Solving the Graph Coloring Problem Using Cuckoo Search

Conference Paper · June 2017

DOI: 10.1007/978-3-319-61824-1_60

3 authors, including:

Claus Aranha
University of Tsukuba
57 PUBLICATIONS **255** CITATIONS

SEE PROFILE

Solving the Graph Coloring Problem using Cuckoo Search

Claus Aranha[1], Keita Toda[2] and Hitoshi Kanoh[1]

[1] Faculty of Engineering, Information and Systems, University of Tsukuba
caranha@cs.tsukuba.ac.jp, kanoh@cs.tsukuba.ac.jp
[2] Graduate School of Systems and Information, University of Tsukuba

Abstract. We adapt the Cuckoo Search (CS) algorithm for solving the three color Graph Coloring Problem (3-GCP). The difficulty of this task is adapting CS from a continuous to a discrete domain. Previous researches used sigmoid functions to discretize the Lévi Flight (LF) operator characteristic of CS, but this approach does not take into account the concept of *Solution Distance*, one of the main characteristics of LF. In this paper, we propose a new discretization of CS that maintains LF's solution distance concept. We also simplify CS's parasitism operator, reducing the number of evaluations necessary. We compare different combinations of the proposed changes, using GA as a baseline, on a set of randomly generated 3-GCP problems. The results show the importance of a good discretization of the LF operator to increase the success rate and provide auto-adaptation to the CS algorithm.

Keywords: Graph Coloring Problem, Cuckoo Algorithm, Lévy Flight

1 Introduction

Cuckoo Search (CS) is an swarm-based optimization meta heuristic developed by Yang et. al [1], inspired by the breeding behaviors of cuckoo birds. One of its main characteristics is the use of the *Lévy flight distribution* as a variation operator. The CS showed excellent results solving optimization problems in the continuous domain. Recently there are efforts to adapt this approach to discrete value domains as well, such as combinatory optimization and constraint satisfaction problems. However, there is not yet an accepted general approach for the application of CS to optimization problems on the discrete domain.

In this context, we are interested in solving the Graph Coloring Problem (GCP) using CS. Former works such as Yongquan et al. [2] and Halima et al. [3] used a Binary representation of the Lévy Distribution that does not take into account the concept of solution distance, which we believe is a key characteristic for the performance of this algorithm. Aoki et al. introduced a formulation of PSO for the GCP which uses the Hamming distance to calculate the distance between solutions [4]. While we consider this to be a better modeling of discrete distance, Aoki's model of Hamming distance is focused on the comparison between two solutions, and not appropriate to calculating distances from a single solution, making it difficult to apply it directly to CS.

In this work we propose a new discretization model of the Lévy flight distribution, allowing us to use CS for the 3-GCP problem. Additionally, we propose adjustments to the parasitism operator to obtain even better results in this problem. We compare the effectiveness of the proposed operators with standard formulations, and include GA as a baseline. Our experiment results show that the proposed methods are an effective way to discretize CS for the 3-GCP.

2 Background

2.1 Lévy Flight Distribution

Various studies have showed that the flight and feeding behavior of animals have the characteristics of the Lévy distribution [5–7]. A *Lévy Flight* (LF) is a random walk which uses the Lévy distribution. Compared with the standard random walk, the LF shows occasional long steps among the many short ones, as illustrated in Figure 1. It has been shown that this behavior allows the Lévy flight to be more effective than the Random Walk when used in a variety of search algorithms [8, 9].

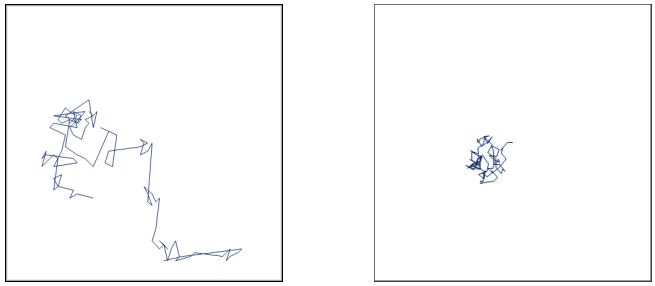

Fig. 1. Lévy Flight (left) versus Random Walk (right) on a 2-D space.

2.2 Cuckoo Search Algorithm

The Cuckoo Search (CS) algorithm is a meta-heuristic search algorithm for optimization problems in the continuous domain. In comparison with other meta-heuristic search algorithms such as GA or PSO, it has fewer control parameters which are simpler to fine-tune [10]. The CS algorithm can be described as follows. Initially, a random population is generated. Then, at each generation, up to two replacement candidates are generated for each individual x_i:

1. A modified individual u_i^1 is generated using the *Lévy Flight* operator;
2. With probability $p_a \in \{0, 1\}$, a second modified individual u_i^2 is generated;

The best among x_i, u_i^1 and u_i^2 is added to the following generation.

The *Lévy Flight* operator generates u_i^1 from x_i as follows [11, 12]:

$$u_i = x_i^{(t)} + \alpha \times \mathrm{L}(\beta), \text{ where } \mathrm{L}(\beta) = \frac{p}{|q|^{1/\beta}}, (0.3 \leq \beta \leq 1.99). \tag{1}$$

In Equation 1, α and β are problem-dependent constants. p and q are random variables sampled from the following normal distributions:

$$p \sim N(0, \sigma_p^2), q \sim N(0, \sigma_q^2), \text{ where } \sigma_p = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma((1+\beta)/2)\beta 2^{(\beta-1)/2}}^{1/\beta} \right\}, \sigma_q = 1$$

The second replacement candidate is generated on the *Parasitism Operator step*. Random generation [1] and the Lévy flight operator [10] have both been suggested for this purpose.

2.3 Graph Coloring Problem

Given a simple, adirected graph $G = \{V, E\}$, the Graph Coloring Problem (GCP) is the problem of assigning a label (color) to every vertex $v_i \in V$ so that no two vertices which share an Edge have the same label. In the n-GCP, the set of labels is fixed with exactly n elements, while in the min-GCP, it is necessary to find the smallest label set for a given graph. In this work, we focus on the 3-GCP problem. The 3-GCP is a NP-complete problem, and often used as a benchmark to evaluate constraint satisfaction algorithms.

2.4 Related Research

Zhou et al. applied CS to the 4-GCP [2]. Instead of the Lévy flight, they used a binary expression obtained by a sigmoid function, which does not model the concept of solution distance. They apply some hybridizations. They compare this algorithm with PSO, MPSO and MTPSO, obtaining improvements against the first two, but not the third. Also, the performance of their discretization of CS without the hybridization was quite low.

Halima et al. applied CS to the min-GCP [3]. They also used a binary expression obtained by a sigmoid function, as in the above work. However, their solution encoding is based on the encoding for the knapsack problem, so it is not applicable to the n-GCP.

Aoki et al. used PSO for solving the 3-GCP problem [4]. They proposed the use of Hamming Distance to calculate the distance vectors between the PSO candidate solutions. This approach showed better performance than the Binary discretizations of previous works, however, their modeling of the hamming distance can only be applied between two candidate solutions, which makes it hard to use in the Lévy Flight operator.

3 Proposed Method

Our implementation of the CS algorithm for solving the 3-GCP obeys the following structure:

1. A candidate solution is represented using the encoding described in Figure 2. One solution is represented as an array where each element corresponds to one vertex in the graph, and contains the color label for that vertex.

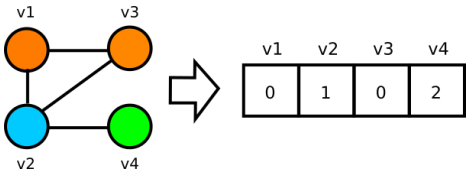

Fig. 2. Encoding of a GCP, each element in the vector represents the color assignment of one vertex in the graph.

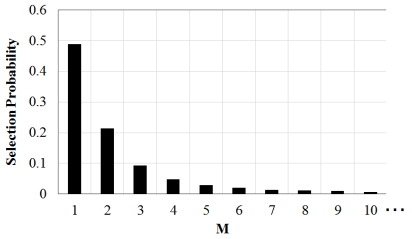

Fig. 3. Distribution of M possible values based on the modified Lévy distribution proposed

2. The Fitness of an individual is defined as the number of edges that end in two vertices of the same color.
3. At every generation, the candidate solutions are modified by the Lévy Flight operator followed by the Parasitism operator, as described below.

The resulting method is summarized in algorithm 1.

3.1 Discrete Lévy Flight Operator

The Discrete Lévy Flight operator is used to modify each candidate x_i in the population. First, it determines a value M as follows:

$$M = \lfloor \alpha \times L(\beta) \rfloor + 1. \tag{2}$$

Then, M vertices from x_i are selected, and their values (colors) are randomly changed, with 0.5 probability for each color. The size of M as selected by the Lévi distribution models the solution distance behavior. In most cases, only a few vertices will be modified. But occasionally, a larger number of vertices will me changed at once. Figure 3 illustrates this behavior.

3.2 Modified Parasitism Operator

Originally, the parasitism operator replaces one individual from the population with a modified one with probability p_a, if the modified one has equal or better fitness. In past works, the modified individual can be fully random [1] or a modification of the original using the Uniform distribution [10]. In the 3-GCP problem, there are multiple local optima, and a smaller range of fitness values, when compared with continuous optimization problems. As a result, we observed that the Parasitism operator rarely replaced existing candidates.

To improve the exploration power of the algorithm and reduce wasted evaluations, we change the Parasitism operator so that the new solution is always adopted without comparison with the current one. To compensate for the randomness of introducing new solutions to the population, we also drastically reduce the value of parameter p_a.

Additionally, we perform the modification in three different ways: 1- Modifying a random number of vertices drawn from a uniform distribution; 2- Modifying a random number of vertices drawn from the Lévy distribution; and 3- Modifying a fixed number of vertices, decided by a fine-tuning experiment.

Algorithm 1 Proposed Cuckoo Search Algorithm for 3-GCP

```
Generate random initial solution set S^0
while Current Evaluations < Max Evaluations do
    for Each candidate solution x_i^(t) ∈ S^(t) do                ▷ Lévy Flight operator
        Select M using the Discrete Lévi distribution
        Generate u_i^1 by randomly changing the color of M vertices in x_i^(t)
        if Fitness(u_i^1) ≥ Fitness(x_i^(t)) then replace x_i^(t) with u_i^1
    end for
    for Each candidate solution x_i^(t+1) ∈ S^(t+1) do             ▷ Parasitism operator
        if random uniform number k_i ∈ (0,1) ≤ p_a then
            Select M using one of {Uniform Distribution, Discrete Lévi, Fixed Value}
            Generate u_i^2 by randomly changing the color of M vertices in x_i^(t)
            if (parasitism_comparison is "No") or (Fitness(u_i^2) ≥ Fitness(x_i^(t))) then
                Replace x_i^(t) with u_i^2
            end if
        end if
    end for
end while
```

4 Experiments

We perform a computational experiment to evaluate the contributions of the various proposed changes. In this experiment we compare four algorithms composed of different combinations of the modifications proposed in the previous section, along with a GA to be used as a baseline. The four algorithms and their descriptions are detailed in Table 1

Each algorithm is tested on a set of random graphs generated based on the formulation by Minton et al. [13]. In our formulation, the vertices in the graphs are divided in three groups, and edges between the groups are randomly added to the graphs until the necessary number is reached. Since the 3-GCP strongly depends on the local structure of the graph, using random instances allows us to avoid local optima.Note that this formulation guarantees that a solution exists for the graph.

Each experiment is generated with a fixed number of vertices, n, and a fixed edge density $d = |E|/|V|$. The edge density can be considered a difficulty index for the 3-GCP problem. Hogg et al. showed that the 3-GCP is most difficult when $2.0 \leq d \leq 2.5$ [14].

4.1 Parameter Setting

To determine the parameter values for each algorithm we perform a preliminary experiment to find the optimal value for each parameter independently. This

Table 1. Algorithm Variations compared in the experiment.

Algorithm	CS-R-Yes	CS-R-NO	CS-L-NO	CS-CONST-NO
Parasitism Variation Operator	Uniform Dist.	Uniform Dist.	Lévy Flight	Constant E
Parasitism Comparison	Yes	No	No	No

experiment was done on a problem set with $n = 120$, $d = 2.5$ and 100 random graphs. The parameter value with highest proportion of successes was adopted. The selected values for each parameter is listed on Table 2.

Table 2. Parameter values used for the comparison experiment

Algorithm	CS-R-Yes	CS-R-NO	CS-L-NO	CS-CONST-NO	GA
Population Size	200	10	10	10	40
p_a	0.2	0.0001	0.0001	0.001	N/A
β			1.5		N/A
α			1		N/A
E	N/A	N/A	N/A	3	N/A
Mutation probability			N/A		0.011
Tournament Size			N/A		2

4.2 Evaluation Experiment

To compare the performance of the proposed methods, we execute them on a series of 3-GCP data sets. The number of nodes in the graphs used are 90, 120, 150 and 180, and the edge density d goes from 1.5 to 9.0, at 0.5 intervals. For each combination of vertex number and edge density, we test each algorithm on a set of 100 random graphs, and report the proportion of solved graphs, and the average number of evaluation functions until a solution was found.

The results of the experiment can be seen on figures 4 to 11. From these results we can see that the CS variants normally outperform GA, specially when the number of vertices in the graph is higher.

Among the proposed variants, CS-L-NO and CS-CONST-NO outperform the CS-R variants, specially in terms of number of evaluations until a solution is found. This shows that the use of Levy Flight when compared to the Random mutation does indeed improve the performance of the CS.

At first glance, the similar results between the CS-L-NO and the CS-CONST-NO may seem to imply that there is no benefit in using the Lévy Flight as opposed to a constant mutation number. However, note that the preliminary experiment necessary to find the optimal constant E is a cost that is not necessary for the Lévy Flight (the Lévy Flight showed very little sensitivity to the choices of α and β. This means that the use of the Lévy Flight gives a degree of self-adaptation to the algorithm.

5 Conclusion

In this paper, we proposed an adaptation of the Cuckoo Search (CS) algorithm for the Graph Coloring Problem with Three colors (3-GCP). The CS is characterized by the Lévy Flight mutation, so we paid special attention to the discretization of the Lévy distribution.

We compared the different proposed changes in a computational experiment with a large variety of graph sizes and difficulties, using the GA as a baseline. The results indicate that the proposed discretization of the Lévy flight allows us to use CS for the 3-GCP without having to worry with fine-tuning parameter values or hybridization with local search operators.

Acknowledgments

This work was supported by JSPS KAKENHI grand number 15K00296.

References

1. Xin-She Yang, Suash Deb, Cuckoo Search via Lévy Flights, Nature & Biologically Inspired Computing, no.37, pp. 210214 (2009).
2. Yongquan Zhou, Hongqing Zheng, Qifang Luo, Jinzhao Wu, An improved Cuckoo Search Algorithm for Solving Planar Graph Coloring Problem, Applied Mathematics & Information Sciences, Vol.7, No.2, pp.785-792 (2013).
3. Halima Djelloul, Abdesslem Layeb, Salim Chikhi, A Binary Cuckoo Search Algorithm for Graph Coloring Problem, International Journal of Applied Evolutionary Computation, Vol. 5, Issue 3, pp. 42–56 (2014).
4. T. Aoki, C. Aranha, H. Kanoh, PSO Algorithm with Transition Probability Based on Hamming Distance for Graph Coloring Problem, IEEE International Conference On Systems, Man, And Cybernetics, pp. 1956–1961 (2015).
5. Brown C., Liebovitch L. S., Glendon R., Lévy flights in Dobe Ju'hoansi foraging patterns, Human Ecol., 35, pp.129-138(2007).
6. Pavlyukevich I., Lévy flights, non-local search and simulated annealing, J. Computational Physics, 226, pp.1830-1844(2007).
7. Reynolds A. M. and Frye M. A., Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search, PLoS One, 2, e354(2007).
8. G.M. Viswanathan, E.P. Raposo, M.G.E. da Luz, Lévy flights and super diffusion in the context of biological encounters and random searches, Physics of Life Reviews, Vol. 5, Issue 3, pp. 133–150 (2008).
9. A. F. Ali, A Hybrid Gravitational Search with Lévy Flight for Global Numerical Optimization, Information Sciences Letters, Vol. 4, No. 2, pp. 71–83 (2015).
10. X.S. Yang, S. Deb, Engineering Optimization by Cuckoo Search, Intl. J. of Mathematical Modeling and Numerical Optimization, pp. 330–343 (2010).
11. Xin-She Yang, Nature-Inspired Metaheuristic Algorithms Second Edition, LUNIVER PRESS (2010).
12. Rosario N. Mantegna, Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes, Physical Review E, Vol. 49, no. 5, pp. 4677–4683 (1994).
13. Steven Minton, Mark D. Johnston, Andrew B. Philips, Philip Laird, Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems, Artificial Intelligence, Vol. 58, pp. 161–205 (1992).
14. Tad Hogg, Colin Williams, The Hardest Constraint Problems: A double Phase Transition, Artificial Intelligence, Vol. 69, pp. 359–377 (1994).

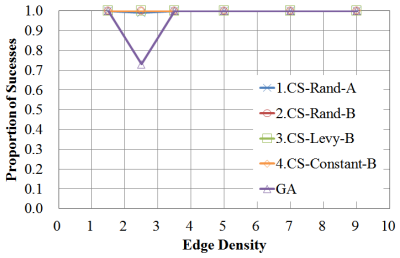

Fig. 4. Proportion of problems solved by each method (n = 90)

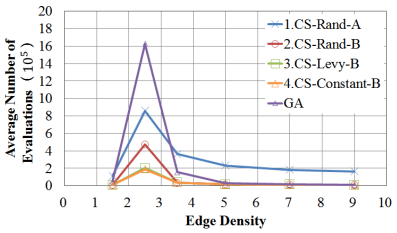

Fig. 5. Average number of evaluations to solve a problem (n = 90)

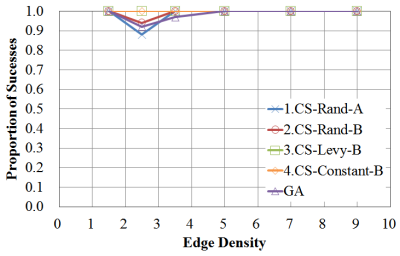

Fig. 6. Proportion of problems solved by each method (n = 120)

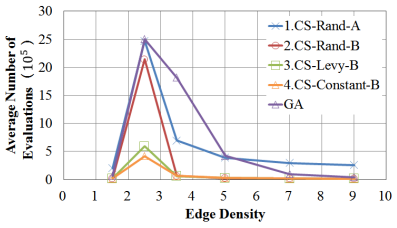

Fig. 7. Average number of evaluations to solve a problem(n = 120)

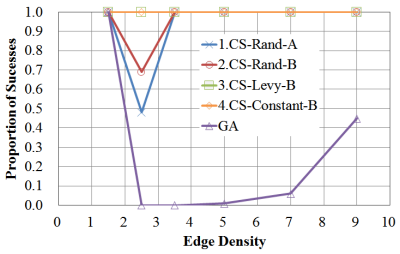

Fig. 8. Proportion of problems solved by each method (n = 150)

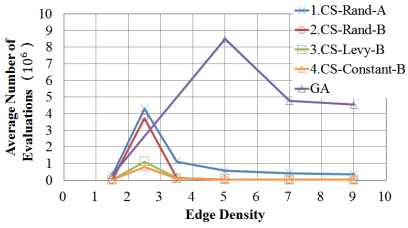

Fig. 9. Average number of evaluations to solve a problem(n = 150)

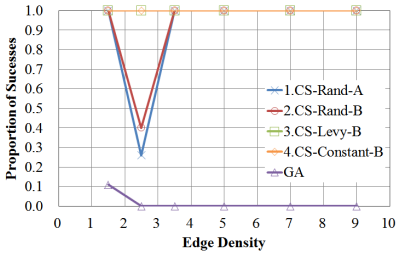

Fig. 10. Proportion of problems solved by each method (n = 180)

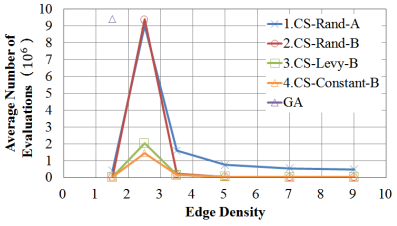

Fig. 11. Average number of evaluations to solve a problem(n = 180)