Modified cuckoo search algorithm for unconstrained optimization problems

Milan TUBA, Milos SUBOTIC, Nadezda STANAREVIC
Faculty of Computer Science
University Megatrend Belgrade
Bulevar umetnosti 29
SERBIA
tuba@ieee.org, milos.subotic@gmail.com, srna@stanarevic.com

Abstract: - This paper presents modified cuckoo search (CS) algorithm for unconstrained optimization problems. Young and Deb`s cuckoo search algorithm was successfully used on some optimization problems and there is also a corresponding code. We implemented a modified version of this algorithm where the step size is determined from the sorted rather than only permuted fitness matrix. Our modified algorithm was tested on eight standard benchmark functions. Comparison of the pure cuckoo search algorithm and our modified one is presented and it shows improved results by our modification.

Key-Words: - Cuckoo Search, Metaheuristic optimization, Unconstrained optimization, Nature inspired algorithms

1 Introduction

Optimization has been an active area of research for several decades. As many real-world optimization problems become more complex, better optimization algorithms were needed. In all optimization problems the goal is to find the minimum or maximum of the objective function. Thus, unconstrained optimization problems can be formulated as minimization or maximization of *D*-dimensional function:

$$Min\ (or\ max)\ f(x),\ x=(x_1,x_2,x_3,\ldots x_D) \qquad (1)$$

where *D* is the number of parameters to be optimized.

Many population based algorithms were proposed for solving unconstrained optimization problems. Genetic algorithms (GA), particle swarm optimization (PSO), and bee algorithms (BA) are most popular optimization algorithms which employ a population of individuals to solve the problem on hand. The success or failure of a population based algorithms depends on its ability to establish proper trade-off between exploration and exploitation. A poor balance between exploration and exploitation may result in a weak optimization method which may suffer from premature convergence, trapping in a local optima and stagnation.

GA is one of the most popular evolutionary algorithms in which a population of individuals evolves (moves through the fitness landscape) according to a set of rules such as selection, crossover and mutation [1].

PSO algorithm is another example of population based algorithms [2]. PSO is a stochastic optimization technique which is well adapted to the optimization of nonlinear functions in multidimensional space and it has been applied to several real-world problems [3].

Several metaheuristics have been proposed to model the specific intelligent behaviour of honey bee swarms [4], [5], [6], [7]. The bee swarm intelligence was used in the development of artificial systems aimed at solving complex problems in traffic and transportation [5]. That algorithm is called bee colony optimization metaheuristic (BCO), which is used for solving deterministic combinatorial problems, as well as combinatorial problems characterized by uncertainty. The artificial bee colony (ABC) algorithm is relatively new population based meta-heuristic approach based on honey bee swarm [8]. In this algorithm possible solution of the problem is represented by food source. Quality of the solution is indicated by the amount of nectar of a particular food source. Exploitation process is carried by employed and onlooker bees, while exploration is done by scouts.

[1] **This research is supported by Ministry of Science, Republic of Serbia, Project No. 44006**

A new metaheuristic search algorithm, called cuckoo search (CS), based on cuckoo bird's behaviour has been developed by Yang and Deb [9]. In this paper, we will introduce our modified version of CS algorithm and validate it against pure version on eight standard unconstrained test functions. For testing purposes, we developed our CS software named CSapp.

2 Cuckoo search algorithm

Cuckoo birds attract attention of many scientists around the world because of their unique behaviour. They have many characteristics which differentiate them from other birds, but their main distinguishing feature is 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 [9]. Cuckoos engage brood parasitism. It is a type of parasitism in which a bird (brood parasite) lays and abandons its eggs in the nest of another species. There are three basic types of brood parasitism: intraspecific brood parasitism, cooperative breeding, and nest takeover [10].

Some host birds do not behave friendly against intruders and engage in direct conflict with them. In such situation host bird will throw those alien eggs away. In other situations, more friendly hosts will simply abandon its nest and build a new nest elsewhere.

Many studies have shown that flight behaviour of many animals and insects has demonstrated the typical characteristics of Lévy flights. Study conducted by Reynolds and Frye on fruit flies shows that fruit fly *Dorsophila* explores its landscape in a quite odd manner. This fly uses a series of straight flight paths with sudden 90^0 turn, which leads to a Lévy-flight-style intermittent scale free search pattern.

Lévy-style search behaviour and random search in general has been applied to optimization and implemented in many search algorithms [11], [12]. One of such algorithms is CS [10]. Preliminary results show its promising capability.

2.1 Description of the original CS algorithm

In order to simplify the description of novel Cuckoo Search algorithm, three idealized rules can be used [10]:

- Only one egg at a time is laid by cuckoo. Cuckoo dumps its egg in a randomly chosen nest.
- Only the best nests with high quality eggs will be passed into the next generation.
- The number of available host nests is fixed. Egg laid by a cuckoo bird is discovered by the host bird with a probability $p_d \in [0,1]$. In this case, the host bird has two options. It can either throw the egg away, or it may abandon the nest.

To make the things even more simple, the last assumption can be approximated by the fraction of p_d of n nests that are replaced by new nests with new random solutions. Considering maximization problem, the quality (fitness) of a solution can simply be proportional to the value of its objective function. Other forms of fitness can be defined in a similar way the fitness function is defined in genetic algorithms and other evolutionary computation algorithms. A simple representation where one egg in a nest represents a solution and a cuckoo egg represent a new solution is used here. The aim is to use the new and potentially better solutions (cuckoos) to replace worse solutions that are in the nests. It is clear that this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions.

When generating new solutions $x^{(t+1)}$ for a cuckoo i, a Lévy flight is performed using the following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \wedge Lévy\ (\lambda), \qquad (2)$$

where α (α>0) represents a step size. This step size should be related to the scales of problem the algorithm is trying to solve. In most cases, α can be set to the value of 1. The above expression is in essence stochastic equation for a random walk which is a *Markov chain,* whose next location (status) depends on two parameters: current location (first term in Eq. 2) and probability of transition (second term in the same expression). The product ^ represents entry-wise multiplications. Something similar to entry-wise product is seen in PSO algorithm, but random walk via Lévy flight is much more efficient in exploring the search space as its step length is much longer in the long run [11].

The random step length is drawn from a Lévy distribution which has an infinite variance with an infinite mean:

$$Lévy \sim u = t^{-\lambda} \qquad (3)$$

where $\lambda \in (0,3]$.

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.

Taking into account basic three rules described above, pseudo code for CS algorithm is:

```
Start
    Objective function $f(x)$, $x= (x_1,x_2...x_u)^T$
    Generating initial population of $n$ host nests $x_i$
     ($i=1,2,...n$)
While ($t<MaxGenerations$) and (! termin.condit.)
    Move a cuckoo randomly via Lévy flights
    Evaluate its fitness $F_i$
     Randomly choose nest among $n$ available nests
     (for example $j$)
    If($F_i > F_j$) Replace $j$ by the new solution;
    Fraction $p_d$ of worse nests are abandoned and
    new nests are being built;
    Keep the best solutions or nests with quality
    solutions;
    Rank the solutions and find the current best
End while
Post process and visualize results
End
```

2.2 Modified cuckoo search algorithm

In the real world, if a cuckoo's egg is very similar to a host's eggs, then this cuckoo's egg is less likely to be discovered, thus the fitness should be related to the difference in solutions. Therefore, it is a good idea to do a random walk in a biased way with some random step sizes.

Both, original, and modified code use random step sizes. Compared to the original code, we use different function set for calculating this step size. In the original code, step size is calculated using following code expression:

*r*nests [permute1 [i]][j]- nests [permute2 [i]][j]* (4)

where *r* is random number in [0,1] range, *nests* is matrix which contains candidate solutions along with their parameters, *permute1* and *permute2* are different rows permutation functions applied on *nests* matrix.

In order to calculate the step size, instead of Equation 4, we used:

*r*nests [sorted [i]][j] - nests [permute [i]][j]* (5)

The difference is that instead of *permute1*, we used *sorted* function. This function sorts *nests* matrix by fitness of contained solutions. In this way, higher fitness solutions have slight advantage over solutions with lower fitness. This method keeps the selection pressure (the degree to which highly fit solutions are selected) towards better solutions and algorithm should achieve better results. That does not mean that high fitness solutions will flood population and the algorithm will stuck in local optimum.

At a first glance, it seems that there are some similarities between CS and hill-climbing in respect with some large scale randomization. But, these two algorithms are in essence very different. Firstly, CS is population-based algorithm in a way similar to GA and PSO, but it uses some sort of elitism and/or selection similar to that used in harmony search. Secondly, the randomization is more efficient as the step length is heavy-tailed, and any large step is possible. And finally, the number of tuning parameters is less than in GA and PSO, and thus CS can be much easier adapted to a wider class of optimization problems.

3 Experiments

In this section, we show experimental results which validated our modified CS algorithm. As mentioned above, we developed our CS software (CSapp), and all tests were run in our testing environment. For testing purposes, we also implemented original version of CS algorithm. We compared results of our modified CS algorithm with the original one. This comparison is shown in the tables within this section.

3.1 Benchmarks

To test the performance of a modified CS, eight well known benchmark functions are used here for comparison, both in terms of optimum solution after a predefined number of iterations and the rate of convergence to the optimum solution. These benchmarks are widely used in evaluating performance of population based methods but some population based methods for unconstrained optimization use different benchmarks [13].

In order to show how our algorithm performs, we used the following set of functions:

- Ackley
- DixonAndPrice
- Griewank
- Penalized
- Rastrigin
- Schwefel
- Sphere
- Step

Ackley function is a continuous, multimodal function obtained by modulating an exponential function with a cosine wave of moderate amplitude. Formulation:

$$f(x) = -20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi Xi)\right) + 20 + e$$

The global minimum value for this function is 0 and the corresponding global optimum solution is $x_{opt} = (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$.

DixonAndPrice is our second test function. The number of parameters is not determinate for this function. Formulation:

$$f(x) = (x_1 - 1)^2 + \sum_{I=2}^{n} i(2x_i^2 - x_i - 1)^2$$

Global minimum is $f5(x) = 0$.

Griewank is third test function. Definition*:*

$$f(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$$

The global minimum value for this function is 0 and the corresponding global optimum solution is $x_{opt} = (x_1, x_2, \ldots, x_n) = (100, 100, \ldots, 100)$.

Penalized function is difficult for optimization because of its combinations of different periods of the sine function. Definition:

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

The global minimum value for this function is 0 and the corresponding global optimum solution is $x_{opt} = (x_1, x_2, \ldots, x_n\text{-}1, x_n) = (-1, -1, \ldots, -1, -5)$.

Rastrigin function is based on *Sphere* function with the addition of cosine modulation to produce many local minima. Definition:

$$f(x) = 10n + \sum_{i=1}^{n}(X_i^2 - 10\cos(2\pi x_i))$$

The global minimum value for this function is 0 and the corresponding global optimum solution is $x_{opt} = (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)$.

Schwefel function as our sixth benchmark. Definition:

$$f(x) = \sum_{i=1}^{n} -X_i \sin(\sqrt{|X_i|}$$

This function has a value - 418.9828 and at its global minimum (420.9867, 420.9867,…, 420.9867). Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction. Test area is usually restricted to hypercube $-500 \le xi \le 500,\ i = 1, \ldots, n$.

Sphere function that is continuous, convex and unimodal. Formulation:

$$f(x) = \sum_{i=1}^{n} X_i^2$$

Global minimum value for this function is 0 and optimum solution is $x_{opt} = (x_1, x_2, \ldots, x_n) = (0, 0, .., 0)$.

Finally, *Step* is our eighth benchmark function. Definition:

$$f(x) = \sum_{i=1}^{n}(|xi + 0.5|)$$

This function represents the problem of flat surfaces. It is very hard for algorithms without variable step sizes to conquer flat surfaces problems because there is no information about which direction can provide optimal solution.

3.2 Experimental results and algorithm's settings

We tried to vary the number of host nests (population size n) and the probability of discovery p_d. We have used different settings for n (5, 10, 15, 20, 50, 100, 150, 250, 500) and for p_d. (0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.4, 0.5) . From test phase simulations, we found that $n = 25$ and $p_d = 0.25$ are sufficient for most optimization problems. This means that the fine parameters tuning are not needed for any given problem. Therefore, we used fixed $n = 25$ and $p_d = 0.25$ for all given problems.

We tested each benchmark function four times with 5, 10, 50 and 100 parameters. For every test, we carried on 30 runs with 500 cycles per each run. We printed out best and mean results as well the standard deviation within the set of 30 runs. Parameter settings are in Table 1.

Parameter	Value
Runtime	30
Max Cycle	500
N	25
D	5/10/50/100
P_d	0.25

Table 1: Parameter settings for benchmark functions.

Tests were done on Intel Core2Duo T8300 mobile processor with 4GB of RAM on Windows 7 x64 Ultimate Operating System and NetBeans 6.9.1 IDE (Integrated Development Environment).

Experimental results with 5, 10, 50 and 100 parameters are shown in Tables 2, 3, 4 and 5 respectively. All tables have two columns with results in order to make side by side comparison. In the first column, we show results of original, and in the second, results of modified algorithm. We showed values for best, mean and standard dev.

Function		Original	Modified
Ackley	Best	1.17E-12	0.05E-12
	Mean	8.52E-11	2.01E-11
	Stdev.	2.23E-10	1.12E-10
DixonAndPrice	Best	3.40E-1	2.32E-2
	Mean	0.057	0.008
	Stdev.	0.097	0.062
Griewank	Best	5.26E-25	1.66E-27
	Mean	4.35E-19	9.05E-20
	Stdev.	1.13E-17	9.31E-17
Penalized	Best	1.29E-6	5.49E-7
	Mean	2.04E-5	1.12E-5
	Stdev.	2.17E-5	0.11E-5
Rastrigin	Best	3.28E-18	1.03E-19
	Mean	1.77E-16	3.52E-18
	Stdev.	9.56E-16	5.52E-18
Schwefel	Best	0.391	0.352
	Mean	134.333	125.886
	Stdev	68.546	56.002
Sphere	Best	8.31E-26	4.63E-28
	Mean	2.50E-22	8.45E-24
	Stdev.	7.88E-22	4.23E-23
Step	Best	3.87E-6	1.23E-6
	Mean	3.57E-5	1.44E-5
	Stdev.	4.60E-5	1.05E-5

Table 2: Experimental results for 5 parameters

As we can see from the Table 2, modified CS algorithm has outperformed the original algorithm in all eight benchmark functions in tests with 5 parameters.

Function		Original	Modified
Ackley	Best	1.65E-7	3.56E-8
	Mean	1.10E-6	9.13E-7
	Stdev.	9.30E-7	0.05E-8
DixonAndPrice	Best	0.593	0.455
	Mean	0.664	0.602
	Stdev.	0.013	0.004
Griewank	Best	3.15E-18	5.67E-19
	Mean	5.03E-15	7.56E-16
	Stdev.	9.89E-15	8.92E-16
Penalized	Best	3.20E-4	8.45E-4
	Mean	2.19E-4	1.22E-3
	Stdev.	9.93E-3	1.02E-2
Rastrigin	Best	1.77E-15	5.03E-17
	Mean	4.72E-9	8.51E-11
	Stdev.	1.16E-8	5.81E-10
Schwefel	Best	661.309	628.205
	Mean	883.518	862.334
	Stdev.	124.294	102.004
Sphere	Best	4.29E-15	1.09E-16
	Mean	6.04E-13	2.32E-14
	Stdev.	9.66E-13	4.29E-14
Step	Best	8.12E-5	1.05E-5
	Mean	3.33E-4	1.23E-4
	Stdev.	7.66E-4	2.52E-4

Table 3: Experimental results for 10 parameters

If we compare test results with 10 and 50 parameters (see Tables 3 and 4), we can observe that the performance is almost the same.

Function		Original	Modified
Ackley	Best	3.91E-5	0.72E-5
	Mean	5.39E-4	3.21E-4
	Stdev.	4.19E-4	2.09E-4
DixonAndPrice	Best	0.667	0.625
	Mean	0.674	0.661
	Stdev.	0.022	0.015
Griewank	Best	1.58E-9	9.92E-10
	Mean	1.60E-8	0.41E-8
	Stdev.	2.50E-8	1.02E-8
Penalized	Best	0.115	0.193
	Mean	0.206	0.288
	Stdev.	0.039	0.076
Rastrigin	Best	8.53E-8	2.39E-14
	Mean	8.01E-6	2.39E-10
	Stdev.	1.23E-5	0.03E-7
Schwefel	Best	9781.549	9653.209
	Mean	10813.984	10012.562
	Stdev.	361.787	307.905
Sphere	Best	2.36E-8	3.02E-10
	Mean	4.64E-6	9.45E-8
	Stdev.	8.43E-6	9.49E-8
Step	Best	3.392	3.013
	Mean	4.141	3.994
	Stdev.	0.532	0.486

Table 4: Experimental results for 50 parameters

Function		Original	Modified
Ackley	Best	1.93E-4	0.31E-4
	Mean	0.035	0.026
	Stdev.	0.005	0.002
DixonAndPrice	Best	0.668	0.598
	Mean	0.698	0.676
	Stdev.	0.057	0.045
Griewank	Best	2.56E-9	0.61E-9
	Mean	2.81E-7	1.05E-7
	Stdev.	3.11E-7	1.85E-7
Penalized	Best	0.402	0.502
	Mean	0.474	0.594
	Stdev.	0.047	0.092
Rastrigin	Best	4.43E-6	1.91E-11
	Mean	1.31E-4	0.06E-8
	Stdev.	1.64E-4	1.12E-5
Schwefel	Best	24983.536	24339.456
	Mean	26902.073	26233.125
	Stdev.	664.865	605.009
Sphere	Best	3.12E-6	9.83E-8
	Mean	4.64E-5	7.24E-7
	Stdev.	5.28E-5	6.09E-6
Step	Best	12.905	12.302
	Mean	15.647	15.133
	Stdev.	0.847	0.701

Table 5: Experimental results for 100 parameters

In tests with 10, 50 and 100 parameters (see Tables 3, 4 and 5 respectively), the original algorithm outperformed modified only in tests for *penalized* function.

As can be seen from presented tables, for almost all test functions, modified CS has performed slightly better than the original algorithm. Although there is no substantial improvement, presented performance benefit should not be neglected.

Modified algorithm, as well as original, establishes a fine balance of randomization and intensification with small number of control parameters. As for any metaheuristic algorithm, a good balance of intensive local search strategy and an efficient exploration of the whole search space will usually lead to a more efficient algorithm [10]. On the other hand, there are only two parameters in this algorithm, the population size n, and p_d. Once n is fixed, p_d essentially controls the elitism and the balance of the randomization and local search. Few parameters make an algorithm less complex and thus potentially more generic.

4 Conclusion

In this paper, we present an improved CS algorithm for unconstrained optimization problems. The capability of this algorithm was investigated through the performance of several experiments on well-known test problems. The results obtained by the modified CS algorithm are satisfying.

As we can from the comparative analysis between original and modified CS algorithm for unconstrained optimization problems, new algorithm has performed slightly better in seven of eight benchmark functions. For only one function, standard CS algorithm outperformed the modified one. Future work will include investigation of the modified CS performance in other benchmark and real life problems.

References:

[1] Mitchell Melanie, *Introduction to Genetic Algorithms,* MIT Press, 1999, pp. 158.

[2] Muhammad S. Yousuf, Hussain N. Al-Duwaish, Zakaryia M. Al-Hamouz, *PSO based Single and Two Interconnected Area Predictive,* WSEAS Transactions on system and control, Vol. 5, Issue 8, 2010, pp. 677-690.

[3] Angel E. Mu˜noz Zavala, Arturo H. Aguirre, Enrique R. Villa Diharce, *Constrained Optimization via Particle Evolutionary Swarm Optimization Algorithm (PESO),* Proceedings of the 2005 conference on Genetic and evolutionary computation, Article in Press, 2005, doi:10.1145/1068009.1068041, pp. 209-216.

[4] Reza A., Alireza M., Koorush Z., *A novel bee swarm optimization algorithm for numerical function optimization,* Communications in Nonlinear Science and Numerical Simulation, Vol. 15, Issue 10, 2009, pp. 3142-3155.

[5] Teodorovic, D., Dell'Orco M., *Bee colony optimization-a cooperative learning approach to complex transportation problems*, Proceedings of the 16th Mini - EURO Conference and 10th Meeting of EWGT - Advanced OR and AI Methods in Transportation, 2005, pp. 51-60.

[6] Drias, H., Sadeg, S., Yahi, S, *Cooperative bees swarm for solving the maximum weighted satisfiability problem,* Lecture notes in computer science, Volume 3512, 2005, pp. 318 – 325.

[7] L. Jiann-Horng, H. Li-Ren, *Chaotic bee swarm optimization algorithm for path planning of mobile robots*, Proceedings of the 10th WSEAS international conference on evolutionary computing, 2009, pp. 84-89.

[8] Behriye Akay, Dervis Karaboga, *A modified Artificial Bee Colony algorithm for real parameter optimization*, Information Sciences, Article in Press, doi:10.1016/j.ins.2010.07.015, 2010.

[9] Yang, X. S. and Deb, S., *Cuckoo search via Lévy flights*, in: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), 2009, pp. 210-214.

[10] Yang, X.S., and Deb, S. *Engineering Optimisation by Cuckoo Search*, Int. J. of Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, 2010, pp. 330–343.

[11] Mahmood M. Nesheli, Othman C., Arash Moradkhani R., *Optimization of Traffic Signal Coordination System on Congestion: A Case Study*, WSEAS Transactions on Advances in Engineering Education, Vol. 6, Issue 7, 2009, pp. 203-212.

[12] L. Ozdamar: *A dual sequence simulated annealing algorithm for constrained optimization,* Proceedings of the 10th WSEAS International Conference on applied mathematics, 2006, pp. 557-564.

[13] Rogerio A. Flauzino, Ivan N. Da Silva, *Tuning of Fuzzy Inference Systems Through Unconstrained Optimization Techniques*, 2002 WSEAS Int. Conf. on System Science, Applied Mathematics & Computer Science, and Power Engineering Systems, 2002, pp. 2611-2616.