), so that a combined best solution $\boldsymbol{g}_*^t$ can be obtained. The non-dominated solutions are then passed onto the next iteration. At the end of a fixed number of iterations, in general n non-dominated solution points can be obtained to approximate the true Pareto front.

In order to do random walks more efficiently, we can find the current best $\boldsymbol{g}_*^t$ which minimizes a combined objective via the weighted sum

$$\psi(\boldsymbol{x}) = \sum_{k=1}^{K} w_k f_k, \quad \sum_{k=1}^{K} w_k = 1. \tag{4}$$

Here $w_k = p_k/K$, where p_k are the random numbers drawn from a uniform distributed Unif[0,1]. In order to ensure that $\sum_k w_k = 1$, a rescaling operation is performed after generating K uniformly distributed numbers. It is worth pointing out that the weights w_k should be chosen randomly at each iteration, so that the non-dominated solution can sample diversely along the Pareto front.

If a firefly is not dominated by others in the sense of Pareto front, the firefly moves

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{g}_*^t + \alpha_t\, \boldsymbol{\epsilon}_i^t, \tag{5}$$

where $\boldsymbol{g}_*^t$ is the best solution found so far for a given set of random weights.

Furthermore, the randomness can be reduced as the iterations proceed, and this can be achieved in a similar manner as that for simulated annealing and other random reduction techniques [11]. We will use

$$\alpha_t = \alpha_0 0.9^t, \tag{6}$$

where α_0 is the initial randomness factor.

2.3 Pareto optimal front

For a minimization problem, a solution vector $\boldsymbol{u} = (u_1, \ldots, u_n)^T$ 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

Table 1 Pseudo code: multiobjective firefly algorithm (MOFA)

```
Define objective functions f_1(x), ..., f_K(x) where x = (x_1, ..., x_d)^T
Initialize a population of n fireflies x_i (i = 1, 2, ..., n)
while (t < MaxGeneration)
quad for i,j = 1:n (all n fireflies)
    Evaluate their approximations PF_i and PF_j to the Pareto front
            if i ≠ j and when all the constraints are satisfied
    if PF_j dominates PF_i,
        Move firefly i towards j using (2)
        Generate new ones if the moves do not satisfy all the constraints
    end if
    if no non-dominated solutions can be found
        Generate random weights w_k (k = 1, ..., K)
        Find the best solution g_*^t (among all fireflies) to minimize ψ in (4)
       Random walk around g_*^t using (5)
    end if
    Update and pass the non-dominated solutions to next iterations
  end
  Sort and find the current best approximation to the Pareto front
  Update t ← t + 1
end while
Postprocess results and visualisation;
```

$\exists i \in \{1, \ldots, n\} : u_i < v_i$. 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

$$\boldsymbol{u} \preceq \boldsymbol{v} \Longleftrightarrow \boldsymbol{u} \prec \boldsymbol{v} \vee \boldsymbol{u} = \boldsymbol{v}. \tag{7}$$

It is worth pointing out that for maximization problems, the dominance can be defined by replacing $\prec$ with $\succ$. Therefore, a point $\boldsymbol{x}_*$ is called a non-dominated solution if no solution can be found that dominates it [5].

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

$$PF = \{\boldsymbol{s} \in \mathcal{S} \,|\, \exists\, \boldsymbol{s}' \in \mathcal{S} : \boldsymbol{s}' \prec \boldsymbol{s}\}, \tag{8}$$

where $\mathcal{S}$ is the solution set.

To obtain a good approximation of the Pareto front, a diverse range of solutions should be generated using efficient techniques [15, 29–31]. For example, Lévy flights ensure the good diversity of the solutions, as we can see from later simulations.

3 Numerical results

We have implemented the proposed MOFA in Matlab, and we have first validated it against a set of multiobjective test functions. Then, we have used it to solve some industrial design of structures. In order to obtain the right algorithm-dependent parameters, we have carried out detailed parametric studies.

3.1 Parametric studies

By varying the parameters α_0, β_0 and γ, we have carried out parametric studies by setting $\alpha_0 = 0$–1 with a step of 0.05, $\beta_0 = 0$–1 with a step of 0.05, and $\gamma = 0.1$–10 with a step of 0.1 and then 1. From simulations, we found that we can use $\alpha_0 = 0.1$–0.5, $\beta_0 = 0.7$–1.0 and $\gamma = 1$ for most problems.

The stopping criterion can be defined in many ways. We can either use a given tolerance or a fixed number of iterations. For a given tolerance, we should have some prior knowledge of the true optimum of the objective function so that we can calculate the differences between the current best solutions and the true optimal solution so that we can assess if the tolerance is met. In reality, we usually do not know the true optimum in advance, except for a few well-tested cases. In addition, the number of functions can vary significantly from function to function even for the same tolerance. 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 2,500, which is sufficient for most

problems. If necessary, we can also increase it to a larger number.

One can generate points of the Pareto front in two ways: increase the population size n or run the program a few more times. Through simulations, we found that increasing n typically leads to a longer computing time than re-running the program a few times. This may be due to the fact that manipulations of large matrices or longer vectors usually take longer. In order to generate M points using a smaller population size n, it requires to run the program M/n times, each run with different, random initial configurations but with the same number of iterations $t = 2{,}500$. For example, to generate $M = 200$ points, we can use $n = 50$, which is easily done within a few minutes. Therefore, in all our simulations, we will use the fixed parameters: $n = 50$, $\alpha_0 = 0.25$, $\beta_0 = 1$ and $\gamma = 1$.

3.2 Multiobjective test functions

There are many different test functions for multiobjective optimization [32–34], but a subset of a few widely used functions provides a wide range of diverse properties in terms of Pareto front and Pareto optimal set. To validate the proposed MOFA, 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 [21, 35]

$$f_1(x) = x^2, \quad f_2(x) = (x-2)^2, \quad -10^3 \le x \le 10^3. \tag{9}$$

- ZDT1 function with a convex front [33, 34]

$$f_1(x) = x_1, \quad f_2(x) = g(1 - \sqrt{f_1/g}),$$
$$g = 1 + \frac{9\sum_{i=2}^{d} x_i}{d-1}, \quad x_i \in [0,1],\ i = 1,\ldots,30, \tag{10}$$

  where d is the number of dimensions. The Pareto optimality is reached when $g = 1$, and thus the true Pareto front is $f_2 = 1 - \sqrt{f_1}$.
- 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 [20, 36]

$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right]^2,$$
$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{d}\right)\right]^2, \tag{11}$$

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

$$x_j = \sin\left(6\pi x_1 + \frac{j\pi}{d}\right), \quad j = 2,3,\ldots,d,\ x_1 \in [0,1]. \tag{12}$$

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

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

$$E_f = ||PF^e - PF^t||^2 = \sum_{j=1}^{N}(PF_j^e - PF_j^t)^2, \tag{13}$$

where N is the number of points. For all the test functions, the true Pareto fronts have analytical forms [20, 33, 35], for example, $f_2 = 1 - \sqrt{f_1}$ for ZDT1, which makes the above calculations straightforward.

The convergence property can be viewed by following the iterations. As this measure is an absolute measure, which depends on the number of points, sometimes, it is easier to use relative measure using generalized distance

$$D_g = \frac{1}{N}\sqrt{\sum_{j=1}^{N}(PF_j - PF_j^t)^2}. \tag{14}$$

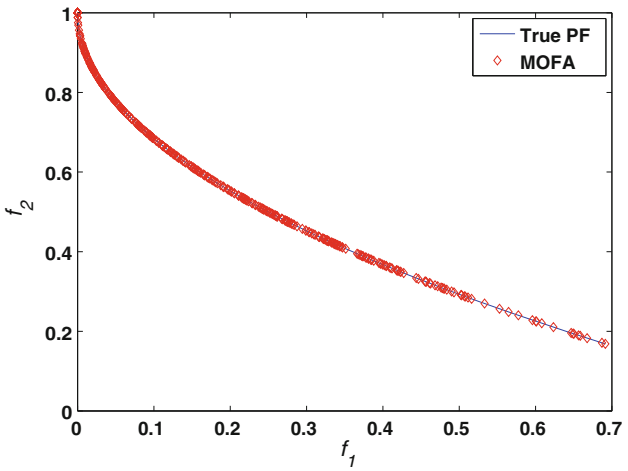

Fig. 1 Pareto front of ZDT1: a comparison of the front found by MOFA and the true Pareto front (true PF). Here the horizontal axis is f_1 while the vertical axis is f_2

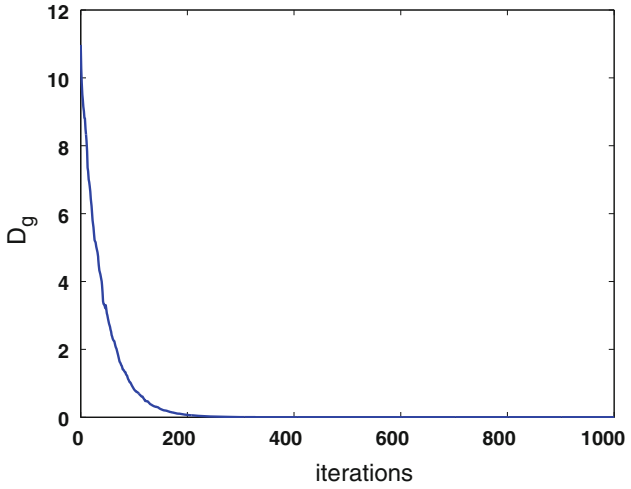

Fig. 2 Convergence of the proposed MOFA. The least-square distance (*vertical axis*) from the estimated front to the true front of ZDT1 for the first 1,000 iterations

Table 2 Summary of results

Functions	Errors (1,000 iterations)	Errors (2,500 iterations)
SCH	5.5E−09	4.0E−22
ZDT1	2.3E−6	5.4E−19
ZDT2	8.9E−6	1.7E−14
ZDT3	3.7E−5	2.5E−11
LZ	2.0E−6	7.7E−12

Figure 2 shows the exponential-like decrease of D_g as the iterations proceed. We can see clearly that our MOFA algorithm indeed converges almost exponentially. The results for all the functions are summarized in Table 2, and the estimated Pareto fronts and true fronts of other functions are shown in Figs. 3 and 4. In all these figures, the vertical axis is f_2 and the horizontal axis is f_1.

3.3 Comparison study

In order to compare the performance of the proposed MOFA with other established multiobjective algorithms, we have carefully selected a few algorithms with available results from the literature. When the results have not been available, we have implemented 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) [21], NSGA-II [37], multiobjective differential evolution (MODE) [23, 38], differential evolution for multiobjective optimization (DEMO) [22], multiobjective bees algorithms (Bees) [39] and strength Pareto evolutionary algorithm (SPEA) [37, 40]. The performance measures in terms of generalized distance D_g are summarized in Table 3 for all the aforementioned major methods.

It is clearly seen from Table 3 that the proposed MOFA obtained better results for almost all five cases, though for ZDT2 function our result is the same order (still slightly better) as that by DEMO.

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 [39, 41, 42]. Some benchmarks

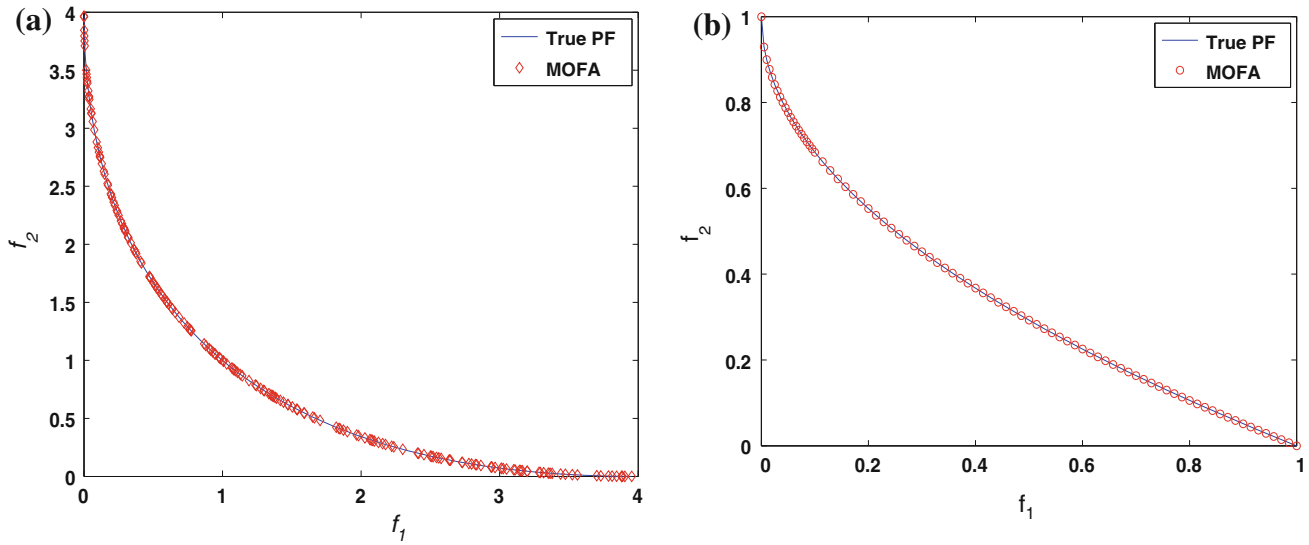

Fig. 3 Pareto front of test function **a** SCH, **b** ZDT2

Table 3 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
MOFA	1.90E−04	1.52E−04	1.97E−04	4.55E−06	8.70E−04

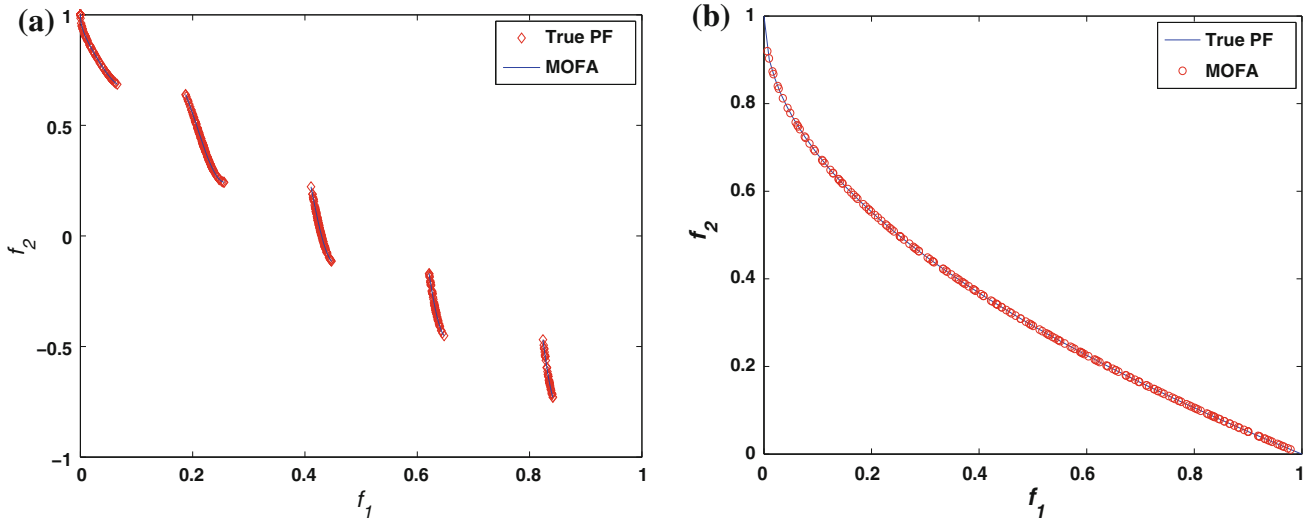

Fig. 4 Pareto front of test function **a** ZDT2, **b** LZ

have been solved by various methods, while others do not have all available data for comparison. Thus, we have chosen the welded beam design and disc brake design among the well-known benchmarks [42–44]. In the rest of this paper, we will solve these two design benchmarks using MOFA.

4.1 Welded beam design

Multiobjective design of a welded beam is a classical benchmark which has been solved by many researchers [6, 12, 44]. 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 to minimize both the overall fabrication cost and the end deflection δ.

The detailed formulation can be found in [6, 12, 44]. Here we only rewrite the main problem as

$$\begin{aligned} &\text{minimise } f_1(\boldsymbol{x}) = 1.10471w^2L + 0.04811dh(14.0 + L),\\ &\text{minimize } f_2 = \delta, \end{aligned} \tag{15}$$

subject to

$$\begin{aligned} &g_1(\boldsymbol{x}) = w - h \le 0,\\ &g_2(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \le 0,\\ &g_3(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13,600 \le 0,\\ &g_4(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30,000 \le 0,\\ &g_5(\boldsymbol{x}) = 0.10471w^2 + 0.04811hd(14 + L) - 5.0 \le 0,\\ &g_6(\boldsymbol{x}) = 0.125 - w \le 0,\\ &g_7(\boldsymbol{x}) = 6,000 - P(\boldsymbol{x}) \le 0, \end{aligned} \tag{16}$$

where

$$\begin{aligned} \sigma(\boldsymbol{x}) &= \tfrac{504,000}{hd^2}, & Q &= 6,000\left(14 + \tfrac{L}{2}\right),\\ D &= \tfrac{1}{2}\sqrt{L^2 + (w+d)^2}, & J &= \sqrt{2}\,wL\left[\tfrac{L^2}{6} + \tfrac{(w+d)^2}{2}\right],\\ \delta &= \tfrac{65,856}{30,000hd^3}, & \beta &= \tfrac{QD}{J},\\ \alpha &= \tfrac{6,000}{\sqrt{2}wL}, & \tau(\boldsymbol{x}) &= \sqrt{\alpha^2 + \tfrac{\alpha\beta L}{D} + \beta^2},\\ P &= 0.61423 \times 10^6\,\tfrac{dh^3}{6}\left(1 - \tfrac{d\sqrt{30/48}}{28}\right). \end{aligned} \tag{17}$$

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

Using the MOFA, we have solved this design problem. The approximate Pareto front generated by the 50 non-dominated solutions after 1,000 iterations are shown in Fig. 5. This is consistent with the results obtained by others [39, 44]. In addition, our results are more smooth with fewer iterations, which shows the efficiency of the proposed MOFA. A comparison of the results with those obtained by other methods is shown in Fig. 6 where we can see MOFA converged faster.

4.2 Design of a disc brake

Design of a multiple disc brake is another benchmark for multiobjective optimization [12, 18, 44]. The objectives are to minimize the overall mass and the braking time by

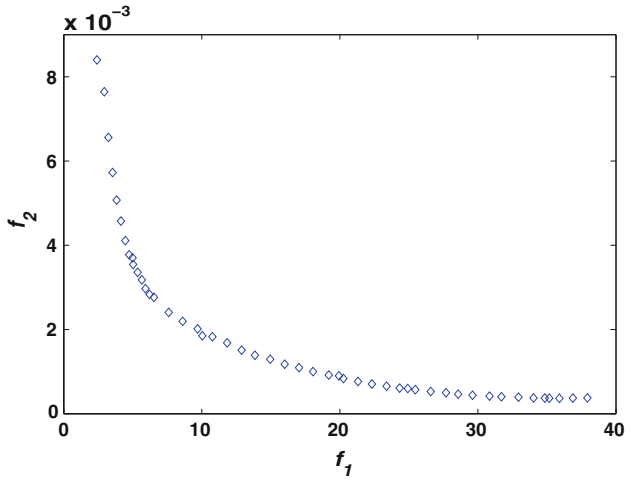

Fig. 5 Pareto front for the bi-objective beam design where the *horizontal axis* corresponds to cost and the *vertical axis* corresponds to deflection

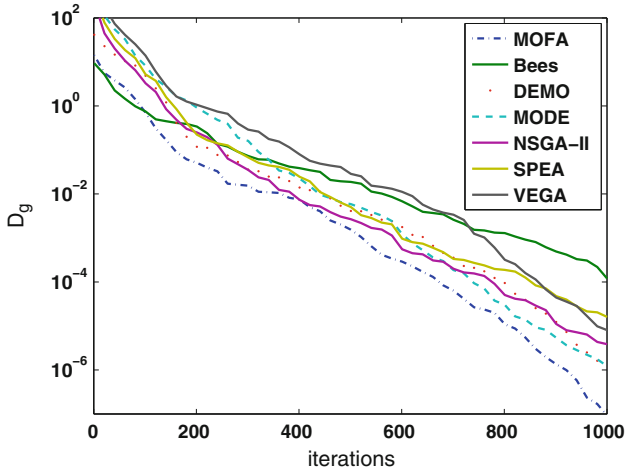

Fig. 6 Convergence comparison for the beam design

choosing optimal design variables: the inner radius r, outer radius R of the discs, the engaging force F and the number of the friction surfaces 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

$$\begin{aligned} &\text{Minimize } f_1(\boldsymbol{x}) = 4.9 \times 10^{-5}(R^2 - r^2)(s-1), \\ &f_2(\boldsymbol{x}) = \frac{9.82 \times 10^6 (R^2 - r^2)}{Fs(R^3 - r^3)}, \end{aligned} \tag{18}$$

subject to

$$\begin{aligned} g_1(\boldsymbol{x}) &= 20 - (R - r) \le 0, \\ g_2(\boldsymbol{x}) &= 2.5(s+1) - 30 \le 0, \\ g_3(\boldsymbol{x}) &= \frac{F}{3.14(R^2 - r^2)} - 0.4 \le 0, \\ g_4(\boldsymbol{x}) &= \frac{2.22 \times 10^{-3} F(R^3 - r^3)}{(R^2 - r^2)^2} - 1 \le 0, \\ g_5(\boldsymbol{x}) &= 900 - \frac{0.0266 Fs(R^3 - r^3)}{(R^2 - r^2)} \le 0. \end{aligned} \tag{19}$$

The simple limits are

$$55 \le r \le 80,\ 75 \le R \le 110,\ 1,000 \le F \le 3,000,\ 2 \le s \le 20. \tag{20}$$

The detailed formulation of this problem and background descriptions can be found in [12, 18, 41, 44]. The Pareto front of 50 solution points after 1,000 iterations obtained by MOFA is shown in Fig. 7, where we can see that the results are smooth and are the same or better than the results obtained in [44]. This can also be seen from the comparison of converge rates shown in Fig. 8.

In order to see how the proposed MOFA performs for the real-world design problems, we also solved the same problems using other available multiobjective algorithms.

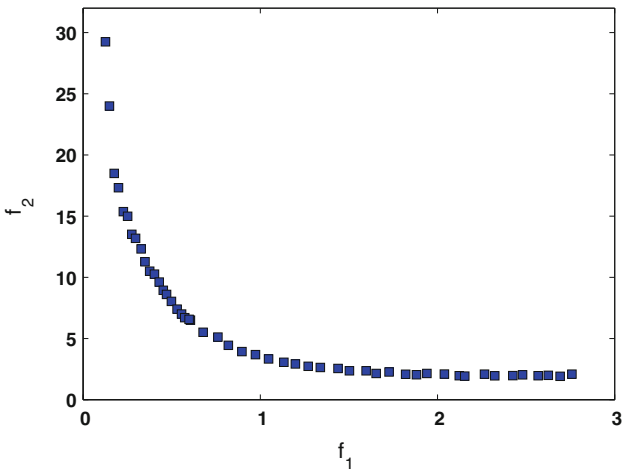

Fig. 7 Pareto front for the disc brake design where f_2 is the *vertical axis* and f_1 is the *horizontal axis*

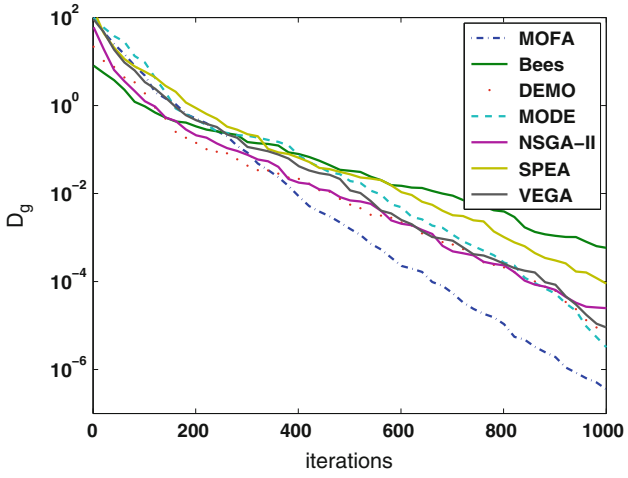

Fig. 8 Convergence comparison for the disc brake design

The comparison of the convergence rates is plotted in the logarithmic scales in Figs. 6 and 8. We can see from these two figures that the convergence rates of MOFA are of the highest in an exponentially decreasing way in both cases. This again suggests that MOFA provides better solutions in a more efficient way.

The simulations for these benchmarks and test functions suggest that MOFA 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. We have successfully formulated a new algorithm for multiobjective optimization, namely, multiobjective firefly algorithm (MOFA), based on the recently developed firefly algorithm. The proposed MOFA has been tested against a subset of well-chosen test functions and then been applied to solve design optimization benchmarks in industrial engineering.

By comparing with other algorithms, the present results suggest that MOFA is an efficient multiobjective optimizer. Further studies can focus on parametric studies which can be very useful to identify the optimal ranges of parameters for various optimization problems.

In addition, convergence analysis of MOFA will also show insight into the working mechanism of the algorithm, and this may also help to improve the proposed algorithm or even design new algorithms. For example, hybridization with other algorithms may also prove to be fruitful. Furthermore, formulation of a discrete MOFA will also be an important topic for further research.

References

1. Cagnina LC, Esquivel SC, Coello CA (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32:319–326
2. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
3. Leifsson L, Koziel S (2010) Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction. J Comput Sci 1:98–106
4. Farina M, Deb K, Amota P (2004) Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Trans Evol Comput 8:425–442
5. Coello CAC (1999) An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In: Proceedings of 1999 congress on evolutionary computation. CEC99. doi:10.1109/CEC.1999.781901
6. Deb K (1999) Evolutionary algorithms for multi-criterion optimization in engineering design. In: Evolutionary algorithms in engineering and computer science. Wiley, New York, pp 135–161
7. Geem ZW (2009) Music-inspired harmony search algorithm: theory and applications. Springer, Berlin
8. Talbi E-G (2009) Metaheuristics: from design to implementation. Wiley, New York
9. Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, Beckington
10. Yang XS (2010) Engineering optimisation: an introduction with metaheuristic applications. Wiley, New York
11. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, SAGA 2009, LNCS, vol 5792, pp 169–178
12. Gong WY, Cai ZH, Zhu L (2009) An effective multiobjective differential evolution algorithm for engineering design. Struct Multidisc Optim 38:137–157
13. Abbass HA, Sarker R (2002) The Pareto differential evolution algorithm. Int J Artif Intell Tools 11(4):531–552
14. Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7:109–124
15. Konak A, Coit DW, Smith AE (2006) Multiobjective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Saf 91:992–1007
16. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76:60–68
17. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ, pp 1942–1948
18. Osyczka A, Kundu S (1995) A genetic algorithm-based multicriteria optimization method. In: Proceedings of 1st world congress structural and multidisciplinary . Optim, pp 909–914
19. Reyes-Sierra M, Coello CAC (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2(3):287–308
20. Zhang QF, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11: 712–731
21. Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of 1st International Conference. Genetic Algorithms, pp 93–100
22. Robič T, Filipič B (2005) DEMO: differential evolution for multiobjective optimization. In: Coello Coello CA et al (eds) EMO 2005, LNCS, vol 3410, pp 520–533
23. Xue F (2004) Multi-objective differential evolution: theory and applications, PhD thesis, Rensselaer Polytechnic Institute

24. Horng M-H, Jiang TW (2010) The codebook design of image vector quantization based on the firefly algorithm. In: Computational collective intelligence, technologies and applications. LNCS, vol 6423, pp 438–447
25. Apostolopoulos T, Vlachos A (2011) Application of the firefly algorithm for solving the economic emissions load dispatch problem. Int J Combin. doi:10.1155/2011/523806http://www.hindawi.com/journals/ijct/2011/523806
26. Gandomi AH, Yang X, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89: 2325–2336
27. Yang X-S (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-inspired Comput 2(2):78–84
28. Yang XS, Deb S (2010) Engineering optimization by cuckoo search. Int J Math Model Numer Optim 1:330–343
29. Erfani T, Utyuzhnikov S (2011) Directed search domain: a method for even generation of Pareto frontier in multiobjective optimization. Eng Optim 43(5):467–484
30. Gujarathi AM, Babu BV (2009) Improved strategies of multi-objective differential evolution (MODE) for multi-objective optimization. In: Proceedings of 4th Indian international conference on artificial intelligence (IICAI-09) December 16–18
31. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidisc Optim 26: 369–395
32. Zhang QF, Zhou AM, Zhao SZ, Suganthan PN, Liu W, Tiwari S (2009) Multiobjective optimization test instances for the CEC 2009 special session and competition, Technical Report CES-487, University of Essex, UK
33. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Evol Comput 3:257–271
34. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8: 173–195
35. Zhang LB, Zhou CG, Liu XH, Ma ZQ, Liang YC (2003) Solving multi objective optimization problems using particle swarm optimization. In: Proceedings of the 2003 congress evolutionary computation (CEC'2003), vol 4. IEEE Press, Australia, pp 2400–2405
36. Li H, Zhang QF (2009) Multiobjective optimization problems with complicated Paroto sets, MOEA/D and NSGA-II. IEEE Trans Evol Comput 13:284–302
37. Deb K, Pratap A, Agarwal S, Mayarivan T (2002) A fast and elitist multiobjective algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197
38. Babu BV, Gujarathi AM (2007) Multi-objective differential evolution (MODE) for optimization of supply chain planning and management. In: IEEE congress on evolutionary computation, (CEC 2007), pp 2732–2739
39. Pham DT, Ghanbarzadeh A (2007) Multi-objective optimisation using the bees algorithm. In: 3rd international virtual conference on intelligent production machines and systems (IPROMS 2007) Whittles, Dunbeath, Scotland
40. Madavan NK (2002) Multiobjective optimization using a pareto differential evolution approach. In: Congress on evolutionary computation (CEC'2002), vol 2, pp 1145–1150
41. Gandomi AH, Yang X (2010) Benchmark problems in structural engineering. In: Koziel S, Yang XS (eds) Computational optimization, methods and algorithms, SCI, vol 356. Springer, Berlin, pp 259–281
42. Kim JT, Oh JW, Lee IW (1997) Multiobjective optimization of steel box girder brige. In: Proceedings 7th KAIST-NTU-KU trilateral seminar/workshop on civil engineering Kyoto
43. Rangaiah G (2008) Multi-objective optimization: techniques and applications in chemical engineering. World Scientific Publishing, USA
44. Ray L, Liew KM (2002) A swarm metaphor for multiobjective design optimization. Eng Optim 34(2):141–153