Engineering optimisation by cuckoo search

Xin-She Yang*

Department of Engineering,
University of Cambridge,
Trumpington Street,
Cambridge CB2 1PZ, UK
E-mail: xy227@cam.ac.uk
*Corresponding author

Suash Deb

Department of Computer Science & Engineering,
C. V. Raman College of Engineering,
Bidyanagar, Mahura, Janla,
Bhubaneswar 752054, India
E-mail: suashdeb@gmail.com

Abstract: A new metaheuristic optimisation algorithm, called cuckoo search (CS), was developed recently by Yang and Deb (2009). This paper presents a more extensive comparison study using some standard test functions and newly designed stochastic test functions. We then apply the CS algorithm to solve engineering design optimisation problems, including the design of springs and welded beam structures. The optimal solutions obtained by CS are far better than the best solutions obtained by an efficient particle swarm optimiser. We will discuss the unique search features used in CS and the implications for further research.

Keywords: algorithm; cuckoo search; engineering optimisation; metaheuristic; particle swarm optimisation.

Reference to this paper should be made as follows: Yang, X-S. and Deb, S. (2010) 'Engineering optimisation by cuckoo search', *Int. J. Mathematical Modelling and Numerical Optimisation*, Vol. 1, No. 4, pp.330–343.

Biographical notes: Xin-She Yang received his DPhil in Applied Mathematics from Oxford University, and he has been the recipient of the Garside Senior Scholar Award, Oxford University. He has been at Cambridge University, and published five books and more than 70 papers.

Suash Deb is a Professor of Computer Sciences and Engineering at C.V. Raman College of Engineering. As a Senior Member of the IEEE, he has been the recipient of Bharat Excellent Award and International Albert Einstein Award for Scientific Excellence. He is the Editor-in-Chief of *International Journal of Soft Computing & Bioinformatics*, and serves as an editor or advisor for several international journals.

1 Introduction

Most design optimisation problems in engineering are often highly non-linear, involving many different design variables under complex constraints. These constraints can be written either as simple bounds such as the ranges of material properties, or as non-linear relationships including maximum stress, maximum deflection, minimum load capacity, and geometrical configuration. Such non-linearity often results in multimodal response landscape. Subsequently, local search algorithms such as hill-climbing and Nelder-Mead downhill simplex methods are not suitable, only global algorithms should be used so as to obtain optimal solutions (Deb, 1995, Arora, 1989, Passino, 2001; Yang, 2005, 2008).

Modern metaheuristic algorithms have been developed with an aim to carry out global search, typical examples are genetic algorithms (GA) (Goldberg, 1989), particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995; Kennedy et al., 2001). The efficiency of metaheuristic algorithms can be attributed to the fact that they imitate the best features in nature, especially the selection of the fittest in biological systems which have evolved by natural selection over millions of years. Two important characteristics of metaheuristics are: intensification and diversification (Blum and Roli, 2003, Gazi and Passino, 2004; Yang, 2009). Intensification intends to search around the current best solutions and select the best candidates or solutions, while diversification makes sure that the algorithm can explore the search space more efficiently, often by randomisation.

Recently, a new metaheuristic search algorithm, called cuckoo search (CS), has been developed by Yang and Deb (2009). Preliminary studies show that it is very promising and could outperform existing algorithms such as PSO. In this paper, we will further study CS and validate it against test functions including stochastic test functions. Then, we will apply it to solve design optimisation problems in engineering. Finally, we will discuss the unique features of CS and propose topics for further studies.

2 Cuckoo search

In order to describe the CS more clearly, let us briefly review the interesting breed behaviour of certain cuckoo species. Then, we will outline the basic ideas and steps of the proposed algorithm.

2.1 Cuckoo breeding behaviour

Cuckoos 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 (Payne et al., 2005). 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 own, 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 specialised in the mimicry in colour and pattern of the eggs of a few chosen host species (Payne et al., 2005). This reduces the probability of their eggs being abandoned and thus increases their reproductivity.

Furthermore, the timing of egg-laying of some species is also amazing. Parasitic cuckoos often choose a nest where the host bird just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by its host bird (Payne et al., 2005). Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunity.

2.2 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 modelled mathematically. For example, various studies have shown that the flight behaviour of many animals and insects has demonstrated the typical characteristics of Lévy flights (Brown et al., 2007; Reynolds and Frye, 2007; Pavlyukevich, 2007).

A recent study by Reynolds and Frye (2007) shows that fruit flies or Drosophila melanogaster, explore their landscape using a series of straight flight paths punctuated by a sudden 90o turn, leading to a Lévy-flight-style intermittent scale-free search pattern. Studies on human behaviour such as the Ju/'hoansi hunter-gatherer foraging patterns also show the typical feature of Lévy flights. Even light can be related to Lévy flights (Barthelemy et al., 2008). Subsequently, such behaviour has been applied to optimisation and optimal search, and preliminary results show its promising capability (Shlesinger, 2006, Pavlyukevich, 2007).

2.3 Cuckoo search

For simplicity in describing our new CS (Yang and Deb 2009), we now use the following three idealised rules:

- 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 simplicity, this last assumption can be approximated by a fraction pa of the n nests being replaced by new nests (with new random solutions at new locations). For a maximisation problem, the quality or fitness of a solution can simply be proportional to the objective function. Other forms of fitness can be defined in a similar way to the fitness function in GA.

Figure 1 Cuckoo search (CS)

```
Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, \dots, x_d)^T$;
Initial a population of n host nests $\boldsymbol{x}_i (i = 1, 2, \dots, n)$;
while ($t$ < MaxGeneration) or (stop criterion);
    Get a cuckoo (say $i$) randomly by Lévy flights;
    Evaluate its quality/fitness $F_i$;
    Choose a nest among n (say $j$) randomly;
    if $(F_i > F_j)$,
        Replace $j$ by the new solution;
    end
    Abandon a fraction $(p_a)$ of worse nests
        [and build new ones at new locations via Lévy flights];
    Keep the best solutions (or nests with quality solutions);
    Rank the solutions and find the current best;
end while
Post process results and visualisation;
```

Based on these three rules, the basic steps of the CS can be summarised as the pseudo code shown in Figure 1.

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

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

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 $\alpha = O(1)$. 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

$$\text{Lévy} \sim u = t^{-\lambda}, (1 < \lambda \leq 3), \tag{2}$$

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.

3 Implementation and validation

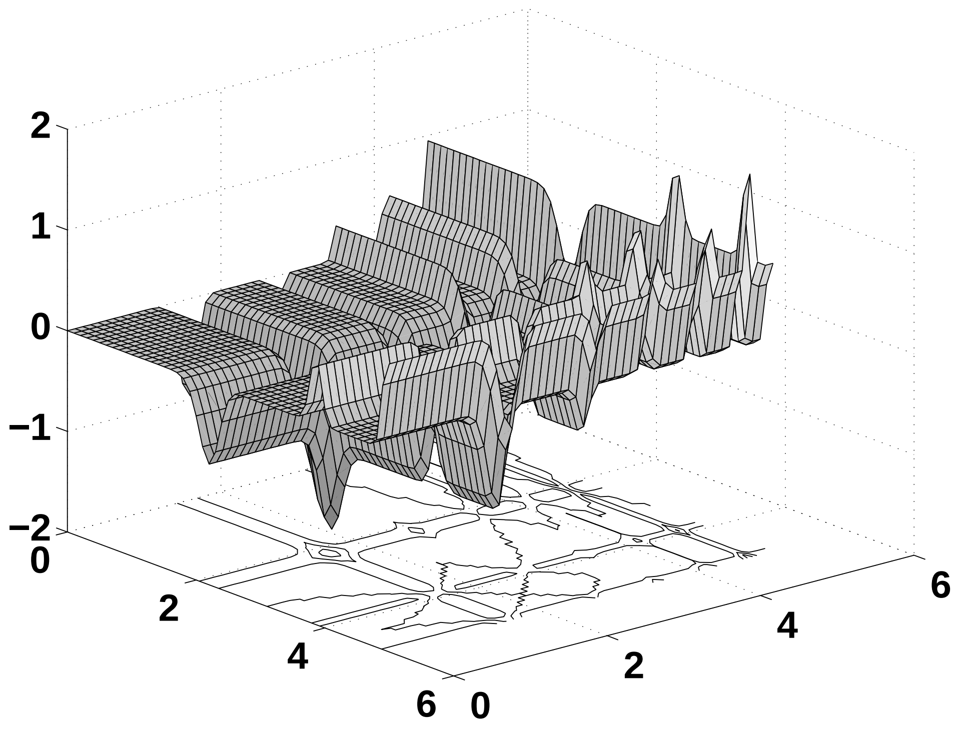
3.1 Validation and parameter studies

It is relatively easy to implement the algorithm, and then we have to benchmark it using test functions with analytical or known solutions. There are many benchmark test functions and there is no standard list or collection, though extensive descriptions of various functions do exist in literature (Floudas et al., 1999; Hedar, 2005; Molga and Smutnicki, 2005). For example, Michalewicz's test function has many local optima

$$f(\boldsymbol{x}) = -\sum_{i=1}^{d} \sin(x_i) \left[\sin\left(\frac{i x_i^2}{\pi} \right) \right]^{2m}, (m = 10), \tag{3}$$

in the domain $0 \le x_i \le \pi$ for $i = 1, 2, \ldots, d$ where d is the number of dimensions. The global minimum $f_* \approx -1.801$ occurs at (2.20319, 1.57049) for $d = 2$, while $f_* \approx -4.6877$ for $d = 5$. In the 2D case, its 3D landscape is shown Figure 2.

Figure 2 The landscape of Michaelwicz's 2D function

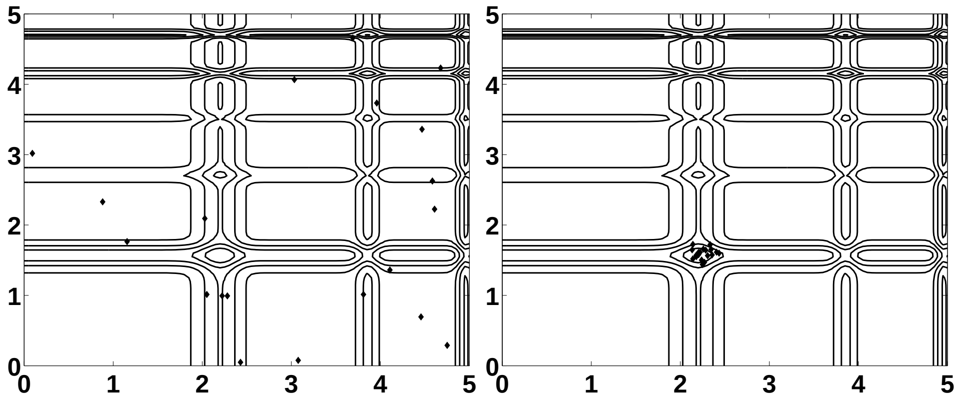


The global optimum in 2D can easily be found using CS, and the results are shown in Figure 3 where the final locations of the nests are marked with ♦. Here we have used $n = 20$ nests, $\alpha = 1$ and $p_a = 0.25$. From the figure, we can see that, as the optimum is approaching, most nests aggregate towards the global optimum. In various simulations, we also notice that nests are also distributed at different (local) optima in the case of multimodal functions. This means that CS can find all the optima simultaneously if the numbers of nests are much higher than the number of local optima. This advantage may become more significant when dealing with multimodal and multi-objective optimisation problems.

We have also tried to vary the number of host nests (or the population size n) and the probability p_a. We have used n = 5, 10, 15, 20, 50, 100, 150, 250, 500 and p_a = 0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.4, 0.5. From our simulations, we found that $n = 15 \text{ to } 25$ and $p_a = 0.15 \text{ to } 0.30$ are sufficient for most optimisation problems.

Results and analysis also imply that the convergence rate, to some extent, is not sensitive to the parameters used. This means that the fine adjustment of algorithm-dependent parameters is not needed for any given problems. Therefore, we will use $n = 20$ and $p_a = 0.25$ in the rest of the simulations, especially for the comparison studies presented later.

Figure 3 Initial locations of 20 nests in CS, and their final locations are marked with ♦



3.2 Standard test functions

Various test functions in literature are designed to test the performance of optimisation algorithms (Chattopadhyay, 1971; Schoen, 1993; Shang and Qiu, 2006). Any new optimisation algorithm should also be validated and tested against these benchmark functions. In our simulations, we have used the following test functions.

De Jong's first function is essentially a sphere function

$$f(\boldsymbol{x}) = -\sum_{i=1}^{d} x_i^2, \quad x_i \in [-5.12, 5.12], \tag{4}$$

whose global minimum $f(\boldsymbol{x}_*) = 0$ occurs at $\boldsymbol{x}_* = (0, 0, \ldots, 0)$. Here d is the dimension.

The generalised Rosenbrock's function is given by

$$f(\boldsymbol{x}) = -\sum_{i=1}^{d-1} \left[(1 - x_i)^2 + 100\left(x_{i+1} - x_i^2\right)^2 \right], \tag{5}$$

which has a unique global minimum $f_* = 0$ at $\boldsymbol{x}_* = (1, 1, \ldots, 1)$.

Schwefel's test function is multimodal

$$f(\boldsymbol{x}) = -\sum_{i=1}^{d} \left[-x_i \sin\left(\sqrt{|x_i|}\right) \right], \ -500 \le x_i \le 500, \tag{6}$$

whose global minimum $f_* = -418.9859d$ is at $x_i^* = 420.9687 (i = 1, 2, \ldots, d)$.

Ackley's function is also multimodal

$$f(\boldsymbol{x}) = -20\exp\left[-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right] + (20+e), \tag{7}$$

with the global minimum $f_* = 0$ at $\boldsymbol{x}_* = (0,0,\ldots,0)$. in the range of $-32.768 \le x_i \le 32.768$ where $i = 1,2,\ldots,d$.

Rastrigin's test function

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right], \tag{8}$$

has a unique global minimum $f_* = 0$ at (0, 0, …, 0) in a hypercube $-5.12 \le x_i \le 5.12$ where $i = 1,2,\ldots,d$.

Easom's test function has a sharp tip

$$f(x,y) = -\cos(x)\cos(y)\exp\left[-(x-\pi)^2 - (y-\pi)^2\right], \tag{9}$$

in the domain $(x,y) \in [-100,100]\times[-100,100]$. It has a global minimum of $f_* = -1$ at (π,π) in a very small region.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{10}$$

but a unique global minimum $f_* = 0$ at (0, 0, …, 0) for all $-600 \le x_i \le 600$ where $i = 1,2,\ldots,d$.

3.3 Stochastic test functions

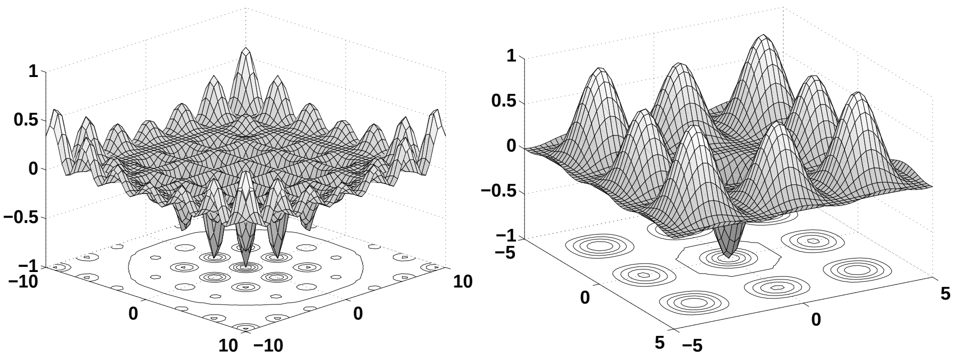
Almost all the test functions in literature are deterministic. It is usually more difficult for algorithms to deal with stochastic functions. We have designed some stochastic test functions for such a purpose.

The first test function designed by Yang (2010) looks like a standing-wave function with a region of defects

$$f(\boldsymbol{x}) = \left[e^{-\sum_{i=1}^{d}(x_i/\beta)^{2m}} - 2e^{-\sum_{i=1}^{d}\epsilon_i(x_i-\pi)^2}\right]\cdot\prod_{i=1}^{d}\cos^2 x_i, \quad m = 5, \tag{11}$$

which has many local minima and the unique global minimum $f_* = -1$ at $\boldsymbol{x}_* = (\pi,\pi,\ldots,\pi)$ for $\beta = 15$ within the domain $-20 \le x_i \le 20$ for $i = 1,2,\ldots,d$. Here the random variables $\epsilon_i (i = 1,2,\ldots,d)$ are uniformly distributed in (0, 1). For example, if all ϵ_i are relatively small (say order of 0.05), a snapshot of the landscape in 2D is shown in Figure 4, while for higher values such as 0.5 the landscape is different, also shown in Figure 4.

Figure 4 Landscape of stochastic function (11) for small ϵ (left) and large ϵ (right)



Yang's second test function is also multimodal but it has a singularity

$$f(\boldsymbol{x}) = \left(\sum_{i=1}^{d} \epsilon_i |x_i| \right) \exp\left[-\sum_{i=1}^{d} \sin\left(x_i^2\right) \right], \tag{12}$$

which has a unique global minimum $f_* = 0$ at $\boldsymbol{x}_* = (0,0,\ldots,0)$ in the domain $-2\pi \le x_i \le 2\pi$ where $i = 1,2,\ldots,d$ (Yang, 2010). This function is singular at the optimum (0, …, 0). Similarly, ϵ_i should be drawn from a uniform distribution in [0, 1] or Unif[0, 1]. In fact, using the same methodology, we can turn many determistic functions into stochastic test functions. For example, we can extend Robsenbrock's function as the following stochastic function

$$f(\boldsymbol{x}) = -\sum_{i=1}^{d-1} \left[\left(1 - x_i\right)^2 + 100\epsilon_i \left(x_{i+1} - x_i^2\right)^2 \right] \tag{13}$$

where ϵ_i should be drawn from Unif[0, 1]. Similarly, we can also extend De Jong's function into its corresponding stochastic form

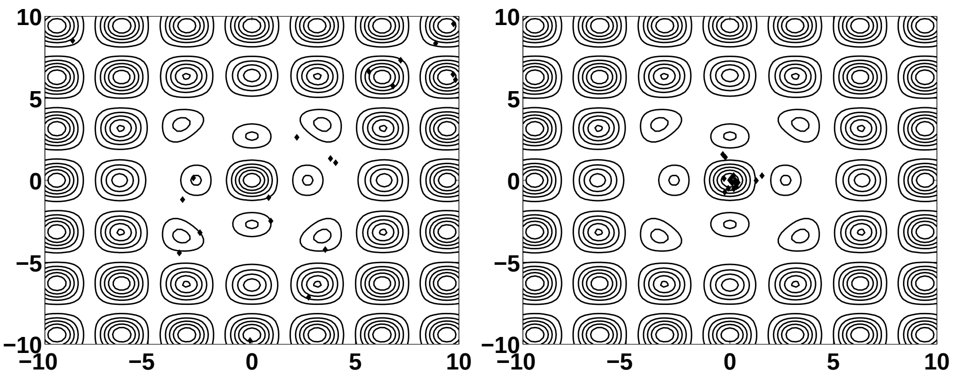
$$f(\boldsymbol{x}) = -\sum_{i=1}^{d} \epsilon_i x_i^2, \tag{14}$$

which still has the same global minimum $f_* = 0$ at (0, 0, …, 0), despite its stochastic nature due to the factor ϵ_i.. For stochastic functions, most deterministic algorithms such as hill climbing and Nelder-Mead downhill simplex method would simply fail. However, we can see later that most metaheuristic algorithms such as PSO and CS are still robust.

3.4 Simulations and comparison

Recent studies indicate that PSO can outperform GA and other conventional algorithms (Goldberg, 1989; Kennedy et al., 2001; Yang 2008).

Figure 5 The initial location of 20 nests (left) for function (11) and their final locations after 15 iterations (right)



This can be attributed partly to the broadcasting ability of the current best estimates, potentially leading to a better and quicker convergence rate towards the optimality. A general framework for evaluating statistical performance of evolutionary algorithms has been discussed in detail by Shilane et al. (2008).

Now we can compare the CS with PSO and GA for various test functions. After implementing these algorithms using Matlab, we have carried out extensive simulations and each algorithm has been run at least 100 times so as to carry out meaningful statistical analysis. The algorithms stop when the variations of function values are less than a given tolerance $\epsilon \leq 10^{-15}$. The results are summarised in Table 1 where the numbers are in the format: average number of evaluations ± one standard deviation (success rate), so 3321 ± 519 (100%) means that the average number (mean) of function evaluations is 3321 with a standard deviation of 519. The success rate of finding the global optima for this algorithm is 100%. The functions used in the Table are

1 Michaelwicz $(d = 16)$.

2 Rosenrbrock $(d = 16)$.

3 De Jong $(d = 32)$.

4 Schwefel $(d = 32)$.

5 Ackley $(d = 128)$.

6 Rastrigin.

7 Easom.

8 Griewank.

9 Yang's first stochastic function.

10 Yang's second stochastic function.

11 Generalised Robsenbrock's function with stochastic components.

12 De Jong's stochastic function. The value of $d = 16$ is used for all other cases, if not stated.

We can see that CS is much more efficient in finding the global optima with higher success rates. Each function evaluation is virtually instantaneous on a modern personal computer. For example, the computing time for 10,000 evaluations on a 3 GHz desktop is about five seconds. In addition, for stochastic functions, GAs do not perform well, while PSO is better. However, CS is far more promising.

4 Engineering design

Design optimisation is an integrated part of designing any new products in engineering and industry. Most design problems are complex and multi-objective, sometimes even the optimal solutions of interest do not exist. In order to see how the CS algorithm may perform, we now use two standard but well-known test problems.

Table 1 Comparison of CS with GAs and particle swarm optimisation

Functions	*GA*	*PSO*	*CS*
1	89325 ± 7914(95%)	6922 ± 537(98%)	3221 ± 519(100%)
2	55723 ± 8901(90%)	32756 ± 5325(98%)	5923 ± 1937(100%)
3	15232 ± 1270(100%)	10079 ± 970(100%)	3015 ± 540(100%)
4	23790 ± 6523(95%)	92411 ± 1163(97%)	4710 ± 592(100%)
5	32720 ± 3327(90%)	23407 ± 4325(92%)	4936 ± 903(100%)
6	110523 ± 5199(77%)	79491 ± 3715(90%)	10354 ± 3755(100%)
7	19239 ± 3307(92%)	17273 ± 2929(90%)	6751 ± 1902(100%)
8	70925 ± 7652(90%)	55970 ± 4223(92%)	10912 ± 4050(100%)
9	79025 ± 6312(49%)	34056 ± 4470(90%)	11254 ± 2733(99%)
10	35072 ± 3730(54%)	22360 ± 2649(92%)	8669 ± 3480(98%)
11	63268 ± 5091(40%)	49152 ± 6505(89%)	10564 ± 4297(99%)
12	24164 ± 4923(68%)	11780 ± 4912(94%)	7723 ± 2504(100%)

4.1 Spring design optimisation

Tensional and/or compressional springs are used widely in engineering (Arora, 1989; Belegundu, 1982). A standard spring design problem has three design variables: the wire diameter w, the mean coil diameter d, and the length (or number of coils) L.

$$\text{Minimise } f(\boldsymbol{x}) = (L+2)w^2 d, \tag{15}$$

subject to

$$g_1(\boldsymbol{x}) = 1 - \frac{d^3 L}{7178 w^4} \le 0,$$
$$g_2(\boldsymbol{x}) = 1 - \frac{140.45 w}{d^2 L} \le 0,$$
$$g_3(\boldsymbol{x}) = \frac{2(w+d)}{3} - 1 \le 0, \tag{16}$$
$$g_4(\boldsymbol{x}) = \frac{d(4d-w)}{w^3(12566d - w)} + \frac{1}{5108 w^2} - 1 \le 0,$$

with the following limits

$$0.05 \le w \le 2.0, \quad 0.25 \le d \le 1.3, \quad 2.0 \le L \le 15.0. \tag{17}$$

Using CS, we have obtained the following optimal solution

$$\boldsymbol{x}_* = (0.0500000, 0.2500000, 9.9876768), \tag{18}$$

with

$$f_{\min} = 0.007492298, \tag{19}$$

which is better or lower than the best solution obtained by Cagnina et al. (2008)

$$f_* = 0.012665 \quad \text{at} \quad (0.051690, 0.356750, 11.287126). \tag{20}$$

4.2 Welded beam design

The so-called welded beam design is another standard test problem for constrained design optimisation (Ragsdell and Phillips, 1976; Cagnina et al., 2008). The problem has four design variables: the width w and length L of the welded area, the depth h and thickness h of the main beam. The objective is to minimise the overall fabrication cost, under the appropriate constraints of shear stress τ, bending stress σ, buckling load P and maximum end deflection.

The problem can be written as

$$\text{Minimise } f(\boldsymbol{x}) = 1.10471 w^2 L + 0.04811 dh(14.0 + L), \tag{21}$$

subject to

$$g_1(\boldsymbol{x}) = w - h \le 0,$$
$$g_2(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \le 0,$$
$$g_3(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13{,}600 \le 0,$$
$$g_4(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30{,}000 \le 0, \tag{22}$$
$$g_5(\boldsymbol{x}) = 0.10471 w^2 + 0.04811 hd(14 + L) - 5.0 \le 0,$$
$$g_6(\boldsymbol{x}) = 0.125 - w \le 0,$$
$$g_7(\boldsymbol{x}) = 6000 - P(\boldsymbol{x}) \le 0,$$

where

$$\begin{aligned}
&\sigma(\boldsymbol{x}) = \frac{504{,}000}{hd^2}, && Q = 6000\left(14 + \frac{L}{2}\right), \\
&D = \frac{1}{2}\sqrt{L^2 + (w+d)^2}, && J = \sqrt{2}wL\left[\frac{L^2}{6} + \frac{(w+d)^2}{2}\right], \\
&\delta = \frac{65{,}856}{30{,}000hd^3}, && \beta = \frac{QD}{J}, \\
&\alpha = \frac{6000}{\sqrt{2wL}}, && \tau(\boldsymbol{x}) = \sqrt{\alpha^2 + \frac{\alpha\beta L}{D} + \beta^2} \\
&P = 0.61423 \times 10^6 \frac{dh^3}{6}\left(1 - \frac{d\sqrt{30/48}}{28}\right).
\end{aligned} \tag{23}$$

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

Using our CS, we have the following optimal solution

$$\boldsymbol{x}_* = (w, L, d, h) = (0.1701244, 10.0000000, 3.1821349, 0.2725074), \tag{24}$$

with

$$f(\boldsymbol{x}_*)_{\min} = 1.32098089. \tag{25}$$

This solution is better than the solution obtained by Cagnina et al. (2008)

$$f_* = 1.724852 \quad \text{at} \quad (0.205730, 3.470489, 9.036624, 0.205729). \tag{26}$$

We have seen that, for both test problems, CS has found the optimal solutions which are better than any solutions found so far in literature.

5 Discussions and conclusions

From the comparison study of the performance of CS with GAs and PSO, we know that our new CS in combination with Lévy flights is very efficient and proves to be superior for almost all the test problems. This is partly due to the fact that there are fewer parameters to be fine-tuned in CS than in PSO and GAs. In fact, apart from the population size n, there is essentially one parameter pa. If we look at the CS algorithm carefully, there are essentially three components: selection of the best, exploitation by local random walk and exploration by randomisation via Lévy flights globally.

The selection of the best by keeping the best nests or solutions is equivalent to some form of elitism commonly used in GAs, which ensures the best solution is passed onto the next iteration and there is no risk that the best solutions are cast out of the population. The exploitation around the best solutions is performed by using a local random walk

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \alpha\varepsilon_t. \tag{27}$$

If ε_t obeys a Gaussian distribution, this becomes a standard random walk indeed. This is equivalent to the crucial step in pitch adjustment in harmony search (Geem et al., 2001;

Yang, 2009). If ε_t is drawn from a Lévy distribution, the step of move is larger and could be potentially more efficient. However, if the step is too large, there is risk that the move is too far away. Fortunately, the elitism by keeping the best solutions makes sure that the exploitation moves are within the neighbourhood of the best solutions locally.

On the other hand, in order to sample the search space effectively so that new solutions to be generated are diverse enough, the exploration step is carried out in terms of Lévy flights. In contrast, most metaheuristic algorithms use either uniform distributions or Gaussian to generate new explorative moves (Geem et al., 2001, Blum and Roli, 2003). If the search space is large, Lévy flights are usually more efficient. A good combination of the above three components can thus lead to an efficient algorithm such as CS.

Furthermore, our simulations also indicate that the convergence rate is insensitive to the algorithm-dependent parameters such as pa. This also means that we do not have to fine tune these parameters for a specific problem. Subsequently, CS is more generic and robust for many optimisation problems, comparing with other metaheuristic algorithms.

This potentially powerful optimisation strategy can easily be extended to study multi-objecitve optimisation applications with various constraints, including NP-hard problems. Further studies can focus on the sensitivity and parameter studies and their possible relationships with the convergence rate of the algorithm. In addition, hybridisation with other popular algorithms such as PSO will also be potentially fruitful. More importantly, as for most metaheuristic algorithms, mathematical analysis of the algorithm structures is highly needed. At the moment, no such framework exists for analyzing metaheuristics in general. Any progress in this area will potentially provide new insight into the understanding of how and why metaheuristic algorithms work.

References

Arora, J. (1989) *Introduction to Optimum Design*, McGraw-Hill.

Barthelemy, P., Bertolotti, J. and Wiersma, D.S. (2008) 'A Lévy flight for light', *Nature*, Vol. 453, pp.495–498.

Belegundu, A. (1982) 'A study of mathematical programming methods for structural optimization', PhD thesis, Department of Civil Environmental Engineering, University of Iowa, USA.

Blum, C. and Roli, A. (2003) 'Metaheuristics in combinatorial optimization: overview and conceptural comparision', *ACM Comput. Surv.*, Vol. 35, pp.268–308.

Brown, C., Liebovitch, L.S. and Glendon, R. (2007) 'Lévy flights in Dobe Ju/'hoansi foragingpatterns', *Human Ecol.*, Vol. 35, pp.129–138.

Cagnina, L.C., Esquivel, S.C. and Coello, C.A. (2008) 'Solving engineering optimization problems with the simple constrained particle swarm optimizer', *Informatica*, Vol. 32, pp.319–326.

Chattopadhyay, R. (1971) 'A study of test functions for optimization algorithms', *J. Opt. Theory Appl.*, Vol. 8, pp.231–236.

Deb, K. (1995) *Optimisation for Engineering Design*, Prentice-Hall, New Delhi.

Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gumus, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A. and Scheiger, C.A. (1999) *Handbook of Test Problems in Local and Global Optimization*, Springer.

Gazi, K. and Passino, K.M. (2004) 'Stability analysis of social foraging swarms', *IEEE Trans. Sys. Man. Cyber. Part B – Cybernetics*, Vol. 34, pp.539–557.

Geem, Z.W., Kim, J.H. and Loganathan, G.V. (2001) 'A new heuristic optimization algorithm: harmony search', *Simulation*, Vol. 76, pp.60–68.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, Reading, Mass.

Hedar, A. (2005) *Test Function Web Pages*, available at http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar files/TestGO files/Page364.htm.

Kennedy, J. and Eberhart, R.C. (1995) 'Particle swarm optimization', *Proc. of IEEE International Conference on Neural Networks*, Piscataway, NJ, pp.1942–1948.

Kennedy, J., Eberhart, R.C. and Shi, Y. (2001) *Swarm Intelligence*, Academic Press.

Molga, M. and Smutnicki, C. (2005) *Test Functions For Optimization Needs*, available at http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf.

Passino, K.M. (2001) *Biomimicry of Bacterial Foraging for Distributed Optimization*, University Press, Princeton, New Jersey.

Pavlyukevich, I. (2007) 'Lévy flights, non-local search and simulated annealing', *J. Computational Physics*, Vol. 226, pp.1830–1844.

Payne, R.B., Sorenson, M.D. and Klitz, K. (2005) *The Cuckoos*, Oxford University Press.

Ragsdell, K. and Phillips, D. (1976) 'Optimal design of a class of welded structures using geometric programming', *J. Eng. Ind.*, Vol. 98, pp.1021–1025.

Reynolds, A.M. and Frye, M.A. (2007) 'Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search', *PLoS One*, Vol. 2, p.e354.

Schoen, F. (1993) 'A wide class of test functions for global optimization', *J. Global Optimization*, Vol. 3, pp.133–137.

Shang, Y.W., Qiu, Y.H. (2006) 'A note on the extended rosenrbock function', *Evolutionary Computation*, Vol. 14, pp.119–126.

Shilane, D., Martikainen, J., Dudoit, S. and Ovaska, S.J. (2008) 'A general framework for statistical performance comparison of evolutionary computation algorithms', *Information Sciences*, Vol. 178, pp.2870–2879.

Shlesinger, M.F. (2006) 'Search research', *Nature*, Vol. 443, pp.281–282.

Yang, X.S. (2005) 'Biology-derived algorithms in engineering optimizaton' in Olarius and Zomaya (Eds.): *Handbook of Bioinspired Algorithms and Applications*, Chapter 32, Chapman & Hall/CRC.

Yang, X.S. (2008) *Nature-Inspired Metaheuristic Algorithms*, Luniver Press.

Yang, X.S. (2009) 'Harmony search as a metaheuristic algorithm', in Z.W. Geem (Ed.): *Music-Inspired Harmony Search: Theory and Applications*, Springer, pp.1–14.

Yang, X.S. (2010) *Engineering Optimisation: An Introduction with Metaheuristic Applications*, John Wiley and Sons.

Yang, X.S. and Deb, S. (2009) 'Cuckoo search via Lévy flights', *Proceeings of World Congress on Nature & Biologically Inspired Computing,(NaBIC 2009, India)*, IEEE Publications, USA, pp.210–214.