Wuhan University Journal of Natural Sciences

2017, Vol.22 No.4, 289-294

Article ID 1007-1202(2017)04-0289-06

DOI 10.1007/s11859-017-1249-y

An Improved Cuckoo Search Algorithm for Multi-Objective Optimization

□ TIAN Mingzheng[1], HOU Kuolin[2], WANG Zhaowei[1] ,WAN Zhongping[1†]

1. School of Mathematics and Statistics, Wuhan University, Wuhan 430072, Hubei, China;

2. Faculty of Foundational Education, Liming Vocational University, Quanzhou 362000, Fujian, China

Abstract: The recently proposed Cuckoo search algorithm is an evolutionary algorithm based on probability. It surpasses other algorithms in solving the multi-modal discontinuous and nonlinear problems. Searches made by it are very efficient because it adopts Levy flight to carry out random walks. This paper proposes an improved version of cuckoo search for multi-objective problems (IMOCS). Combined with nondominated sorting, crowding distance and Levy flights, elitism strategy is applied to improve the algorithm. Then numerical studies are conducted to compare the algorithm with DEMO and NSGA-II against some benchmark test functions. Result shows that our improved cuckoo search algorithm convergences rapidly and performs efficienly.

Key words: multi-objective optimization; evolutionary algorithm; Cuckoo search; Levy flight

CLC number: O 224

Received date: 2017-01-10

Foundation item: Supported by the National Natural Science Foundation of China (71471140)

Biography: TIAN Mingzheng, male, Master candidate, research direction: optimization algorithm. E-mail: alvin_tian@foxmail.com

† To whom correspondence should be addressed. E-mail: zpwan-whu@126.com

0 Introduction

Multi-objective optimization (also vector optimization) plays an extremely important part whether in scientific research or engineering. A multi-objective optimization problem with n decision variables and m objectives can be defined as follows[1]:

$$\begin{cases}\min & \boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x})=(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_m(\boldsymbol{x}))^{\mathrm{T}}\\ \text{s.t.} & g_i(\boldsymbol{x})\leqslant 0,\ \ i=1,2,\cdots,q\\ & h_i(\boldsymbol{x})=0,\ \ i=1,2,\cdots,p\end{cases}\tag{1}$$

$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)\in X$ is a decision vector in the decision space. $\boldsymbol{y}=(y_1,y_2,\cdots,y_m)\in Y$ is the objective vector in the objective space. q and p are the numbers of inequality and equality constraints, respectively.

In general, there are two ways to tackle with multi-objective optimization. One is to combine the objective functions to form a single objective function, such as weighted sum method and utility theory. But the problem is that we cannot describe the decision maker's preference accurately, and we may lose the solutions on a concave Pareto front.

The other method is trying to find all the solutions in the Pareto optimal set, which contains all the non-dominated solutions. The Pareto optimal set is more acceptable because decision maker can check the trade-offs among these solutions with respect to their own preference.

To find the Pareto optimal set is not an easy job as some traditional methods is inefficient. But the metaheuristic algorithms based on mimicking nature outperform other algorithms. Lots of multi-objective evolutionary algorithms were proposed such as MOGA [2], NSGA [3], SPEA2 [4] etc. Especially, NSGA-II [5] and MOPSO[6] are famous untill today. New strategies such as Pareto based selection and crowding distance were introduced to these population based algorithms to achieve a higher performance.

Cuckoo search is a relatively new algorithm, which has many advantages such as fast convergence, diversity in distribution of solutions, exploitation as well as exploration. This probability based metaheuristic algorithms was proved to be efficient [7].

In this paper, combining the elitism strategy, we propose an improved multi-objective cuckoo search (IMOCS) algorithm. Numerical studies are performed against some benchmark problems, which also shows the proposed algorithm is efficient and time-saving.

The rest of the paper is organized as follows. In Section 1 we introduce the original cuckoo search algorithm and explain why it is efficient. In Section 2 we propose IMOCS and compare it with other famous algorithms. Numerical studies are presented in Section 3 together with our analysis. Section 4 contains conclusion and further discussion about the algorithm.

1 Cuckoo Search

1.1 Original Cuckoo Search

Cuckoo search imitates the parasite behaviors of cuckoo to search for the optimal solutions. At first, a fixed amount of nests are placed in the search space, and each nest has one "egg (candidate solution)". Cuckoos search the whole decision space and record the fitness value of all the encountered candidate solution to find the optimal. The search pattern used is called Levy flight, which is widely adopted by birds, insect, herbivores and fishes in real world [8]. Its character is a series of straight flight paths punctuated by a sudden 90° turn. Drawn from the levy distribution, the step length can be occasionally very big, which makes it more suitable for global search. As Yang indicated, Levy flight is more efficient than random walk and Brownian movement [9].

Single objective cuckoo search explores the whole search space by Levy flight and exploits the local area intensively by random walk. A parameter called "Discovery probability"(Pr) is introduced to balance the global and local search, namely the intensification and diversification. The formula below describes how a Levy flight is performed.

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \alpha \oplus \text{Levy}(\beta) \tag{2}$$

where α is the step length decided by the specific problem. In most cases, Yang suggests we can use $O(1)$ [10]. So α can be set by the difference of different solutions as:

$$\alpha = \alpha_0 (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}) \tag{3}$$

where α_0 is a constant. Two solutions are randomly chosen from the current population. This comes from the fact that similar eggs are less likely to be discovered.

Levy flight is supposed to provide a "random walk", where its step sizes are drawn from a Levy distribution.

$$\text{Levy} \sim u = t^{-1-\beta}, \ 0 \leqslant \beta \leqslant 2 \tag{4}$$

Levy distribution was found to have an infinite variance and an infinite mean, which actually makes the consecutive jumps be a random walk process which obeys a power-law step-length distribution with a heavy tail.

Furthermore, some of the worst solutions will be discarded with a probability of Pr to make room for new nests. New solutions can be obtained by random walk and mixing. Different permutation of existing solutions generates new mixing solutions.

Apparently, our focus now is to generate the step lengths needed by the Levy flights. Yang *et al* gave a simple way below[7]:

$$s = \alpha_0 (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}) \oplus \text{Levy}(\beta) \sim \alpha_0 \frac{u}{|v|^{1/\beta}} (\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}) \tag{5}$$

where u and v obey the normal distributions below:

$$u \sim N(0, \sigma_u^2), \ v \sim N(0, \sigma_v^2) \tag{6}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \ \sigma_v = 1 \tag{7}$$

where Γ is the standard Gamma function.

1.2 Standard Multi-Objective Cuckoo Search

By modifying the Levy flight and the domination rules, we can get the multi-objective cuckoo search algorithms. The MOCS proposed by Yang *et al* [7] asserts to keep the offspring only when they dominate their parents. This is too harsh for offspring because there are child solutions on the Pareto front, but they may not dominate their parents. Obviously we want to keep those child solutions.

The following MOCS algorithm combing the non-dominated sort and crowding distance was proposed by Syberfeldt *et al* [11] and Wang *et al* [12]. Here we present some basic concepts of the non-dominated sort and crowding distance. This process aims to compare and select better solutions generated by Levy flights for further iterations.

① **Non-dominated sort:** For solution p in population, test every other solution in the population. Record the solutions that p dominates and record the number of solutions dominating p. Rank the non-dominated solutions as the first front. Then for solutions dominated by the first front, the number which dominates it minuses the number in the first front dominates it. Rank the non-dominated solutions as the second front, and so on.

② **Crowding distance:** This approach is used to select solutions in the same front. Sort each solution in front i by the value of m-th objective. The difference of the m-th value prior and inferior to it divided by the range of m-th objective is the m-th distance. Crowding distance is the sum of all the objectives' distance [5]. Here is the pseudo-code for the multi-objective cuckoo search [11]:

```
Generate N parent solutions randomly and evaluate them
while stopping criterion not met
  Levy flight among N parent population to generate N child population
  Evaluate the child population
  Combine the parent and child population and sort
  for (each solution in 2N) and (next generation<N)
    add solutions to next generation according to their front
    number till next generation = N
    sort by crowding distance to fill the next generation to N
  end for
  Generate N solutions randomly with the discovery probability Pr
  Sort the 2 N population to select N solutions for the next step
  the chosen N solutions are the next generation
 end while
print the results
```

1.3 Advantages of the Cuckoo Search

In fact cuckoo search makes a good combination of all the efficient techniques in the literatures, which makes it more promising. For instance, particle swarm optimization (PSO) algorithm updates the speed vector by calculating the difference between the current solution and the current global best. This is the main procedure to ensure randomization. But this technique actually constrains the step size by using difference. In cuckoo search algorithms, due to the infinite mean and variance of Levy distribution, the step size consists of many small ones and occasionally big ones make the search even more efficient, especially for nonlinear problem with many local optimums [7].

2 Improved Multi-Objective Cuckoo Search

2.1 Main Ideas

Due to the step sizes drawn from the Levy distribution, many small steps already make the local search efficient. There is no need to use the "Discovery Probability" Pr to generate random solutions in the process of exploiting local areas. This strategy not only searches repeatedly but also costs one more sort and selection operation. This leads to a relatively high computational complexity, and takes more time.

In the case of multi-objective optimization, the relation between solutions is more complicated. We cannot do the Levy flights referring to the global best, because there is no global best. There is a situation where no solution dominates any other, e.g. solutions on the same Pareto front. Ordinary methods re-permute the current population, and then Levy flights are performed referring to the randomly permuted solutions. But the solution we referred to may not be non-dominated, even there might be many other solutions dominating it. Without moving towards the better solutions, we have a reason to doubt its efficiency.

Given this, we propose the improved MOCS. It has the following characters.

- Abandon the discovery process to reduce computational complexity. Local searches are carried out at a reasonable level due to the small steps of Levy flights.
- Apply the elitism strategy to adapt to the learning pattern of multi-objective optimization. We first sort the whole population to get the better half, or elites. Levy flights are then performed between these elites instead of randomly permutation to search more efficiently.

2.2 Structure of the IMOCS

Here is the pseudo-code for the IMOCS:

```
Generate N parent solutions randomly and evaluate them
Sort the evaluated population, and rank them
while stopping criterion not met
    Screen sorted N population and keep the best N/2 elites
    Levy flight between the elites to generate N child population
    Evaluate the child population
    Combine the parent and child population, sort and rank them
    Screen sorted 2N population and keep the best N as new generation
end while
print the results
```

In the while loop, original MOCS needs to evaluate and sort twice because of the discovery process. But our IMOCS only has to evaluate the child population generated by Levy flight, and sort the combined population only once.

The screen process, which does not take long, could pick out better individuals with lower ranks calculated by the non-dominated sort process. Further, the screen process is conducted twice equally in IMOCS and MOCS, that costs almost the same time.

Also, the new searching strategy abandons discovery process, keeping only one random solutions generation process (Levy flight), thereby saving lots of time.

2.3 Process Comparison with Other Algorithms

Table 1 compares the IMOCS with MOCS, Multi-objective Differential Evolution (MODE) and Non-dominated Sort Genetic Algorithm (NSGA-II).

Table 1 Process comparison with other algorithms

Algorithm	Mutation	Crossover	Selection
IMOCS	Levy flights	Learn from each other among elites	Non-dominated sort, crowding distance
MOCS	Levy flights	Random Discovery	Non-dominated sort, crowding distance
MODE	Linear combination of parent solutions	Fixed probability	Abandon the parent if child dominates it, abandon the child conversely. Keep all of them if nondominated to each other, waiting for truncate
NSGA-II	Polynomial mutation	Simulated binary cross-over	Non-dominated sort, crowding distance

3 Experiments

3.1 Test Functions

Two objectives are already capable of reflecting the algorithms character, so here we choose 2-dimension objective functions. These are the benchmark test functions.

ZDT1

$$f_1(\boldsymbol{x})=x_1,\ f_2(\boldsymbol{x})=g(\boldsymbol{x})(1-\sqrt{x_1/g(\boldsymbol{x})}) \tag{8}$$

$$g(\boldsymbol{x})=1+\frac{9\sum_{i=2}^{n}x_i}{n-1},\ x_i\in[0,1],\ i=1,2,\cdots,30 \tag{9}$$

ZDT1 has a convex front. n is the dimension of the decision space, Pareto optimality is reached when g=1. Here n is set to 30.

ZDT2

$$f_1(\boldsymbol{x})=x_1,\ f_2(\boldsymbol{x})=g(\boldsymbol{x})(1-(x_1/g(\boldsymbol{x}))^2) \tag{10}$$

$$g(\boldsymbol{x})=1+\frac{9\sum_{i=2}^{n}x_i}{n-1},\ x_i\in[0,1],\ i=1,2,\cdots,30 \tag{11}$$

ZDT2 has a concave front. n is the dimension of the decision space. Also Pareto optimality is reached when g=1.

ZDT3

$$f_1(\boldsymbol{x})=x_1,\ f_2(\boldsymbol{x})=g(\boldsymbol{x})(1-\sqrt{\frac{x_1}{g(\boldsymbol{x})}}-\frac{x_1}{g(\boldsymbol{x})}\sin(10\pi x_1)) \tag{12}$$

$$g(\boldsymbol{x})=1+\frac{9\sum_{i=2}^{n}x_i}{n-1},\ x_i\in[0,1],\ i=1,2,\cdots,30 \tag{13}$$

ZDT3 has a discontinuous front. n is the dimension of the decision space. Also Pareto optimality is reached when g=1.

The test functions chosen are representative enough to show the algorithms' features. In calculation, 300 uniformly distributed points on the Pareto are chosen to imitate the real Pareto front.

3.2 Metrics

There are three main goals in multi-objective optimization [13]:

- The best known Pareto front should be as close to the real front as possible, say solutions on the Pareto front are the most wanted ones.
- Solutions in the best known Pareto set should be evenly and diversely distributed on the Pareto front, which gives the decision maker more choices.
- Best known Pareto front should depict the whole picture real front, including some extreme points in the objective space.

These goals contradict each other on a regular basis. The first goal focuses on a particular region on the front while the second goal emphasizes on the global search to improve diversity on the front known. The third goal extends the search to the both ends of the front, to cover as many solutions as possible.

Convergence metric: $P^*=(\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_{|P^*|})$ are the solutions on the real Pareto front, $A=(\boldsymbol{a}_1,\boldsymbol{a}_2,\boldsymbol{a}_3,\cdots,\boldsymbol{a}_{|A|})$ is the best-known Pareto front estimated by our algorithm. For each $\boldsymbol{a}_i$ in A, the regular minimum distance to P^* is defined as:

$$d_i=\min_{1\leqslant j\leqslant|P^*|}\sqrt{\sum_{m=1}^{k}\left(\frac{f_m(\boldsymbol{a}_i)-f_m(\boldsymbol{p}_j)}{f_m^{\max}-f_m^{\min}}\right)^2} \tag{14}$$

Here $f_m^{\max}$ is the maximum of the P^*'s m-th objective, while $f_m^{\min}$ is the minimum. Convergence metric is the average regularized distance of all the points in A.

$$C(A)\triangleq\frac{\sum_{i=1}^{|A|}d_i}{|A|} \tag{15}$$

Convergence metric reflects the distance between best known Pareto front and the real front. A lower value of it represents a better approximation.

Spacing metric: A is the best-known Pareto front estimated by algorithm. Function S:

$$S \triangleq \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|}(\bar{d}-d_i)^2} \tag{16}$$

where

$$d_i = \min_j \left\{ \sum_{m=1}^{k} \left| f_m(\boldsymbol{a}_i) - f_m(\boldsymbol{a}_j) \right| \right\}, \boldsymbol{a}_i, \boldsymbol{a}_j \in A, i, j=1,2,\cdots,|A| \tag{17}$$

$\bar{d}$ is the average of all the d_i. k is the number of objectives. If $k = 0$, then all the non-dominated solutions are evenly distributed in the objective space. As Manhattan distance is used in the definition, this metric works well in the non-continuous front's cases too.

3.3 Results

We tested our algorithm along with 3 other well-known multi-objective algorithms. The software is MATLAB R2015b. The computer hardware: Processor Intel(R) Core(TM) i7-4712MQ CPU @2.30GHz 2.30 GHz, RAM: 8.00GB. Parameters of the algorithms are set below:

Population: 50; Maximum iteration: 500; Step length: 1; Number of times running independently: 20; Discovery probability of the standard MOCS (Pr): 0.25.

Tables 2-7 show the statistics of the results at the 100-th generation on test problem ZDT1, ZDT2 and ZDT3. t means the average time (s) needed running through all 500 generations. We run each algorithm against each problem 20 times independently.

Table 2 Convergence metrics on ZDT1

Algorithm	Max	Min	Mean	Variance
IMOCS	0.0057	0.0022	**0.0031**	7.7089E−07
MOCS	**0.0048**	**0.0021**	0.0037	**4.3428E−07**
MODE	0.0464	0.0227	0.0323	2.9891E−05
NSGA-II	1.7136	1.1239	1.5011	0.0198

Table 3 Spacing metrics on ZDT1

Algorithm	Max	Min	Mean	Variance	t/s
IMOCS	0.0173	0.0105	0.0141	3.8848E−06	2.4755
MOCS	**0.0166**	**0.0101**	**0.0132**	**3.3793E−06**	4.8566
MODE	0.0516	0.0136	0.0245	8.2196E−05	**0.9351**
NSGA-II	0.0331	0.0128	0.0232	3.4149E−05	2.7153

Table 4 Convergence metrics on ZDT2

Algorithm	Max	Min	Mean	Variance
IMOCS	**0.0037**	**0.0021**	**0.0029**	1.9669E−07
MOCS	0.0042	0.0028	0.0035	**1.4518E−07**
MODE	0.0598	0.0245	0.0399	9.4248E−05
NSGA-II	2.4114	1.3504	1.7923	0.0681

Table 5 Spacing metrics on ZDT2

Algorithm	Max	Min	Mean	Variance	t/s
IMOCS	**0.016**	0.0104	0.0143	**1.9631E−06**	2.6144
MOCS	0.0185	0.0100	0.0138	4.6533E−06	5.0495
MODE	0.1264	0.0305	0.066	4.7218E−04	**0.938**
NSGA-II	0.0907	**2.4491 E−05**	**0.012**	3.8469E−04	5.5116

Table 6 Convergence metrics on ZDT3

Algorithm	Max	Min	Mean	Variance
IMOCS	**0.0048**	**0.0018**	0.0030	8.4771E−07
MOCS	0.0049	**0.0018**	**0.0029**	**4.7790E−07**
MODE	0.0413	0.0212	0.0300	3.4829E−05
NSGA-II	1.1599	0.7099	0.9363	0.0186

Table 7 Spacing metrics on ZDT3

Algorithm	Max	Min	Mean	Variance	t/s
IMOCS	0.0227	0.0120	0.0176	7.4071E−06	4.8159
MOCS	**0.0209**	0.0119	**0.016**	**6.8069E−06**	9.0642
MODE	0.0799	0.0304	0.0507	1.9437E−04	**0.9413**
NSGA-II	0.032	**0.0113**	0.0188	3.1902E−05	3.5062

We can clearly see that solutions generated by MODE and NSGA-II are far from the real Pareto front at 100-th generation, therefore their spacing metrics make no sense. MODE runs fast, but it converges slowly in the first 100 generations.

We know from the results that the performance of IMOCS and MOCS are very close to each other. Sometimes IMOCS works even better. Because of the elitism strategy and the abandonment of the discovery process, IMOCS spends less time than MOCS.

4 Conclusion

We abandoned the discovery process of the standard MOCS and proposed an IMOCS using elite strategy. High efficiency is achieved by referring to better elite solutions when doing Levy flights. Considering that the

Levy distribution has already taken local searches into account, we delete the random discovery process in order to save time. Numerical studies have illustrated that IMOCS would take only half the needed by MOCS. The evidence proves that our IMOCS is a more efficient multi-objective algorithm.

For further study, more test functions are welcomed to test the algorithm. Other learning mechanisms are also worth considering. Distributions other than Levy distribution should also be examined to see if we can improve its efficiency.

References

[1] Purshouse R C, Deb K, Mansor M M, *et al*. A review of hybrid evolutionary multiple criteria decision making methods[C]// *Evolutionary Computation.* Piscataway: IEEE Press, 2014: 1147-1154.

[2] Fonseca C M, Fleming P J. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization [C] // *International Conference on Genetic Algorithms.* New York: Morgan Kaufmann Publishers Inc, 1999: 416-423.

[3] Srinivas N, Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms [J]. *Evolutionary Computation*, 2014, **2**(3): 221-248.

[4] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm [C]// *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Berlin: Springer-Verlag, 2001: 95-100.

[5] Deb K, Pratap A, Agarwal S, *et al*. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. *IEEE Transactions on Evolutionary Computation*, 2002, **6**(2): 182-197.

[6] Coello C A C, Lechuga M S. MOPSO: A proposal for multiple objective particle swarm optimization [C] // *Evolutionary Computation*, 2002. *CEC* '02. *Proceedings of the* 2002 *Congress on.* Piscataway: IEEE Xplore, 2002: 1051-1056.

[7] Yang X S, Deb S. Multiobjective cuckoo search for design optimization [J]. *Computers & Operations Research*, 2013, **40**(6): 1616-1624.

[8] Viswanathan G M, Buldyrev S V, Havlin S, *et al*. Optimizing the success of random searches [J]. *Nature*, 1999, **401**(6756): 911-914.

[9] Yang X S. *Nature-Inspired Metaheuristic Algorithms*: *Second Edition* [M]. Bristol: Luniver Press, 2010.

[10] Yang X S, Deb S. Engineering optimisation by Cuckoo search [J]. *International Journal of Mathematical Modelling & Numerical Optimisation*, 2010, **1**(4): 330-343.

[11] Syberfeldt A, Ng A, John R I, *et al*. Multi-objective evolutionary simulation-optimization of a real-world manufacturing problem [J]. *Robotics and Computer-Integrated Manufacturing*, 2009, **25**(6): 926-931.

[12] Wang Q, Liu S M, Wang H, *et al*. Multi-Objective cuckoo search for the optimal design of water distribution systems [C] // *Civil Engineering and Urban Planning*. Yantai: American Society of Civil Engineers, 2012: 402- 405.

[13] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results [J]. *Evolutionary Computation*, 2006, **8**(2):173.

□