Neural Comput & Applic
DOI 10.1007/s00521-013-1367-1

ORIGINAL ARTICLE

Cuckoo search: recent advances and applications

Xin-She Yang · Suash Deb

Received: 2 February 2013 / Accepted: 21 February 2013

Abstract Cuckoo search (CS) is a relatively new algorithm, developed by Yang and Deb in 2009, and the same has been found to be efficient in solving global optimization problems. In this paper, we review the fundamental ideas of cuckoo search and the latest developments as well as its applications. We analyze the algorithm and gain insight into its search mechanisms and find out why it is efficient. We also discuss the essence of algorithms and its link to self-organizing systems, and finally, we propose some important topics for further research.

Keywords Cuckoo search · Convergence · Swarm intelligence optimization · Metaheuristic · Nature-inspired algorithm

1 Introduction

Optimization concerns many disciplines with a wide range of applications. As time, money and resources are always limited, optimization is ever-increasingly more important. For example, energy-saving designs and green solutions to many industrial problems require a paradigm shift in thinking and design practice. On the other hand, people want smart and intelligent products, and computational intelligence has emerged as a promising area with potentially wide-ranging impact. Nowadays machine learning often uses optimization algorithms to enhance its learning performance, while optimization also borrows ideas from machine learning such as statistical learning theory and neural networks. In this article, we will focus on the introduction of cuckoo search as a powerful, nature-inspired metaheuristic algorithm for optimization and computational intelligence.

In almost all applications in engineering and industry, we are always trying to optimize something—whether to minimize the cost and energy consumption, or to maximize the profit, output, performance and efficiency [19, 37, 40]. The optimal use of available resources of any sort requires a paradigm shift in scientific thinking; this is because most real-world applications have far more complicated factors and parameters to affect the behavior of the system. For any optimization problem, the integrated components of the optimization process are the optimization algorithm, an efficient numerical simulator and a realistic representation of the physical processes we wish to model and optimize. This is often a time-consuming process, and in many cases, the computational costs are usually very high. Once we have a good model, the overall computation costs are determined by the optimization algorithms used for search and the numerical solver used for simulation.

Optimization algorithms are the tools and techniques for solving optimization problems with an intention to find its optimality, though such optimality is not always reachable. This search for optimality is complicated further by the fact that uncertainty is almost always present in the real-world systems. Therefore, we seek not only the optimal design but also robust design in engineering and industry. Optimal solutions, which are not robust enough, are not practical in reality. Suboptimal solutions or good robust solutions are often the choice in such cases. Optimization problems can be formulated in many ways. For example, the commonly used method of least squares is a special case of maximum-

X.-S. Yang
School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, UK

S. Deb (✉)
Cambridge Institute of Technology, Cambridge Village, Tatisilwai, Ranchi 835103, Jharkhand, India
e-mail: suashdeb@gmail.com

Published online: 09 March 2013

likelihood formulations. By far the most widely formulation is to write a nonlinear optimization problem as

$$\text{minimize } f_i(\boldsymbol{x}), (i = 1, 2, \ldots, M), \tag{1}$$

subject to the constraints

$$h_j(\boldsymbol{x}) = 0, \quad (j = 1, 2, \ldots, J), \tag{2}$$

$$g_k(\boldsymbol{x}) \leq 0, \quad (k = 1, 2, \ldots, K), \tag{3}$$

where f_i, h_j and g_k are in general nonlinear functions. Here, the design vector or design variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ can be continuous, discrete or mixed in d-dimensional space. The functions f_i are called objective or cost functions, and when $M > 1$, the optimization is multiobjective or multicriteria [37]. It is possible to combine different objectives into a single objective, though multiobjective optimization can give far more information and insight into the problem. It is worth pointing out that here we write the problem as a minimization one, it can also be written as a maximization by simply replacing $f_i(\boldsymbol{x})$ by $-f_i(\boldsymbol{x})$.

When all functions are nonlinear, we are dealing with nonlinear constrained problems. In some special cases when f_i, h_j and g_k are linear, the problem becomes linear, and we can use the widely linear programming techniques such as the simplex method. When some design variables can only take discrete values (often integers), while other variables are real continuous, the problem is of mixed type, which is often difficult to solve, especially for large-scale optimization problems.

2 The essence of an optimization algorithm

2.1 Optimization algorithm

Optimization is a process of searching for the optimal solutions to a particular problem of interest, and this search process can be carried out using multiple agents which essentially form a system of evolving agents. This system can evolve by iterations according to a set of rules or mathematical equations. Consequently, such a system will show some emergent characteristics, leading to self-organizing states which correspond to some optima in the search space. Once the self-organized states are reached, we say the system converges. Therefore, design of an efficient optimization algorithm is equivalent to mimicking the evolution of a self-organizing system [1, 17].

2.2 The essence of an algorithm

Mathematically speaking, an algorithm is a procedure to generate outputs for a given set of inputs. From the optimization point of view, an optimization algorithm generates a new solution $\boldsymbol{x}^{t+1}$ to a given problem from a known solution $\boldsymbol{x}^t$ at iteration or time t. That is

$$\boldsymbol{x}^{t+1} = \boldsymbol{A}(\boldsymbol{x}^t, \boldsymbol{p}(t)), \tag{4}$$

where $\boldsymbol{A}$ is a nonlinear mapping from a given solution d-dimensional vector $\boldsymbol{x}^t$ to a new solution vector $\boldsymbol{x}^{t+1}$. The algorithm $\boldsymbol{A}$ has k algorithm-dependent parameters $\boldsymbol{p}(t) = (p_1, \ldots, p_k)$ which can be time dependent and can thus be tuned.

To find the optimal solution $\boldsymbol{x}_*$ to a given optimization problem S with an often infinitely number of states is to select some desired states ϕ from all states ψ, according to some predefined criterion D. We have

$$S(\psi) \xrightarrow{\boldsymbol{A}(t)} S(\phi(x_*)), \tag{5}$$

where the final converged state ϕ corresponds to an optimal solution $\boldsymbol{x}_*$ of the problem of interest. The selection of the system states in the design space is carried out by running the optimization algorithm $\boldsymbol{A}$. The behavior of the algorithm is controlled by the parameters $\boldsymbol{p}$, the initial solution $\boldsymbol{x}^{t=0}$ and the stopping criterion D. We can view the combined $S + \boldsymbol{A}(t)$ as a complex system with a self-organizing capability.

The change of states or solutions of the problem of interest is through the algorithm $\boldsymbol{A}$. In many classical algorithms such as hill-climbing, gradient information of the problem S is used so as to select states, say, the minimum value of the landscape, and the stopping criterion can be a given tolerance or accuracy, or zero gradient, etc.

An algorithm can act like a tool to tune a complex system. If an algorithm does not use any state information of the problem, then the algorithm is more likely to be versatile to deal with many types of problem. However, such black-box approaches can also imply that the algorithm may not be efficient as it could be for a given type of problem. For example, if the optimization problem is convex, algorithms that use such convexity information will be more efficient than the ones that do not use such information. In order to be efficient to select states/solutions efficiently, the information from the search process should be used to enhance the search process. In many case, such information is often fed into the selection mechanism of an algorithm. By far the most widely used selection mechanism to select or keep the best solution found so far. That is, the simplest form of 'survival of the fittest'.

From the schematic representation (5) of the optimization process, we can see that the performance of an algorithm may also depend on the type of problem S it solves. On the other hand, the final, global optimality is achievable

or not (within a given number of iterations) will also depend on the algorithm used. This may be another way of stating the so-called no-free-lunch theorems.

Optimization algorithms can be very diverse, with several dozens being widely used. The main characteristics of different algorithms will only depend on the actual, nonlinear, often implicit form of $\boldsymbol{A}(t)$ and its parameters $\boldsymbol{p}(t)$.

2.3 Efficiency of an algorithm

An efficient optimization algorithm is very important to ensure that the optimal solutions are reachable. The essence of an algorithm is a search or optimization process implemented correctly so as to carry out the desired search (though not necessarily efficiently). It can be integrated and linked with other modeling components. There are many optimization algorithms in the literature and no single algorithm is suitable for all problems [36].

Algorithms can be classified as deterministic or stochastic. If an algorithm works in a mechanically deterministic manner without any random nature, it is called deterministic. For such an algorithm, it will reach the same final solution if we start with the same initial point. Hill-climbing and downhill simplex are good examples of deterministic algorithms. On the other hand, if there is some randomness in the algorithm, the algorithm will usually reach a different point every time we run the algorithm, even though we start with the same initial point. Genetic algorithms and hill-climbing with a random restart are good examples of stochastic algorithms.

Analyzing current metaheuristic algorithms in more detail, we can single out the type of randomness that a particular algorithm is employing. For example, the simplest and yet often very efficient method is to introduce a random starting point for a deterministic algorithm. The well-known hill-climbing with random restart is a good example. This simple strategy is both efficient in most cases and easy to implement in practice. A more elaborate way to introduce randomness to an algorithm is to use randomness inside different components of an algorithm, and in this case, we often call such algorithms heuristic or more often metaheuristic [37, 38].

Metaheuristic algorithms are often nature-inspired, and they are now among the most widely used algorithms for optimization. They have many advantages over conventional algorithms [13, 37]. Metaheuristic algorithms are very diverse, including genetic algorithms, simulated annealing, differential evolution, ant and bee algorithms, bat algorithm, particle swarm optimization, harmony search, firefly algorithm, cuckoo search and others [14, 18, 38, 39, 41]. Here, we will introduce cuckoo search in great detail.

3 Cuckoo search and analysis

3.1 Cuckoo search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang and Suash Deb [42–44]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Levy flights [24], rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms [42]. 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. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species).

For simplicity in describing the standard Cuckoo Search, we now use the following three idealized rules:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
- The best nests with high-quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in (0, 1)$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by a fraction p_a of the n host nests that are replaced by new nests (with new random solutions). For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms.

For the implementation point of view, we can use the following simple representations that each egg in a nest represents a solution, and each cuckoo can lay only one egg (thus representing one solution), the aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. Obviously, this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions. For this present introduction, we will use the simplest approach where each nest has only a single egg. In this case, there is no distinction between egg, nest or cuckoo, as each nest corresponds to one egg which also represents one cuckoo.

This algorithm uses a balanced combination of a local random walk and the global explorative random walk,

controlled by a switching parameter p_a. The local random walk can be written as:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{6}$$

where $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ϵ is a random number drawn from a uniform distribution and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha L(s, \lambda), \tag{7}$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{8}$$

Here, $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest. In most cases, we can use $\alpha = O(L/10)$, where L is the characteristic scale of the problem of interest, while in some cases $\alpha = O(L/100)$ can be more effective and avoid flying too far. The above equation is essentially the stochastic equation for a random walk. In general, a random walk is a Markov chain whose next status/location only depends on the current location (the first term in the above equation) and the transition probability (the second term). However, a substantial fraction of the new solutions should be generated by far field randomization and their locations should be far enough from the current best solution; this will make sure that the system will not be trapped in a local optimum [42, 43].

The literature on cuckoo search is expanding rapidly. There has been a lot of attention and recent studies using cuckoo search with a diverse range of applications [5, 6, 9–11, 13, 16, 45]. Walton et al. improved the algorithm by formulating a modified cuckoo search algorithm [34], while Yang and Deb extended it to multiobjective optimization problems [44].

3.2 Why cuckoo search is so efficient?

Theoretical studies of particle swarm optimization have suggested that PSO can converge quickly to the current best solution, but not necessarily the global best solutions [7, 15, 35]. In fact, some analyses suggest that PSO updating equations do not satisfy the global convergence conditions, and thus there is no guarantee for global convergence [26]. On the other hand, it has proved that cuckoo search satisfy the global convergence requirements and thus has guaranteed global convergence properties [35]. This implies that for multimodal optimization, PSO may converge prematurely to a local optimum, while cuckoo search can usually converge to the global optimality.

Furthermore, cuckoo search has two search capabilities: local search and global search, controlled by a switching/discovery probability. As mentioned in Sect. 3.1, the local search is very intensive with about 1/4 of the search time (for $p_a = 0.25$), while global search takes about 3/4 of the total search time. This allows that the search space can be explored more efficiently on the global scale, and consequently the global optimality can be found with a higher probability.

A further advantage of cuckoo search is that its global search uses Lévy flights or process, rather than standard random walks. As Lévy flights have infinite mean and variance, CS can explore the search space more efficiently than algorithms by standard Gaussian process. This advantage, combined with both local and search capabilities and guaranteed global convergence, makes cuckoo search very efficient. Indeed, various studies and applications have demonstrated that cuckoo search is very efficient [8, 13, 14, 28, 34, 43].

4 Applications

Cuckoo search has been applied in many areas of optimization and computational intelligence with promising efficiency. For example, in the engineering design applications, cuckoo search has superior performance over other algorithms for a range of continuous optimization problems such as spring design and welded beam design problems [13, 14, 43].

In addition, a modified cuckoo search by Walton et al. [34] has demonstrated to be very efficient for solving nonlinear problems such as mesh generation. Vazquez [33] used cuckoo search to train spiking neural network models, while Chifu et al. [5] optimized semantic web service composition processes using cuckoo search. Furthermore, Kumar and Chakarverty [20] achieved optimal design for reliable embedded system using cuckoo search, and Kaveh and Bakhshpoori [16] used CS to successfully design steel frames. Yildiz [45] has used CS to select optimal machine parameters in milling operation with enhanced results, and while Zheng and Zhou [46] provided a variant of cuckoo search using Gaussian process.

On the other hand, a discrete cuckoo search algorithm has been proposed by Tein and Ramli [30] to solve nurse scheduling problems. Cuckoo search has also been used to generate independent paths for software testing and test data generation [6, 25, 28]. In the context of data fusion and wireless sensor network, cuckoo search has been shown to be very efficient [9, 10]. Furthermore, a variant of cuckoo search in combination with quantum-based approach has been developed to solve Knapsack problems efficiently [21]. From the algorithm analysis point of view,

a conceptual comparison of cuckoo search with particle swarm optimization (PSO), differential evolution (DE), artificial bee colony (ABC) by Civicioglu and Desdo [8] suggested that cuckoo search and differential evolution algorithms provide more robust results than PSO and ABC. Gandomi et al. [13] provided a more extensive comparison study for solving various sets of structural optimization problems and concluded that cuckoo search obtained better results than other algorithms such as PSO and genetic algorithms (GA). Speed [27] modified the Lévy cuckoo search and shown that CS can deal with very large-scale problems. Among the diverse applications, an interesting performance enhancement has been obtained by using cuckoo search to train neural networks as shown by Valian et al. [31] and reliability optimization problems [32].

For complex phase equilibrium applications, Bhargava et al. [2] have shown that cuckoo search offers a reliable method for solving thermodynamic calculations. At the same time, Bulatović et al. [3] have solved a six-bar double dwell linkage problem using cuckoo search, and Moravej and Akhlaghi [22] have solved DG allocation problem in distribution networks with good convergence rate and performance. Taweewat and Wutiwiwatchai have combined cuckoo search and supervised neural network to estimate musical pitch with reduced size and higher accuracy [29].

As a further extension, Yang and Deb [44] produced a multiobjective cuckoo search for design engineering applications. For multiobjective scheduling problems, a significant progress was made by Chandrasekaran and Simon [4] using cuckoo search algorithm, which demonstrated the superiority of their proposed methodology.

Recent studies have demonstrated that cuckoo search can performance significantly better than other algorithms in many applications [13, 23, 45, 46].

5 Discussion and concluding remarks

Swarm intelligence-based algorithms such as cuckoo search and particle swarm optimization are very efficient in solving a wide range of nonlinear optimization problems, and thus have diverse applications in sciences and engineering. Some algorithms (e.g., cuckoo search) can have very good global convergence. However, there are still some challenging issues that need to be resolved in the future studies.

One key issue is that there is a significant gap between theory and practice. Most metaheuristic algorithms have good applications in practice, but mathematical analysis of these algorithms lacks far behind. In fact, apart from a few limited results about the convergence and stability about algorithms such as particle swarm, genetic algorithms and cuckoo search, many algorithms do not have theoretical analysis. Therefore, we may know they can work well in practice, we hardly understand why it works and how to improve them with a good understanding of their working mechanisms.

Another important issue is that all metaheuristic algorithms have algorithm-dependent parameters, and the actual values/setting of these parameters will largely influence the performance of an algorithm. Therefore, the proper parameter tuning itself becomes an optimization problem. In fact, parameter tuning is an important area of research [12], which deserves more research attention.

In addition, even with very good applications of many algorithms, most of these applications concern the cases with the number of design variables less than a few hundreds. It would be more beneficial to real-world applications if the number of variables can increase to several thousands or even to the order of millions.

All these challenging issues may motivate more research in the near future. There is no doubt more applications of cuckoo search will be seen in the expanding literature in the coming years.

References

1. Ashby WR (1962) Principles of the self-organizing system. In: Von Foerster H, Zopf GW Jr (eds) Principles of self-organization: transactions of the University of Illinois Symposium. Pergamon Press, London, UK, pp 255–278
2. Bhargava V, Fateen SEK, Bonilla-Petriciolet A (2013) Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations. Fluid Phase Equilibria 337:191–200
3. Bulatović RR, Bordević SR, Dordević VS (2013) Cuckoo search algorithm: a metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. Mech Mach Theory 61:1–13
4. Chandrasekaran K, Simon SP (2012) Multi-objective scheduling problem: hybrid approach using fuzzy assisted cuckoo search algorithm. Swarm Evol Comput 5(1):1–16
5. Chifu VR, Pop CB, Salomie I, Suia DS, Niculici AN (2012) Optimizing the semantic web service composition process using cuckoo search. Intell Distributed Comput V Stud Computat Intell 382:93–102
6. Choudhary K, Purohit GN (2011) A new testing approach using cuckoo search to achieve multi-objective genetic algorithm. J Comput 3(4):117–119
7. Clerc M, Kennedy J (2002) The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73
8. Civicioglu P, Besdok E (2011) A conception comparison of the cuckoo search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif Intell Rev. doi:10.1007/s10462-011-92760, 6 July (2011)
9. Dhivya M, Sundarambal M, Anand LN (2011) Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (CBPA). Int J Commun Netw Syst Sci 4(4):249–255
10. Dhivya M, Sundarambal M (2011) Cuckoo search for data gathering in wireless sensor networks. Int J Mobile Commun 9:642–656

11. Durgun I, Yildiz AR (2012) Structural design optimization of vehicle components using cuckoo search algorithm. Mater Test 3:185–188
12. Eiben AE, Smit SK (2011) Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol Comput 1:19–31
13. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35. doi:10.1007/s00366-011-0241-y
14. Gandomi AH, Yang XS, Talatahari S, Deb S (2012) Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Comput Math Appl 63(1):191–200
15. Jiang M, Luo YP, Yang SY (2007) Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Inf Process Lett 102:8–16
16. Kaveh A, Bakhshpoori T (2011) Optimum design of steel frames using cuckoo search algorithm with Levy flights. Structural design of tall and special buildings, vol 21, online first 28 Nov 2011. http://onlinelibrary.wiley.com/doi/10.1002/tal.754/abstract
17. Keller EF (2009) Organisms, machines, and thunderstorms: a history of self-organization, part two. Complexity, emergence, and stable attractors. Hist Stud Nat Sci 39(1):1–31
18. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. Piscataway, NJ, pp 1942–1948
19. Koziel S, Yang XS (2011) Computational optimization, methods and algorithms. Springer, Germany
20. Kumar A, Chakarverty S (2011) Design optimization for reliable embedded system using Cuckoo search. In: Proceedings of 3rd international conference on electronics computer technology (ICECT2011), pp 564–568
21. Layeb A (2011) A novel quantum-inspired cuckoo search for Knapsack problems. Int J Bio-inspir Comput 3(5):297–305
22. Moravej Z, Akhlaghi A (2013) A novel approach based on cuckoo search for DG allocation in distribution network. Elect Power Energy Syst 44:672–679
23. Noghrehabadi A, Ghalambaz M, Vosough A (2011) A hybrid power series—Cuckoo search optimization algorithm to electrostatic deflection of micro fixed-fixed actuators. Int J Multidiscip Sci Eng 2(4):22–26
24. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226:1830–1844
25. Perumal K, Ungati JM, Kumar G, Jain N, Gaurav R, Srivastava PR (2011) Test data generation: a hybrid approach using cuckoo and tabu search, swarm, evolutionary, and memetic computing (SEMCCO2011). Lect Notes Comput Sci 7077:46–54
26. Ren ZH, Wang J, Gao YL (2011) The global convergence analysis of particle swarm optimization algorithm based on Markov chain. Control Theory Appl (in Chinese) 28(4):462–466
27. Speed ER (2010) Evolving a Mario agent using cuckoo search and softmax heuristics. Games innovations conference (ICE-GIC), pp 1–7
28. Srivastava PR, Chis M, Deb S, Yang XS (2012) An efficient optimization algorithm for structural software testing. Int J Artif Intell 9(S12):68–77
29. Taweewat P, Wutiwiwatchai C (2013) Musical pitch estimation using a supervised single hidden layer feed-forward neural network. Expert Syst Appl 40:575–589
30. Tein LH, Ramli R (2010) Recent advancements of nurse scheduling models and a potential path. In: Proceedings of 6th IMT-GT conference on mathematics, statistics and its applications (ICMSA 2010), pp 395–409
31. Valian E, Mohanna S, Tavakoli S (2011) Improved cuckoo search algorithm for feedforward neural network training. Int J Artif Intell Appl 2(3):36–43
32. Valian E, Tavakoli S, Mohanna S, Haghi A (2013) Improved cuckoo search for reliability optimization problems. Comput Ind Eng 64:459–468
33. Vazquez RA (2011) Training spiking neural models using cuckoo search algorithm. 2011 IEEE congress on evolutionary computation (CEC'11), pp 679–686
34. Walton S, Hassan O, Morgan K, Brown MR (2011) Modified cuckoo search: a new gradient free optimization algorithm. Chaos Solitons Fractals 44(9):710–718
35. Wang F, He X-S, Wang Y, Yang SM (2012) Markov model and convergence analysis based on cuckoo search algorithm. Comput Eng 38(11):180–185
36. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
37. Yang XS (2010) Engineering optimisation: an introduction with metaheuristic applications. Wiley, New York
38. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, SAGA 2009. Lect Notes Comput Sci 5792:169–178
39. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-inspir Comput 2(2):78–84
40. Yang XS, Deb S, Fong S (2011) Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: Networked digital technologies 2011. Commun Comput Inf Sci 136:53–66
41. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):1–18
42. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. Proceedings of world congress on nature and biologically inspired computing (NaBIC 2009). IEEE Publications, USA, pp 210–214
43. Yang XS, Deb S (2010) Engineering optimization by cuckoo search. Int J Math Modell Num Opt 1(4):330–343
44. Yang XS, Deb S (2012) Multiobjective cuckoo search for design optimization. Comput Oper Res. Accepted October (2011). doi:10.1016/j.cor.2011.09.026
45. Yildiz AR (2012) Cuckoo search algorithm for the selection of optimal machine parameters in milling operations. Int J Adv Manuf Technol. doi:10.1007/s00170-012-4013-7
46. Zheng HQ, Y Zhou (2012) A novel cuckoo search optimization algorithm based on Gauss distribution. J Comput Inf Syst 8: 4193–4200