A New Cuckoo Search

Zhigang Lian, Lihua Lu, Yangquan Chen

A New Cuckoo Search

Zhigang Lian[1] and Lihua Lu[1] Yangquan Chen[2]

[1] School of Electronic and Information Engineering, Shanghai DianJi University, Shanghai, 200240,China,
lianzg@sdju.edu.cn, lulihua@sdju.edu.cn
[2] School of Engineering, University of California, Merced, CA 95343, USA
ychen53@ucmerced.edu

Abstract. In this paper, we intend to formulate a new Cuckoo Search (NCS) for solving optimization problems. This algorithm is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior of some birds and fruit flies, at the same time, combine particle swarm optimization (PSO), evolutionary computation technique. It is tested with a set of benchmark continuous functions and compared their optimization results, and we validate the proposed NCS against test functions with big size and then compare its performance with those of PSO and original Cuckoo search. Finally, we discuss the implication of the results and suggestion for further research.

Keywords: Cuckoo search, Lévy distribution, Particle swarm optimization

1 Introduction

In factual production and management, there are many complicated optimization problems. So many scientists have constantly proposed the new intelligent algorithms to solve them. For example, PSO was inspired by fish and bird swarm intelligence [1] [2]. These nature-inspired metaheuristic algorithms have been used in a wide range of optimization problems including NP-hard problems such as the travelling salesman problem (TSP) and scheduling problems [3] [4]. Based on the interesting breeding behavior such as brood parasitism of certain species of cuckoos, Yang and Deb [5] has formulated the Cuckoo Search (CS) algorithm. Yang and Deb in [6] review the fundamental ideas of CS and the latest developments as well as its applications. They analyze the algorithm, gain insight into its search mechanisms and find out why it is efficient.

In this paper, we intend to formulate a new Cuckoo Search (NCS) with different evolution mode, for solving function optimization problems. The NCS is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior of some birds and fruit flies. Moreover it integrates the PSO with some evolutionary computation technique. The PSO has particularly gained prominence due to its relative ease of operation and capability to quickly arrive at an optimal/near-optimal solution. This algorithm

is considered with the advantages of CS and PSO, avoid tendency to get stuck in a near optimal solution in reaching optimum solutions especially for middle or large size optimization problems. This work differs from existing ones at least in three aspects:

- it proposes the iterative formula of NCS, in which combines the iterative equations of CS and PSO.
- it finds the best combine parameters of NCS for different size optimization problems.
- by strictly analyzes the performance of NCS, we validate it against test functions and then compare its performance with those of PSO and CS with different random coefficient generate.

Finally, we discuss the implication of the results and suggestion for further research.

2 The new CS

2.1 The model of new CS algorithm

Yang [7] provides an overview of CS and firefly algorithm as well as their latest developments and applications. In this paper, we research on different random distribution number and their influence for algorithm. Furthermore, we also present a new CS with evolutionary pattern.

A New Cuckoo Search (NCS) combining CS and PSO is presented in this paper, in which is based on the obligate brood parasitic behavior of some cuckoo species and considered with the Lévy flight behavior of some birds and fruit flies. In the process of evolution, nests/particles/solutions of next generation share the historical and global best of the ith nests/particles/solutions. NCS improves upon the PSO and CS variation to increase accuracy of solution without sacrificing the speed of search solution significantly, and its detailed information will be given in following.

Suppose that the searching space is D-dimensional and m nests/particles/solutions form the colony. The ith nest/particle represents a D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}), (i = 1, 2, \ldots, m)$, and it means that the ith nest/particle located at X_i in the searching is a potential solution. Calculate the nest/particle/solution fitness by putting it into a designated objective function. The historical best of the ith nests/particles/solutions denotes as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, called $IBest$, and the best of the global as $P_g = (p_1, p_2, \ldots, p_D)$, called $GBest$ respectively. At the same time, some of the new nest/particle/solution should be generated combining Lévy walk around the best solution obtained so far. It will speed up the local search.

After finding above two best values, and with Lévy flight the nests/particles/solutions of NCS updates as formulas (1).

$$X_i^{(t+1)} = X_i^{(t)} + \alpha \oplus \delta * L'evy(\lambda) + (1-\delta)(R_1(P_i^{(t)} - X_i^{(t)}) + R_2(P_g^{(t)} - X_i^{(t)})); \quad (1)$$

In (1), Lévy is same as (**??**) and its step size follows a random walk Lévy distribution. A part from distant random position solution is far from the optimal solution, so it can make sure that the system does not fall into local optimal solution. Where the $\delta \in [0, 1]$ is weight index that is chosen according to different optimization problem. It reflects relatively important degree of the t generation Lévy fly, the best nests/particles/solutions of individual historical $P_i^{(t)}$ and the best nest/particle/solution of global $P_g^{(t)}$. In addition, the NCS evolution process, the global best nest/particle/solution $P_g^{(t)}$ may in $K\%$ forced preserved, $K \in (0, 100)$. Others parameters such as α and $R_1, R_2 \in (0, 1)$ are same as the ones in[2].

The search is a repeated process, and the stop criteria are that the maximum iteration number is reached or the minimum error condition is satisfied. The stop condition depends on the problem to be optimized. In the NCS evolution process, the nests/particles/solutions will be mainly updated through the three parts:

- Lévy walk;
- the distance between the best nests/particles/solutions of individual historical $P_i^{(t)}$ and its current nests/particles/solutions;
- the distance between the best nest/particle/solution of individual historical $P_i^{(t)}$ and its current nest/particle/solution

There are some significant differences of NCS, CS and PSO. Firstly, their iterative equations are not the same. The NCS integrates the advantages of CS and PSO algorithm, which share the excellent information of nests/particles/solutions. It uses some sort of elitism and/or selection which is similar to the ones used in harmony search (HS). Secondly, the randomization is more efficient as the step length is heavy-tailed, and any large step is possible. Thirdly, the parameter δ is to be turned and easy to find the highest efficiency parameters which are adapted to a wider class of optimization problems. In addition, the NCS can thus be extended to the type of meta-population algorithm.

2.2 Pseudo code of the NCS Algorithm

3 Numerical simulation

3.1 Test functions

To proof-test the effectiveness of NCS for optimization problems, 14 representative benchmark functions with different dimensions are employed to compare with CS and PSO described in Table 1, where the f_{10} is given as formula (2).

$$\begin{aligned} f_{10}(x) =& 0.1\{(sin(\pi y_1))^2 + \sum_{i=1}^{n-1}(y_i - 1)^2 \times [1 + 10(sin(\pi y_{i+1}))^2] \\ &+ (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4) \end{aligned} \tag{2}$$

Algorithm 1 Pseudo code of NCS: Part 1

Require:
nests/particles/solutions population size: PS;
maximum of generation $Endgen$;
weight index δ;
forced preserved percent;
step size α;

Ensure: optimization results;

procedure
  Generate stochastically PS size initial population;
  Evaluate each nest/particle/solutions fitness/quality;
  Generate initial global best population $P_g^{(t)}$ with the lowest fitness nest/particle/solution in the whole population;
  Generate randomly initial individual historical best population $P_i^{(t)}$;
  t:=0;
  while (t ¡ $Endge$) or (stop criterion) **do**
    t:=t+1;
    Generate next nests/particle/solutions by equations (1);
    $\beta = 3/2$;
    $\sigma = (gamma(1+\beta) * sin(\pi * \beta/2)/(gamma(1+\beta)/2) * \beta * 2^{((\beta-1)/2)))^{(1/\beta)}}$;
    $\mu = randn() * \sigma$, $\upsilon = randn()$;
    $step = \mu./abs(\varrho).^{(1/beta)}$;
    $stepsize = 0.01 * step. * (X_i^{(t)} - P_g^{(t)})$;
    $X_i^{(t+1)} = X_i^{(t)} + \delta * stepsize. * randn() + (1-\delta) * (randn() * (P_g^{(t)} - X_i^{(t)}))$;
    Evaluate nests/particles/solutions;
    {Compute each nest/particle/solution's fitness F_i in the population;
    Find new $P_g^{(t)}$ of nest/particle/solution by comparison, and update $P_g^{(t)}$;}
    Keep the best nest/particle/solution;
    Rank solutions and find the current best;
    t:=t+1;
    Choose a nest/particle/solution in population randomly;
    if $(F_i > F_j)$ **then**
      Replace j by the new nest/particle/solution;
    end if
    A fraction p_α of worse nests/particles/solutions are abandoned and new ones are built;
    The iteration calculation are as follows (X_{randn} is generated randomly):
    $K = randn() > p_\alpha$;
    $stepsize = randn() * (X_{randn} - X'_{randn})$;
    $X_i^{t+1} = X_i^t + stepsize. * K$;
    Evaluate nests/particles/solutions;
    {Compute each nest/particle/solution‘s fitness F_i in the population;
    Find new $P_g^{(t)}$ and $P_i^{(t)}$ by comparison, and update $P_g^{(t)}$ and $P_i^{(t)}$;}
    Keep the best nest/particle/solution;
    Rank solutions and find the current best;
    Randomly selected nests/particles/solutions of populations K%, forced instead them by the highest quality nest/particle/solution $P_g^{(t)}$;
  end while
end procedure

Table 1. Benchmark functions

$sphere: f_1(x)$;	$Quadric: f_2(x)$;	$Schwefe: f_3(x)$;
$Rosenbrock: f_4(x)$;	$Schwefel: f_5(x)$;	$Rastrigin: f_6(x)$;
$Ackley: f_7(x)$;	$Griewank: f_8(x)$;	$Generalized\ Penalized: f_9(x)$;
$f_{10}(x)$	$Axis\ parallel\ hyper-ellipsoid: f_{11}(x)$;	
$Sum\ of\ different\ power: f_{12}(x)$;	$Michalewicz: f_{13}(x)$;	$Schaffer\ f_7: f_{14}(x)$;

where

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a, y_i=1+\frac{1}{4}(x_i+1), x_i\in[-50,50]\\ k(-x_i-a)^m, & x_i<-a\end{cases} \tag{3}$$

3.2 Experimental results and comparison used against test function with big size

To more scientifically evaluate the proposed algorithm, we run the algorithm 32 times and compare their minimum values, mean values and standard deviations searched by NCS, CS and PSO. For each test function, the parameters and results used in experiments are listed in table 2.

Table 2. The comparison results of the PSO,CS [6]and the NCS algorithm

In CS, the p_α=0.25, and in NCS the p_α=0.25 and $k=10\%(e\sim n \quad is\times 10^n)$							
$Fun=f_1$		δ	Min/Average/Std	$Fun=f_2$		δ	Min/Average/Std
Dim=100	NCS	0.2	0.0407/5.3738/6.1344	Dim=50	NCS	0.2	5.5837/136.355/104.5021
Best=0		0.5	4.1881/4.1752/4.1881	Best=0		0.5	**2.2119**/112.231/103.8505
PS=200		**0.8**	**0.0161**/3.2514/2.9194	PS=200		0.8	3.3954/109.8777/116.003
EG=2000	PSO		4.2734e+4/6.4205e+4/1.7802e+4	EG=2000	PSO		8.0679e+3/1.837e+4/5.6383e+3
	CS		1.4453e+3/ 1.9655e+3/311.6468		CS		9.8117e+3/1.1428e+4/938.2576
$Fun=f_3$		δ	Min/Average/Std	$Fun=f_4$		δ	Min/Average/Std
Dim=50	NCS	0.2	**0.041**/0.2531/0.14356.1344	Dim=50	NCS	0.2	**3.3818**/173.9222/140.0647
Best=0		0.5	0.0439/0.2491/0.1356	Best=0		0.5	22.6772/148.6759/88.6797
PS=200		**0.8**	0.0678/0.2543/0.106	PS=200		0.8	6.7335/147.4183/81.444
EG=2000	PSO		2.8602e+3/1.7809e+4/7.2644e+3	EG=2000	PSO		9.4946e+3/1.8978e+4/7.0683e+3
	CS		18.5934/20.3249/1.2497		CS		1.0000e+10/1.0000e+10/0
$Fun=f_5$		δ	Min/Average/Std	$Fun=f_6$		δ	Min/Average/Std
Dim=30	NCS	0.2	-12569.5/-12565.5/4.8859	Dim=50	NCS	0.2	0.0394/1.1456/1.0407
Best=-12569.5		0.5	**-12569.5**/-12567/2.7239	Best=0		0.5	0.0244/0.9167/0.7278
PS=200		**0.8**	-12569.5/-12566/4.0731	PS=200		0.8	**0.0204**/1.1315/0.9799
EG=2000	PSO		0/1.2656e+4/4.9802e+4	EG=2000	PSO		1.0071e+5/1.0272e+5/1.1186e+3
	CS		-9.5134e+3/-8.9445e+3/199.3234		CS		166.4727/203.5985/16.2855
$Fun=f_7$		δ	Min/Average/Std	$Fun=f_8$		δ	Min/Average/Std
Dim=100	NCS	0.2	2.5087/51.0446/29.1489	Dim=100	NCS	0.2	0.0944/0.7814/0.3262
Best=0		0.5	**0.0328**/0.4947/0.3140	Best=0		0.5	0.2225/0.8737/0.2328
PS=200		**0.8**	0.0437/0.4701/0.3093	PS=200		0.8	**0.0777**/0.8240/0.2938
EG=2000	PSO		6.6817e+4/8.8147e+4/9.6893e+3	EG=2000	PSO		4.3883e+4/1.0700e+6/5.1800e+5
	CS		13.0822/16.9746/1.3364		CS		9.1896/11.6123/1.2995
$Fun=f_9$		δ	Min/Average/Std	$Fun=f_{10}$		δ	Min/Average/Std
Dim=100	NCS	0.2	4.9369e-5/0.0045/0.0038	Dim=100	NCS	0.2	2.1331e-4/0.0084/0.0061
Best=0		0.5	1.0745e-4/0.0034/4.9369e-5	Best=0		0.5	9.1625e-4/0.0075/0.006
PS=200		**0.8**	**2.5771e-5**/0.0034/0.0037	PS=200		**0.8**	**1.8272e-5**/0.0119/0.0156
EG=2000	PSO		2.391e+4/5.0278e+4/1.5785e+4	EG=2000	PSO		1.1415e+4/5.3205e+4/1.6434e+4
	CS		8.3128/10.159/0.8293		CS		24.833/31.9707/3.7447
$Fun=f_{11}$		δ	Min/Average/Std	$Fun=f_{12}$		δ	Min/Average/Std
Dim=100	NCS	0.2	0.0129/0.6487/0.8072	Dim=100	NCS	0.2	1.7951e-4/0.0069 /0.0077
Best=0		**0.5**	0.0057/0.4278/0.3422	Best=0		0.5	6.4493e-5/0.0069/0.0114
PS=200		**0.8**	0.0176/0.5174/0.4947	PS=200		**0.8**	1.4869e-5/0.0064/0.0064
EG=2000	PSO		2.1285e+5/2.1521e+5/1.7257e+3	EG=2000	PSO		1.6815e+4/4.8188e+4/1.6373e+4
	CS		57.2611/80.8869/10.0869		CS		1.0000e+10/1.0000e+10/0
$Fun=f_{13}$		δ	Min/Average/Std	$Fun=f_{14}$		δ	Min/Average/Std
Dim=100	NCS	**0.2**	**-39.6594**/-35.0528/1.7772	Dim=100	NCS	0.2	0.0014/0.098/0.0852
Best=?		0.5	-39.3823/-35.7054/1.9454	Best=0		**0.5**	**6.1878e-4**/0.1044/0.0954
PS=200		0.8	-39.5466/-35.3446/1.8935	PS=200		**0.8**	0.0012/0.0822/0.0723
EG=2000	PSO		2.2069e+5/2.2175e+5/493.2826	EG=2000	PSO		3.7512e+4/7.1659e+4/1.6774e+4
	CS		-34.8234l/-32.8746/0.9454		CS		168.0519/199.9071/9.6655

Remark 1: In table 2, the parameters of *Fun* and *Dim* denote function and its dimension respectively. The *Best* is this functions optimum value. The *PS* and *EG* indicate algorithm population size and their terminate generation number. The better solutions and corresponding parameters found in NCS algorithm are illustrated with bold letters. The best minimum average and standard deviation are shown in italic and underline respectively.

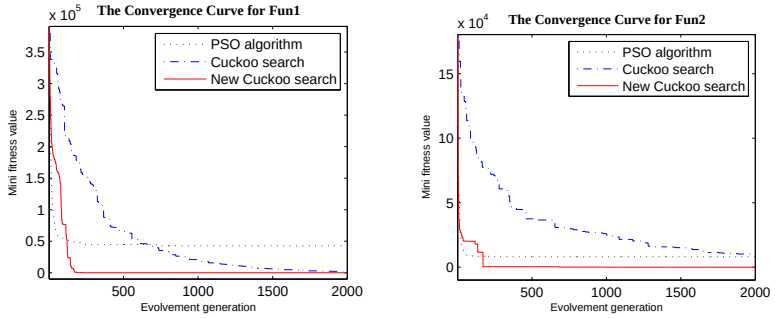
From table 2, in general it can observe that the $\delta \in [0.5, 0.8]$ has the highest performance since using them have smaller minimum and arithmetic mean in relation to solutions obtained by others. Especially the $\delta \approx 0.5$ has better search efficiency. In NCS, to optimize different problem should select various parameter δ, as a whole, it is better when $\delta \approx 0.5$. The CS optimization function f_4 and f_{12} is almost divergence. The NCS in all kinds of function optimization show excellent performance. In above function tests, the minimum value of NCS searched is NS and PSO searched 1/1000, or even 1/10000. Especially for large size problem, and function argument value in large range, optimal is strong search ability. NCS From simulation results we can obtain that the NCS is clearly better than PSO and CS for continuous non-linear function optimization problem. The NCS algorithm searches performance is strong, which can get better *the minimum*, *mean* and *standard deviation* relatively, but it computes more part $(1-\delta) * (randn() * (P_i^{(t)} - X_i^{(t)}) + randn() * (P_g^{(t)} - X_i^{(t)}))$, therefore more cost of hardware resources, although each running use a little more time, which is negligible. The convergence figures of most effective, distribution figures of run 32 times to search the optimal value of NCS comparing with PSO and CS for 14 instances are shown in Figure 1 Figure 4.

From Figure 1 Figure 4, it can discover that the convergence rate of NCS is clearly faster than the PSO and CS on every benchmark function. Especially it is more efficacious than PSO for middle and large size optimization problem. Accordingly, we can do state that the NCS is more effective than PSO and CS.

4 Conclusions and perspectives

According to shortcoming of CS and PSO algorithm especially solving the middle or large size problem, we proposed NCS. Using 14 representative instances with different dimensions and compared with the PSO and CS, the performance of the NCS shows that is efficacious for solving optimization problems.

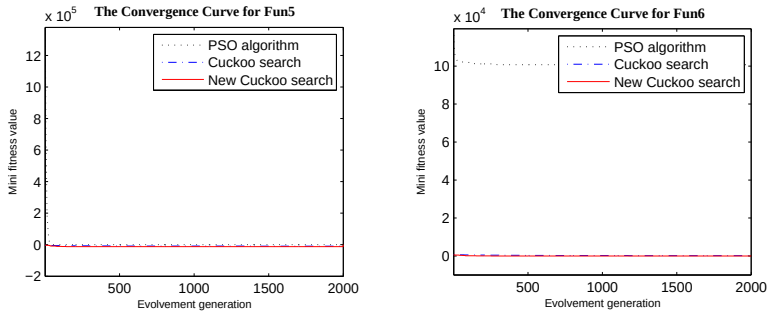
The proposed NCS algorithm can be considered as effective mechanisms from this point of view. There are a number of research directions which can be regarded as useful extensions of this research. Although this algorithm is tested with 14 representative instances, a more comprehensive computational study should be made to measure it. In the future it is maybe do experiments with different parameters and evaluate the performance of NCS. Furthermore, it finds the best parameters and usage scenarios such as TSP, scheduling, etc.



(a) f_1 (b) f_2

(c) f_3 (d) f_4

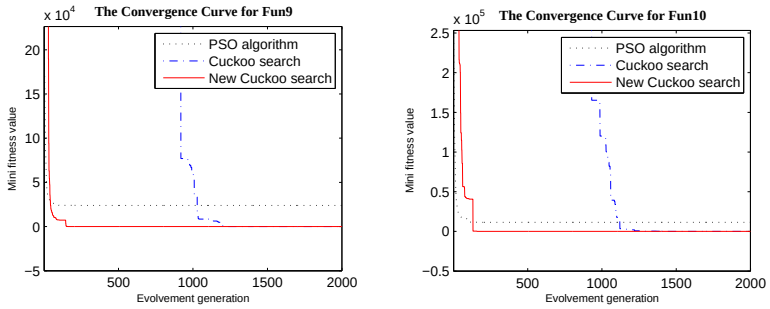
Fig. 1. Convergence figure of NCS comparing with PSO and CS for $f_1 - f_4$.



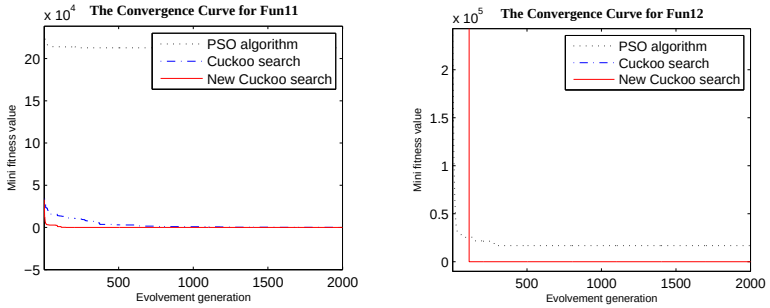
(a) f_5 (b) f_6

(c) f_7 (d) f_8

Fig. 2. Convergence figure of NCS comparing with PSO and CS for $f_5 - f_8$.

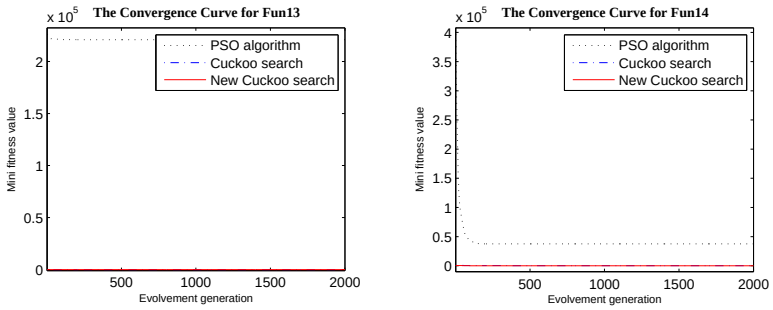


(a) f_9 (b) f_{10}



(c) f_{11} (d) f_{12}

Fig. 3. Convergence figure of NCS comparing with PSO and CS for $f_9 - f_{12}$.



(a) f_{13} (b) f_{14}

Fig. 4. Convergence figure of NCS comparing with PSO and CS for $f_{13} - f_{14}$.

5 Acknowledgements

This work is supported by Academic Discipline Project of Shanghai Dianji University (Number: 16YSXK04) the Shanghai Natural Science Foundation (Number: 14ZR1417300).

References

1. Eberhart R. Kennedy J.: A New Optimizer Using Particle Swarm Theory. In: Proc of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan. 39–43 (1995)
2. Kennedy J, Eberhart R.: Particle Swarm Optimization. In: IEEE Int1 Conf on Neural Networks, Perth, Australia. 1942–1948 (1995)
3. Lian Zhigang, Lin Weitian, Gao Yejun, Jiao bin: A Discrete Particle Swarm Optimization Algorithm for Job-Shop Scheduling Problem to Maximizing production. The International Journal of Innovative Computing, Information and Control. International Journal of Innovative Computing, Information and Control. 10(2), 729–740 (2014)
4. Lian Zhigang: A United Search Particle Swarm Optimization Algorithm for Multi-Objective Scheduling Problem. Applied Mathematical Modelling. 34, 3518–3526 (2010)
5. X.-S. Yang, S. Deb: Cuckoo search via Lvy flights. World Congress on Nature & Biologically Inspired Computing, IEEE Publications. 210C-214 (2009)
6. Xin-She Yang, Suash Deb: Cuckoo search: recent advances and applications. Neural Computing and Applications, 24(1), 169–174, (2014)
7. Xin-She Yang: Cuckoo Search and Firefly Algorithm: Overview and Analysis. Cuckoo Search and Firefly Algorithm. Studies in Computational Intelligence. 516, 1–26 (2014)