Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Graph Coloring using Evolutionary Computation: A case study of Blind Naked Mole Rat Algorithm

Fahad Maqbool*[1] | Muhammad Fahad[1] | Muhammad Ilyas[1] | Hajira Jabeen[2]

[1]Computer Science & IT, University of Sargodha, Pakistan
[2]Big Data Analytics, GESIS-Leibniz Institute for the Social Sciences, Cologne, Germany

Correspondence
*Fahad Maqbool. Email: fahad.maqbool@uos.edu.pk

Present Address
University of Sargodha

Summary

Graph Coloring Problem (GCP) is an NP-complete optimization problem. It's famous for its applications in scheduling, register allocation, and map coloring. In recent years, biological inspired and especially Swarm Intelligence (SI) techniques have gained popularity for solving complex optimization problems. In this paper, we have proposed Blind Naked Mole Rat based Coloring (BNMR-Col) for graphs. BNMR-Col uses both exploitation and exploration to find the best solution in search space. Exploitation uses both local moves and global moves to find a better solution in the surroundings of an existing solution. On the other hand, exploration generates new solution by combining various solutions from the search space. BNMR-Col shows better convergence rate and approaches the lowest color value in 83% of the cases when tested on standard benchmark graph instances.

KEYWORDS:
meta heuristic, intelligent system, optimization, swarm intelligence

1 | INTRODUCTION

Evolutionary algorithms have gained popularity in the last few years due to their capability to solve complex optimization problems efficiently and effectively. Researchers have proposed numerous applications of these algorithms in resource optimization [1], economic dispatch [2], large scale global optimization [3], parameter estimation [4], and engineering problems [5]. These population-based algorithms are inspired by the mechanism of biological evolution and maintain a set of candidate solutions. Crossover, mutation, and reproduction are central to many of these algorithms. Genetic Algorithm (GA) is a widely used population-based evolutionary algorithm that also has its applications in graph problems. The problems include graph partitioning [6,7,8], steiner graph problem [9,10], and Graph Coloring Problem (GCP) [11].

Let $G = (V, E)$ be a simple graph, where V and E represent the set of vertices and the set of edges respectively. GCP is an assignment of minimum colors (labels) to the vertices of a graph such that the adjacent vertices do not have the same color. Solving GCP with the minimum possible colors makes it a hard problem. GCP has been mapped to numerous application areas including resource scheduling [12], Parking Algorithm [13], wireless network channel allocation problem [14], video synopsis problem [15], and multiple stacks traveling salesman person problem [16]. A specialized case of GCP is k-GCP. Let k be the number of colors assigned to vertices of a graph then the conflict optimization version of k-GCP minimizes the number of conflicts in the graph. The optimal solution of k-GCP is a coloring configuration with zero conflicts and is termed as proper k-coloring.

Blind Naked Mole-Rats (BNMR) algorithm [17] is a meta-heuristic that is inspired by the social behavior and population rich colony culture of blind naked mole rats. BNMR uses both exploration and exploitation to search the optimal solution in the search space. Exploration helps to explore the far away areas in search space while exploitation helps to converge to a solution by exploring the neighborhood of a fittest solution. Attenuation coefficient controls the rate of exploitation and decreases with number of iterations. It has been observed empirically that BNMR prevents from trapping in local optima and outperforms several competing algorithms (such as particle swarm optimization, artifical bee colony , genetic algorithm,

[0]**Abbreviations:** GCP, graph coloring problem; BNMR-Col, blind naked mole rat based coloring; SDO, saturation degree ordering

stem cells algorithm, and simulated annealing) when applied to numerical function optimization [17], data clustering [18], traveling salesman person problem [19], and electromagnetic designing [20].

In this article, we have proposed BNMR-Col algorithm to solve k-GCP optimization problem. BNMR-Col addresses numerous research questions: (i) How BNMR algorithm can be modified to solve GCP? (ii) Will BNMR-Col always converge for benchmark graph instances? (iii) Can BNMR-Col outperform Chromaticats and ANTCOL algorithms? We have tested BNMR-Col on DIMACS graphs instances [11,21] and compared with the relevant techniques (i.e ANTCOL [22] and chromaticats [23]). This paper is organized as follows. In section 2 we have discussed the relevant techniques that have been applied to GCP. Section 3 presents the BNMR algorithm and major components of BNMR-Col algorithm. In section 4, we have explained BNMR-Col algorithm. Section 5 presents the results of bnmr-col, ANTCOL, Chromaticats on popular graph instances. Finally, we conclude our work and discuss the future ideas in section 6.

2 | LITERATURE REVIEW

Swarm Intelligence (SI) [24] and evolutionary algorithms [25,26] are the famous bio-inspired algorithms that have gained popularity in the last few years and have been successfully applied to many real world problems (including but not limited to link prediction [27], clustering [28], and model tuning & hyper-parameter optimization [29]). SI technique comprises a population of candidate solutions and have shown competitive results for GCP. Gravitational Swarm Intelligence (GSI) was applied to solve GCP [30] and was found that the chromatic number found by GSI was either equivalent or better than Degree of Saturation [31], Tabu Search [32], Simulated Annealing [33], and Ant Colony Optimization (ACO) [34]. A famous SI algorithm discrete cuckoo optimization was also applied to GCP that achieved a success rate but was unable to produce optimal results for all benchmark graph instances [35]. In another effort a proper coloring of a graph was generated using the collective experience of an artificial ant colony ANTCOL [36]. Each ant in the colony iteratively builds a coloring with the probability of selecting a vertex in each step based on the accumulated coloring experience of the ant colony. Empirical results revealed that the number of colors used by ANTCOL is closer to the probabilistically estimated chromatic number for a given graph [37].

A hybrid algorithm comprising discrete version of PSO and local search was also proposed to solve GCP [38]. The hybrid technique used a distance measure defined over a discrete solution space to evaluate the difference between two particle positions (i.e. coloring). Once the position of each particle is updated, the Tabucol algorithm [39] is used to reduce the number of conflicts. However, this hybrid technique worked on limited instances of discrete solution space. In another approach, ABC algorithm used an order based technique for solving GCP. Food sources were represented as a set of n-component vectors where each component corresponds to a node in the graph. The algorithm decodes each food source as a potential solution to GCP by ordering each component in ascending order and then coloring the graph using the first-fit strategy. The ABC algorithm exhibits a slightly better performance than Largest Degree Ordering (LDO) and Saturation Degree Ordering (SDO). LDO gives preference to the nodes that have the highest number of edges incident to them while SDO prioritizes the nodes based on the number of colors in the neighborhood of a node. However, the authors selected a few graphs instances that were of limited edge density [40]. Karim Baiche et. al. enhanced the dragonfly algorithm to ensure the diversity of solutions that potentially avoids local optima [41]. The approach was tested on a limited number of DIMACS graph instances.

A Cat Swarm Optimization (CSO) based technique chromaticats was proposed that comprises two phases (i.e. seeking and tracing mode). The seeking mode is designed to identify proper coloring and further optimizing the solution by minimizing the number of used colors. The tracing mode explores new possible colorings by using random mutation. Chromaticats converged in fewer iterations when compared to Cultural Algorithm (CA) and Evolutionary Programming (EP). However, it takes more computational time to complete each iteration [23]. Kui Chen and Hitoshi Kanoh used adaptive artificial bee colony algorithm to find better graph coloring compared to different variants of artificial bee colony algorithms [42]. Authors tested the effectiveness of the proposed approach on 30 graph instances.

Although these already attempted SI algorithms effectively find good solutions for limited graph instances (e.g. Hybrid algorithm [38] performs for limited graph instances of discrete solution space, ABC algorithm [40] performs better better for graph instances with limited edge density, and enhanced binary dragonfly algorithm [41] was also tested for a limited number of DIMACS graph instances) but they suffer with slow convergence rate. Moreover few SI algorithms like chromaticats [23] has high computational cost per iteration.

Taherdangkoo, M. and Taherdangkoo, R. proposed a modified BNMR algorithm (i.e. M-BNMR) to deal Loney's Solenoid Optimization (LSO) problem and incorporated the idea of searching areas which have not been examined due to the stochastic nature of the search process. The algorithm exhibits a better convergence rate and effectively avoids the local optima [43]. M-BNMR revealed better results compared to other SI algorithms (including PSO, ABC, Gaussian ABC, and stem cells algorithm). Faster convergence rate and better computational time of BNMR and M-BNMR inspired us to explore the potential of BNMR algorithm for graph coloring problem. In this paper we have proposed BNMR-Col to solve graph coloring problem that have competitive results for various graph instances.

3 | PRELIMINARIES

3.1 | Blind Naked Mole Rat (BNMR)

BNMR is a meta-heuristic based on social behavior of mole-rats. They searched the food and protects the colony against the attacks. BNMR worked at random on the complete problem space. Number of mole rats(i.e. population) are twice the number of food sources. Whereas each food source represents a response for problem space i.e. target to be found by mole-rats. BNMR has focused to overcome the issues faced by the conventional optimization algorithms like low convergence rate and getting stuck in local optima. Algorithm 1 presents the optimization process of BNMR algorithm[38].

Algorithm 1 Blind Naked Mole Rat (BNMR) Algorithm

Require:

Define Objective function $f(X_i)$

Define constants $\alpha,\ \beta,\ \zeta$, and maximum number of iterations T

Initialize population $X = \{X_i\}$, removal rate b_i where $i = 1 \cdots N$ and $X_i = [x_1^i \cdots x_d^i]$ ▷ d represents the dimensions of the search space

Ensure: X_{best} ▷ Optimal or Near Optimal Solution

Sort X_i such that $f(X_i) \geq f(X_{i+1})$

$t \leftarrow 1$ ▷ Reset iteration counter

while $t \leq T$ **do** ▷ While stopping criteria not met i.e. maximum iterations

  $X_{new} \leftarrow [x_j^{new}] \leftarrow x_j^{Min} + \beta(x_j^{Max} - x_j^{Min})$

  if $rand \leq A_i$ **then** ▷ Exploitation phase

    Select a solution X_{best} among the best solutions using the probability $p_i = \frac{f(X_i)=FS_i * R_i}{\sum_{k=1}^{N} f(X_k)}$

    Find X_{local} around X_{best}

  end if

  Generate a new solution X_{rand} using random search ▷ Exploration phase

  if $rand \leq B_i \quad \& \quad f(X_i) < f(X_{opt})$ **then**

    Accept and save the new solutions in the memory of mole-rat

  end if

end while

Let M_i be a mole rat with a food source X_i and removal rate b_i, where $i = 1 \cdots m$. M_i performs both exploration and exploitation in each iteration of BNMR. In exploitation M_i select another mole-rat M_j with food source X_j (based on relative fitness probability). M_i generates a neighboring food source X_j' for M_j. In case of exploration, BNMR considers M_i a potential invader. Therefore, it generates a new random food source X_i'. If $f(X_i')$ is better than $f(X_i)$ then M_i is mark as invader and replaces its food source X_i with X_i'.

Table 1 Average number of conflicts using 10 runs of Random-k and k-ColorWalk for Leighton graphs with known chromatic number ($\triangle$)

Instance	$\triangle$	Number of conflicts	
		Random-k	k-ColorWalk
le450_5a	5	1144	815
le450_5b	5	1127	838
le450_5c	5	1988	1619
le450_5d	5	1927	1583
le450_15a	15	548	156
le450_15b	15	543	158
le450_15c	15	1124	698
le450_15d	15	1106	685
le450_25a	25	334	31
le450_25b	25	328	27
le450_25c	25	703	231
le450_25d	25	693	236

3.2 | Population (P)

Population $P = M_1 \cdots M_N$ consists of N mole-rats with no division of labor mole-rats as suggested by M. Taherdangkoo et al.[18]. Each mole-rat $M_j = X_j, T_j, s_j, b_j$ consist of an assigned food source X_j, a restricted list of vertices T_j, a stagnation counter s_j, and removal rate b_j. Each food source X_j represents a potential solution to the problem from the solution space S. Each food source X_j is initialized by a randomized algorithm, k-ColorWalk as described in Algorithm 2. The algorithm uses the structure of the graph to initially build k color classes without any conflict. After that remaining uncolored vertices, if any, are assigned random color c_r such that $c_r \in \{c_1 \cdots c_k\}$

Algorithm 2 k-ColorWalk for Population Initialization

Require:

$Graph\ G \leftarrow (V, E)$ ▷ Graph with vertex set V and edge set E
$Initialize\ k$ ▷ Maximum number of colors
$c_i\ where\ i\ \leftarrow 0 \cdots k$ ▷ Number of colors currently assigned
$C_j \leftarrow 0\ where\ j = \{1 \cdots |V|\}$ ▷ Color of vertex u
$U \leftarrow V$ ▷ Set of vertices to visit
$S \leftarrow \phi$ ▷ Set of visited vertices
$T \leftarrow \phi$ ▷ Set of neighbor vertices that still needs to be explore
$N(u) \leftarrow \{v\}\ such\ that\ \forall v\ there\ exists\ (u, v)$ ▷ Neighborhood of a vertex u

Ensure:

while $i \leq k$ **do**
  $i \leftarrow i + 1$
  while $U \neq \phi$ **do**
    $Randomly\ select\ u\ \in\ U$
    if $c_i \notin C_{N(u)}$ **then**
      $C_u = c_i$
    end if
    $T \leftarrow N(u)$
    $S \leftarrow S \cup u$
    $U \leftarrow U - u$
    while $T \neq \phi$ **do**
      $Randomly\ select\ t\ \in T$
      if $c_i \notin C_{N(t)}$ **then**
        $C_t = c_i$
      end if
      $S \leftarrow S \cup t$
      $T \leftarrow T \cup N(t)$
      $T \leftarrow T - S$
    end while
  end while
end while

Another approach Random-k is used to create the initial solutions by assigning assignment colors to the vertices. Random-k results in higher number of conflicts as compared to k-ColorWalk. Table 1 displays the average number of conflicts using 10 runs of Random-k and k-ColorWalk for Leighton graphs. Generating k-colorings for the given graph using k-ColorWalk helps to keep large-sized color classes and generates the coloring with fewer conflicts compared to Random-k. This strategy helps to improve the convergence speed. However, number of conflicts in initial coloring does not effect the final solution[44].

3.3 | Attenuation Coefficient

Attenuation coefficient A is a variable that controls the probability of a mole-rat to select one of the best solutions and improve it. Attenuation rate starts from a higher initial value and gradually settles down to a lower probability value. It is defined as $A = \alpha * e^{\frac{-\alpha i}{z}}$, where α is a user-specified value that controls the rate of decrement, i is the current iteration and z is half of the maximum iteration value set by the user.

3.4 | Conflict Resolution

Conflict resolution in GCP checks the adjacent vertices having the same color. Various conflict resolution strategies have been proposed. We have used SDO [31] for conflict resolution due to it's simplicity and effectiveness. In SDO, saturation degree of a vertex is defined as the number of it's adjacent differently colored vertices. Let C be the set of all conflicting vertices in the graph then set of available conflicting vertices ($C_A = C - l$) is obtained by removing all tabu vertices(l) from C. Tabu vertices are the vertices that have been visited recently and belongs to set C. To generate a neighboring solution, a vertex $v_i \in C_A$ is randomly selected. Using probability β, v_i is assigned either lowest color ($c = 1$) or a random color cr such that $2 \leq cr \leq k$. Then we uncolor the vertices adjacent to v_i. After this, we iteratively select a vertex $vi \in C_A$ with largest saturation degree and assign it either a lowest legal color or a random color in the range $[2 \cdots k]$. After coloring all the vertices vi, these are added in the restricted list. The length of restricted list l is adjusted dynamically using $l = \sum_{c=1}^{k} h(v_i^c) + r$. Here r is a random number in the range [1-10] and $h(v_i^c)$ is the number of conflicts in the food source X_i given by equation 2.

Figure 1 shows the step by step working of One k-SDO move using a k-coloring graph with two conflicting vertices v_2 and v_5 as shown in figure 1a. Figure 1b shows the first step of One k-SDO move which is to select a conflicting vertex and assign it the lowest color. In the next step we uncolor all the vertices in the neighborhood of selected conflicting vertex as shown in figure 1c. In the next step (figure 1d), we iteratively color all the vertices in the neighborhood of the selected conflicting vertex v2 using Saturation Degree Ordering (SDO) with k as a fixed bound not to exceed while assigning a color value. Then we select uncolored vertex with maximum saturation degree (i.e. v_5) and assign it lowest legal color (c_1) as shown in figure 1e. In a similar fashion all the remaining uncolored vertices are assigned colors using saturation degree as the order of priority (figure 1f - 1i). If no legal color is available for assignment to a given vertex, any random color is assigned within 1 to k value. In the final step we assign lowest legal color to all uncolored vertices as shown in figure 1j.

3.5 | Exploitation

Exploitation is the process of improving assigned food source towards a better solution by a given mole-rat. It is further divided into two phases, (i.e. Self-Exploitation (SE) and Probabilistic-Exploitation (PE)). In SE, a mole-rat M_i applies the local search procedure on its assigned food source X_i to generate a neighboring solution. While in PE, a mole-rat M_i generates neighboring solutions for other mole-rats M_j using tabu search [45]. Both of these phases are briefly discussed in following sub sections.

3.5.1 | Self-Exploitation

In self-exploitation phase, a mole-rat M_j generates a neighborhood of its assigned food source X_j using SDO technique. A neighboring solution of X_j is generated by selecting a conflicting vertex $v_i \in C$ having maximum conflict degree $d_c(v_i)$. Conflict degree of a vertex v_i is the number of vertices adjacent to v_i, which have the same color label as of v_i. After generating Y Neighboring Solutions $\{X_{j^1}, X_{j^2} \cdots X_{j^Y}\}$ for X_j, a neighboring solution $X_{j^{best}}$ with the least number of conflicts (i.e. $\sum_{c=1}^{k} h(v_{j^{best}}^c) > \sum_{c=1}^{k} g(v_{j^y}^c)$ where $best \neq n$ and $y = 1 \cdots Y$) compared to X_j is selected to replace X_j.

Figure 2 shows the working of self-exploitation using an example. Let X_j be the initial coloring of the mole-rat M_j as shown in figure 2a. A set of conflicting vertices $C = 1, 2, 4, 6, 7, 8, 10, 11, 12, 13, 14, 15, 18, 19$ represents all the distinct vertices that have conflicting colors in adjacent vertices. A set of restricted list T consist of all those vertices that are forbidden to be selected and is empty at the start. Set of available conflicting vertices C_A is obtained by removing all vertices in T from C (i.e. $C_A = C - T$). A neighbor move is generated by selecting a vertex from C_A and performing one k-SDO move to that vertex. Let us consider that the first neighbor move is generated by selecting the vertex with the largest conflict degree. All the other moves are generated by selecting a vertex randomly from CA. Two moves N1 using vertex v11 and N2 using vertex v4 have been generated in Figure 2a. Here v11 is the vertex with largest conflict degree and v4 is a randomly selected vertex. N1 reduces the conflicts to 7 and N2 results in 8 conflicts. Subsequently, the best move among the neighbor moves is selected. It is clear that N1 has better fitness than N2. Also, the fitness of N1 is better than initial coloring (i.e. 13 conflicts) of mole-rat M_j. Thus, current coloring X_j of mole-rat is replaced with N1. Conflicting vertex v11 selected to generate N1 is considered as tabu and added in T. This completes one single Local Move step for a mole-rat in an iteration. In the second iteration, the same procedure is repeated and two neighbor moves are generated using modified coloring of mole-rat. Figure 2b shows the modified coloring X_j and two new neighbor moves N1 and N2. Note that v11 is not available as a conflicting vertex

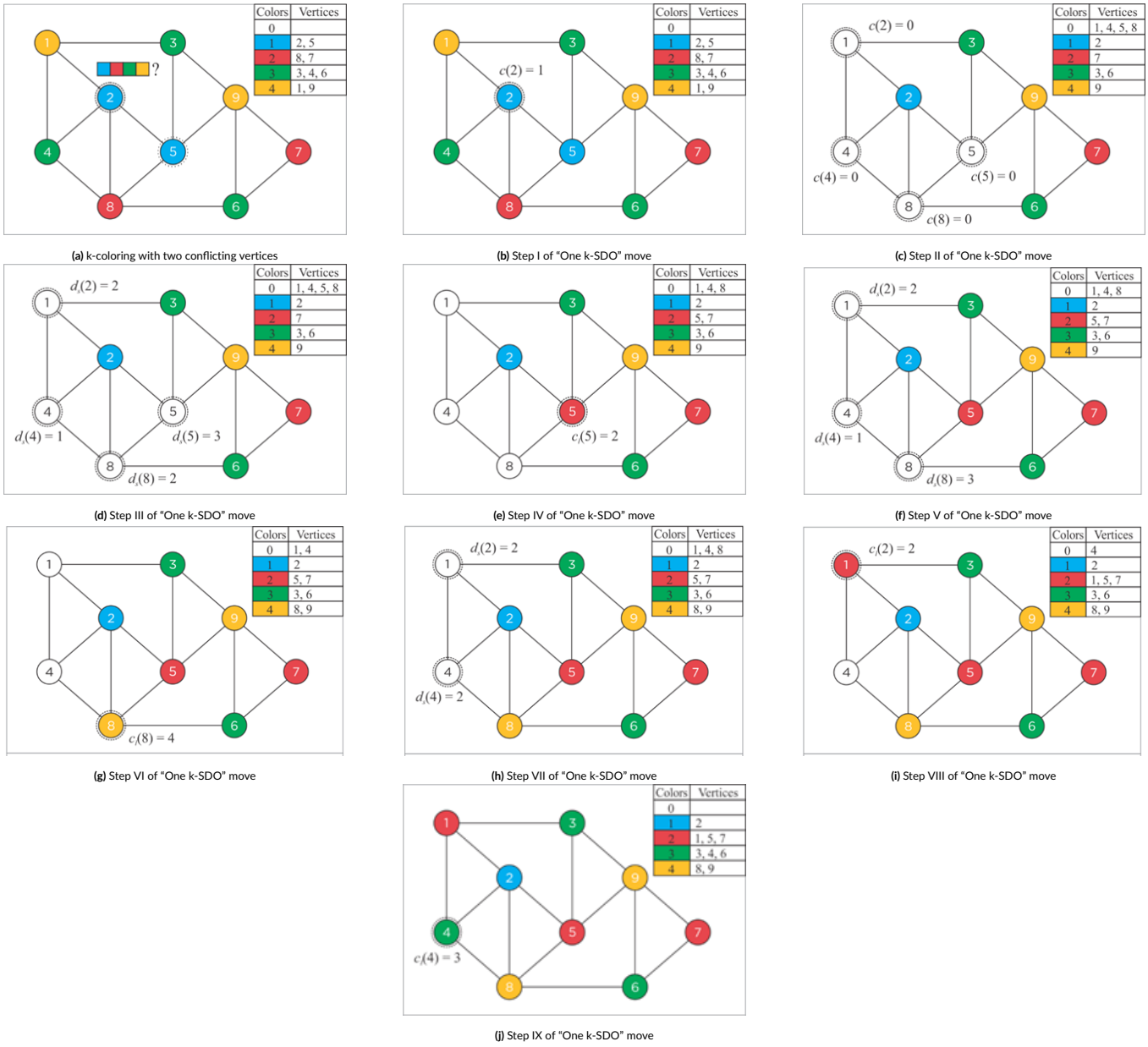


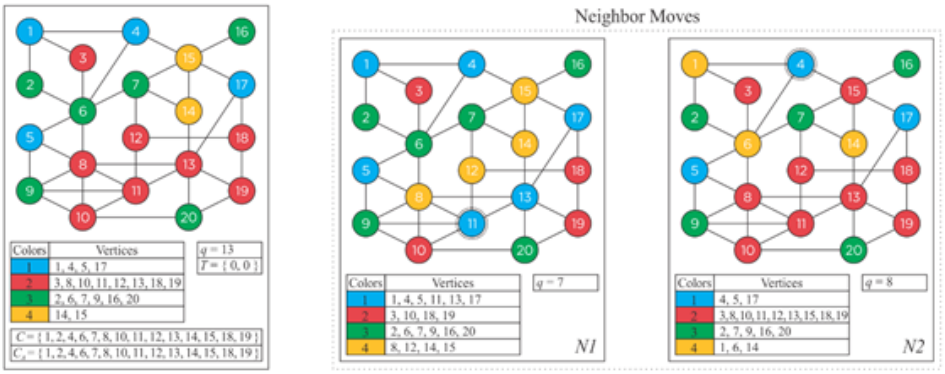
Figure 1 Conflict resolution steps through One k-SDO move

as it is in T. New neighbor moves N1 and N2 are generated using v6 and v13. Both N1 and N2 produce coloring with the same number of conflicts and both are better than current coloring of the mole-rat. Therefore, any one of them can be selected. Let N2 be the coloring which updates the mole-rat. Figure 2c shows the next iteration of local move that results in elimination of all the conflicts.

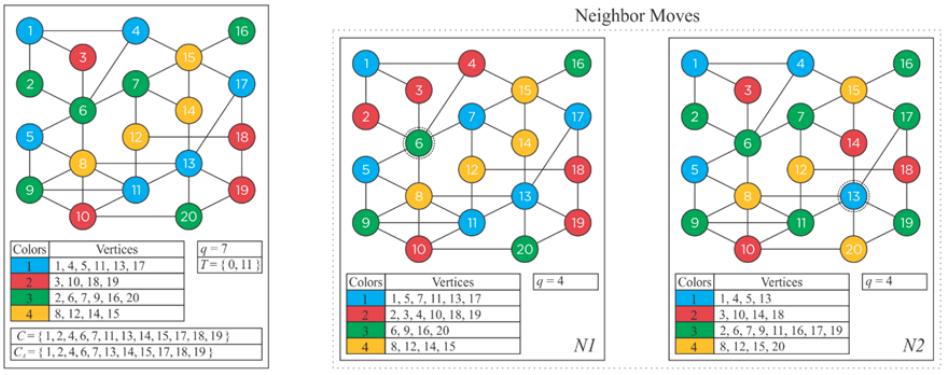
3.5.2 | Probabilistic-Exploitation

In this phase, a mole-rat M_j selects another target mole-rat M_t from the population. Selection is based on probability which is highest for the fittest mole-rat. The probability ρ for each mole-rat in the population is defined as $\rho((M_t)) = \frac{f(M_t)}{\sum_{u=1}^{N} f(M_u)}$.

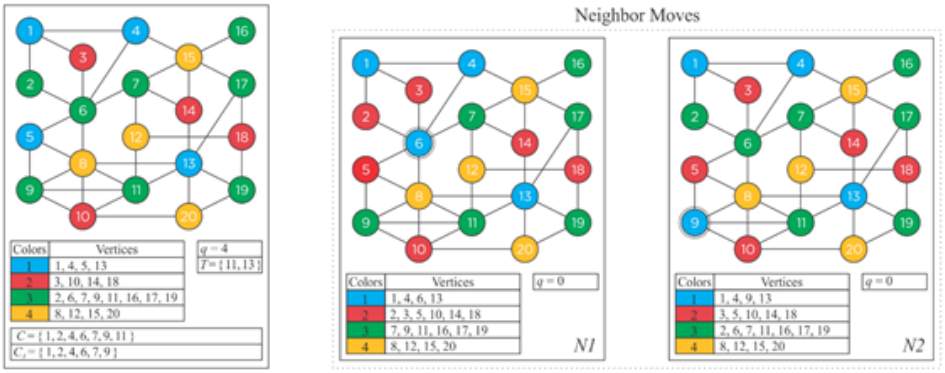
After selection of M_t, tabu search[45] is applied to food source X_t for a limited number of iterations. Combining self-exploitation with tabu search helps M_t to avoid getting stuck in local optima. Attenuation coefficient A controls the probability of a mole-rat to apply tabu search on another mole-rat's food source.



(a) Step I - Neighborhood move



(b) Step II - Neighborhood move



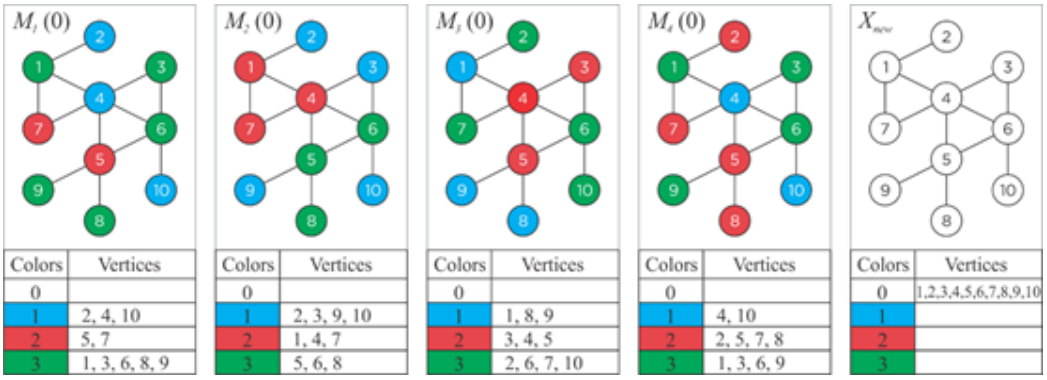
(c) Step III - Neighborhood move

Figure 2 Self exploitation of a given food source

3.6 | Exploration

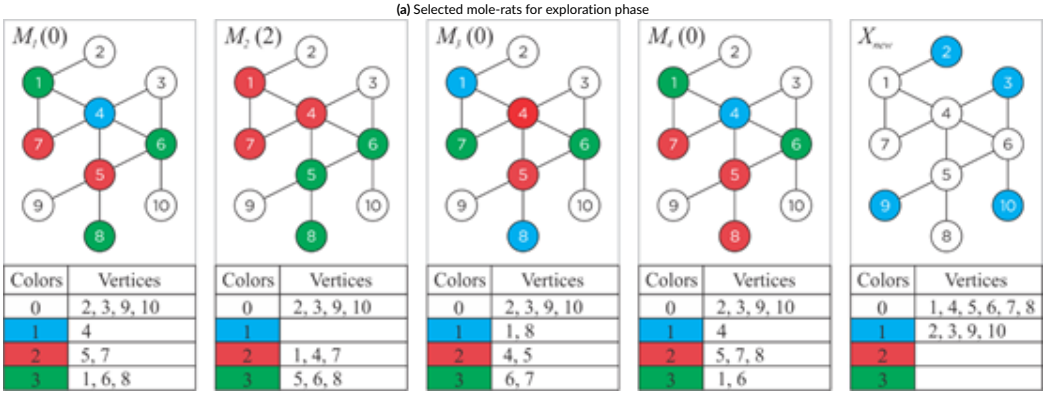
Crossover operator is employed to build a new food source X_{new} utilizing the knowledge of best mole-rats in the colony. We have used Adaptive Multi-Parent Crossover (AMPaX) operator[46] with a modified class evaluation function. AMPaX considers only the cardinality of the color class. However, we extend AMPaX to count the no of conflicts also in order to determine the quality of the color class. If a mole-rat M_i has below average fitness in the colony and is unable to improve it's fitness for λ number of iterations, then the food source Xi is replaced with $Xnew$. In order to create X_{new}, we select m mole-rats as parents (i.e. $M_{selected-1}, M_{selected-2} \cdots M_{selected-m}$), such that $m \geq 2$, and the selected mole-rats are the fittest mole-rats of the population. We evaluate each color class c in X_j (where $j = 1 \cdots m$) using equation 1.

$$g(V_j^c) = |V_j^c| \times h(V_j^c) \tag{1}$$



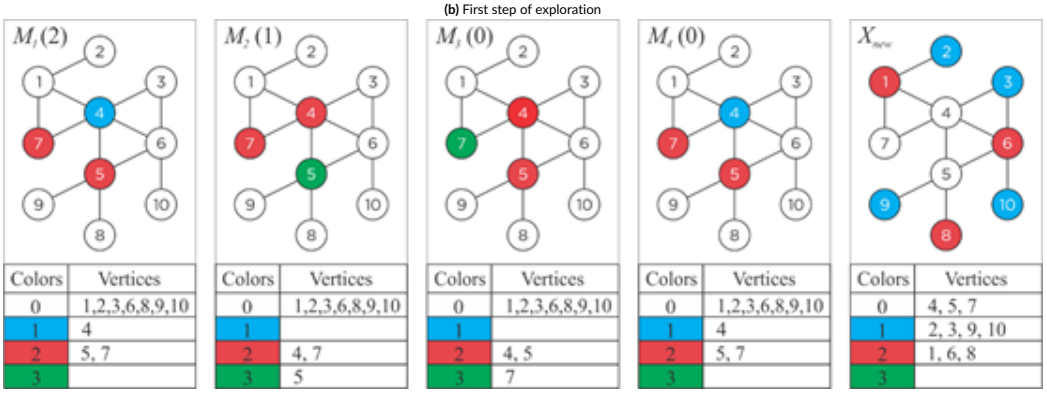
$M_1(0)$		$M_2(0)$		$M_3(0)$		$M_4(0)$		X_{new}	
Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices
0		0		0		0		0	1,2,3,4,5,6,7,8,9,10
1	2, 4, 10	1	2, 3, 9, 10	1	1, 8, 9	1	4, 10	1	
2	5, 7	2	1, 4, 7	2	3, 4, 5	2	2, 5, 7, 8	2	
3	1, 3, 6, 8, 9	3	5, 6, 8	3	2, 6, 7, 10	3	1, 3, 6, 9	3	

(a) Selected mole-rats for exploration phase



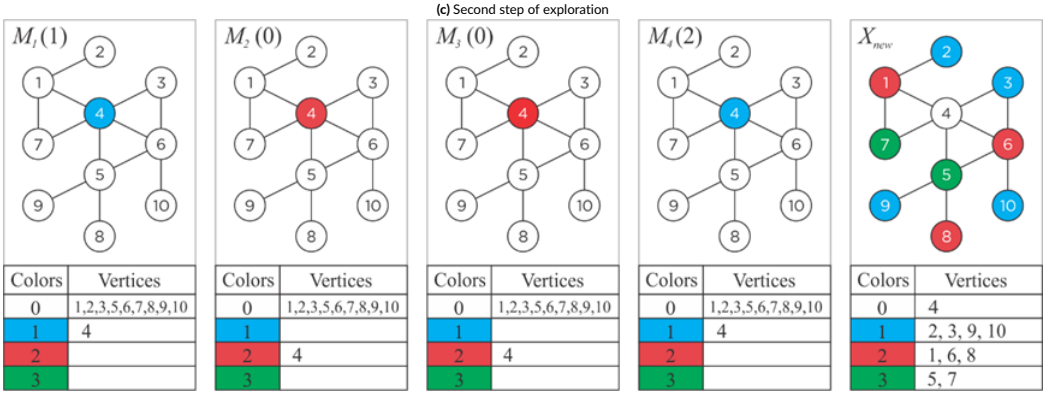
$M_1(0)$		$M_2(2)$		$M_3(0)$		$M_4(0)$		X_{new}	
Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices
0	2, 3, 9, 10	0	2, 3, 9, 10	0	2, 3, 9, 10	0	2, 3, 9, 10	0	1, 4, 5, 6, 7, 8
1	4	1		1	1, 8	1	4	1	2, 3, 9, 10
2	5, 7	2	1, 4, 7	2	4, 5	2	5, 7, 8	2	
3	1, 6, 8	3	5, 6, 8	3	6, 7	3	1, 6	3	

(b) First step of exploration



$M_1(2)$		$M_2(1)$		$M_3(0)$		$M_4(0)$		X_{new}	
Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices
0	1,2,3,6,8,9,10	0	1,2,3,6,8,9,10	0	1,2,3,6,8,9,10	0	1,2,3,6,8,9,10	0	4, 5, 7
1	4	1		1		1	4	1	2, 3, 9, 10
2	5, 7	2	4, 7	2	4, 5	2	5, 7	2	1, 6, 8
3		3	5	3	7	3		3	

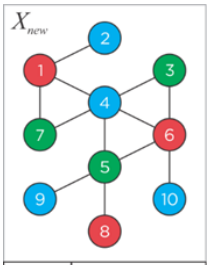
(c) Second step of exploration



$M_1(1)$		$M_2(0)$		$M_3(0)$		$M_4(2)$		X_{new}	
Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices	Colors	Vertices
0	1,2,3,5,6,7,8,9,10	0	1,2,3,5,6,7,8,9,10	0	1,2,3,5,6,7,8,9,10	0	1,2,3,5,6,7,8,9,10	0	4
1	4	1		1		1	4	1	2, 3, 9, 10
2		2	4	2	4	2		2	1, 6, 8
3		3		3		3		3	5, 7

(d) Third step of exploration

Figure 3 Exploration Phase comprising four parent mole rats to generate new food source.



Colors	Vertices
0	
1	2, 4, 9, 10
2	1, 6, 8
3	3, 5, 7

Figure 4 Solution obtained from exploration phase

Where $|V_j^c|$ is the number of vertices in X_j that belongs to color class c, and $h(V_j^c)$ is a function that evaluates the fitness of color class V_j^c based on $\complement_j^c$ number of conflicting vertices in V_j^c as elaborated in equation 2 and equation 3.

$$h(V_j^c) = \begin{cases} 1 & if\ \complement_j^c = 0 \\ 0.667 & if \complement_j^c = 1 \\ \frac{1}{\complement_j^c} & otherwise \end{cases} \tag{2}$$

$$\complement_j^c = \sum_{i=1}^{E} \left| \forall (v_a, v_b) \in E \mid v_a, v_b \in V_j^c \right| \tag{3}$$

After evaluating the color classes for all the selected parent mole-rats, a mole-rat $M_a \in M_{selected-1}, M_{selected-2} \cdots M_{selected-m}$ is selected that has the best color class $c*$ i.e. $h(V_a^{c*}) \geq h(V_j^c)$, where $j \neq a$ and $j = 1 \cdots m$. All the vertices V_a^{c*} of the best-selected color class are then transferred to X_new as its first color class. A restricted list with size $m/2$ is maintained to keep diversity in selection. Mole-rat M_a is added in the restricted list and vertices in V_a^c* are then removed from all the selected parent mole-rats. In next step, another parent mole-rat $M_b \in M_{selected-1}, M_{selected-2} \cdots M_{selected-m}$ with best color class V_b^{c**} among all the parent mole-rats is selected. Note that M_b should not be the part of restricted list. All the vertices in V_b^{c**} are then transferred to X_{new} as its second color class. In similar manner, k number of color classes are transferred to X_{new},. Any uncolored vertices in X_{new} are then assigned random color c_r such that $1 \leq c_r \leq k$. The new food source X_{new} finally replaces X_i.

We have explained the concept of exploration using an example as shown in figure 3. we have selected 4 mole-rats $M_1, M_2, M_3, and M_4$ to build a new graph coloring X_{new}. Three color classes has been transferred from selected parent mole-rats to X_{new} as $k = 3$. In the first step, the best color class is determined among all the mole-rats. Consider mole-rat M_1 and its color classes C_1, C_2 and C_3. Quality of a color class is determined using equation 1. There are three vertices in C_1 and no conflict in the class, that results in the quality of color class $g(C_1) = 3$. Similarly, in C_2, there are two vertices and no conflict. Quality of color class C_2 is given as $h(C_2) = 2$. Lastly, quality of color class C_3 is equal to $h(C_3) = 3.335$ as there are five vertices and one conflict between v_3 and v_6. Therefore, the best color class of M_1 is C_3. In the similar fashion, we determine the best color classes of all mole-rats. The best color classes of M_2 is C_1 with $h(C_1) = 4$, M_3 is C_1 with $h(C_1) = 3$, and M_4 is C_2 with $g(C_2) = 2.668$. The best color class among mole-rats $M_1 \cdots M_4$ is C_1 of M_2. This color class is then transferred to X_{new} as its first color class. M_2 is considered as tabu for the next two iterations. All the vertices of C_1 (v_2, v_3, v_9, v_{10}) are uncolored for mole-rats $M_1 \cdots M_4$. This completes first step of exploration, with a single class transferred to X_{new}. Figure 3b shows the first color class of X_{new} after transferring C_1 of M_2.

In following steps C_3 of M_1 and C_2 of M_4 is transferred to X_{new} as shown in figure 3c and figure 3d respectively. After transferring all three classes to X_{new}, one vertex v_4 still remains uncolored. This vertex is assigned a random color within bounds of $1 \cdots k$. Figure 4 shows the final coloring solution.

3.7 | Invader Removal

Colony defense mechanism[18] considers low-cost data points as invaders. They are replaced by randomly selected data points from search space, thus causing mutation. In BNMR-Col vertices with the highest conflict degree are considered as invaders. In mutation we select a conflicting vertex v_i (with color c_s) that has maximum conflict degree. We replace color of v_i with a random color c_r such that $1 \leq c_r \leq k$ and $c_r \neq c_s$.

3.8 | Removal Rate

Removal rate b_i of a mole-rat M_i defines the number of times mutation is performed. Initially, the removal rate is determined as in equation 4.

$$b_j = 1 + (r \times |V|) \tag{4}$$

$$b_j = 1 + (\zeta \times b_j) \tag{5}$$

Where r is a random number in the range $0.1 - 0.25$ and $|V|$ is the number of vertices in the graph. In each iteration, the removal rate is updated as mentioned in equation 5. ζ specifies the rate of decrement or increment in the removal rate and can vary in the range of $0 - 1$.

3.9 | Fitness Function

The fitness of each mole-rat $f(M_i)$ is determined on the basis of the number of conflicts in its assigned food source X_i. We use equation 6 to find the fitness of each mole-rat.

$$f(M_i) = \sum_{c=1}^{k} h(V_i^c) \tag{6}$$

4 | BNMR-COL

Natural inspiration for the BNMR algorithm comes from the social behavior of blind naked mole-rats in a colony. Two key aspects of mole-rat colony behavior depicted in the BNMR meta-heuristic are digging of the tunnels and colony defense. Mole-rats dig tunnels in search of large tubers to feed on. This behavior is modeled as a stochastic search for food sources. The colony defense mechanism is formulated as a mutation process to enforce exploration in a search space.

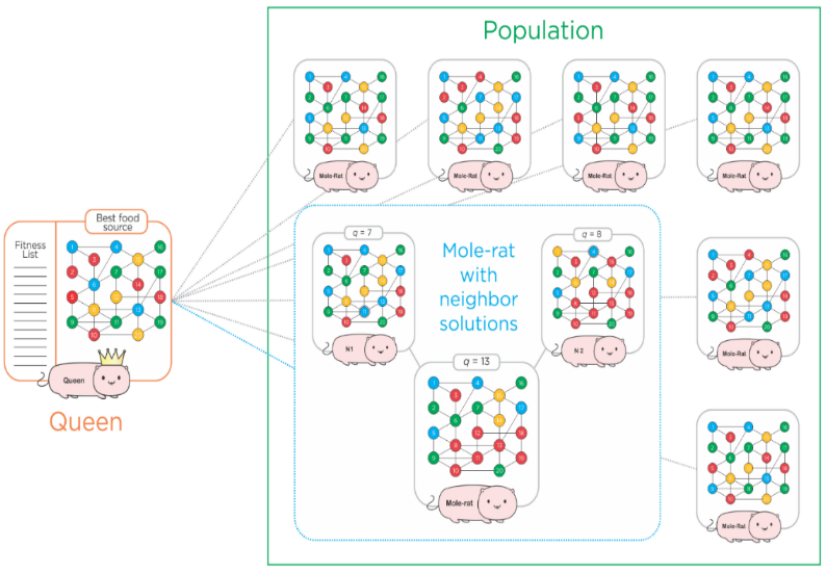


Figure 5 Overview of BNMR colony for k-GCP

Each mole-rat in a colony independently searches for a configuration (k-coloring) with fewer conflicts. The queen mole-rat acts as a space to store the best coloring as well as the fitness values of all the mole-rats as shown in figure 5. This enables queen mole-rat to facilitate other mole-rats to coordinate with each other. BNMR-Col consists of four phases: (i) Population Initialization, (ii) Exploitation of Food Sources, (iii) Mutation, and (iv) Crossover. Detailed working of BNMR-Col has been explained in algorithm 3.

Algorithm 3 Blind Naked Mole Rat Coloring (BNMR-Col) Algorithm

Require:

$G \leftarrow (V, E)$ ▷ V is the set of vertices, E is the set of edges

Initialize k, α, β, ζ ▷ Number of colors, decrement factor for A, mutation probability, mole-rat removal rate's decrement factor

Initialize $X_i = \{C_1^i, C_1^i \cdots C_k^i\}$ ▷ i^{th} Solution with vertices using k colors

Define M_i with food source X_i ▷ Each mole rat initailed with a k coloring solution

Initialize b_i and s_i ▷ Initialize removal rate and stagnation counter of M_i

Define $t = 0, T, A$ ▷ Define current iteration, maximum iteration, and attenuation variable

Ensure: $X_{best} = \{C_1^{best}, C_2^{best} \cdots C_k^{best}\}$ ▷ Optimal or Near Optimal Solution

1: **while** $t \leq T$ **do** ▷ While stopping criteria not met i.e. maximum iterations
2: **for each** M_i **do** ▷ where $i = 1 \cdots n$
3: **if** $rand \leq A_i$ **then** ▷ Exploitation phase
4: M_i performs probabilistic exploitation on M_j
5: **end if**
6: M_i performs self exploitation on X_i
7: **if** $rand < \beta$ and $f(M_i) < f(M_{best})$ **then**
8: Assign random colors to b_i vertices
9: **end if**
10: $b_i \leftarrow 1 + (\zeta + b_i)$
11: **if** $s_i > s_l$ **then**
12: **if** $f(M_i) < \frac{\sum_{j=1}^{n} f(M_j)}{n}$ **then**
13: Apply invader removal mechanism to M_i
14: $s_j \leftarrow 0$
15: **end if**
16: **end if**
17: Determine and update best mole rat M_{best}
18: $A = \alpha \, exp^{\frac{-\alpha t}{0.5T}}$
19: $t \leftarrow t + 1$
20: **end for**
21: **end while**

5 | RESULTS AND DISCUSSION

We have implemented BNMR-Col in C#. Moreover, we have also implemented BNMR-Col, ANTCOL and Chromaticats for comparison. We have conducted experiments on a machine having 3.4 GHz Intel Core i3 4130 processor and 16 GB RAM, using 64-bit Windows 10 Operating System. We have selected graph instances for experiments from a more recent collection of DIMACS instances by "COLOR02/03/04:

Table 2 Characteristics of graph instances from different graph families.

Mycielski Graphs (MYC)	These triangle free graphs are based on Mycielski transformation.
Register Graphs (REG)	Graph files are based on problem of register allocation for variables in real codes.
Book Graphs (SGB)	Each vertex in a graph represents a fictional character and is connected by an edge with another character.
Game Graph (SGB)	Each vertex represents a college football team and two teams are connected by an edge if they play a match.
Queen Graphs (SGB)	Each vertex represents a square on chessboard and has an edge with all other vertices in same row, column, or diagonal.
Miles Graphs (SGB)	Each vertex represents a USA city. Two vertices are connected by an edge if they are close enough, based on road mileage.
Insertion Graphs (CAR)	These graphs are generalization of Mycielski graphs with inserted vertices to increase graph size but not density.
Random Graphs (DSJ)	These are randomly generated graphs by David Johnson.
Multi Coloring Graphs	These are quasi-random graphs intended for multi coloring problem and used for graph coloring by ignoring node weights.

Graph Coloring and its Generations" symposium[47]. Graph instances are grouped together into various families based on their sources. A brief description of graph families are discussed in table 2. Table 3 shows the characteristics of various graph instances from different graph families.

Where $|V|$ denotes the number of vertices in graph, $|E|$ denotes the number of edges in graph, $\triangle$ gives the degree of graph, and lowest k is the known minimum value of colors for a graph with which it can be properly colored.

Table 3 Characteristics of various graph instances from MYC, REG, CAR and DSJ families along with uncategorized instances. $|V|$ denotes the number of vertices in the graph, |E| denotes the number of edges in the graph, $\triangle$ gives the degree of the graph, and lowest k is the known minimum value of colors for a graph with which it can be properly colored.

Sr.	instance	family	|V|	|E|	$\triangle$	lowest k	Sr.	instance	family	|V|	|E|	$\triangle$	lowest k
1	myciel3	MYC	11	20	5	4	21	1-FullIns_5	CAR	282	3247	95	5
2	myciel4	MYC	23	71	11	5	22	2-FullIns_3	CAR	52	201	15	4
3	myciel5	MYC	47	236	23	6	23	2-FullIns_4	CAR	212	1621	55	5
4	myciel6	MYC	95	755	47	7	24	3-FullIns_3	CAR	80	346	19	5
5	myciel7	MYC	191	2360	95	8	25	4-FullIns_3	CAR	114	541	23	6
6	mulsol.i.1	REG	197	3925	121	49	26	5-FullIns_3	CAR	154	792	27	7
7	mulsol.i.2	REG	188	3885	156	31	27	R100_1g	-	100	509	20	5
8	mulsol.i.3	REG	184	3916	157	31	28	R100_5g	-	100	2456	61	15
9	zeroin.i.1	REG	211	4100	111	49	29	R100_9g	-	100	4438	45	36
10	zeroin.i.2	REG	211	3541	140	30	30	R75_1g	-	70	251	12	4
11	zeroin.i.3	REG	206	3540	140	30	31	R75_5g	-	75	1407	48	14
12	1-Insertions_4	CAR	67	232	22	5	32	R75_9g	-	75	2513	71	33
13	1-Insertions_5	CAR	202	1227	67	6	33	R50_1g	-	50	108	8	3
14	2-Insertions_3	CAR	37	72	9	4	34	R50_5g	-	50	612	36	10
15	2-Insertions_4	CAR	149	541	37	5	35	R50_9g	-	50	1092	47	21
16	3-Insertions_3	CAR	56	110	11	4	36	r125.1	-	125	209	8	5
17	3-Insertions_4	CAR	281	1046	56	5	37	r125.5	-	125	3838	99	36
18	4-Insertions_3	CAR	79	156	13	3	38	DSJC125.1	DSJ	125	736	23	5
19	1-FullIns_3	CAR	30	100	11	3	39	DSJC125.5	DSJ	125	3891	75	12
20	1-FullIns_4	CAR	93	593	32	4	40	DSJC125.9	DSJ	125	6961	120	30

Table 4 Characteristics of selected graph instances from Donald Knuth's Stanford Graph Base

games120	SGB	120	638	26	9
queen5_5	SGB	25	320	32	5
queen6_6	SGB	36	580	38	7
queen7_7	SGB	49	952	48	7
queen8_8	SGB	64	1456	54	9
queen8_12	SGB	96	2736	64	12
queen9_9	SGB	81	2112	64	10
queen10_10	SGB	100	2940	70	11
queen11_11	SGB	121	3960	80	11
queen12_12	SBG	144	5192	86	13
miles250	SGB	128	774	32	8
miles500	SGB	128	2340	76	20
miles750	SGB	128	4226	128	31
miles1000	SGB	128	6432	172	42
miles1500	SGB	128	10396	212	73
anna	SGB	138	493	142	11
huck	SGB	74	301	106	11
jean	SGB	80	254	72	10
david	SGB	87	406	164	11

Graph instances from Donald Knuth's Stanford Graph Base (SGB) are further divided into book, queen, miles graphs and a game graph. Table 4. shows number of vertices, number of edges, degree of graph and best known k for selected SGB graphs. Two basic strategies are employed in BNMR-Col i.e. fixing k when chromatic number is known, and sequentially decreasing k until algorithm termination criteria is met where the upper bound on k is provided by LDO. Rational behind using these graph instances for result comparison was the popularity and common use of

these graph instances by state of the art graph coloring techniques. Table 5 depicts the parameters for ANTCOL, BNMR-Col and Chromaticats. Parameters for ANTCOL and Chromaticats are used as suggested in their respective implementations.

Table 5 Parameters for AntCol, BNMR-Col and Chromaticats.

BNMR-Col Parameters	Chromaticats Parameters	ANTCOL Parameters
max_iteration = 100	Number of Cats = 30	nants = 100
N = 10	Number of Cycles = 100	$ncycles_{max}$ = 100
‘α = 0.5	MR=2%	α=2
β = 0.1	CDC = 80	β=4
ζ=0.95	SMP=20	ρ=0.5
ρ = 40%		
sl = 20		
move_count = 50		
depth_of_search= 20		

ANTCOL, Chromaticats and BNMR-Col are executed five times on 18 graph instances for maximum 100 iterations. Success ratio is defined as the number of times algorithm is able to reach best known number of colors denoted by k. Results have been shown in Table 6.

Table 7 show that none of the algorithm was able to find k for FullIns and DSJC125.1 graph instances. MSPGCA and W-GA were unable to find optimal coloring for queen6_6 and queen8_8 instances. W-GA cannot find optimal coloring for queen7_7. MSPGCA reported coloring with 14 colors for queen8_12 instance, for which best known coloring is 12. BNMR-Col is able to match k for all of these instances. For queen10_10 instance MA-GCP and MSPGCA reported coloring of 13 and 14 respectively. BNMR-Col was unable to match best known coloring (i.e. 11) for this instance. However, BNMR-Col reported coloring of 12, which is better than MA-GCP and MSPGCA. These results show that BNMR-Col is able to match or produce better colorings than MSPGCA, MA-GCP and W-GA.

Table 6 Computational results of ANTCOL, Chromaticats, and BNMR-Col.

instance	k	BNMR-Col		ANTCOL		Chromaticats	
		lowest k	Success Ratio	lowest k	Success Ratio	lowest k	Success Ratio
myciel5	6	6	5/5	6	5/5	6	5/5
myciel6	7	7	5/5	7	5/5	7	5/5
queen5_5	5	5	5/5	5	5/5	5	5/5
queen6_6	7	7	5/5	7	5/5	7	3/5
queen7_7	7	7	3/5	7	1/5	7	1/5
miles250	8	8	5/5	8	5/5	8	2/5
miles500	20	20	5/5	20	5/5	20	2/5
jean	10	10	5/5	10	5/5	10	5/5
huck	11	11	5/5	11	5/5	11	5/5
1-FullIns_3	3	4	0/5	4	0/5	4	0/5
2-FullIns_3	4	5	0/5	5	0/5	5	0/5
R50_1g	3	3	5/5	3	5/5	3	1/5
R50_5g	10	10	5/5	10	5/5	10	5/5
R50_9g	21	21	5/5	21	5/5	21	2/5
R75_1g	4	4	5/5	4	5/5	4	2/5
R75_5g	13	13	5/5	13	5/5	15	0/5
R75_9g	33	33	5/5	33	5/5	34	0/5
mulsol.i.1	49	49	5/5	49	5/5	49	3/5

6 | CONCLUSION

We have proposed a new algorithm BNMR-Col, an implementation of Blind Naked Mole-Rats metaheuristics for GCP. BNMR-Col is tested on a wide range of DIMACS graph instances. We compared the results with other state ot the art swarm intelligence and evolutionary algorithms. BNMR-Col is competitive and in 83% of the cases it approaches the best known value of k. Further research can be conducted to investigate the

Table 7 Results of BNMR-Col, MA-GCP, MSPGCA and W-GA.

instance	k	MA-GCP[20]	MSPGCA[18]	W-GA[19]	BNMR-Col
1-FullIns_5	5	-	6	-	6
2-FullIns_4	5	-	6	-	6
3-FullIns_4	6	-	7	-	7
4-FullIns_3	6	-	7	-	7
5-FullIns_3	7	-	8	-	8
1-Insertions_5	6	-	6	-	6
2-Insertions_4	5	-	5	-	5
3-Insertions_4	5	-	5	-	5
4-Insertions_4	5	-	5	-	5
myciel3	4	4	-	4	4
myciel4	5	5	-	5	5
myciel5	6	6	6	6	6
myciel6	7	7	7	-	7
myciel7	8	8	8	-	8
games120	9	9	9	9	9
huck	11	11	11	11	11
jean	10	10	10	10	10
david	11	11	11	11	11
queen5_5	5	5	5	5	5
queen6_6	7	7	8	8	7
queen7_7	7	7	7	8	7
queen8_8	9	-	11	10	9
queen8_12	12	-	14	-	12
queen9_9	10	-	10	-	10
queen10_10	11	13	14	-	12
miles250	8	8	-	8	8
miles500	20	20	-	-	20
miles750	31	31	31	-	31
miles1000	42	42	42	42	42
miles1500	73	73	73	-	73
anna	11	11	11	11	11
DSJC125.1	5	-	6	-	6
mulsol.i.1	49	49	-	-	49
zeroin.i.1	49	49	-	-	49
fpsol2.i.1	65	-	-	65	65

various aspects that this study was unable to address. More extensive computational experimentation involving large graph instances and using more efficient graph library implementation is always desirable. Research can be conducted by using different and more complex fitness functions. To improve diversification in population, a distance criterion can be used to distinguish between similar individuals of BNMR-Col. Reduction of neighborhood size with the reduction of conflicts is a major issue for local move which can be explored further. Another area of possible future research is to investigate cyclic behaviors and develop methods to avoid cycling in the search space. SI algorithms are intrinsically parallel in nature and recent development of cluster computing frameworks like Apache Spark also opens new horizons for the development of scalable BNMR-Col algorithm.

References

1. Deng W, Ni H, Liu Y, Chen H, Zhao H. An adaptive differential evolution algorithm based on belief space and generalized opposition-based learning for resource allocation. *Applied Soft Computing* 2022; 127: 109419.

2. Qu BY, Zhu Y, Jiao Y, Wu M, Suganthan PN, Liang JJ. A survey on multi-objective evolutionary algorithms for the solution of the environmental/economic dispatch problems. *Swarm and Evolutionary Computation* 2018; 38: 1–11.

3. Maqbool F, Razzaq S, Yar A, Jabeen H. Large Scale Distributed Optimization using Apache Spark: Distributed Scalable Shade-Bat (DistSSB). In: IEEE. ; 2021: 2559–2566.

4. Gao S, Wang K, Tao S, Jin T, Dai H, Cheng J. A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models. *Energy Conversion and Management* 2021; 230: 113784.

5. Slowik A, Kwasnicka H. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications* 2020; 32(16): 12363–12379.

6. Bui TN, Moon BR. Genetic algorithm and graph partitioning. *IEEE Transactions on computers* 1996; 45(7): 841–855.

7. Talbi EG, Bessiere P. A parallel genetic algorithm for the graph partitioning problem. In: ACM. ; 1991: 312–320.

8. Mishra A, Vakharia D, Hati AJ, Raju KS. Hardware software partitioning of task graph using genetic algorithm. In: IEEE. ; 2014: 1–5.

9. Esbensen H. Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm. *Networks* 1995; 26(4): 173–185.

10. Rojas F, Meza F. A parallel distributed genetic algorithm for the prize collecting steiner tree problem. In: IEEE. ; 2015: 643–646.

11. Zhang K, Qiu M, Li L, Liu X. Accelerating genetic algorithm for solving graph coloring problem based on CUDA architecture. In: Springer. 2014 (pp. 578–584).

12. De S. An efficient technique of resource scheduling in cloud using graph coloring algorithm. *Global Transitions Proceedings* 2022; 3(1): 169–176.

13. Agizza M, Balzano W, Stranieri S. An Improved Ant Colony Optimization Based Parking Algorithm with Graph Coloring. In: Springer. ; 2022: 82–94.

14. Marappan R. A New Multi-Objective Optimization in Solving Graph Coloring and Wireless Networks Channels Allocation Problems. *Int. J. Advanced Networking and Applications* 2021; 13(02): 4891–4895.

15. He Y, Gao C, Sang N, Qu Z, Han J. Graph coloring based surveillance video synopsis. *Neurocomputing* 2017; 225: 64–79.

16. Demange M, Ekim T, Ries B, Tanasescu C. On some applications of the selective graph coloring problem. *European Journal of Operational Research* 2015; 240(2): 307–314.

17. Mohammad TMHS, Mohammad HB. A novel meta-heuristic algorithm for numerical function optimization: Blind, naked mole-rats (BNMR) algorithm. *Scientific Research and Essays* 2012; 7(41): 3566–3583.

18. Taherdangkoo M, Shirzadi MH, Yazdi M, Bagheri MH. A robust clustering method based on blind, naked mole-rats (BNMR) algorithm. *Swarm and Evolutionary Computation* 2013; 10: 1–11.

19. Yildirim T, Kalayci CB, Mutlu Ö. A novel metaheuristic for traveling salesman problem: blind mole-rat algorithm. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi* 2016; 22(1): 64–70.

20. Taherdangkoo M. Modified blind naked mole-rat algorithm applied to electromagnetic design problems. *Archives of Electrical Engineering* 2021; 70(2).

21. Johnson DS, Trick MA. Introduction to the second DIMACS challenge: cliques, coloring, and satisfiability. *Cliques, coloring, and satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 1996; 26: 1–7.

22. Ferrández JM, Lorente V, Cuadra J, Álvarez-Sánchez JR, Fernández E, others . A biological neuro processor for robotic guidance using a center of area method. *NeuroComputing* 2011; 74(8): 1229–1236.

23. Bacarisas ND, Yusiong JPT. The Effects of Varying the Fitness Function on the Efficiency of the Cat Swarm Optimization algorithm in Solving the Graph Coloring Problem.. *Annals. Computer Science Series* 2011; 9(2).

24. Fister Jr I, Yang XS, Fister I, Brest J, Fister D. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186* 2013.

25. Yu X, Gen M. *Introduction to evolutionary algorithms*. Springer Science & Business Media . 2010.

26. Maier HR, Razavi S, Kapelan Z, Matott LS, Kasprzyk J, Tolson BA. Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental modelling & software* 2019; 114: 195–213.

27. Chen B, Chen L. A link prediction algorithm based on ant colony optimization. *Applied Intelligence* 2014; 41(3): 694–708.

28. Razzaq S, Maqbool F, Hussain A. Modified cat swarm optimization for clustering. In: Springer. ; 2016: 161–170.

29. Jabeen H, Baig AR. Particle swarm optimization based tuning of genetic programming evolved classifier expressions. In: Springer. 2010 (pp. 385–397).

30. Rebollo-Ruiz I, Graña M. An empirical evaluation of Gravitational Swarm Intelligence for graph coloring algorithm. *Neurocomputing* 2014; 132: 79–84.

31. Brélaz D. New methods to color the vertices of a graph. *Communications of the ACM* 1979; 22(4): 251–256.

32. Li C, Xu H, Liao X, Yu J. Tabu search for CNN template learning. *Neurocomputing* 2003; 51: 475–479.

33. Johnson DS, Aragon CR, McGeoch LA, Schevon C. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations research* 1991; 39(3): 378–406.

34. Ge F, Wei Z, Tian Y, Huang Z. Chaotic ant swarm for graph coloring. In: . 1. IEEE. ; 2010: 512–516.

35. Mahmoudi S, Lotfi S. Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem. *Applied soft computing* 2015; 33: 48–64.

36. Costa D, Hertz A. Ants can colour graphs. *Journal of the operational research society* 1997; 48(3): 295–305.

37. Johri A, Matula DW. Probabilistic bounds and heuristic algorithms for coloring large random graphs. Master's thesis. Southern Methodist University. Dallas, Texas, USA: 1982.

38. Qin J, Yin Y, Ban X. Hybrid Discrete Particle Swarm Algorithm for Graph Coloring Problem.. *J. Comput.* 2011; 6(6): 1175–1182.

39. Galinier P, Hao JK. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization* 1999; 3(4): 379–397.

40. Tomar RS, Singh S, Verma S, Tomar GS. A novel abc optimization algorithm for graph coloring problem. In: IEEE. ; 2013: 257–261.

41. Baiche K, Meraihi Y, Hina MD, Ramdane-Cherif A, Mahseur M. Solving graph coloring problem using an enhanced binary dragonfly algorithm. *International Journal of Swarm Intelligence Research (IJSIR)* 2019; 10(3): 23–45.

42. Chen K, Kanoh H. Solving the graph coloring problem using Adaptive Artificial Bee Colony. *Transactions of the Society for Evolutionary Computation* 2019; 9(3): 103–114.

43. Taherdangkoo M, Taherdangkoo R. Modified BNMR algorithm applied to Loney's solenoid benchmark problem. *International Journal of Applied Electromagnetics and Mechanics* 2014; 46(3): 683–692.

44. Lü Z, Hao JK. A memetic algorithm for graph coloring. *European Journal of Operational Research* 2010; 203(1): 241–250.

45. Hertz A, Werra dD. Using tabu search techniques for graph coloring. *Computing* 1987; 39(4): 345–351.

46. Chowdhury HAR, Farhat T, Khan MH. Memetic algorithm to solve graph coloring problem. *International Journal of Computer Theory and Engineering* 2013; 5(6): 890.

47. Bensouyad M, Guidoum N, Saïdouni DE. A New and Fast Evolutionary Algorithm for Strict Strong Graph Coloring Problem. *Procedia Computer Science* 2015; 73: 138–145.

How to cite this article: Williams K., B. Hoskins, R. Lee, G. Masato, and T. Woollings (2016), A regime analysis of Atlantic winter jet variability applied to evaluate HadGEM3-GC2, *Q.J.R. Meteorol. Soc.*, *2017;00:1–6.*