A Binary Cuckoo Search Algorithm for Graph Coloring Problem

Halima Djelloul, MISC Laboratory, Constantine 2 University, Constantine, Algeria

Abdesslem Layeb, MISC Laboratory, Constantine 2 University, Constantine, Algeria

Salim Chikhi, MISC Laboratory, Constantine 2 University, Constantine, Algeria

ABSTRACT

The Graph Coloring Problem (GCP) is one of the most interesting, studied, and difficult combinatorial optimization problems. That is why several approaches were developed for solving this problem, including exact approaches, heuristic approaches, metaheuristics, and hybrid approaches. This paper tries to solve the graph coloring problem using a discrete binary version of cuckoo search algorithm. To show the feasibility and the effectiveness of the algorithm, it has used the standard DIMACS benchmark, and the obtained results are very encouraging.

Keywords: *Binary Cuckoo Search, Cuckoo, Evolutionary Computation, Graph Coloring Problem, Heuristics, Optimization, Search Algorithm*

1. INTRODUCTION

The Graph Coloring Problem (GCP) is one of the most interesting, studied and difficult combinatorial optimization problems. The GCP consists in coloring each vertex of a given graph by using a minimum number of colors called the chromatic number (Matula et al., 1972), so that no two adjacent vertices are colored with the same color. Unfortunately, GCP has been shown to be NP-hard (Garey & Jonhson, 1979), hence, several approaches were developed to handle this problem, that we can classify into three classes; exact approaches, heuristic approaches and metaheuristics approaches. There exist quite a few of exact approaches for this problem; they are generally based on the implicit enumeration algorithms (Kubale & Jackowski, 1985), as well as branch- and- bound and its variants (Méndez-Diaz & Zabala, 2006;Mehrotra & Trick, 1996). On the other hand, the constructive approaches have been widely proposed to resolve the graph coloring problem. The constructive approaches progressively build the solution, in this category we can mention the algorithm developed by Welsh and Powell (Welsh & Powell, 1967), the degree of saturation (DSATUR) (Brélaz, 1979) and the recursive largest first algorithm (RLF) (Leighton,1979).). Moreover, many kinds of metaheuristics and

DOI: 10.4018/ijaec.2014070103

their hybridizations have been used to solve GCP like tabu Search (Hertz & de werra,1987), Simulated annealing (Johnson et al., 1991), genetic algorithm (Fleurent & Ferland, 1996), ant colony (Dowsland & Thompson, 2008), variable neighborhood search VNS (Avanthay et al., 2003), Variable Search Space (VSP) that is an extension of the VNS (Hertz et al., 2008),), a memetic algorithm (Zhipeng & Jin-Kao,2010), a hybrid Artificial Bee Colony Algorithm for Graph 3-Coloring(Fister, I.Jr., Fister, I., & Brest, 2012), etc.

Far from the graph coloring problem, Cuckoo Search (CS) is an optimization algorithm developed by Xin-She Yang and Suash Deb (Yang & Deb,2010). It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some bird's host can involve direct conflicts with the intruding cuckoos. For example, if a bird's host discovers that the eggs are strange eggs, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere(Barthelemy et al., 2008).

The cuckoo's behavior and the mechanism of Lévy flights (Payne et al.,2005 ; Pavlyukevich 2007) have led to the design of an efficient inspired algorithm performing optimization search. The recent applications of Cuckoo Search for optimization problems have shown its promising effectiveness. Moreover, a promising Binary Cuckoo Search algorithm (BCS) is recently proposed to deal with discrete problems (Gherboudj et al., 2012) which used a sigmoid function similar to that used in the binary particle swarm optimization algorithm to obtain a binary solution.

The present study was designed to investigate the use of the BCS algorithm to deal with the graph coloring problem. The main features of the proposed approach consist in adopting a binary representation of the search space and using a sigmoid function and probability model in order to generate binary values (Gherboudj et al., 2012). Moreover, we have modified the BSR algorithm proposed in (Gherboudj et al., 2012) to avoid the infeasible solution like having an uncolored node. Finally, we have tested our algorithm on some DIMACS instances taken from (http://mat.gsia.cmu.edu/COLOR/instances.html) and the results found are promising.

The reminder of the paper is organized as follows. In section 2, a formulation of the tackled problem is given. In section 3, an overview of cuckoo search is presented. In section 4, the proposed method is described in section 5. Experimental results are discussed in section 6. Finally, conclusions and future work are drawn.

2. PROBLEM FORMULATION

Graph Coloring Problem (GCP) is a well-known combinatorial problem, and important task in solving many real problems such as the frequency assignment problem (Smith et al., 1998), crew scheduling(Gamache et al., 2007), register allocation (de Werra et al., 1999), etc. graph is k-colorable if and only if it can be colored using k colors.Formally, a k-coloring will be represented by a set H = $\{C(v_1), C(v_2),\ldots,C(v_n)\}$ such as $C(v_i)$ is the color assigned to the vertex v_i. If for all $\{u,v\} \in E$, $C(u) \neq C(v)$, then H is a legal coloring ; otherwise, H is an unfeasible k-coloring. In the optimization version of the GCP, the principal objective is to minimize the total number of colors used to color a given graph.

Formally, the graph coloring problem can be formulated as follows:

Given a k-coloring H = $\{C(v_1)C(v_2),\ldots,C(v_n)\}$ with the set V = $\{v_1, ..., v_n\}$ of vertices, the evaluation function f counts the number of conflicting vertices produced by H such that:

$$f(H) = \sum_{\{u,v\} \in E} \delta_{uv}$$

Where:

$$\delta_{uv} = \begin{cases} 1, & if C(u) = C(v) \\ 0, & otherwise. \end{cases}$$

By consequent, a coloring H with $f(H) = 0$ corresponds to a feasible k-coloring.

3. CUCKOO SEARCH ALGORITHM

In order to solve complex problems, ideas gleaned from natural mechanisms have been exploited to develop heuristics. Nature inspired optimization algorithms has been extensively investigated during the last decade paving the way for new computing paradigms such as neural networks, evolutionary computing, swarm optimization, etc. The ultimate goal is to develop systems that have ability to learn incrementally, to be adaptable to their environment and to be tolerant to noise. One of the recent developed bioinspired algorithms is the Cuckoo Search (CS)(Yang & Deb,2010) which is based on style life of Cuckoo bird. Cuckoos use an aggressive strategy of reproduction that involves the female hack nests of other birds to lay their eggs fertilized. Sometimes, the egg of cuckoo in the nest is discovered and the hacked birds discard or abandon the nest and start their own brood elsewhere. The Cuckoo Search proposed by Yang and Deb (Yang & Deb, 2010) is based on 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 of eggs (solutions) will carry over to the next generations;
- The number of available host nests is fixed, and a host can discover an alien egg with a probability pa ∈ [0, 1]. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

The last assumption can be approximated by a fraction pa of the n nests being replaced by new nests (with new random solutions at new locations). The generation of new solutions x(t+1) is done by using a Lévy flight (eq.3). Lévy flights essentially provide a random walk while their random steps are drawn from a Lévy distribution for large steps which has an infinite variance with an infinite mean (eq.4). 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 (Yang & Deb,2010).

$$x_i^{t+1} = x_i^t + \alpha \oplus Lévy(\lambda)$$
$$Lévy \sim u = t^{-\lambda}$$

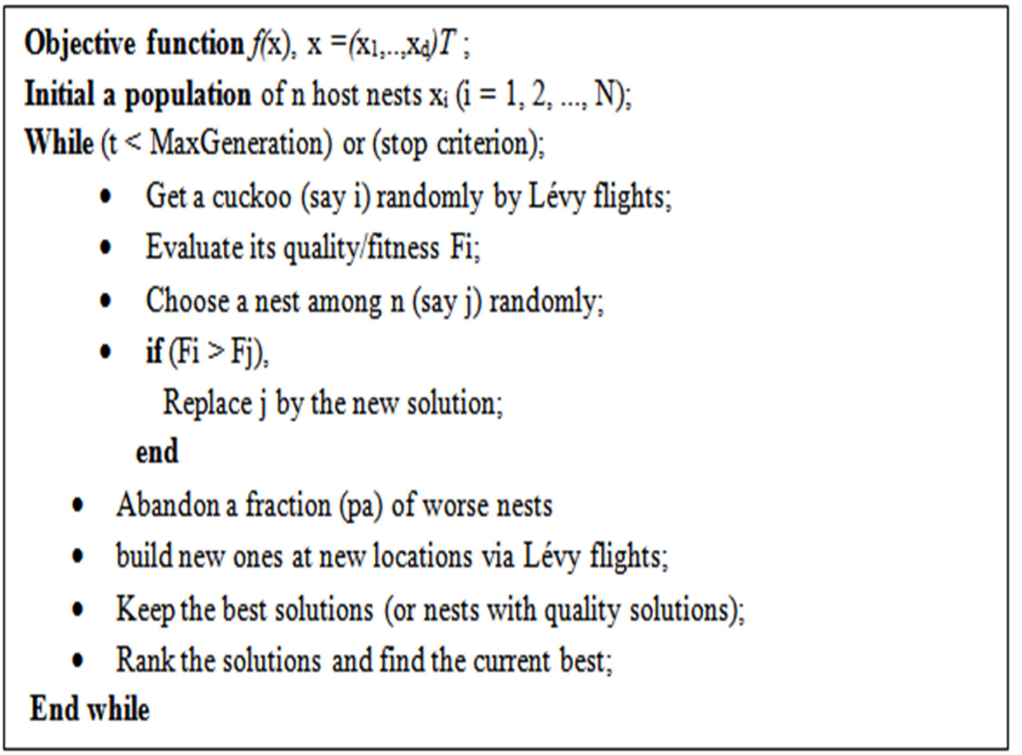
Where x_i^{t+1} and x_i^t represent the solution i at times t+1 and t respectively. $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. Generally we take $\alpha = O(1)$. The above equation is essentially the stochastic equation for random walk. The product $\oplus$ means entry-wise multiplications. This entry-wise product is similar to those used in PSO, but here the random walk via Lévy flight is more efficient in exploring the search space as its step length is much longer in the long run.

The main characteristic of CS algorithm is its simplicity. In fact, comparing with other population or agent-based metaheuristic algorithms such as particle swarm optimization and harmony search, there are few parameters to set. The applications of CS into engineering optimization problems have shown its encouraging efficiency. For example, a promising discrete cuckoo search algorithm is recently proposed to solve scheduling problem (Tein & Ramli, 2010). Another binary version of cuckoo search is proposed in (Gherboudj et al., 2012) to solve the knapsack problems. An efficient computation approach based on cuckoo search has been proposed for data fusion in wireless sensor networks (Dhivya, 2011), and a recent cuckoo search algorithm was proposed for coloring the planar graph (Yongquan et al., 2013), a binary cuckoo search algorithm for feature selection (Douglas et al., 2013), Cuckoo search algorithm for the selection of optimal machining parameters in milling operations (Yildiz, 2013).

In more details, the proposed cuckoo search algorithm can be described in Figure 1.

Cuckoo search algorithm was originally designed for solving the continuous optimization problems where the solution is represented by a set of real numbers. However, in discrete optimization problems, the solution is represented by a set of integer numbers. Discrete binary optimization problems are a sub-class of the discrete optimization problems class in

Figure 1. Cuckoo search schema



which a solution is represented by a set of bits. Many optimization problems can be modeled as a discrete binary search space such as flow-shop scheduling problem (Liao et al., 2007), job-shop scheduling problem (Pongchairerks, 2009), routing problems (Zhan & Zhang, 2009), knapsack problem(Gherboudj et al,2012) and so one.

The original CS algorithm is based on Lévy flights, and it operates in continuous search space. Consequently, CS algorithm gives a set of real numbers as a solution of the handled problem. However, a binary optimization problem needs a binary solution and the real solutions are not acceptable, because they are considered as illegal solutions. Therefore, the solutions must be converted from real values to binary values. In BCS algorithm, the problem solution must be represented by a set of bits that's why we have used a binary representation of the CGP solution.

4. THE BINARY CUCKOO SEARCH ALGORITHM FOR GRAPH COLORING PROBLEM

In this section, we present how the Binary Cuckoo search algorithm has been tailored for the graph coloring problem. The proposed BCS architecture contains two essential modules. The first module contains the main binary cuckoo dynamics. This module is composed of two main operations: Lévy flights and Binary Solution Representation (BSR) operations. These two operations combine the CS algorithm and the sigmoid function to obtain a BCS. In the first operation, Lévy flight is used to get a new cuckoo. In the second operation, the sigmoid function is used to calculate the flipping chances of each cuckoo. Then, the binary value of each cuckoo is computed using their flipping chances. The second module contains the objective function and the selection operator. The selec-

tion operator is similar to the elitism strategy used in genetic algorithms. Figure 2 shows the flowchart of the proposed architecture. In the following, we will explain in more detail the proposed BCS for the graph coloring problem (Gherboudj et al., 2012).

4.1. Binary Solution Representation of Graph Coloring Solution

The main objective of the BCS algorithm is to deal with the binary optimization problems. Therefore, we need to map the graph coloring solution into a binary representation that could be manipulated by BCS operators. So, the graph coloring solution is represented as binary matrix (Figure 3) satisfying the following criteria:

- For a graph of n nodes and k colors, the size of the binary matrix is k*n.

The columns represent the nodes and the rows represent the colors.

- The presence of 1 in the position *(i, j)* indicates that the node *j* colored with the color *i.*
- In each column there is a single 1, i.e. the node is colored with one color.

The main feature of BCS algorithm is to transform the matrix *x* which represents the

Figure 2. Flowchart of the BCS architecture

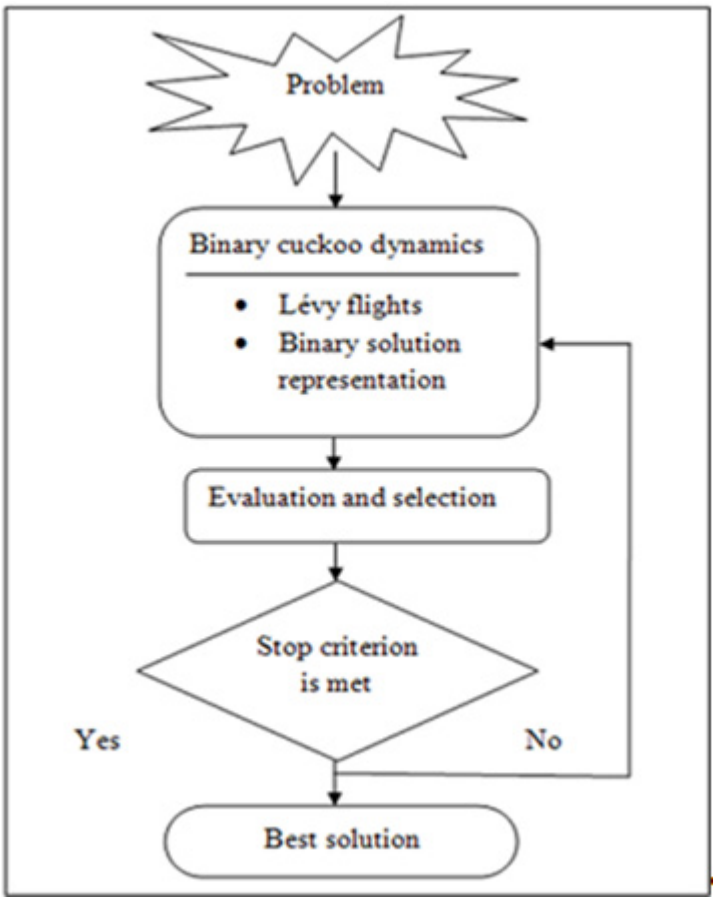
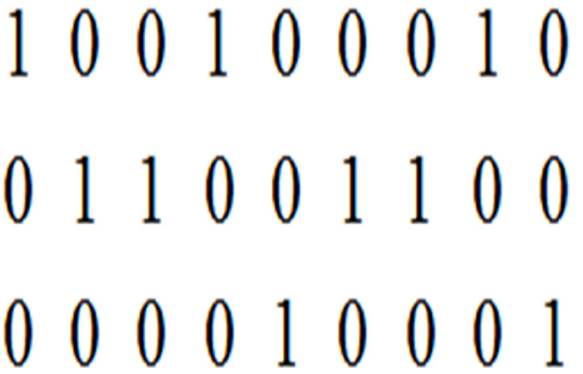


Figure 3. Feasible binary representation of the solution



solution from real search area to binary search area, and consequently obtain the binary solution representation (BSR) of x (x').To meet this need, the sigmoid function is used to constrain the solution x to be in the interval [0, 1] as follow (Gherboudj et al., 2012):

$$S(x_{ij}) = \frac{1}{\left(1 + e^{-x_{ij}}\right)}$$

where $S(x_{ij})$ is the flipping chance of bit x_{ij}'. It represents the probability of bit x_{ij}' to take the value 1.To obtain the binary solution x', a random matrix r(k, n) from the interval [0, 1] is generated and compared with the flipping chance $S(x_{ij})$ as mentioned below in equation (6). If r_{ij} is lower than the flipping chance of bit x_{ij}', than x_{ij}' takes the value 1. Otherwise, x_{ij}' takes the value 0. (Gherboudj et al.,2012)

$$x'_{ij} = \begin{cases} 1 & \text{if } r_{ij} < S(x_{ij}) \\ 0 & \text{otherwise} \end{cases}$$

Consequently, having a solution x encoded as a matrix of real numbers, the sigmoid function is used to transform the solution x into a set of probabilities that represents the chance for each bit of the matrix to be flipping. The flipping chance is then used to compute the binary solution x'.

For example, assuming that we have a graph with 9 vertices and 3 colors are used for coloring the graph, x is the obtained solution with original CS algorithm; $S(x)$ is the set of flipping chances (probabilities) of each bit x_{ij} calculated by the Sigmoid function; In order to obtain a binary solution, we must generate (3*9) random matrix r from the interval [0, 1] .

Following the defined instructions in equation (6), x' is the binary solution obtained. Figures 4, 5, 6, and 7 below show x, $S(x)$, r, x' respectively.

However, the use of BSR proposed in (Gherboudj et al., 2012) for solving the graph coloring problem can lead to infeasible solutions as it shown in Figure 7 and then increase the computational time of the algorithm to find good solutions. Indeed, by using the standard BSR, we can get an uncolored node like the fifth node or a node having more than one color like the first and the second node. To correct this kind of solution we have proposed a modified version of BSR as following. for the uncolored vertex, we have introduced a randomly assignment of colors to the uncolored vertices in the core of the BSR algorithm, and the value 1 is obtained if the random number r_{ij} is lower than the flipping chance of bit x_{ij} and any neighbor of the current node have already the same color i. A pseudo code of the modified BSR algorithm is shown in Figure 8.

Figure 4. The obtained solution with original CS

2.8750	-1.9160	3.1680	-2.3150	0.6758	3.2769	-0.2637	1.5864	-2.3845
1.9314	2.5570	-3.8740	0.3412	-1.6378	-2.8634	3.3647	-2.5481	-0.6957
3.5486	2.9482	-0.8196	0.3457	-3.2197	1.7487	-1.5463	0.9873	-0.9871

Figure 5. Flipping chances calculated by the Sigmoid

0.9466	0.1283	0.9596	0.0899	0.6628	0.9636	0.4345	0.8301	0.0844
0.8734	0.9280	0.0204	0.5845	0.1628	0.0540	0.9666	0.0726	0.3328
0.9720	0.9502	0.3058	0.5856	0.0384	0.8518	0.1756	0.7286	0.2715

Figure 6. Random (3,9) generated matrix r

0.8147	0.9134	0.2785	0.9649	0.9572	0.1419	0.7922	0.0357	0.6787
0.9058	0.6324	0.5469	0.1576	0.4854	0.4218	0.9595	0.8491	0.7577
0.1270	0.0975	0.9575	0.9706	0.8003	0.9157	0.6557	0.9340	0.7431

Figure 7. Binary solution representation

1	0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0	0
1	1	0	0	0	0	0	0	0

Figure 8. The modified BSR algorithm

Input: real solution representation x, random matrix r (k, n)

Output: x'

Begin

For (i =1 to k) {

For (j =1 to n) {

If $S(x_{ij}) > r_{ij}$ and any neighbor of the node j have already the same color i

$x'_{ij}=1$

Else

$x'_{ij}=0$

}

}

- Choose a color randomly for the uncolored nodes.

End

4.2. Constructive Heuristic for Generating the Initial Population

To generate the initial population, we have used a modified version of the constructive RLF algorithm proposed in (Mabrouk et al., 2009), which colors the vertices one node at a time, in the following greedy way:

Let *Ncl* to be the next color to be assigned, *Y* the set of uncolored vertices that can be assigned to the color *Ncl*, and *B* the set of uncolored vertices that cannot be assigned to the color *Ncl*.

- If *Ncl* > k then the remaining nodes will be colored randomly using the set of colors {1, 2,…, k}.
- Otherwise, choose the first vertex v $\in$ *Y* randomly, color v with the color *Ncl*, and move its neighbors included in *Y* from *Y* to B.

The integration of the randomness in the RLF heuristic helps to obtain a population with good diversity, consequently reduce the convergence time.

In more details, the heuristic used can be described in Figure 9:

4.3. Outline of the Proposed Algorithm

Now, we describe how the sigmoid function has been embedded within a cuckoo search algorithm to obtain a binary solution to a binary optimization problem. The powerful of BCS algorithm is situated essentially in three components: selection of the best solution, local exploitation by random walk, and global exploration by randomization via Lévy flights. A pseudo code of the proposed BCS algorithm is shown in Figure 10. Like any other population-based meta-heuristics, the first step in the developed

algorithm called BCSCOL (Binary Cuckoo Search algorithm for graph COLoring) involves setting the parameters for the algorithm. The main advantage of the BCSCOL algorithm is that there are fewer parameters to be set in this algorithm than in PSO. In BCSCOL, there are essentially two main parameters to be initialized, population size N and a fraction pa of the worst nests to be rejected and replaced. In the second step, a swarm of N host nests is created by using the modified RLF algorithm (Mabrouk et al., 2009). The algorithm progresses through a number of generations according to the BCS dynamics. During each iteration, the following main tasks are performed. A new cuckoo is built using the Lévy flights operator. The Lévy flight provides a random walk that consists of taking successive random steps in which the step lengths are distributed according to a heavy tailed probability distribution. The next step is to evaluate the current cuckoo. For that, we apply the modified BSR algorithm (Figure 3) in order to get a feasible binary solution. The binary solution is evaluated by using the number of edge violated as objective function. After this step, we replace some worst nests by the current cuckoo if it is better, or by new random nests generated by Lévy flights. The selection phase in BCS of the best nests or solutions is comparable to some form of elitism selection used in genetic algorithms, which ensures that best solution is kept always in the next iteration, if a feasible k-coloring is found the number of colors k will be reduced by one, and the procedure will be iterated until a legal k-coloring cannot be found after achieving a stopping criterion.

Figure 9. Constructive modified RLF. (adapted from [Mabrouk et al., 2009])

$B = \phi$, $Y = V$, $Ncl = 0$

While ($Y \neq \phi$) do

$Ncl = Ncl + 1$

If ($Ncl > k$)

Color the remaining vertices with colors randomly chosen in the set {1, 2, . . . ,k}

Else choose randomly $v \in Y$

$C(v) = Ncl$, $B = B \cup N_{G(Y)}(v)$, $Y = Y \setminus (\{v\} \cup N_G(v))$

While ($Y \neq \phi$)

Choose $v \in Y$

$C(v) = Ncl$, $B = B \cup N_{G(Y)}(v)$, $Y = Y \setminus (\{v\} \cup N_G(v))$

Endwhile

Endif

Endwhile

End_InitPop

Figure 10. Binary cuckoo search

Input: N and p_a
Output: the last best solution
Objective function $f(x)$, $x = (x1, \ldots, xd)^T$;
Initial a population of N host nests xi (i = 1, 2, ..., N);
while (t < MaxGeneration) or (stop criterion){
• Get a cuckoo (say i) randomly by Lévy flights;
• Get its binary representation by the modified BSR algorithm;
• Evaluate its quality/fitness Fi;
• Choose a nest among N (say j) randomly;
• Get its binary representation by the modified BSR algorithm;
• Evaluate its quality/fitness Fj;
• if ($F_i > F_j$){
Replace j by the new solution;
}
• Get the binary representation by the modified BSR algorithm of all the nests and evaluate their qualities;
• Abandon a fraction (pa) of worse nests;
• Build new ones at new locations via Lévy flights;

• Get there binary representation by the modified BSR

5. IMPLEMENTATION AND VALDATION

BCS for graph coloring (BCSCOL) is implemented in Matlab 7.9 with Intel core i3 processor and 4 GB of memory. The algorithm has been tested 30 times for each instance.To assess the efficiency of our approach, a set of standard DIMACS benchmark have been used. The results are given in Table 1, the first column is the name of the instance, the second column contains the number of vertices, the third contains the number of edges, the fourth column contains the chromatic number, the fifth column contains the results of the first version of our approach (BCSCOL1) that generates the initial population randomly, the sixth column contains the results of the second version that uses the modified RLF for generation the initial solution (BCSCOL2),the seventh column contains the results of the third version that uses the BSR proposed in (Gherboudj et al., 2012),the eighth column contain the result of the heuristic used for generating the initial population, and the final columns contain the results of two genetic algorithms proposed in (Hindi & Yampolskiy, 2012) and (Abbasian & Mouhoub, 2011) respectively. In all experiments, we have set the parameters of BCS as follows: *pa* = 65%, the number of cuckoos is 8. Finally, Friedman test is used to compare statistically the results found.

First, there are clear differences between the different versions of the proposed approach and the best solutions given by the randomized constructive heuristic InitPop (Figure 11).

Moreover, BCSCOL2 which uses a modified BSR operator is the closest to the exact results, followed by the BCSCOL3 and BCSCOL1. According to Table 1, the BCSCOL2 was able to reach the optimum in 11 cases out

Table 1. Results on DIMACS graphs

Instance	|V|	|E|	χ	BCSCOL1	BCSCOL2	BCSCOL3	InitPop	GA	PGA
mycel3	11	20	4	4	4	4	4	4	4
mycel4	23	71	5	5	5	5	5	5	5
queen5_5	25	160	5	5	5	5	7	5	5
queen6_6	36	290	7	9	8	9	9	7	8
myciel5	47	236	6	6	6	6	6	6	6
Huck	74	301	11	11	11	11	11	11	11
Jean	80	254	10	10	10	10	11	10	10
David	87	406	11	11	11	11	12	11	11
games120	120	638	9	9	9	9	9	9	9
miles250	128	387	8	8	8	8	11	8	8
miles500	128	1170	20	22	21	22	22	-	-
anna	138	493	11	11	11	11	11	11	11
fpsol2.i.1	496	11654	65	65	65	65	66	65	65

of 13. For the graph queen6_6 and miles500 the difference was only one. We also note the effectiveness of the introduction of the Constructive heuristic for generating the initial population in BCSCOL2 since it has improved the results achieved by BCSCOL1 that has not a statistically significant difference compared to the results obtained by InitPop (Figure 11), We also note the effectiveness of the use of the modified BSR in BCSCOL2 since it has improved the results achieved by BCSCOL3 which use the BSR proposed in (Gherboudj et al., 2012).

The statistical Friedman test in Figure 12 represents a comparison of the Exact, the BCSCOL, the GA and the PGA results. This statistical test shows that the difference between Exact and BCSCOL results is not statistically significant. Consequently, the obtained results confirm that the proposed algorithm gives good and promising results, compared to the genetic algorithm GA(Hindi & Yampolskiy, 2012) and to the parallel version PGA(Abbasian & Mouhoub, 2011), BCSCOL has the same performances to the PGA and comparable results to GA algorithm that have managed to obtain the optimal results through using a more sophisticated specific operators.

The examination of the results shows that very good results can be obtained with a good construction algorithm and more sophisticated operations in order to avoid exploring unpromising areas in the search space, and so accelerate the algorithm convergence.

6. CONCLUSION

In this work, we have presented a discrete binary version of CS algorithm for the graph coloring problem called BCSCOL. The main contributions of our approach are: the defini-

Figure 11. Friedman test compares the different versions of the proposed algorithm

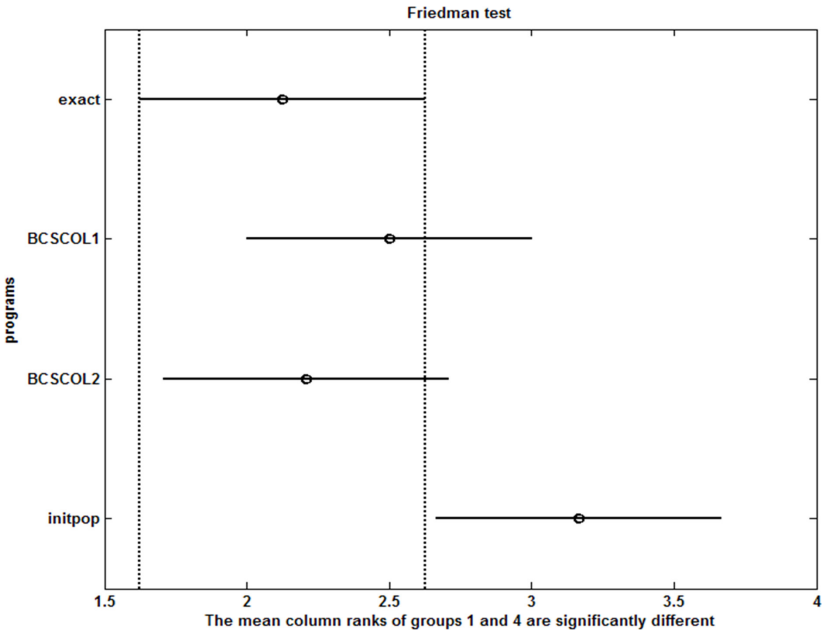
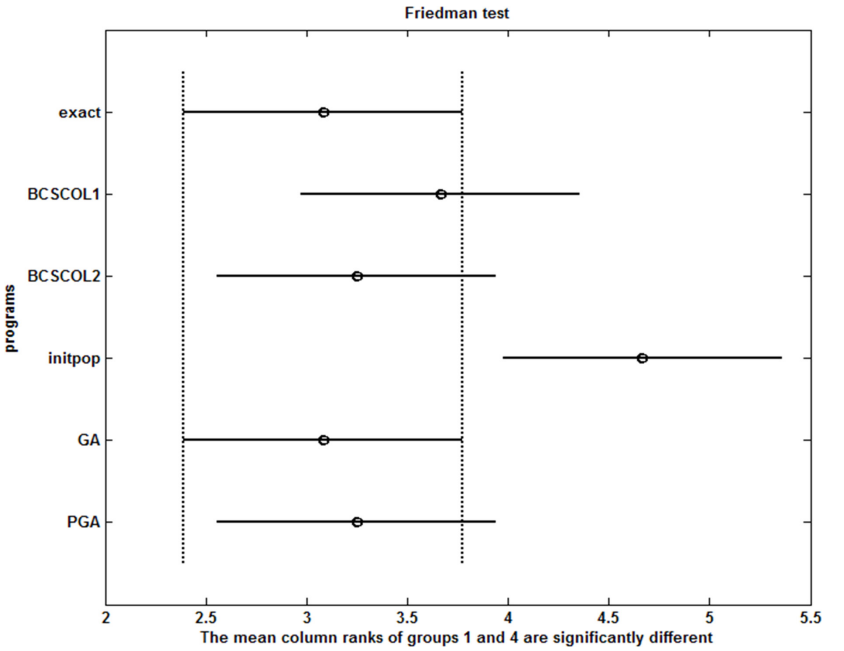


Figure 12. Friedman test compares the proposed algorithm against genetic algorithms



tion of a new binary representation for the graph coloring solution, the use of a modified Binary Solution Representation operation to avoid the infeasible solution, and the use of a modified RLF to create the initial solution. The approach has been thoroughly assessed with different instance types and problem sizes taken from the DIMACS benchmark. The proposed algorithm reduces efficiently the population size, and the number of iterations to have the optimal solution. However, there are several issues to improve our algorithm. Firstly, in order to improve the performance of our algorithm, it is better to integrate a local search method like tabu search in the core of the algorithm. In addition, integration of other operations inspired from other popular algorithms such as PSO or GA will also be potentially fruitful. the importance of the project management skills.

REFERENCES

Abbasian, R., & Mouhoub, M. (2011, July). An efficient hierarchical parallel genetic algorithm for graph coloring problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 521-528). ACM. doi:10.1145/2001576.2001648

Avanthay, C., Hertz, A., & Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, *151*(2), 379–388. doi:10.1016/S0377-2217(02)00832-9

Barthelemy, P., Bertolotti, J., & Wiersma, D. S. (2008). A Lévy flight for light. *Nature, 453*(7194), 495–498. doi:10.1038/nature06948 PMID:18497819

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, *22*(4), 251–256. doi:10.1145/359094.359101

de Werra, D., Eisenbeis, C., Lelait, S., & Marmol, B. (1999). On a graph-theoretical model for cyclic register allocation. *Discrete Applied Mathematics*, *93*(2), 191–203. doi:10.1016/S0166-218X(99)00105-5

Dhivya, M., Sundarambal, M., & Anand, L. N. (2011). Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (CBPA). *Int'l J. of Communications. Network and System Sciences*, *4*(04), 249–255. doi:10.4236/ijcns.2011.44030

Dowsland, K. A., & Thompson, J. M. (2008). An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics*, *156*(3), 313–324. doi:10.1016/j.dam.2007.03.025

Fister, I. Jr, Fister, I., & Brest, J. (2012). A hybrid artificial bee colony algorithm for graph 3-coloring. In *Swarm and Evolutionary Computation* (pp. 66–74). Springer Berlin Heidelberg. doi:10.1007/978-3-642-29353-5_8

Fleurent, C., & Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, *63*(3), 437–461. doi:10.1007/BF02125407

Gamache, M., Hertz, A., & Ouellet, J. O. (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers & Operations Research*, *34*(8), 2384–2395. doi:10.1016/j.cor.2005.09.010

Gherboudj, A., Layeb, A., & Chikhi, S. (2012). Solving 0–1 knapsack problems by a discrete binary version of cuckoo search algorithm. *International Journal of Bio-inspired Computation*, *4*(4), 229–236. doi:10.1504/IJBIC.2012.048063

Hertz, A., & de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, *39*(4), 345–351. doi:10.1007/BF02239976

Hertz, A., Plumettaz, M., & Zufferey, N. (2008). Variable space search for graph coloring. *Discrete Applied Mathematics*, *156*(13), 2551–2560. doi:10.1016/j.dam.2008.03.022

Hindi, M. M., & Yampolskiy, R. V. (2012, April). Genetic algorithm applied to the graph coloring problem. In *Proc. 23rd Midwest Artificial Intelligence and Cognitive Science Conf* (pp. 61-66). Academic Press.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, *39*(3), 378–406. doi:10.1287/opre.39.3.378

Kubale, M., & Jackowski, B. (1985). A generalized implicit enumeration algorithm for graph coloring. *Communications of the ACM*, *28*(4), 412–418. doi:10.1145/3341.3350

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, *84*(6), 489–506. doi:10.6028/jres.084.024

Liao, C. J., Tseng, C. T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research*, *34*(10), 3099–3111. doi:10.1016/j.cor.2005.11.017

Lü, Z., & Hao, J. K. (2010). A memetic algorithm for graph coloring. *European Journal of Operational Research*, *203*(1), 241–250. doi:10.1016/j.ejor.2009.07.016

Mabrouk, B. B., Hasni, H., & Mahjoub, Z. (2009). On a parallel genetic–tabu search based algorithm for solving the graph colouring problem. *European Journal of Operational Research*, *197*(3), 1192–1201. doi:10.1016/j.ejor.2008.03.050

Matula, D. W., Marble, G., & Isaacson, J. D. (1972). Graph coloring algorithms. *Graph Theory and Computing*, 109-122.

Mehrotra, A., & Trick, M. A. (1996). A column generation approach for graph coloring. *informs Journal on Computing, 8*(4), 344-354.

Méndez-Díaz, I., & Zabala, P. (2006). A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, *154*(5), 826–847. doi:10.1016/j.dam.2005.05.022

Michael, R. G., & David, S. J. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: WH Freeman & Co.

Pavlyukevich, I. (2007). Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, *226*(2), 1830–1844. doi:10.1016/j.jcp.2007.06.008

Payne, R. B. (2005). *The cuckoos* (Vol. 15). Oxford University Press.

Pongchairerks, P. (2009). Particle swarm optimization algorithm applied to scheduling problems. *ScienceAsia*, *35*(1), 89–94. doi:10.2306/scienceasia1513-1874.2009.35.089

Rodrigues, D., Pereira, L. A., Almeida, T. N. S., Papa, J. P., Souza, A. N., Ramos, C. C., & Yang, X. S. (2013, May). BCS: A binary cuckoo search algorithm for feature selection. In *Proceedings of Circuits and Systems (ISCAS),* (pp. 465-468). IEEE. doi:10.1109/ISCAS.2013.6571881

Smith, D. H., Hurley, S., & Thiel, S. U. (1998). Improving heuristics for the frequency assignment problem. *European Journal of Operational Research*, *107*(1), 76–86. doi:10.1016/S0377-2217(98)80006-4

Tein, L. H., & 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* (pp. 395-409). IMT-GT.

Welsh, D. J., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, *10*(1), 85–86. doi:10.1093/comjnl/10.1.85

Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, *1*(4), 330–343. doi:10.1504/IJMMNO.2010.035430

Yildiz, A. R. (2013). Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *International Journal of Advanced Manufacturing Technology*, *64*(1-4), 55–61. doi:10.1007/s00170-012-4013-7

Zhan, Z. H., & Zhang, J. (2009). Discrete particle swarm optimization for multiple destination routing problems. In *Applications of Evolutionary Computing* (pp. 117–122). Springer Berlin Heidelberg. doi:10.1007/978-3-642-01129-0_15

Zhou, Y., Zheng, H., Luo, Q., & Wu, J. (2013). An improved Cuckoo search algorithm for solving planar graph coloring problem. *Applications of Mathematics*, *7*(2), 785–792.

Halima Djelloul received her Masters degree from Mentouri University of Constantine, Algeria in 2011 and she is a PhD student at Mentouri University of Constantine, Algeria in 2012. Her current research interests include optimisation methods and their applications to solve several combinatorial optimisation problems.

Abdesslem Layeb is an Associate Professor in the Department of Computer Science at the Mentouri University of Constantine, Algeria. He is a member of MISC Laboratory. He receivedhis PhD in Computer Science from the University Mentouri of Constantine, Algeria. He isinterested by combinatorial optimisation methods and their applications to solve several problems from different domains like bioinformatics, imagery, formal methods, etc.

Salim Chikhi received his PhD in Computer Science from the University of Constantine. In 2005,he is currently a Professor at Constantine University 2, Algeria. He is the Leader of the SCALTeam of MISC Laboratory. His research areas include soft computing and artificial life techniques and their application in several domains.