Applied Soft Computing 33 (2015) 48–64

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem

Shadi Mahmoudi [a,*], Shahriar Lotfi [b]

[a] Computer Engineering Department, College of Nabi Akram, Tabriz, Iran
[b] Computer Science Department, University of Tabriz, Tabriz, Iran

ARTICLE INFO

Article history:
Received 28 May 2014
Received in revised form 6 December 2014
Accepted 5 April 2015
Available online 18 April 2015

Keywords:
Modified cuckoo optimization algorithm (MCOA)
Optimization
Graph coloring problem
Non-linear optimization

ABSTRACT

In recent years, various heuristic optimization methods have been developed. Many of these methods are inspired by swarm behaviors in nature, such as particle swarm optimization (PSO), firefly algorithm (FA) and cuckoo optimization algorithm (COA). Recently introduced COA, has proven its excellent capabilities, such as faster convergence and better global minimum achievement. In this paper a new approach for solving graph coloring problem based on COA was presented. Since COA at first was presented for solving continuous optimization problems, in this paper we use the COA for the graph coloring problem, we need a discrete COA. Hence, to apply COA to discrete search space, the standard arithmetic operators such as addition, subtraction and multiplication existent in COA migration operator based on the distance's theory needs to be redefined in the discrete space. Redefinition of the concept of the difference between the two habitats as the list of differential movements, COA is equipped with a means of solving the discrete nature of the non-permutation. A set of graph coloring benchmark problems are solved and its performance is compared with some well-known heuristic search methods. The obtained results confirm the high performance of the proposed method.

1. Introduction

Optimization problems are very important in many branches of sciences and their objectives are to find the best possible answer for a specific problem. Thus the need to search in solving optimization problems, has led to the emergence of different search algorithms. Evolutionary algorithms are a category of algorithms that has made remarkable progress in solving optimization problems. Genetic algorithms [1], particle swarm optimization [2], differential evolution [3], imperialist competitive algorithm [4], gravitational search algorithm [5], social-based algorithm [6], are most popular evolutionary optimization algorithms.

Most of evolutionary algorithms are considered to be continuous from the very early emergence; however, there are many issues with a discrete nature, such as scheduling applications, minimum distance in routing applications, graph partitioning and graph coloring problem. Solving such problems requires a series of changes on continuous algorithms to make them suitable for discrete problems.

The graph coloring problem (GCP) is one of the well-studied NP-hard problems in graph's theory [7]. The reasons why the graph coloring problem is important are divided into two parts. First, in terms of its application, samples of applications include timetabling and scheduling [8,9], radio frequency assignment [10], computer register allocation [11,12], printed circuit board testing [13], channel routing [14], and etc. Another reason is that the graph coloring problems in hard calculations at different levels are shown. Not only does this kind of stuff is making NP-Complete, but rather as an approximate version of the proposed NP-hard.

These two reasons to justify efforts to provide different ways of solving graph coloring problem are important enough and quickly new Meta heuristic methods for these problems have been provided and experimented. On the other hand, the graph coloring problem is a discrete problem and because of its use as outlined above, even the Meta heuristic methods in order to solve this problem continue to provide discrete version. So algorithm discretization is important. Also, most of approaches available to solve graph coloring problems are usually in two classes: (1) exact algorithms and (2) approximate algorithms. Due to the fact that exact algorithms can solve instances with up to 100 vertices; so for larger instances we need heuristic methods.

* Corresponding author. Present address: Faculty of Engineering, College of Nabi Akram, Tabriz, Iran.
E-mail addresses: mahmoudi.shadi@yahoo.com, mahmoudi.sh2@gmail.com (S. Mahmoudi), shahriar_lotfi@tabrizu.ac.ir (S. Lotfi).

http://dx.doi.org/10.1016/j.asoc.2015.04.020
1568-4946/

In this paper, we modify the most recent evolutionary algorithm called cuckoo optimization algorithm to make it able to solve graph coloring problem. Inspired by the lifestyle of a bird which is called cuckoo, cuckoo optimization algorithm (COA) was developed by Rajabioun in 2011 [15]. Specific egg laying and breeding of cuckoos is the basis of this novel optimization algorithm. Cuckoos used in this modeling exist in two forms: mature cuckoos and eggs. Mature cuckoos lay eggs in some other birds' nest and if these eggs are not recognized and not killed by host birds, they grow and become a mature cuckoo. Environmental features and the migration of societies (groups) of cuckoos hopefully lead them to converge and find the best environment for breeding and reproduction. This best environment is the global objective function. The main cuckoo optimization algorithm (COA) is introduced basically for continuous problems. Based on the performance of the COA algorithm in optimizing continues problems, it is predictable that this algorithm would be impressive in discrete optimization problems. The idea is that in order to solve graph coloring problem by COA, the standard arithmetic operators such as addition, subtraction and multiplication of COA migration operator, need to be redefined on discrete space.

Rest of the paper is organized as follows: In Section 2, a review of related works is presented. In Section 3, original cuckoo optimization algorithm (COA) is reviewed and it's different parts are studied in glance. The idea to modify basic COA and solving graph coloring problem is described in Section 4. The proposed algorithm is tested with some DIMACS CHALLENGE benchmarks in Section 5. Finally the conclusions and future work are presented in Section 6.

2. The related work

There are many several approaches for approximate algorithms, such as Greedy constructive, Local Search heuristics and Meta heuristics. The most of Greedy constructive methods are Dsature [16] and RLF [17], which are recently used to generate initial solutions to advanced Meta heuristic algorithms. Local Search method such as TS which is generally used in hybrid approach. Dorne and Hao [18] and Hertz and Werra [19] suggested Two Tabu Search algorithms. The most common approach for approximate algorithms are Meta heuristics or same evolutionary method. Costa and Hertz [20] present Ant Colony Optimization for graph coloring problem which embedded two constructive heuristics RLF and DSATUR.

Blöchliger and Zufferey [21] proposed an approach called FOOPARTIALCOL, which is based on Tabu Search. The method considers feasible, but partial solutions and tries to increase the size of the current partial solution. Ray et al. [22] proposed an evolutionary algorithm (GA) with double point Guided Mutation for the Graph Coloring problem, which could advance the performance level of simple GA dramatically. Lu and Hao [23] present a memetic algorithm (MACOL) to solve the graph coloring problem. The proposed MACOL algorithm integrates several distinguished features such as adaptive multi-parent crossover (AMPaX) operator and a distance-and-quality based replacement criterion for pool updating. Ge et al. [24] proposed a hybrid chaotic ant swarm approach for the graph coloring problem (CASCOL). This approach was based on a novel swarm intelligence technique called chaotic ant swarm (CAS) and a simple greedy sequential coloring, First-Fit algorithm.

Bessedik et al. [25] presented a marriage in honey bees optimization (MBO) approach to graph coloring problem (GCP). They proposed, a worker, in Bees colony of the following methods: local search, tabu search or a proposed-based ant colony system algorithm (IACSCol). The worker intervened at two levels; it improved initial and crossed solutions.

Qin et al. [26] introduced a Hybrid Discrete Particle Swarm Algorithm for Graph Coloring problem. To apply PSO to discrete problems, the standard arithmetic operators of PSO were required to be redefined over discrete space. A conception of distance over discrete solution space was introduced. Under this notion of distance, the PSO operators were redefined. After reinterpreting the composition of velocity of a particle, a general discrete PSO algorithm was proposed. In order to solve graph coloring problem with the discrete PSO algorithm, an algorithm to implement the crucial PSO operator difference of two positions (solutions) was designed. Then, a hybrid discrete PSO algorithm for graph coloring problem was proposed by combining a local search. The local search is Tabucol.

Fister and Brest [27] used a hybrid self-adaptive differential evolution algorithm for graph coloring that is comparable with the best heuristics for graph coloring. Abbasian et al. [28] presented an algorithm which combined a novel Sequential Graph Coloring Heuristic Algorithm (SGCHA) with a non-systematic method based on a cultural algorithm to solve the graph coloring problem. In this approach, they first use an estimator which is implemented with SGCHA to predict the minimum colors. Then, in the non-systematic part which has been designed using cultural algorithms, they improved the prediction. Various components of the cultural algorithm have been implemented to solve the GCP with a self-adaptive behavior in an efficient manner. Faraji [29] proposed a new algorithm for graph coloring based on Bees behavior in nature (BEECOL).

Hsu et al. [30] proposed a modified turbulent particle swarm optimization (named MTPSO) model for solving planar graph coloring problem based on particle swarm optimization. The proposed model is composed of the walking one strategy, assessment strategy, and turbulent strategy for improving the discrete PSO in the planar graph coloring. It was found that the walking one strategy and assessment strategy can improve the performance of MTPSO algorithm. The turbulent strategy also can be helpful to drive those lazy particles (i.e., those velocities of the particles smaller than threshold), and hence they can find a better solution. In addition, the experimental results indicated that the MTPSO model is more efficient and accurate than the modified PSO algorithm proposed by Cui et al.

Xiong, Bao, and Hu [31] evaluated the performances of multi-step-ahead forecasting of crude oil prices that were generated using the revised EMD-based feed-forward neural network (FNN) modeling framework. In addition, the three leading multi-step-ahead prediction strategies proposed in the literature were reviewed, and their performances in generating high-quality multi-step-ahead forecasting in the context of the crude oil market were compared. The quantitative and comprehensive assessments were performed on the basis of prediction accuracy and computational cost. The procured results demonstrated that the proposed EMD–SB M–FNN model using the MIMO strategy was a very promising prediction technique with high-quality forecasts and accredited computational loads.

Bao, Hu, and Xiong [32] proposed a particle swarm optimization algorithm (PSO) and pattern search (PS) based on memetic algorithm. In the proposed memetic algorithm, PSO was used to explore of the search space and detect the potential regions by PSO. To balance the exploration of PSO and exploitation of PS, a probabilistic selection strategy was also introduced to select the appropriate individuals among the current population to undergo local refinement. Experimental results supported that the local refinement with PS and the proposed selection strategy were effective, and finally indicated that the proposed PSO-PS based MA could be a promising alternative for SVMs parameters optimization.

Using multi-output support vector regression (MSVR), Xiong, Bao, and Hu [33] examined the possibility of forecasting an interval-valued stock price index series over short and long horizons. Furthermore, a firefly algorithm (FA)-based approach, built on the

established MSVR, was proposed to determine the parameters of MSVR (abbreviated as FA-MSVR). Three globally traded broad market indices were used to compare the performance of the proposed FA-MSVR method with selected counterparts. The results obtained in this study indicated the effectiveness of the proposed FA-MSVR method for forecasting interval-valued financial time series.

In another study, Xiong, Bao, and Hu [34] proposed the incorporation of end condition methods for time series prediction and provided large scale experimental evidences for the purpose of justification. Four well-established end condition methods including Mirror method, Coughlin's method, Slope-based method, and Rato's method were chosen, and support vector regression (SVR) was employed as the modeling technique. The obtained experimental results revealed a significant improvement by the proposed EMD-based SVR models with end condition methods. The EMD–SBM–SVR model and EMD–Rato–SVR model, in particular, achieved the most stable prediction performances regarding rank-based measure, indicating the superiority of slope-based method and Rato's method as end condition methods.

Using a hybrid Ant Colony Optimization (ACO)/particle swarm optimization (PSO) technique, Patel, Kabat, and Tripathy [35] presented a swarming agent based intelligent hybrid algorithm to optimize the multicast tree. The performance of the proposed algorithm was evaluated through extensive simulation. The proposed algorithm used the collective and coordination process for the mobile agents attached to each pattern. The simulation results were compared with two existing algorithms PSO-TREE and TGBACA and indicated that the proposed algorithm functioned better than the existing algorithms. It was also found that the proposed algorithm constructed the multicast tree patterns more sensibly such that the tree patterns not only satisfy the QoS constraints, but also tries to minimize the tree cost.

Considering that the re-entrant permutation flowshop scheduling problem minimizes the makespan, more recently, Xu et al. [36] adopted the CPLEX solver and developed a memetic algorithm (MA) to cope with the problem. Because it requires a large amount of computer memory to use CPLEX to solve the corresponding MIP model. They conducted computational experiments to test the effectiveness of the proposed algorithm and compared it with two existing heuristics. It was found that CPLEX could solve mid-size problem instances in a reasonable computing time, and the proposed MA performed better than CPLEX in terms of solution quality and solution time. In addition, the proposed solution provided an approach to handle the considered re-entrant flowshop problem, which is relevant to manufacturing and assembly processes.

In the same vein, Zhou et al. [37] proposed a new exact algorithm with clause learning for the graph coloring problem. The proposed algorithm cdclGCP extends a backtracking algorithm backGCP by applying the CDCL technology to discover the implicit constraints among vertices. The implicit constraints are utilized to prune the search space. The experimental results revealed that this algorithm outperformed other algorithms on many instances. More specifically, this algorithm allowed to close the open DIMACS instance 4-Fulli ns _5. The results also showed that the search tree size was significantly reduced.

In their study, Rahmani and Mir Hassani [38] proposed a discrete firefly algorithm to solve the capacitated facility location problem. Moreover, they employed an improvement method based on GA for transforming a trial solution into a better one; in fact, the method combines the discrete firefly algorithm with the standard (GA). The results of implementation have been compared with the well-known software, CPLEX (for small and medium scale test problems) and PSO algorithm (for large scale test problems). The preliminary results suggested that the proposed method performed the same as CPLEX for small size problems (because of negligible gap), and was also applicable for large scale problems.

3. Basic cuckoo optimization algorithm

This algorithm is one of the newest and most powerful evolutionary optimization techniques that have already been introduced. Cuckoo algorithm, inspired by the life of a bird named cuckoo which in 2009 by Yang and Deb has been developed [39]. This algorithm is based on the life of a cuckoo. This algorithm has been developed by flaying levy instead of isotropic random walk. Cuckoo algorithm later in 2011 by Rajabioun fully examined in more detail [15]. In this paper COA algorithm will be studied and before COA details study, discuss why choosing this method.

3.1. Incentives to choose COA algorithm

Argue that an optimization algorithm for solving optimization problems is the best method is not correct. Because every year new algorithms to cover the weaknesses of previous approaches have been emerged. Maybe later when GA and PSO optimization algorithm were considered the best of its kind. However, one can still see it in many of his articles. These methods are also sometimes referred to as the criteria used to compare the performance of the new method.

Some researchers believe that there is no reason that an evolutionary optimization method is better than another because each method because of evolution should be able to find the right answer. The point is that there are some methods like their real models, slow to evolve in nature. For example, in genetic evolution which is inspired by the true gene, the evolution occurs slowly, over many years. Perhaps this is why the GA requires a more iterations to find a suitable answer. In contrast, there is PSO algorithm, because of the inspiration of birds to act quickly to feed and act faster than GA in solving problems and this makes users more willing to use faster and more accurate evolutionary optimization algorithm. This is true of the cuckoo optimization algorithm and in applications that have been tested so far [40–46] this algorithm has been much better than the rest and show its efficiency. This can be a good motivation for the use of COA in solving many problems with high dimensions and very complex.

Fig. 1 shows the flowchart of cuckoo optimization algorithm introduced in [15]. COA is a global continues search method which inspired from a bird called cuckoo. Like other evolutionary algorithms, COA starts with an initial population, which is composed of a cuckoos.

This population of the cuckoos have eggs that lay them in host bird nest. Some of these eggs that are more similar to the host bird's eggs, have better chance to grow and become an adult cuckoo. Other eggs were identified and destroyed by the host bird. The grown eggs, show nests suitability of the area. The more eggs in the environment and are able to be saved, equally more profit (desire) is assigned to that area. The situation in which the maximum number of eggs to be saved, will be a parameter that COA wants to optimize it.

To maximize their eggs saving more, cuckoos looks for the best area. After the chicks were hatched and became mature, make some groups. Each group has its specific habitat. The best habitat is the next destination of cuckoos in other groups. All groups move to the best existent habitat. Each group resides in the area near the current location. Given the number an eggs a cuckoo will lay and cuckoos distant from current optimized habitat, a number of spawning radius calculated. Then cuckoos begin to lay egg in some nests within spawning radius. This process continues to reach the best hatchery. This optimized place is where that more cuckoos come together.

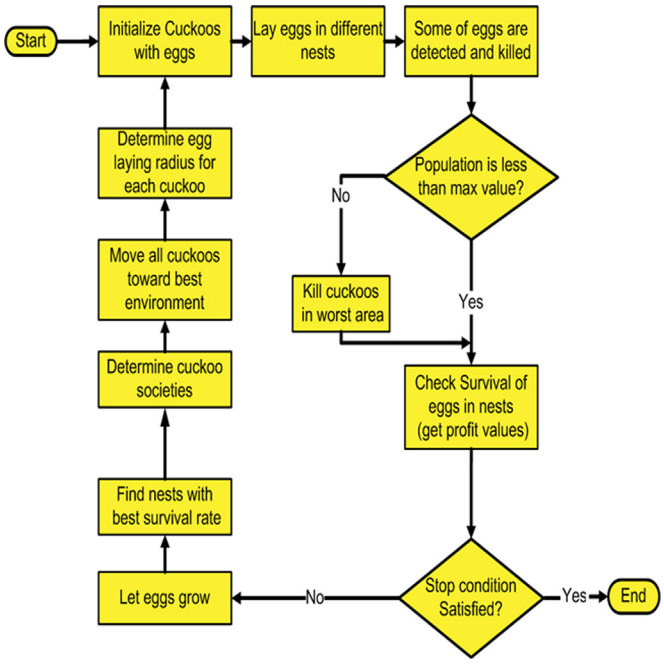


Fig. 1. Flowchart of cuckoo optimization algorithm.

3.2. Generating initial cuckoo habitat

In order to solve an optimization problem, it's necessary that the values of problem variables be formed as an array. In GA and PSO, this array specifies with Chromosome and the particle situation but in cuckoo optimization algorithm is called habitat or location. In an N_{var} dimensional optimization problem, a habitat will be a $1 \times N_{\text{var}}$ that shows the current location of cuckoos. A sample array is defined as the following equation:

$$\text{habitat} = [x_1, x_2, \ldots, x_{N_{var}}] \tag{1}$$

The suitability of current habitat achieves by (f_p) benefit function in habitat. According to the following equation:

$$\text{Profit} = f_p(\text{habitat}) = f_p(x_1, x_2, \ldots, x_{N_{var}}) \tag{2}$$

As it can be seen, COA is an algorithm that maximizes the benefit function. To use COA for Minimization problem solving, it is sufficient to multiply a negative mark in cost function, according to the following equation:

$$\text{Profit} = -\text{Cost}(\text{habitat}) = -f_c(x_1, x_2, \ldots, x_{N_{var}}) \tag{3}$$

To start optimization algorithm, a matrix with $N_{\text{pop}} \times N_{\text{var}}$ is created and then for each habitat, randomly egg assigns. In nature, each cuckoo lays from 5 to 20 eggs. So each cuckoo has an upper and a lower limit for egg laying at different iterations. Another habitat of real cuckoos is that they lay eggs within a maximum distance from their habitat. This maximum range is be called "Egg Laying Radius (ELR)". In an optimization problem with upper limit of varhi and lower limit of varlow for variables, each cuckoo has an ELR which is proportional to the total number of eggs; number of current cuckoo's eggs and also variable limits of varhi and varlow. The ELR is defined as in the following equation:

$$\text{ELR} = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times (var_{\text{hi}} - var_{\text{low}}) \tag{4}$$

where α is an integer, supposed to handle the maximum value of ELR.

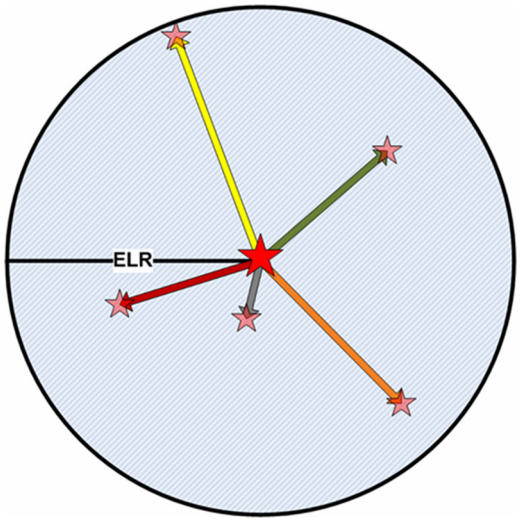


Fig. 2. Random egg laying in ELR, central red star is the initial habitat of the cuckoo with 5 eggs; pink stars are the eggs' new nest [15]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.3. Cuckoos' style for egg laying

Each cuckoo starts laying eggs randomly in some other host birds' nests within her ELR.

Fig. 2 gives a clear view of this concept.

After all cuckoos' eggs are laid in host birds' nests, some of them, less similar to host birds' own eggs are detected and thrown out of the nest. So after egg laying process, $p\%$ of all eggs (usually 10%), which their utility function is lower are destroyed. These eggs have no chance to grow. Rest of the eggs are fed and grow in the host's nest.

Another interesting thing about the cuckoo chick is that only one egg per nest is allowed to grow because when the cuckoo chick hatch from the egg, throw out host bird eggs and if host bird chicks are hatched sooner, cuckoo chick eat the most food host bird brings and after some days, host bird chicks eat die from hungry and just cuckoo chick survive.

3.4. Migration of cuckoos

After the young cuckoos grow and become mature, they live in their own area and society for some time. At the time of spawning, they migrate to new and better habitats that there is more chance of eggs surviving. After the formation of cuckoo groups in different areas of environmental (the search space of the problem), the group holds a position as a target for other cuckoo to migrate is selected.

When the mature cuckoos live in different areas of environment, determining that a cuckoo belongs to any group is a tough work to solve this problem, cuckoo grouping id done by k-means (a k between 3 and 5 is sufficient). Now that the cuckoo groups were formed, the average income is calculated to achieve the relative optimality of that group habitats. Then group with the highest amount of profit (optimality) is selected as a target group and other groups migrate toward it.

When moving toward goal point, the cuckoos don't fly all the way to the destination habitat. They only fly a part of the way and also have a deviation. This movement is clearly shown in Fig. 3.

As is evident from Fig. 3, each cuckoo travels only λ% of the total direction toward the ideal target and also has a deviation φ radians. These two parameters help cuckoo to search the environment more λ is a random number between 0 and 1, and φ is a number between $\pi/6$ and $-\pi/6$.

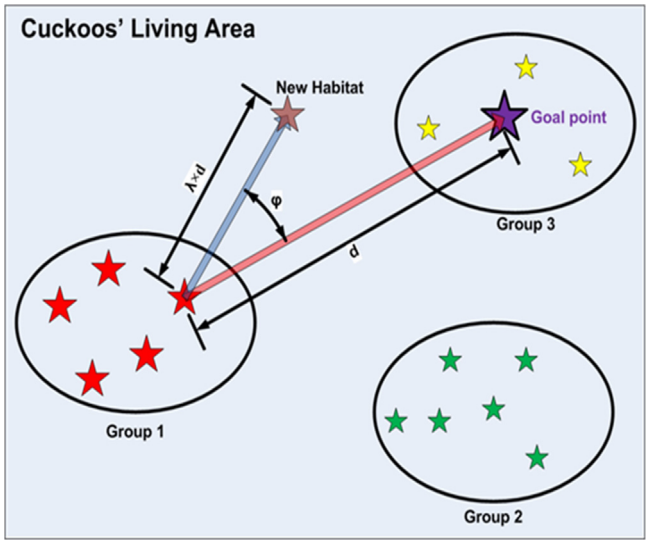


Fig. 3. Migration of a sample cuckoo toward goal habitat [15].

When all cuckoos migrate to target location and also new locations are specified, each cuckoo is an owner of some eggs. Given the number of eggs for each cuckoo, an ELR is specified for it and then pawning starts.

The Formula for migration operator in cuckoo optimization algorithm is as follow in the following equation:

$$X_{\text{NextHabitat}} = X_{\text{CurrentHabitat}} + F \times (X_{\text{GoalPoint}} - X_{\text{CurrentHabitat}}) \quad (5)$$

F is a parameter that causes deviation.

3.5. Eliminating cuckoos in worst habitats

Given that there is always equilibrium in birds' population, a number like $N_{\max}$, limits the maximum number of cuckoos that can live in the environment. This balance exists because of food limitations, being killed by predators and also inability to find proper nest for eggs.

The main steps of COA are presented in Fig. 4 as a pseudo-code. In the next section, COA is modified in a way suits for discrete optimization applications.

1. *Initialize cuckoo habitats with some random points on the profit function*
2. *Dedicate some eggs to each cuckoo*
3. *Define ELR for each cuckoo*
4. *Let cuckoos to lay eggs inside their corresponding ELR*
5. *Kill those eggs that are recognized by host birds*
6. *Let eggs hatch and chicks grow*
7. *Evaluate the habitat of each newly grown cuckoo*
8. *Limit cuckoos' maximum number in environment and kill those who live in worst habitats*
9. *Cluster cuckoos and find best group and select goal habitat*
10. *Let new cuckoo population immigrate toward goal habitat*
11. *if stop condition is satisfied stop, if not go to 2*

Fig. 4. Pseudo-code for cuckoo optimization algorithm [15].

4. Modified cuckoo optimization algorithm to solve graph

4.1. Select the method and graph coloring problem for COA

Before examining how to discrete COA for graph coloring problem, first we discuss why choosing discrete method. Since solvable problems in discrete space divide in various groups, that among these are Permutation and non-Permutation problems.

In permutation problems, each solution in search space corresponds to $\pi : \{1 \dots n\} \to \{1 \dots n\}$. In this case, numbers are assigned to problems answer that are just a code for their identification. Then an array of these codes constitute problem answer. Permutation view has many applications in discrete optimization. Traveling salesman problem and the scheduling problem are clear examples. Besides permutation problems, there are problems that are non-permutation. Here, each solution in search space corresponds to integral vector that gets discrete amount and maybe redundant numbers are exist in these vectors. Clear example is graph coloring problem.

Now, due to graph application mentioned in introduction, the main motivation to write this paper was to solve graph coloring problem by COA. Since COA is continues in nature, we changed it test the graph by that and because suggested algorithm doesn't have generality for all discrete problems, we named it the discrete algorithm. This suggested algorithm, is for non-permutation problems.

4.2. Notion of distance over discrete solution space

COA is essentially an algorithm which is presented to search continuous space and combinational optimization problems that can be solved in a discrete space. To apply COA on a discrete space, standard arithmetic operators of COA need to be redefined on discrete space which so that concept of space and geometric laws for solutions of discrete spaces is introduced and based on the theory of distance, operators of COA are redefined. Generally, in this section, necessary change in the basic algorithm COA to optimize the discrete problems is the relationship of migration.

Before performing discretization on migration operator, we must redefine some concepts in combinational optimization problems involving permutation of integer vectors, direct use of arithmetic operations is invalid because mathematic operations on symbols don't produce a meaningful direction. In addition, usually invalid responses are obtained. Distance is a good criterion for difference statement between two points in search space. Based on article [26,47].

We introduce a distance concept that is appropriate for COA.

Search space: (S, O) is a discrete search space which S is a set of solutions and O is the operations that apply to S members enroll others.

Metric space: A metric space is a pair (M, d), where M is a set and d is the distance in M, means that: $d : M \times M \to R$

Distance: given search space (S, O), distance between two solutions $s_i \in S$ and $s_j \in S$ is equal to the minimum number of sequence of actions of operator required for converting s_i *to* s_j. This definition refers to an interval available to the operator, which is shown in the following equation:

$$O(s_i) = s_k, O(s_k) = s_t, \dots, O(s_l) = s_j \quad (6)$$

According to this definition, a discrete search space (S, O) can be given as the corresponding metric space where $M = S$, and the distance between the solution $s_i \in S$ and $s_j \in S$ means; $d = (s_i, s_j)$ equals to *Distance* Definition. In other words, in metric space (M, d) introduced by the search space (S, O), geometric concept for

$d=(s_i, s_j)$ distance is the shortest length of route between two nodes; respectively that s_i and s_j are displayed.

4.2.1. Generalization of COA to discrete space

Now, in order to generalize COA to discrete search space, we need to redefine all arithmetic operators in the migration operator that the major is the difference between two habitats. The basic concept of migration in the COA is that each cuckoo tries to migrate into their best bio-region (target point) and reduce the gap with the best position in the space.

According to the definition of distance on discrete search space, for each position s_i and s_j, s_i can be turn to s_j through operator which is the shortest path between them. Each operator action reduces the distance between s_i and s_j a single unit. So, distance between two positions (candidate solutions) from the cuckoos is defined as the sequence of operator actions converting a position to the other.

In migration phase of the COA, objective is cuckoo motion toward best population. First, we will change the migration equation in COA continuous that is shown in Eq. (5).

F is a random number determines the pace of movement toward the target point. Thus, in the discrete COA, a method is required to model momentum and achieve to the target at each repeat. Then, concepts of subtraction; multiplication and additions in Eq. (5) for discrete space are redefined:

The difference between the two situations: for each position s_i and s_j, their difference $(s_i - s_j)$ equals to the sequence of the minimum number of times to apply the operator.

$$s_i - s_j = M = O(s_i), O(s_k), \ldots, O(s_l) = s_j \tag{7}$$

So

$$O(s_i) = s_k, O(s_k) = s_t, \ldots, O(s_l) = s_j \tag{8}$$

Eq. (8) means that difference operator in the cuckoo algorithm for non-permutation problems is equal to the sequence number of times applies to change operator of elements. In other words, to obtain the difference between two vectors in a discrete space, list of necessary motions to change from s_j to s_i, entitled $M_{j\to i}$ that obtains through element changes is defined. The difference between the two solutions is shown in the following equation:

$$M_{j\to i} = S_i \ominus S_j \tag{9}$$

Eq. (9) means that if the motion in the list $M_{j\to i}$ on the solution S_j is imposed, the solution S_i will be obtained. List $M_{j\to i}$ models the difference between the target point and the current position of the cuckoo. So, the difference between two points equals to the movement list that by triple component term (a, b, c) is displayed; means that in the indices a vector, changes b number into c.

Multiplication: Considering $F \in [0, 1]$ that is a decimal number and $M' = F \times |M|$. Here, multiplication operator means that the number of movements M is selected in the list. So to complete the operation, parameter F, as a random step toward the target point is defined and used. This idea just imposes a random numbers of movements within the list of $M_{j\to i}$ that is effective.

Summation: if s_i be position and M' be the list of movements, $s_i + M'$ means M' movement list imposes on vector s_i that creates a new position in the space.

So the migration operator is transformed into the following equation:

$$\begin{aligned} M_{j\to i} &= X_{\text{GoalPoint}} \ominus X_{\text{CurrentHabitat}} \\ X_{\text{NextHabitat}} &= X_{\text{CurrentHabitat}} \oplus F \otimes M_{j\to i} \end{aligned} \tag{10}$$

To explain the idea, we consider the following example; having two following situations in Fig. 5:

Clearly, highlighted bits between the current position of mother cuckoo and the target point is different. $X_{\text{GoalPoint}} - X_{\text{CurrentHabitat}}$

$X_{GoalPoint}$ =	1	4	3	3	2	3	4
$X_{CurrentHabitat}$ =	1	3	2	3	2	4	4

Fig. 5. An example.

$X_{NextHabitat}$ =	1	4	3	3	2	4	4

Fig. 6. New achieved condition.

must be defined as the list of motion that when imposes on $X_{\text{CurrentHabitat}}$, $X_{\text{GoalPoint}}$ will be generated. But on the other hand, according to the theory of cuckoo migration, as we know that only a percentage of migration distance from the current point to the target is traveled and there is a detour on the route. So, only a part of M on the $X_{\text{CurrentHabitat}}$ is applied. List of motions for this example is as follows in the following equation:

$$M = X_{\text{GoalPoint}} - X_{\text{CurrentHabitat}} = \{(2, 3, 4); (3, 2, 3); (6, 4, 3)\} \tag{11}$$

Imposing these movements on $X_{\text{CurrentHabitat}}$, in effect produce $X_{\text{GoalPoint}}$. But, in the cuckoo migration in the COA, only a percentage of the movements to $X_{\text{CurrentHabitat}}$ should be applied.

For this purpose, according to that $F \in [0, 1]$, though $F = 0.7$, $M' = \lceil F \times |M| \rceil$ equals to the first two moves of list M to the form of the following equation:

$$M' = \{(2, 3, 4); (3, 2, 3)\} \tag{12}$$

By applying the $X_{\text{NextHabitat}} = X_{\text{CurrentHabitat}} \oplus M'$, new position is obtained as follows in Fig. 6:

Pseudo-code of this model of discretization for other non-permutation problems, called MCOA is shown in the following algorithm:

Algorithm 1: modified cuckoo optimization algorithm (MCOA)
Input: Non-permutation problems with discrete nature
Output: A best solution
1. Initialize cuckoo habitats with some random points on the profit function in range
2. Dedicate some eggs to each cuckoo
3. Define ELR for each cuckoo
4. Let cuckoos to lay eggs inside their corresponding
5. Kill those eggs that are recognized by host birds
6. Let eggs hatch and chicks grow
7. Evaluate the habitat of each newly grown cuckoo through objective function
8. Limit cuckoos' maximum number in environment and kill those who live in worst habitats
9. Cluster cuckoos and find best group and select goal habitat
10. Let new cuckoo population immigrate toward goal habitat by Eq. (10)
11. If stop condition is satisfied stop, if not go to 2

4.3. Solving graph coloring problem using MCOA algorithm

The graph coloring problem can be defined as follows: Given a graph $G = (V, E)$ with vertex set V and edge set E and given an integer k, a k-coloring of G is a function of C: $V \to \{1 \ldots k\}$. The value $C(x)$ of vertex x is called the color of x. The vertices with color i $(1 \le i \le k)$, define a color class denoted as V_i. If two adjacent vertices x and y have the same color i, vertices x and y, the edge $[x, y]$ and color i are said to be conflicting. So the k-coloring without conflicting edges is said legal. A stable set is a subset of vertices that none of these are adjacent. Hence, a k-coloring is legal, if and only if its color classes are stable sets. Now in graph coloring problem, we want to color all nodes in a way that no two adjacent nodes have the same color and use the least amount of color. The least possible color number as a good coloring from graph G achieves is called chromatic

number and denoted by $\chi(G)$. Reference [48] identifies four different strategies for defining the search space which includes: legal strategy, penalty strategy, k-fixed partial legal strategy and k-fixed penalty strategy. The legal strategy and k-fixed partial legal strategy involves choosing the first k-coloring and then tries to minimize the number of conflicts for candidate k [28]. In this paper we used of k-fixed strategy mode.

This section describes how to solve the problem of graph coloring by algorithm MCOA we take the idea of paper [29,49]. Graph with a fixed number K as input entered into algorithm and output of the problem is finding the right coloring k. Initially $K=|V|$ is valued. Solution using the function is colored initiatively then neighborhood search in the section of egg laying algorithm to minimize the number of colors that avoid the collision is applied. Work is described in more detail.

COA algorithm contains several sections to produce a cuckoo or same candidate answer to the questioned: production of original responses; production of cuckoo's chicks as a child of cuckoo mother and immigrant to the optimal current location.

In the cuckoo algorithm, the initial population is usually generated randomly, but in this part, because of particular condition problem of graph coloring, the initial population is produced by a coloring function called Coloring Vertex in the following algorithm:

Algorithm 2: Initialize a solution with coloring_vertex
Input: currentColoring and V_a
Output: a k-coloring $C=\{V_1, V_2, \ldots, V_k\}$ of currentColoring
Create adjacentNodesSet (V_a);
2. Create adjacentNodesColors (adjacentNodesSet);
Set $N=|V|$;
4. FOR $j=1$ to N
IF $j \notin adjacentNodesColors$
Set $C(V_a)=j$;
ENDIF
ENDFOR

First, a node is randomly selected and colored by the Coloring Vertex in Algorithm 2. Then, other nodes are sorted based on the degree to have and according to the color of first node, and each node has its own specific color by Coloring Vertex algorithm respectively.

In other words, if V is a set of graph nodes and V_a a target node for coloring, in this case, $V_a \in V$ and $\deg(V_a)$, V_a node degree; $\deg(V_a)=m$ and $\{V_1, V_2, \ldots, V_m\}$ equals adjacent nodes to V_a; $C=\{c(V_1), c(V_2), \ldots, c(V_m)\}$ also are assigned colors to V_a adjacent nodes.

In Algorithm 2, N number of graph nodes and $c(V_a)$ is assigned color to the V_a node. In fact, this assigned the smallest color is not exist in the set C (colors assigned to V_a adjacent nodes) to the V_a node. In this algorithm, the set of used colors to graph coloring is a subset of natural numbers that the color begins by number one.

Now for start, we have answers that do not have any interference color. COA should only attempt to reduce the number of used colors. For this purpose, the laying egg part changes in a special way that is suitable for graph coloring.

Now, for understanding how mother cuckoos are produced, we follow this example. A produced habitat by MCOA is in a form as follows in Fig. 7:

The number of positions in habitat equals with the number of nodes in the graph. Each number in a position of a habitat shows the color code for that node.

For Fig. 8, Algorithm 2 works as follow. The solution that is supposed to being colored and the first node that also is supposed to

1	2	...	N-1	N

Fig. 7. A solution view on MCOA.

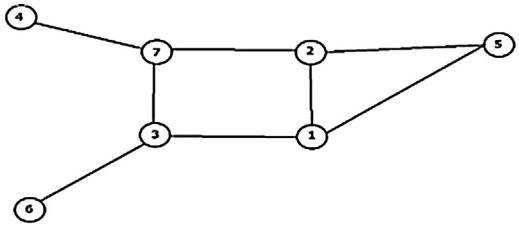


Fig. 8. Algorithm 2 test graph.

currentColoring=	7	7	7	7	7	7	7

Fig. 9. Preliminary currentColoring.

being colored are given to Algorithm 2. In Fig. 8 which have 7 nodes, a number randomly selects for coloring the first node.

For example, $V_a=7$ and in which we consider the solution to Fig. 9 that for all nodes choose the highest color and give to Algorithm 2 to color.

Now in coloring vertex algorithm, first, we hold adjacent node of V_a for node 7 and allocated color to this node like Fig. 10.

Now, As Algorithm 2 implies, coloring vertex starts from the least color means 1 and if it is not within the color of adjacent nodes of V_a, allocates that color to V_a that in this example, V_a gets color 1 and the solution changes to Fig. 11.

If V be the set of initial nodes of the graph, as it was shown in the following equation:

$$V = \{1, 2, 3, 4, 5, 6, \} \tag{13}$$

After coloring the V_a node, this will eliminates from V set of this node; according to the following equation:

$$V = \{1, 2, 3, 4, 5, 6\} \tag{14}$$

Now, for remaining nodes, calculate the adjacent nodes for each node using adjacent matrix, Fig. 12 which derived from graph; means that for each column, we sum amounts 1 to specify each node degree and show that in array $\deg v_i$ in the following equation:

$$Deg v_i = \begin{matrix} 3 \\ 3 \\ 3 \\ 1 \\ 2 \\ 1 \end{matrix} \qquad i = 1, 2, \ldots, 6 \tag{15}$$

Then, arrange the $\deg v_i$ in an ascending form and in array index, hold the sorting index as Eq. (16):

Then, for coloring based on degsorted, select the node with the highest degree from index as next node that here, we send node 1 with degree 3 and current coloring like Fig. 12 to Algorithm 2 and all steps within the algorithm repeats that derive the numbers of

Adjacentnodenumbers=	2	3	4

Adjacentnodecolors=	7	7	7

Fig. 10. Adjacent node numbers and adjacent node colors.

currentColoring=	7	7	7	7	7	7	1

Fig. 11. CurrentColoring after node 7 coloring.

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

Fig. 12. Adjacent matrix of the graph in Fig. 8.

adjacent nodes $V_a = 1$ with their colors from adjacent matrix like Fig. 13:

$$\text{degsorted} = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \end{bmatrix}, \quad \text{index} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix} \tag{16}$$

Now, compares the least color means 1 with colors within the adjacent node colors and not make interference so, color allocates to V_a and solution comes to the form of Fig. 14.

So, for coloring remained nodes, each next node selects degsorted and sends with solution to the algorithm that by this process, finally below solution produces like Fig. 15:

So by this trend, mother cuckoo is produced in this way by algorithm instead of being produced randomly.

4.3.1. Lay egg algorithm

In this section of the COA algorithm according to the current population status of cuckoos, we are attempting to produce an optimal permutation for cuckoo eggs. For this purpose, we search the neighborhood that was defined as in Algorithm 3:

To explain this phase, we suggest the following definition:

Definition 1. If $\deg(V_a) = m$ and $\{V_1, V_2, \ldots, Vm\}$ are adjacent nodes of V_a nodes then:

$$Hdeg(Va) = deg(V1) + deg(V2) + \cdots + deg(Vm) \tag{17}$$

currentColoring=	1	7	7	7	7	7	1

Fig. 14. CurrentColoring after node1 coloring.

currentColoring=	1	2	2	2	3	1	1

Fig. 15. Produced mother cuckoo.

In Eq. (17), $Hdeg(V_a)$ is the total degrees of adjacent nodes of V_a node.

Definition 2. If $\{V_1, V_2, \ldots, V_{D_T}\}$ are nodes in the solutions S cuckoo with the same color and have T color then:

$$Q_{S_T} = \sum_{i=1}^{D_T} Hdeg(V_i) \tag{18}$$

In Eq. (18) D_T is equal to the number of nodes in the solution S cuckoo and has T color.

Definition 3. If N is the number of nodes of graph and $k(S)$ is the number of colors S cuckoo uses for coloring the graph then:

$$F_S = \frac{(N/k(S))}{3} \tag{19}$$

Definition 4. If the solution of S cuckoo uses $k(S)$ color for coloring the graph so then

$$X_{s_i} = \begin{cases} Q_{S_i} - Q_{S_{i+1}} & 0 < i < K(S) \\ Q_{s_i} - Q_{s_1} & i = K(S) \end{cases} \tag{20}$$

At this stage, each cuckoo seeks its neighborhood solutions. The S cuckoo is used for all the colors to solution and forms X_{S_i} using Definition 4.

IfnumberOfNodes4EachColor = Q_{S_T} numberOfNodes4EachColor = length (nodesWithColors)

First i color will choose. After selecting first color, these steps repeat to build the first new solutions S cuckoo.

Step 1: If $i > 1$ and $i < K(S)$ and number of Nodes for Each $\text{Color}_{i-1} > F_S$ and $i - 1 \notin b_S$, then all nodes in the solution S cuckoo are colored $i - 1$ and coloring by Coloring Vertex function and $i - 1$can be added to the b_s set. B_S collection represents colors that are used in neighborhood search by S cuckoo. In this phase, F_S is obtained from the Definition 3 and $k(S)$ is the number of colors for coloring the graph which used by cuckoo S.

Step 2: If $i > 1$ and $i < K(S)$ and number of Nodes for Each $\text{Color}_{i+1} > F_S$ and $i + 1 \notin b_S$ then all nodes S cuckoo solutions have $i + 1$ color can be colored by Coloring Vertex function and $i + 1$ is added to the b_S set.

Step 3: If $i \notin b_S$ then all nodes that S cuckoo solutions have i color can be colored by Coloring Vertex function and i is added to the b_S set.

Step 4: The next color of nodes to be randomly selected and by Coloring Vertex algorithm allocates a new color to each node. Color choosing is completely random. The random selection, causes a new population of eggs, which are produced on the mother population, should be produced and placed in a better place. In this section, according to each mother, only one egg is produced and placed. It is a variation on the basic COA algorithm. After obtaining color i, go back to step 1 and repeat steps 1 to 4 until all nodes of

Adjacentnodenumbers=	2	3	5

Adjacentnodecolors=	7	7	7

Fig. 13. The view of adjacent node numbers and adjacent node colors.

currentColoring=	1	2	2	1	3	1	3

Fig. 16. Mother cuckoo samples generated by the algorithm 1.

graph are colored and neighboring solutions are generated. And its algorithm is as follow:

Algorithm 3: search neighbourhood of each position
Input: currentColoring which colored by Algorithm 1
Output: a k-coloring $C = \{V_1, V_2, \ldots, V_k\}$ of currentColoring
1. Set K = max(currentColoring);//K is the number of colors that used for currentColoring
2. Set $N = |V|$;
3. FOR colorNumber = 1 to K
find nodes with color of colorNumber
ENDFOR
4. CalculateF_s by Eq. (19)
5. Calculate X_{si} by Eq. (20)
6. Set[Xs_sorted Xs_index] = sort(Xs,'descend');
7. Set newTmpColoring = generate n random number between 1 and N
8. WHILE any(newTmpColoring = =N)
num = Generate Permutation of (length(Xs_index));
index = Xs_index(num(1));
Apply step 1 by Algorithm 3.a for color $i-1$
Apply step 2 by Algorithm 3.b for color $i+1$
Apply step 3 by Algorithm 3.c for color i
ENDWHILE

Algorithm 3.a: step 1
Input: index
Output: a coloring vertex index-1 of newTmpColoring
If $i > 1$ and $i < K(S)$ and numberOfNodes4EachColor$_{i-1} > F_S$ and $i - 1 \notin b_S$ then
1. Set allNodesWithSameColorss = nodesWithColor(index-1);
2. FOR counterColor = 1to length(allNodesWithSameColorss)
coloring newTmpColoring by Algorithm 2 with this
inputs(allNodesWithSameColorss(counterColor),newTmpColoring);
ENDFOR
ENDIF

Algorithm 3.b: step 2
Input: index
Output: a coloring vertex index + 1 of newTmpColoring
If $i > 1$ and $i < K(S)$ andnumberOfNodes4EachColor$_{i+1} > F_S$ and $i - 1 \notin b_S$ then
1. Set allNodesWithSameColorss = nodesWithColor(index + 1);
2. FOR counterColor = 1 to length(allNodesWithSameColorss)
coloring newTmpColoring by Algorithm 2 with this
inputs(allNodesWithSameColorss(counterColor),newTmpColoring);
ENDFOR
ENDIF

Algorithm 3.c: step 3
Input: index
Output: a coloring vertex index of newTmpColoring
If $i \notin b_S$ then
1. Set allNodesWithSameColorss = nodesWithColor(index);
2. FOR counterColor = 1 to length(allNodesWithSameColorss)
coloring newTmpColoring by Algorithm 2 with this
inputs(allNodesWithSameColorss(counterColor),newTmpColoring);
ENDFOR
ENDIF

Now, in order to understand a neighborhood search, we consider an example:

Neighborhood search of mother cuckoo is passing to this part and the following is extracted:

For example, if currentColoring is as follow, neighborhood search for making eggs of cuckoo children, to the form of Fig. 16 and by Eq. (21) acts

$$K = 3, \quad \text{nodesWithColor} = \begin{bmatrix} 1, 4, 6 \\ 2, 3 \\ 5, 7 \end{bmatrix} \tag{21}$$

Newtempcoloring=	7	7	7	7	7	7	7

Fig. 17. Preliminary Newtempcoloring.

In Eq. (21), nodes with color is a matrix that shows nodes which get the same color, means; nodes 1,4,6 from considered graph that located in row 1 have color 1 and nodes 2,3 in row 2 have color 2 and so as nodes 5,7,color 3 and k is the used colors in mother cuckoo means, Fig. 16.

$$\begin{aligned} \text{numberOfNodes4EachColor} &= \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix}, X_S = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, X_{S_{sorted}} \\ &= \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, X_S\_index = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \end{aligned} \tag{22}$$

$$F_S = \frac{7}{3/3} = 0.7778 \tag{23}$$

A random permutation can be constructed based on the number of colors cuckoo's mother, as the following equation:

$$\text{Num} = \{3, 2, 1\} \tag{24}$$

Now, based on index = $Xs\_index(num(1)) = Xs\_index(3) = 2$, implement defined procedure in Algorithm 3 for producing child cuckoo, means now that index = 2.

Step 1: extract all nodes that have color index-1 form matrix nodes with color in Eq. (21) and have Eq. (25):

$$\text{All nodesWithSameColors} = 1, 4, 6 \tag{25}$$

And this is the order of nodes that coloring vertex function do in Algorithm 1 on solution Newtempcoloring for coloring in Fig. 17 that the Process is shown in Fig. 18:

And Color index-1 is set in Bs As in the following equation:

$$B_S = \{1\} \tag{26}$$

Step 2: then extract all nodes that have color index + 1 from matrix nodes with color and have Eq. (27):

$$\text{all nodes with the same color} = 5, 7 \tag{27}$$

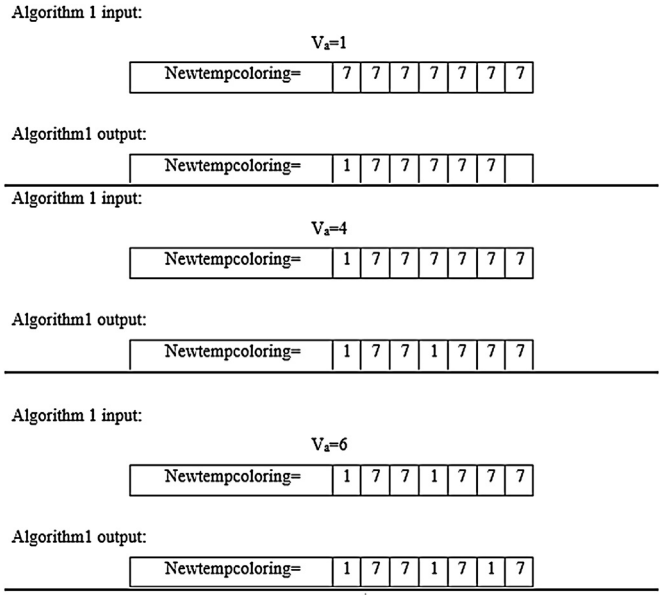


Fig. 18. Step1 stages view.

Newtempcoloring=	1	7	7	1	7	1	7

Fig. 19. Final output of step 1.

Algorithm 1 input:

V_a=5

Newtempcoloring=	1	7	7	1	7	1	7

Algorithm 1 input:

Newtempcoloring=	1	7	7	1	2	1	7

Algorithm 1 input:

V_a=7

Newtempcoloring=	1	7	7	1	2	1	7

Algorithm 1 input:

Newtempcoloring=	1	7	7	1	2	1	2

Fig. 20. Step 2 stages view.

Newtempcoloring=	1	7	7	1	2	1	2

Fig. 21. Final output of step 2.

And proceed step 1 output which is Fig. 19 that the process is shown in Fig. 20:

Bs set updates with color index + 1 according to the following equation (28):

$$B_s = \{1, 2\} \tag{28}$$

Step 3: now extract all nodes that have color index from matrix nodes with color and have Eq. (29):

$$\text{All nodes with the same color} = 2, 3 \tag{29}$$

And proceed Fig. 21 which is the output of step 2 that the process is shown in Fig. 22:

Bs set updates with color index according to the following equation:

$$B_s = \{1, 2, 3\} \tag{30}$$

By this, from mother cuckoo, means Fig. 16, produce a child in that local region in the form of Fig. 23 according to neighborhood search.

Algorithm 1 input:

V_a=2

Newtempcoloring=	1	7	7	1	2	1	2

Algorithm 1 input:

Newtempcoloring=	1	3	7	1	2	1	2

Algorithm 1 input:

V_a=3

Newtempcoloring=	1	3	7	1	2	1	2

Algorithm 1 input:

Newtempcoloring=	1	3	3	1	2	1	2

Fig. 22. Step 3 stages view.

Newtempcoloring=	1	3	3	1	2	1	2

Fig. 23. Mother cuckoo child.

And this repeats for all mother cuckoos. Finally, Algorithm 5 shows the imposed MCOA algorithm on coloring problem that is called MCOACOL. Complementary algorithms don't giver near optimal responses for such problems. Therefore, for responses optimization, we combine MCOACOL algorithm with local search Tabucol of Algorithm 4.

Algorithm 4: TabuCol for GCP
Input: A graph $G = (V, E)$, an integer $k > 0$
Output: Solution C^*
1. Build a random solution C
2. Set $C^* = C$ and $iter = 0$
3. Set the Tabu List to empty set
4. WHILE $f(C) \sim = 0$ or $iter \leq MaxIter$
   Set $iter = iter + 1$
   Choose a candidate *1-move(v,i)* with minimum value $\delta(v,i)$
   Introduce *move(v,C(v))* into Tabu List or $L + \lambda f(C)$
   Set $C = C + (v,i)$
   if $f(C) < f(C^*)$ *then* set $C^* = C$
ENDWHILE

Algorithm 5: Apply MCOA to Graph Coloring Problem (MCOACOL)
Input: A graph $G = (V, E)$
Output: A best k-coloring of G
1. Initialize cuckoo habitats with Algorithm 2
2. Dedicate some eggs to each cuckoo
3. Define ELR for each cuckoo
4. Let cuckoos to lay eggs inside their corresponding with Algorithm 3
5. Kill those eggs that are recognized by host birds
6. Let eggs hatch and chicks grow
7. Evaluate the habitat of each newly grown cuckoo through objective function
8. Limit cuckoos' maximum number in environment and kill those who live in worst habitats
9. Cluster cuckoos and find best group and select goal habitat
10. Let new cuckoo population immigrate toward goal habitat by Eq. (10)
11. Apply Algorithm 4 for local search
12. If stop condition is satisfied stop, if not go to 2

4.3.2. Defined cost function

For graph coloring by this algorithm, it is sufficient to design a cost function for the problem and first, give known number of each graph as the dimensions of V optimization as input to algorithm MCOACOL that the aim is color numbers reduction based on penalty. Our defined cost function contains two different sections and is as follow:

(1) If there is no edge between nodes i and j and proposed colors for these two nodes are not the same, penalty increases by one unit. This penalty is used to reduce the number of colors used.
(2) The number of proposed colors for coloring the graph is the second value used, for which the first trace of color is stronger and more effective. Despite that these words are just auxiliary tool for penalty that makes a stronger impact; If not treated, but the number of colors used increases.

These two components by assigning weights to each of the cost function to be as below:

$$\text{Cost} = P_1 \times \text{Penalty} + P_2 \times \text{Number of used Colors} \tag{31}$$

where P_1 and P_2 are user-defined weights. The number of highest level for maximum colors maybe used and the minimum color is found by two departments of cost function is obtained. The final algorithm called MCOACOL and cost function of this algorithm is equal to Eq. (31).

Table 1
The parameters of the algorithm MCOACOL.

Parameters of MCOACOL	
Number of initial cuckoos	5
Maximum number of iterations	3000
Number of clusters for KNN	1
Maximum number of living cuckoos	50
varhi	Length(graph)
varlow	1

5. Results analysis of MCOACOL algorithm

In solving graph coloring problems to evaluate the algorithms, test function DIMACS (2002) of graphs are usually used [48]. Of course, most of these graphs, before this year, under various categories have been introduced by scientists. Some graphs of the set, were randomly and using programs of graph producer such as DSJ series David Johnson graphs is made. Some graphs also would be result of practical problems for example series graphs SCH Gary Londusky that involves the scheduling of class in each of the states with no spaces and spaces or series graphs LAT, which is related to the Latin square. Most important characteristics of each test graphs are the number of its vertices and the density of ledges. Labeling graphs is based on the characteristics and chromatic number of a graph, if known. For example, DSJC250.5 graph name involves 250 vertices and 0.5 density is of DSJ series. For this graphs,the time to answer, frequency of successful performance than the total numbers of performance and algorithm repetition among the most important of measurement criteria which have been used by researchers.

In this section, results of MCOACOL algorithm will be evaluated. Cost function parameters are approximate values in the range, parameters shall be set as trial and error for each graph sample by the user.

$0.0001 < p_1 < 0.1$

$1000 < p_2$

Table 1 shows the parameters MCOA for graph coloring.

The implementation of the proposed algorithm is tested on 97 samples. In 67 samples of graph, the optimal solution in 20 times will be obtained and in the 20 remaining, results of implementation includes digressions that in the attached Table 6 shown in bold and in Fig. 24,we show the success rate of these samples in terms of percentage. In effect, the success rate; the number of times that the algorithm is obtained a valid coloring of the graph is measured. As it can be seen from Fig. 24, solving classes of graphs DSJ and queen n_n seems hard. Algorithms with success rates above 60% have accurately been reported in accordance with color value and in most cases, the success rate is close to 100%.

The whole table of obtained results from algorithm MCOACOL that time and repetition is mentioned is attached in Appendix A.

In Table 2, the results obtained from implementing the proposed algorithm (MCOACOL) with the results of algorithms ABAC [50] and BEECOL [49] are compared. The first column shows the name of the graph, the second column shows the number of vertices, best known chromatic number of a graph is shown in the third column and the next column indicates the chromatic number results of the three algorithms are compared. As seen in the table, based on the number of vertices of the graph are classified.

The Performance of MCOACOL proposed algorithm with BEECOL and ABAC algorithms have been compared on 92 subjects of graph. As it can be seen from Table 2, MCOACOL algorithms as two algorithms that have been compared, easily and with accuracy and high speed are able to find known chromatic number of the graph.

In 8 cases: (qg.order40; queen6_6; queen8_8; queen9_9; queen10_10; queen15_15; queen16_16 and ash331GP1A) known chromatic number of our proposed algorithm is equal to known chromatic number of BEECOL algorithm in comparison with ABAC algorithms different are 1.

In 9 cases: (DSJC250.1; DSJC125.5; DSJC125.9; queen8_12; queen11_11; quenn12_12; queen13_13; queen14_14 and will199GP1A) chromatic number have 1 different with two algorithms is compared. For abb313GPIA sample have two different, for DSJ250.5 have 3 different with two algorithms is compared.

In five samples of Graph (DSJR500.1, le450_5c, le450_5d, qg.order30 and queen7_7) our proposed algorithm in finding the chromatic number, obtains the better results than the algorithm BEECOL. MCOACOL algorithm for solving the graph coloring problem is inspired by BEECOL method and as can been seen MCOACOL algorithm shows better results than BEECOL.

5.1. Algorithms MCOACOL reliability

To ensure the accuracy of performance of MCOACOL algorithm and demonstrate its abilities in solving graph coloring problems, this section compares the proposed algorithms in solving graph coloring problems.

As it can be seen from Table 3, MCOACOL algorithm in 12 samples of graph is compared to culture algorithms. in queen7–7 sample, our approach concludes 7 that, than cultural algorithm has achieved better outcomes and in 16 samples were compared with ANN in samples of 5-FullIns-4, 2-FullIns-5 and queen8_8 our proposed algorithm gives better results.

We will also have other comparison with SAGCP and MSAGCP [52] algorithms that are reported in Table 4.

Table 4 shows the comparison graph for 29 samples. As it can be seen, MCOACOL algorithm in comparison with or SAGCP algorithms in 21 instances gives better response in comparison with MSAGCP algorithm for 6 samples (1-insertion_5, 3-fullins_4, 4-inertion_4, miles1000, miles1500, mulsol. *i*.1). So our method has achieved a better answer. Our proposed algorithm in comparison with MSAGCP algorithm for queen11_11will results in chromatic number in 3 different.

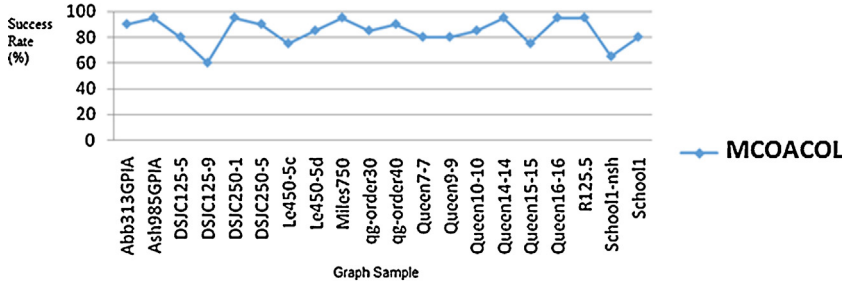


Fig. 24. Displays the success rate in terms of percentage.

Table 2
Comparison of MCOACOL algorithm with ABAC and BEECOL algorithms.

Graph name	|V|	Best known	MCOACOL	ABAC [50]	BEECOL [49]
1-FullIns_3	[1,100]	4	**4**	4	4
1-FullIns_4		5	**5**	5	5
1-Insertions_4		4	**5**	5	5
2-FullIns_3		5	**5**	5	5
2-Insertions_3		4	**4**	4	4
3-FullIns_3		6	**6**	6	6
3-Insertions_3		4	**4**	4	4
4-Insertions_3		3	**4**	4	4
David		11	**11**	11	11
Huck		11	**11**	11	11
Jean		10	**10**	10	10
mug100_1		4	**4**	4	4
mug100_25		4	**4**	4	4
mug88_1		4	**4**	4	4
mug88_25		4	**4**	4	4
myciel3		4	**4**	4	4
myciel4		5	**5**	5	5
myciel5		6	**6**	6	6
myciel6		7	**7**	7	7
queen5_5		5	**5**	5	5
queen6_6		7	**8**	7	8
queen7_7		7	**7**	7	8
queen8_12		12	**13**	12	12
queen8_8		9	**10**	9	10
queen9_9		10	**11**	10	11
queen10_10		11	**12**	11	12
2-Insertions_4	(100,200]	4	**5**	5	5
4-FullIns_3		7	**7**	7	7
5-FullIns_3		8	**8**	8	8
Anna		11	**11**	11	11
DSJC125.1		5	**6**	6	6
DSJC125.5		17	**19**	18	18
DSJC125.9		44	**45**	44	44
games120		9	**9**	9	9
miles1000		42	**42**	42	42
miles1500		73	**73**	73	73
miles250		8	**8**	8	8
miles500		20	**20**	20	20
miles750		31	**31**	31	31
mulsol.i.1		49	**49**	49	49
mulsol.i.2		31	**31**	31	31
mulsol.i.3		31	**31**	31	31
mulsol.i.4		31	**31**	31	31
mulsol.i.5		31	**31**	31	31
myciel7		8	**8**	8	8
queen11_11		11	**14**	13	13
queen12_12		13	**15**	14	14
queen13_13		13	**16**	15	15
queen14_14		16	**17**	16	16
1-FullIns_5	(200,300]	6	**6**	6	6
1-Insertions_5		6	**6**	6	6
2-FullIns_4		6	**6**	6	6
3-Insertions_4		5	**5**	5	5
DSJC250.1		8	**10**	9	9
DSJC250.5		28	**33**	30	30
queen15_15		16	**18**	17	18
queen16_16		18	**19**	18	19
zeroin.i.1		49	**49**	49	49
zeroin.i.2		30	**30**	30	30
zeroin.i.3		30	**30**	30	30
school1_nsh	(300,400]	14	**14**	14	14
school1		14	**14**	14	14
3-FullIns_4	(400,500]	7	**7**	7	7
4-Insertions_4		5	**5**	5	5
fpsol2.i.1		65	**65**	65	65
fpsol2.i.2		30	**30**	30	30
fpsol2.i.3		30	**30**	30	30
le450_5c		5	**6**	5	7
le450_5d		5	**7**	5	8
le450_25a		25	**25**	25	25
le450_25b		25	**25**	25	25
DSJR500.1		12	**12**	12	13
2-Insertions_5	(500,600]	6	**6**	6	6
Homer		13	**13**	13	13
1-Insertions_6	(600,700]	7	**7**	7	7
4-FullIns_4		8	**8**	8	8
ash331GPIA		4	**5**	4	5
inithx.i.2		31	**31**	31	31
inithx.i.3		31	**31**	31	31
will199GPIA	701	7	**8**	7	7
2-FullIns_5	(800,900]	7	**7**	7	7
inithx.i.1		54	**54**	54	54
qg.order30		30	**30**	30	31
wap05a	905	50	**50**	50	50
3-Insertions_5	(1000,2000]	6	**6**	6	6
5-FullIns_4		9	**9**	9	9
abb313GPIA		9	**12**	10	10
ash608GPIA		4	**5**	5	5
ash958GPIA		4	**5**	5	5
qg.order40		40	**41**	40	41
3-FullIns_5	2030	8	**8**	8	8
4-FullIns_5	4146	9	**9**	9	9

Queen *n_n* graphs are graphs and if there isn't any collision, their chromatic number will be n. Including graphs of MIZ categories, are evaluated that is good samples for testing because hard graphs are well known and our approach like popular algorithms in the field, results in best chromatic number with high accuracy.

As mentioned, MCOACOL algorithm for DSJ categories, is unable to find known chromatic number that may be seen in comparison with CLAVCA algorithm [53] and to find near-optimal results is adequate. But, for the last two examples CLAVCA algorithm easily finds the chromatic number. Table 5 shows this well.

5.2. MCOACOL algorithm stability

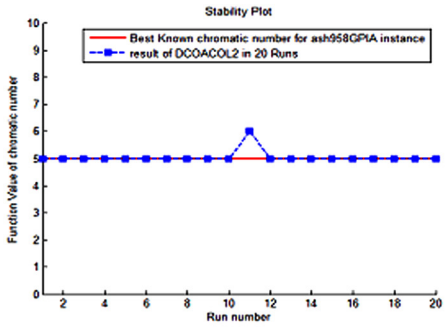
To demonstrate the stability of the algorithm, stability diagram for each chromatic number obtained in the 20 times is drawn and Fig. 25 shows stability of the algorithm on several graphs as below.

5.3. The convergence of the MCOACOL algorithm

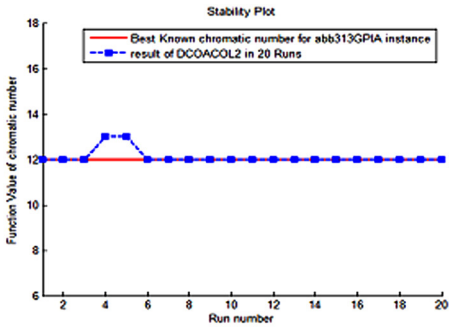
In order to demonstrate the accuracy and speed of convergence of the algorithm, charts of convergence in 20 times implementation of the algorithm on sample of graphs, than cost function are plotted. Fig. 26 shows faster convergence of algorithm with high precise to obtain chromatic number.

Table 3
Comparison of MCOACOL algorithm with culture algorithm, and ANNGCP algorithm.

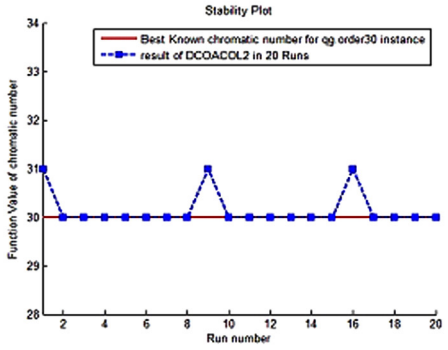
Graph name	Algorithm in [28]	ANNGCP [51]	MCOACOL
Myciel3	4	–	4
Myciel4	5	–	5
Queen5_5	5	–	5
Queen6_6	7	–	7
Myciel5	6	–	6
Miles250	8	–	8
Queen7_7	8	7	7
Huck	11	11	11
Jean	10	10	10
David	11	11	11
Games120	9	9	9
Anna	11	11	11
1-fullins-5	–	6	6
2-FullIns-5	–	11	7
3-FullIns-4	–	7	7
5-FullIns-4	–	11	9
2-Insertion-3	–	4	4
2-Insertion-4	–	5	5
3-Insertion-5	–	6	6
Homer	–	13	13
Miles1000	–	42	42
Queen8_8	–	11	10



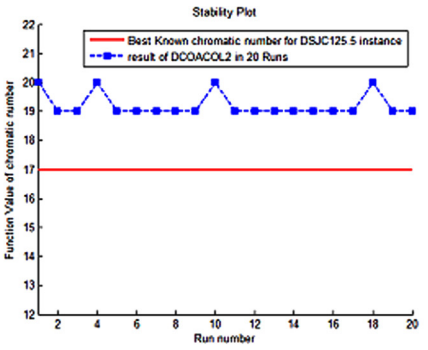
A) ash958GPIA Graph



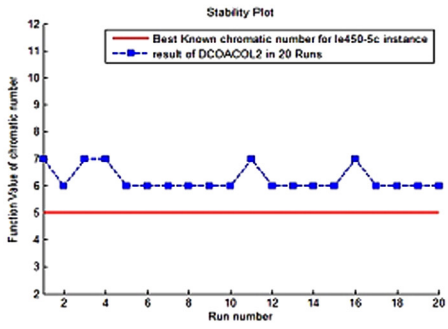
B) abb313GPIA Graph



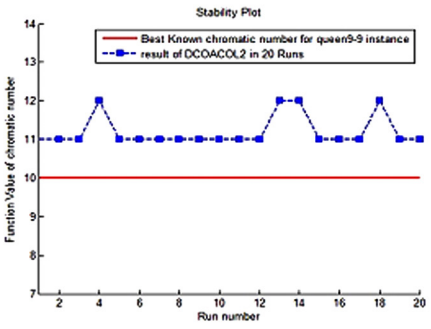
C) qg.order30 Graph



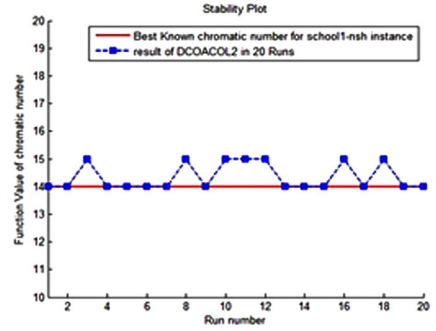
D) DSJC125.5 Graph



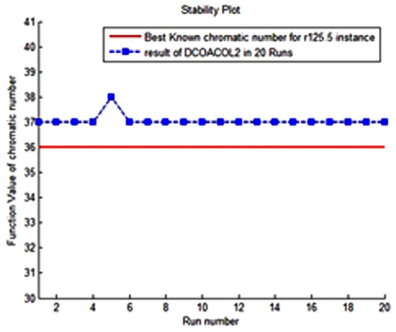
E) le450-5c Graph



F) queen 9-9 Graph



G) school1-nsh Graph



H) r125.5 Graph

Fig. 25. Stability diagram of MCOACOL algorithm for the sample of graph.

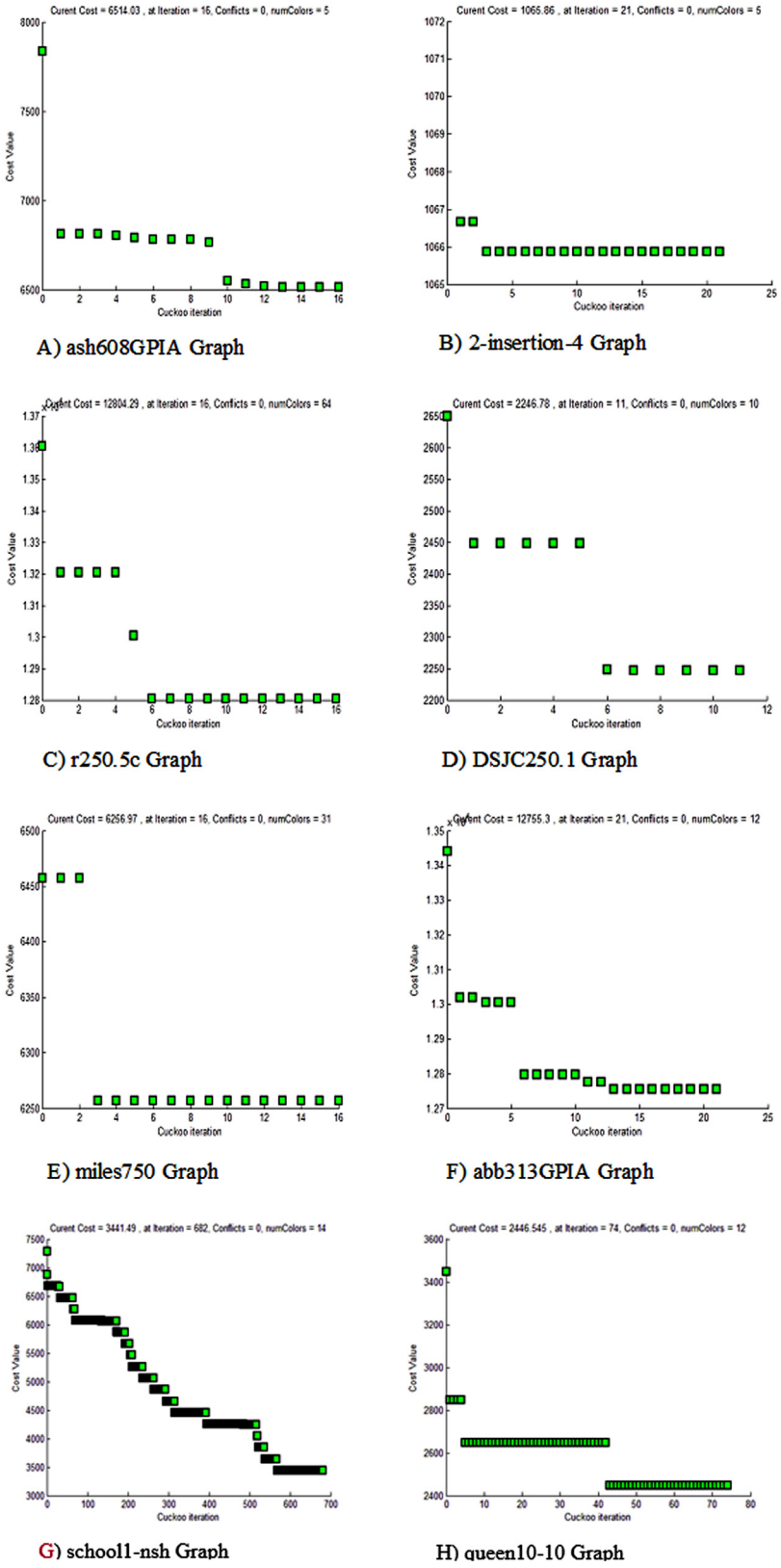


Fig. 26. Graph of MCOACOL algorithms convergence for graphs.

Table 4
Comparison of MCOACOL algorithm with SAGCP and MSAGCP algorithms.

Graph name	Best known	SAGCP [52]	MSAGCP [52]	MCOACOL
1-FullIns_5	6	8	6	6
1-Insertions_5	6	9	7	6
2-FullIns_4	6	6	6	6
2-Insertions_4	4	6	5	5
3-FullIns_4	7	12	9	7
3-Insertions_4	5	5	5	5
4-FullIns_3	7	7	7	7
4-Insertions_4	5	7	6	5
5-FullIns_3	8	8	8	8
Anna	11	13	11	11
David	11	11	11	11
DSJC125.1	5	10	6	6
games120	9	10	9	9
homer	13	17	13	13
Huck	11	11	11	11
Jean	10	11	10	10
miles1000	42	50	45	42
miles1500	73	80	75	73
miles750	31	35	31	31
mug100_25	4	4	4	4
mug88_25	4	4	4	4
mulsol.i.1	49	58	52	49
myciel6	7	8	7	7
queen5_5	5	7	5	5
queen7_7	7	9	7	7
queen9_9	10	14	11	11
queen10_10	11	17	12	12
queen11_11	11	15	11	14
zeroin.i.2	30	44	30	30

Table 5
Comparison of the MCOACOL algorithm with CLAVCA algorithm.

Graph name	Best known	CLAVCA [53]	MCOACOL
DSJC125.1	5	5	6
DSJC125.5	17	17	19
DSJC125.9	44	44	45
DSJC250.1	8	8	10
DSJC250.5	28	28	33
le450_25a	25	25	25
le450_25b	25	25	25

5.4. Functionality of the cuckoo in results

The main reason why COA works better than other evolutionary algorithms, is in multiple performance of COA like spawning and migration in other evolutionary algorithms, we face operators that jut have one goal but in cuckoo optimization algorithm, defined operators simultaneously realize several goals. For example, clustering in COA helps cuckoo to divides the space into several sectors and choose the best place approximately. This area is likely to contain the global optimums. Then all cuckoo migrate to this area and better search within that area. This leads to much faster convergence of the cuckoo algorithm.

Now, the question may arise why COA premature convergence does not occur?

The answer is hidden in the process spawning of cuckoos. Unlike other algorithms, in cuckoo optimization algorithm, cuckoo mother's eggs are in different locations. Spawning particular model used in this algorithm plays an essential role in the COA:

1. The eggs were distributed around the current optimal point helps COA not stuck in a local optimum.
2. The process of laying alone is a local search process.

Table 6
The whole table of obtained results from algorithm MCOACOL.

Graph name	Best known	MCOACOL	No. of iterations	Time (s)	Success rate
1-FullIns_3	4	4	4	0.4	20/20
1-FullIns_4	5	5	4	0.5	20/20
1-FullIns_5	6	6	6	1.9	20/20
1-Insertions_4	4	5	6	0.5	20/20
1-Insertions_5	6	6	6	1.2	20/20
1-Insertions_6	7	7	7	8.1	20/20
2-FullIns_3	5	5	5	0.4	20/20
2-FullIns_4	6	6	5	1.2	20/20
2-FullIns_5	7	7	5	10.7	20/20
2-Insertions_3	4	4	3	0.4	20/20
2-Insertions_4	4	5	5	1.1	20/20
2-Insertions_5	6	6	6	6.5	20/20
3-FullIns_3	6	6	2	0.4	20/20
3-FullIns_4	7	7	4	1.6	20/20
3-FullIns_5	8	8	4	30.5	20/20
3-Insertions_3	4	4	3	0.5	20/20
3-Insertions_4	5	5	7	2.1	20/20
3-Insertions_5	6	6	9	45	20/20
4-FullIns_3	7	7	6	0.7	20/20
4-FullIns_4	8	8	8	7.7	20/20
4-FullIns_5	9	9	5	147.5	20/20
4-Insertions_3	3	4	7	0.6	20/20
4-Insertions_4	5	5	5	3.7	20/20
5-FullIns_3	8	8	4	0.5	20/20
5-FullIns_4	9	9	7	28	20/20
abb313GPIA	9	12	20	224	18/20
anna	11	11	5	0.8	20/20
ash331GPIA	4	5	8	42.2	20/20
ash608GPIA	4	5	9	177.3	20/20
ash958GPIA	4	5	10	471.6	19/20
david	11	11	5	0.5	20/20
DSJC125.1	5	6	8	0.8	20/20
DSJC125.5	17	19	433	120	16/20
DSJC125.9	44	45	800	164.7	11/20
DSJC250.1	8	10	8	3	19/20
DSJC250.5	28	33	1100	600	18/20
DSJR500.1	12	12	9	6.9	20/20
fpsol2.i.1	65	65	5	3.6	20/20
fpsol2.i.2	30	30	7	7.4	20/20
fpsol2.i.3	30	30	8	6.4	20/20
games120	9	9	6	0.8	20/20
homer	13	13	7	4.3	20/20
huck	11	11	5	0.5	20/20
inithx.i.1	54	54	7	19.4	20/20
inithx.i.2	31	31	6	9	20/20
inithx.i.3	31	31	5	6	20/20
jean	10	10	5	0.5	20/20
le450_5c	5	6	253	279	15/20
le450_5d	5	7	100	80.8	17/20
le450_25a	25	25	3	4.4	20/20
le450_25b	25	25	7	7	20/20
miles1000	42	42	4	1	20/20
miles1500	73	73	6	1.2	20/20
miles250	8	8	4	1.1	20/20
miles500	20	20	6	1.2	20/20
miles750	31	31	5	1.5	19/20
mug100_1	4	4	5	0.8	20/20
mug100_25	4	4	3	0.5	20/20
mug88_1	4	4	4	1.1	20/20
mug88_25	4	4	6	1.3	20/20
mulsol.i.1	49	49	6	1.2	20/20
mulsol.i.2	31	31	5	1.1	20/20
mulsol.i.3	31	31	4	1.3	20/20
mulsol.i.4	31	31	5	1.7	20/20
mulsol.i.5	31	31	4	1.9	20/20
myciel3	4	4	2	0.2	20/20
myciel4	5	5	4	0.3	20/20
myciel5	6	6	6	0.3	20/20
myciel6	7	7	5	0.5	20/20
myciel7	8	8	10	3.8	20/20
qg.order30	30	30	71	326.4	17/20
qg.order40	40	41	84	746.3	18/20
queen5_5	5	5	3	0.3	20/20
queen6_6	7	8	8	5.7	20/20
queen7_7	7	7	401	30.5	17/20

Table 6 (*Continued*)

Graph name	Best known	MCOACOL	No. of iterations	Time (s)	Success rate
queen8_12	12	13	5	1.1	20/20
queen8_8	9	10	18	1.4	20/20
queen9_9	10	11	511	94	16/20
queen10_10	11	12	100	9.6	17/20
queen11_11	11	14	18	2.4	20/20
queen12_12	13	15	20	2.8	20/20
queen13_13	13	16	51	9.3	20/20
queen14_14	16	17	90	22.2	19/20
queen15_15	16	18	70	17.9	15/20
queen16_16	18	19	171	53.6	19/20
r125.1	5	5	5	1.2	20/20
r125.1c	46	46	7	1.8	20/20
r125.5	36	37	50	6.7	19/20
r250.1	8	8	7	2.4	20/20
r250.1c	64	64	10	5	20/20
school1_nsh	14	14	2874	1352.6	13/20
school1	14	14	1700	1934.2	18/20
wap05a	50	50	5	12.4	20/20
will199GPIA	7	8	7	21.8	20/20
zeroin.i.1	49	49	9	1.7	20/20
zeroin.i.2	30	30	7	2.1	20/20
zeroin.i.3	30	30	6	1.6	20/20

6. Conclusion and future solutions

Capability of cuckoo optimization algorithm in continuous space will provide an incentive in this paper presents the idea to discretization of cuckoo algorithms that conducted by implementation, the effective results of this discretization is obtained. Discretization on the most popular combinational problems (graph coloring problem) is tested and evaluated, graph coloring problem have done results on a range of benchmark function of this problem and analyzed and we compared the proposed algorithms in this context. Benchmark functions of graph coloring problem in various categories are classified from simple to difficult; in this study it was attempted to a wide range with all the features of the graph selected to test.

Since the graph coloring problem is a NP-complete problem as well, some graphs require more time to solve that we based on a maximum iterative algorithm and it can be improved by increasing the quality and better results is obtained. The results show that MCOACOL algorithms is the main focus of this paper in most cases success rate is nearly 100% but as we have shown that compared with other methods, we cannot conclude that our algorithm is best in all cases of graphs. It can be said that the proposed algorithm is able to compete with the other algorithms in this field. Our method provides a good balance between diversification and centralizing, because using a neighborhood search in radius of the lay egg cause the algorithm hardly trapped in local minimum and producing new eggs, the diversity of population is maintained. The results of this study could be useful for solving practical problems of graph problems. To continue offered three suggestions as follows:

First suggestion: Our algorithm when reach a fix state, it does not stop calculations because the termination condition based on the number of iterations of the algorithm originally given as input to the algorithm. For the graph coloring problem also occur when some external conditions, alter the relationship between nodes, finally, the algorithm is automatically activated and to restore the balance status, it works better.

Second suggestion: Offer more parallelism in migration operator in the discrete cuckoo algorithm that cuckoo group in same time to move toward best current group that it is effective in the algorithm.

Third suggestion: In this paper we have tested the algorithm for graph coloring problem. It is possible to continue with any proposed cuckoo discrete algorithms with a few changes, on the other discrete problems such as identification and tracking of social networks or similar problems, testing and analyzed the results.

Appendix A.

Table A.6.

Table A.6
The whole table of obtained results from algorithm MCOACOL.

Graph name	Best known	MCOACOL	No. of iterations	Time (s)	Success rate
1-FullIns_3	4	4	4	0.4	20/20
1-FullIns_4	5	5	4	0.5	20/20
1-FullIns_5	6	6	6	1.9	20/20
1-Insertions_4	4	5	6	0.5	20/20
1-Insertions_5	6	6	6	1.2	20/20
1-Insertions_6	7	7	7	8.1	20/20
2-FullIns_3	5	5	5	0.4	20/20
2-FullIns_4	6	6	5	1.2	20/20
2-FullIns_5	7	7	5	10.7	20/20
2-Insertions_3	4	4	3	0.4	20/20
2-Insertions_4	4	5	5	1.1	20/20
2-Insertions_5	6	6	6	6.5	20/20
3-FullIns_3	6	6	2	0.4	20/20
3-FullIns_4	7	7	4	1.6	20/20
3-FullIns_5	8	8	4	30.5	20/20
3-Insertions_3	4	4	3	0.5	20/20
3-Insertions_4	5	5	7	2.1	20/20
3-Insertions_5	6	6	9	45	20/20
4-FullIns_3	7	7	6	0.7	20/20
4-FullIns_4	8	8	8	7.7	20/20
4-FullIns_5	9	9	5	147.5	20/20
4-Insertions_3	3	4	7	0.6	20/20
4-Insertions_4	5	5	5	3.7	20/20
5-FullIns_3	8	8	4	0.5	20/20
5-FullIns_4	9	9	7	28	20/20
abb313GPIA	9	12	20	224	18/20
anna	11	11	5	0.8	20/20
ash331GPIA	4	5	8	42.2	20/20
ash608GPIA	4	5	9	177.3	20/20
ash958GPIA	4	5	10	471.6	19/20
david	11	11	5	0.5	20/20
DSJC125.1	5	6	8	0.8	20/20
DSJC125.5	17	19	433	120	16/20
DSJC125.9	44	45	800	164.7	11/20
DSJC250.1	8	10	8	3	19/20
DSJC250.5	28	33	1100	600	18/20
DSJR500.1	12	12	9	6.9	20/20
fpsol2.i.1	65	65	5	3.6	20/20
fpsol2.i.2	30	30	7	7.4	20/20
fpsol2.i.3	30	30	8	6.4	20/20
games120	9	9	6	0.8	20/20
homer	13	13	7	4.3	20/20
huck	11	11	5	0.5	20/20
inithx.i.1	54	54	7	19.4	20/20
inithx.i.2	31	31	6	9	20/20
inithx.i.3	31	31	5	6	20/20
jean	10	10	5	0.5	20/20
le450_5c	5	6	253	279	15/20
le450_5d	5	7	100	80.8	17/20
le450_25a	25	25	3	4.4	20/20
le450_25b	25	25	7	7	20/20
miles1000	42	42	4	1	20/20
miles1500	73	73	6	1.2	20/20
miles250	8	8	4	1.1	20/20
miles500	20	20	6	1.2	20/20
miles750	31	31	5	1.5	19/20
mug100_1	4	4	5	0.8	20/20
mug100_25	4	4	3	0.5	20/20
mug88_1	4	4	4	1.1	20/20
mug88_25	4	4	6	1.3	20/20
mulsol.i.1	49	49	6	1.2	20/20
mulsol.i.2	31	31	5	1.1	20/20
mulsol.i.3	31	31	4	1.3	20/20
mulsol.i.4	31	31	5	1.7	20/20
mulsol.i.5	31	31	4	1.9	20/20
myciel3	4	4	2	0.2	20/20

Table A.6 (*Continued*)

Graph name	Best known	MCOACOL	No. of iterations	Time (s)	Success rate
myciel4	5	5	4	0.3	20/20
myciel5	6	6	6	0.3	20/20
myciel6	7	7	5	0.5	20/20
myciel7	8	8	10	3.8	20/20
qg.order30	30	30	71	326.4	17/20
qg.order40	40	41	84	746.3	18/20
queen5_5	5	5	3	0.3	20/20
queen6_6	7	8	8	5.7	20/20
queen7_7	7	7	401	30.5	17/20
queen8_12	12	13	5	1.1	20/20
queen8_8	9	10	18	1.4	20/20
queen9_9	10	11	511	94	16/20
queen10_10	11	12	100	9.6	17/20
queen11_11	11	14	18	2.4	20/20
queen12_12	13	15	20	2.8	20/20
queen13_13	13	16	51	9.3	20/20
queen14_14	16	17	90	22.2	19/20
queen15_15	16	18	70	17.9	15/20
queen16_16	18	19	171	53.6	19/20
r125.1	5	5	5	1.2	20/20
r125.1c	46	46	7	1.8	20/20
r125.5	36	37	50	6.7	19/20
r250.1	8	8	7	2.4	20/20
r250.1c	64	64	10	5	20/20
school1_nsh	14	14	2874	1352.6	13/20
school1	14	14	1700	1934.2	18/20
wap05a	50	50	5	12.4	20/20
will199GPIA	7	8	7	21.8	20/20
zeroin.i.1	49	49	9	1.7	20/20
zeroin.i.2	30	30	7	2.1	20/20
zeroin.i.3	30	30	6	1.6	20/20

References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, U Michigan Press, 1975.
[2] R. Ebenhart, J. Kennedy, Particle swarm optimization, Perth, Australia, Piscataway, Proceeding IEEE Inter Conference on Neural Networks, 1995, pp. 1942–1948.
[3] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[4] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, IEEE, 2007, pp. 4661–4667.
[5] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[6] F. Ramezani, S. Lotfi, Social-based algorithm (SBA), Appl. Soft Comput. 13 (2013) 2837–2856.
[7] M.R. Garey, D.S. Johnson, Computers and intractability, in: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979, ©1979 ISBN:0716710447.
[8] D. de Werra, An introduction to timetabling, Eur. J. Oper. Res. 19 (1985) 151–162.
[9] K. Dowsland, J. Thompson, Ant colony optimization for the examination scheduling problem, J. Oper. Res. Soc. 56 (2005) 426–438.
[10] A. Gamst, Some lower bounds for a class of frequency assignment problems, IEEE Trans. Veh. Technol. 35 (1986) 8–14.
[11] G.J. Chaitin, M.A. Auslander, A.K. Chandra, J. Cocke, M.E. Hopkins, P.W. Markstein, Register allocation via coloring, Comput. Lang. 6 (1981) 47–57.
[12] F.C. Chow, J.L. Hennessy, The priority-based coloring approach to register allocation, ACM Trans. Program. Lang. Syst. (TOPLAS) 12 (1990) 501–536.
[13] M.R. Garey, D. Johnson, H. So, An application of graph coloring to printed circuit testing, IEEE Circuits Syst. 23 (1976) 591–599.
[14] S. Sen Sarma, R. Mandal, A. Seth, Some sequential graph colouring algorithms for restricted channel routeing, Int. J. Electron. 77 (1994) 81–93.
[15] R. Rajabioun, Cuckoo optimization algorithm, Appl. Soft Comput. 11 (2011) 5508–5518.
[16] D. Brélaz, New methods to color the vertices of a graph, Commun. ACM 22 (1979) 251–256.
[17] F.T. Leighton, A graph coloring algorithm for large scheduling problems, J. Res. Nat. Bur. Stand. 84 (1979) 489–506.
[18] R. Dorne, J.-K. Hao, Tabu search for graph coloring, T-colorings and set T-colorings, in: Meta-heuristics, Springer, 1999, pp. 77–92.
[19] A. Hertz, D. de Werra, Using tabu search techniques for graph coloring, Computing 39 (1987) 345–351.
[20] D. Costa, A. Hertz, Ants can colour graphs, J. Oper. Res. Soc. 48 (1997) 295–305.
[21] I. Blöchliger, N. Zufferey, A graph coloring heuristic using partial solutions and a reactive tabu scheme, Comput. Oper. Res. 35 (2008) 960–975.
[22] B. Ray, A.J. Pal, D. Bhattacharyya, T. Kim, An efficient ga with multipoint guided mutation for graph coloring problems, Int. J. Signal Process. Image Process. Pattern Recogn. 3 (2010) 51–58.
[23] Z. Lü, J.-K. Hao, A memetic algorithm for graph coloring, Eur. J. Oper. Res. 203 (2010) 241–250.
[24] F. Ge, Z. Wei, Y. Tian, Z. Huang, Chaotic ant swarm for graph coloring, in: Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on, IEEE, 2010, pp. 512–516.
[25] M. Bessedik, B. Toufik, H. Drias, How can bees colour graphs, Int. J. Bio-Inspired Comput. 3 (2011) 67–76.
[26] J. Qin, Y.-x. Yin, X.-j. Ban, Hybrid discrete particle swarm algorithm for graph coloring problem, J. Comput. 6 (2011) 1175–1182.
[27] I. Fister, J. Brest, Using differential evolution for the graph coloring, in: Differential Evolution (SDE), 2011 IEEE Symposium on, IEEE, 2011, pp. 1–7.
[28] R. Abbasian, M. Mouhoub, A. Jula, Solving graph coloring problems using cultural algorithms, in: FLAIRS Conference, 2011.
[29] M. Faraji, H.H.S. Javadi, Proposing a new algorithm based on bees behavior for solving graph coloring, Int. J. Contemp. Math. Sci. 6 (2011) 41–49.
[30] L.-Y. Hsu, S.-J. Horng, P. Fan, M.K. Khan, Y.-R. Wang, R.-S. Run, J.-L. Lai, R.-J. Chen, MTPSO algorithm for solving planar graph coloring problem, Expert Syst. Appl. 38 (2011) 5525–5531.
[31] T. Xiong, Y. Bao, Z. Hu, Beyond one-step-ahead forecasting: evaluation of alternative multi-step-ahead forecasting models for crude oil prices, Energy Econ. 40 (2013) 405–415.
[32] Y. Bao, Z. Hu, T. Xiong, A PSO and pattern search based memetic algorithm for SVMs parameters optimization, Neurocomputing 117 (2013) 98–106.
[33] T. Xiong, Y. Bao, Z. Hu, Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting, Knowledge Based Syst. 55 (2014) 87–100.
[34] T. Xiong, Y. Bao, Z. Hu, Does restraining end effect matter in EMD-based modeling framework for time series prediction? Some experimental evidences, Neurocomputing 123 (2014) 174–184.
[35] M.K. Patel, M.R. Kabat, C.R. Tripathy, A hybrid ACO/PSO based algorithm for QoS multicast routing problem, Ain Shams Eng. J. 5 (2014) 113–120.
[36] J. Xu, Y. Yin, T. Cheng, C.-C. Wu, S. Gu, A memetic algorithm for the re-entrant permutation flowshop scheduling problem to minimize the makespan, Appl. Soft Comput. 24 (2014) 277–283.
[37] Z. Zhou, C.-M. Li, C. Huang, R. Xu, An exact algorithm with learning for the graph coloring problem, Comput. Oper. Res. 51 (November) (2014) 282–301.
[38] A. Rahmani, S. MirHassani, A hybrid firefly-genetic algorithm for the capacitated facility location problem, Inf. Sci. 283 (2014) 70–78.
[39] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, IEEE, 2009, pp. 210–214.
[40] H. Shokri-Ghaleh, A. Alfi, A comparison between optimization algorithms applied to synchronization of bilateral teleoperation systems against time delay and modeling uncertainties, Appl. Soft Comput. 24 (2014) 447–456.
[41] M. Khajeh, A.R. Golzary, Synthesis of zinc oxide nanoparticles–chitosan for extraction of methyl orange from water samples: cuckoo optimization algorithm-artificial neural network, Spectrochim. Acta, A:, Mol. Biomol. Spectrosc. 131 (2014) 189–194.
[42] R. Stryczek, A metaheuristic for fast machining error compensation, J. Intell. Manuf. (2014) 1–12, http://dx.doi.org/10.1007/s10845-014-0945-0
[43] M.K. Ahirwal, A. Kumar, G.K. Singh, EEG/ERP adaptive noise canceller design with controlled search space (CSS) approach in cuckoo and other optimization algorithms, IEEE/ACM Trans. Comput. Biol. Bioinf. (TCBB) 10 (2013) 1491–1504.
[44] B. Abolpour, A. Mohebbi, Estimation of the compressive strength of 28-day-old concrete by use of an adaptive cuckoo—fuzzy logic model, Res. Chem. Intermed. 39 (2013) 4001–4009.
[45] R. Teimouri, H. Sohrabpoor, Application of adaptive neuro-fuzzy inference system and cuckoo optimization algorithm for analyzing electro chemical machining process, Front. Mech. Eng. 8 (2013) 429–442.
[46] M.A. Mellal, E.J. Williams, Parameter optimization of advanced machining processes using cuckoo optimization algorithm and hoopoe heuristic, J. Intell. Manuf. (2014) 1–16, doi:10.1007/s10845-014-0925-4, Print ISSN 0956-5515, Online ISSN 1572-8145, Publisher Springer US.
[47] A. Moraglio, C. Di Chio, J. Togelius, R. Poli, Geometric particle swarm optimization, J. Artif. Evol. Appl. 2008 (2008) 11.
[48] P. Galinier, A. Hertz, A survey of local search methods for graph coloring, Comput. Oper. Res. 33 (2006) 2547–2562.
[49] M. Faraji, Proposing a New Algorithm Based on Bees Behavior for Solving Graph Coloring, Azad Islamic University, Science and Research Brach, Iran, Tehran, 2011 (Thesis in Persian).
[50] T.N. Bui, T.H. Nguyen, C.M. Patel, K.-A.T. Phan, An ant-based algorithm for coloring graphs, Discrete Appl. Math. 156 (2008) 190–200.
[51] T. Maitra, A.J. Pal, T.-H. Kim, D. Bhattacharyya, Hybridization of genetic algorithm with bitstream neurons for graph coloring, Int. J. u-and e-Serv. 3 (2010).
[52] A.J. Pal, B. Ray, N. Zakaria, S.S. Sarma, Comparative performance of modified simulated annealing with simple simulated annealing for graph coloring problem, Procedia Comput. Sci. 9 (2012) 321–327.
[53] J. Akbari Torkestani, M.R. Meybodi, A cellular learning automata-based algorithm for solving the vertex coloring problem, Expert Syst. Appl. 38 (2011) 9237–9247.