Received: 30 November 2025 | Revised: 13 May 2026 | Accepted: 23 June 2026 | Published online: 17 September 2026

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–13
DOI: 10.47852/bonviewJDSIS62028596

BON VIEW PUBLISHING

A New Metaheuristic Algorithm for Large Graph Coloring Problems

Tansel Dokeroglu[1] **and Deniz Canturk**[1,*]

[1] *Software Engineering Department, TED University, Turkey*

Abstract: The Graph Coloring Problem (GCP) is an NP-hard combinatorial optimization problem that involves assigning colors to the vertices of a graph such that no two adjacent vertices share the same color. Due to its computational complexity, exact algorithms are often impractical for solving large-scale instances within reasonable time constraints. Consequently, soft computing techniques—particularly metaheuristics—have been widely employed and demonstrated high efficiency in addressing large GCP instances. In this study, we propose a novel island-based parallel metaheuristic algorithm, referred to as PEM-Color, designed to tackle large instances of the GCP. Ensemble learning, a recent paradigm in machine learning, enhances model performance by aggregating the outputs of multiple distinct models rather than relying on a single one. Building on this idea, PEM-Color integrates three state-of-the-art metaheuristic algorithms—Harris Hawk Optimization, Artificial Bee Colony, and Teaching–Learning-Based Optimization—within a parallel ensemble framework using the Message Passing Interface for efficient distributed computation. To the best of our knowledge, this is the first study to employ an ensemble-based approach that combines multiple metaheuristics for solving the GCP in a parallel computing environment. Extensive experiments were conducted on large-scale graph instances from the well-known DIMACS benchmark set, utilizing 64 processors. The results demonstrate substantial reductions in execution time, with near-linear scalability and speed-up. Moreover, the proposed algorithm achieved superior solution quality, outperforming 13 state-of-the-art algorithms, highlighting its effectiveness and potential for further research in large-scale GCPs.

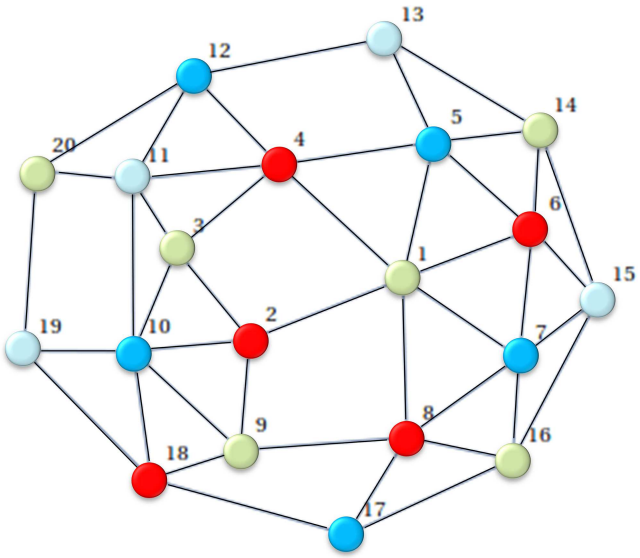
Keywords: graph coloring, ensemble, optimization, TabuCol, parallelization

1. Introduction

The Graph Coloring Problem (GCP) stands as a challenging NP-Hard combinatorial optimization problem in graph theory [1]. The GCP, namely, vertex coloring problem, is a labeling problem and can be defined as minimizing the number of colors (the chromatic number) of vertices such that no two neighboring vertices have the same color in an undirected graph. (Figure 1 illustrates an undirected graph that satisfies the constraints of the GCP, with 20 vertices and painted with four colors.) There are many application areas of the GCP in the real world such as optimization problems of timetabling [2], scheduling [3], computer science [4], biology [5], sociology, etc.

Finding an optimal solution for the GCP is still challenging due to the time complexity of the exact algorithms that can only find solutions for graphs with up to 100 vertices. Often, more intelligent techniques such as metaheuristics need to be employed for larger problems. Many metaheuristic algorithms have been proposed to obtain near-optimal solutions for the GCP recently. Genetic algorithms (GAs), Artificial Bee Colony (ABC) Optimization [6, 7], Ant Colony Optimization (ACO) [8, 9], Cuckoo Optimization Algorithm, and Particle Swarm Optimization (PSO) [10, 11] are the most well-known metaheuristics applied to the solution of the GCP.

Figure 1
Illustration of GCP: an undirected graph (having 20 vertices) labeled using four colors



In essence, all metaheuristics are heuristic methods designed to solve generic combinatorial optimization problems. Often,

***Corresponding author:** Deniz Canturk, Software Engineering Department, TED University, Turkey. Email: deniz.canturk@tedu.edu.tr

metaheuristic approaches aim to find solutions to complex problems using a three-step process applied iteratively. The first step of this three-step process involves initialization of the metaheuristic with a set of random or semi-random solutions called the *candidate solutions*. Depending on the metaheuristic, the representation of the candidate solutions may vary significantly. The initialization is then followed by a fitness calculation where each candidate solution is assigned a quality score according to its resemblance to the optimization objective. Last, metaheuristics employ various selection/adaptation operations to modify the existing candidate solutions so that the solution space is scanned sufficiently well (i.e., exploration) and regions in the solution space that have relatively high potential for an optimal solution are searched well (i.e., exploitation). In the literature, many different selection/adaptation mechanisms exist. For example, Genetic Algorithm (GA) mimics crossover and mutation mechanics in Darwinian biology, ABC and ACO adopt animal behavioral patterns, and PSO uses a simulated social behavior mechanism for the purpose. By adopting an iterative approach, metaheuristics explore the solution space for an optimal solution, and this process continues until a specified criterion is satisfied. Figure 2 illustrates the representation of a solution (a chromosome for GA) used by metaheuristic algorithms for the graph given in Figure 1.

Ensemble learning is a machine learning technique that combines multiple models to improve the overall performance and accuracy of the solution [12, 13]. Instead of relying on a single model, ensemble methods use a collection of diverse models and aggregate their predictions to make more robust and reliable predictions than individual models alone. The diversity among the models can be achieved through different algorithms, variations in training data, or by tweaking the model parameters. Ensemble methods can leverage multiple learning methods to enhance the performance beyond what could be achieved by any individual learning algorithm in isolation [14]. They have evolved as a means to increase the generality of search and optimization algorithms, and in our study, they work on top of a set of metaheuristics to solve a particular optimization problem. They can be used as distinct methodologies developed to solve challenging NP-Hard problems. In essence, ensemble algorithms can learn (decide) the most appropriate metaheuristic to solve a problem. The metaheuristic algorithms can obtain (near)-optimal solutions when exact algorithms face challenges due to the long execution times [15]. According to the No Free Lunch (NFL) theorem [16], the performance of a metaheuristic for a specific problem is balanced by its performance on another class of problems. Therefore, for problems with different complexities, different metaheuristics may perform better. This means different metaheuristics may perform better than others for different GCPs. Combining multiple metaheuristics with ensemble algorithms and using parallel computing to reduce the costs may provide a novel but intuitive approach to complex GCPs with a state-of-the-art approach. This was the most important motivation for our study.

In addition to this, the fitness evaluation of candidate solutions constitutes a major computational burden in metaheuristic algorithms, as it must be executed thousands of times at each iteration. In many studies, this process can take hours or even days, which is a well-known limitation of metaheuristic approaches. While parallel metaheuristic algorithms have been proposed to alleviate this issue by exploiting high computational capacity [17], most existing studies either focus on parallelizing a single algorithm or employ ensemble strategies without tight integration between heterogeneous metaheuristics. In contrast, the proposed framework distinguishes itself by jointly integrating Harris Hawk Optimization (HHO), ABC, and Teaching–Learning-Based Optimization (TLBO) within a unified parallel ensemble structure, where each algorithm operates concurrently while contributing to a shared search process. Unlike conventional approaches, this design not only distributes the computational load but also enhances exploration–exploitation balance through algorithmic diversity. Furthermore, the proposed model explicitly addresses challenges such as inter-process dependency, communication overhead, and load balancing through coordinated information exchange mechanisms, thereby achieving improved scalability and convergence performance compared to existing parallel or ensemble metaheuristic methods.

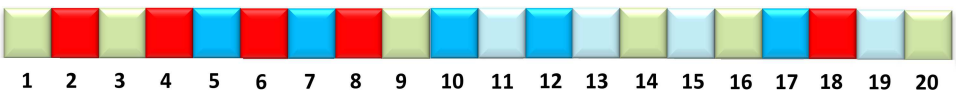
The selection of the employed metaheuristics is not arbitrary; rather, it is motivated by their complementary search behaviors and proven effectiveness in solving complex optimization problems. Specifically, HHO is known for its dynamic transition between exploration and exploitation, ABC provides strong global exploration and diversification capabilities, and TLBO is effective in rapid convergence with minimal parameter dependency. The combination of these algorithms is therefore designed to balance exploration, exploitation, and convergence speed within a unified framework, which is a common strategy supported by the metaheuristic literature.

In this study, we proposed a new and scalable parallel ensemble algorithm (PEM-Color) for the GCP optimization of large graphs. The proposed approach combined three state-of-the-art metaheuristics to achieve the best possible result quality for a given GCP. To this end, we carried out this study by using the most effective and up-to-date metaheuristics, HHO [18–20], with a new approach that we have not encountered in this field before. We use high-performance parallel computing techniques and Message Passing Interfaces (MPI) libraries to facilitate distributed memory computation and report significant improvements on the well-known Center for the Discrete Mathematics and Theoretical Computer Science (DIMACS) benchmark graph problem instances. We report the best results in 37 out of 43, and for the rest of the problems, our results are only marginally different from the reported best solutions.

The contributions of our study are as follows:

1) A new parallel ensemble metaheuristic algorithm is proposed for the GCP for the first time in the literature.
2) HHO, ABC, and TLBO are combined for the first time in an ensemble metaheuristic algorithm for the GCP.
3) HHO is applied to the GCP for the first time.
4) Almost a linear speed-up and strong scalability are achieved with the new parallel algorithm.

Figure 2
The representation of the GCP solution in Figure 1. The indices represent vertex IDs in the graph



5) Stagnation is controlled with different seeding mechanisms for each population in the computing environment.
6) The execution time of the metaheuristics is significantly reduced using MPI libraries.

Related studies for solving the GCP are given in Section 2. The proposed parallel ensemble metaheuristic algorithm is presented in Section 3. The experimental setup and obtained results are reported in Section 4. The conclusion and future works are provided in the last section.

2. Literature Review

Within this section, we conducted an extensive review of the most effective exact, metaheuristic, ensemble, parallel, tabu search, and parallel local search algorithms proposed for addressing the GCP [21, 22].

Bravyi et al. applied the recursive quantum optimization algorithm (RQAOA) to the k-vertex graph coloring problem [4]. The algorithm is compared to the classical and hybrid quantum algorithms. They reported that the RQAOA is surprisingly competitive. Gaspers and Lee [23] presented a polynomial algorithm to compute the number of independent sets in a graph. This algorithm is a fast one for the GCP. Shimizu and Mori [24] presented a polynomial quantum algorithm using quantum random access memory. The algorithm is based on quantum dynamic programming. Hoeve introduced an iterative framework using decision diagrams for the GCP [25]. The decision diagram represents color classes. A constrained minimum network flow model is used to compute conflicts. They performed evaluations on 137 DIMACS graph coloring instances. They solved 52 of the instances optimally.

We can briefly summarize the prominent parallel metaheuristic algorithms in the literature. Dokeroglu and Sevinc [26] proposed a memetic TLBO algorithm (TLBO-Color) combined with a tabu search. A parallel TLBO-Color is developed for painting large graphs having millions of edges and thousands of vertices. The optimization times are polynomial, and most of the optimal results of the DIMACS graphs are reported. They suggested a new ordering of the vertices. Hijazi et al. [27] introduced a parallel heterogeneous ensemble feature selection method incorporating three metaheuristics (genetic, PSO, and gray wolf optimizer). The approach involves the phases—distribution, parallel ensemble feature selection, combining and aggregation, and testing. A sequential approach on a CPU, a parallel approach on a multi-core CPU, and a parallel approach utilizing both a multi-core CPU and a GPU are implemented. The proposed parallel approach significantly enhanced predictive outcomes and reduced running time. This is the study most similar to our proposed algorithm. We use MPI instead of a GPU parallel computation environment, which is more scalable.

Bogle et al. [28] presented new MPI and GPU coloring techniques on the distributed memory architectures. The algorithms use Kokkos Kernels for CPUs and GPUs. They extended the algorithm to compute distance-2 coloring problems. They proposed a novel heuristic to reduce communication. Alabandi and Burtscher [29] presented an approach that increases the parallelism, but not the quality of coloring. The technique is observed to yield a 3.4 times speed-up. The CUDA version on a Titan V is observed to be 2.9 times faster. Giannoula et al. [30] proposed the ColorTM algorithm that detects inconsistencies of coloring between vertices. The ColorTM algorithm has a speculative synchronization to minimize costs and improve parallelism.

Thadani et al. [31] proposed applications of graph coloring for team building, scheduling, and network analysis problems. Kole et al. [32] compared ACO, simulated annealing, and quantum annealing for the GCP. Datta et al. [33] proposed a new graph coloring algorithm for scheduling examinations for universities and colleges across. Ananda et al. [34] developed a tabu search graph coloring algorithm for the scheduling problem of nurses in a hospital. Goudet et al. [35] developed a new framework of deep neural networks with heuristics for the GCP. The algorithm was able to obtain competitive results.

Barenboim and Elkin [36] have published a book about distributed GCP and its fundamental theories and recent studies. Xu et al. [37] proposed a distribution evolutionary algorithm based on a population of probability model (DEA-PPM). The DEA-PPM algorithm employs a novel probability model to search the distribution space. A tabu search method is used in this study. The algorithm leads to a nice balance between exploration and exploitation. De Freitas et al. [38] consider coloring problems with distance constraints as weighted edges and modeling them as distance geometry problems. The authors proposed new variations of vertex coloring problems in graphs. Seker et al. [39] studied the NP-Hard selective GCP. The authors proposed an algorithm to generate perfect graphs and produce a collection of instances. The experiments demonstrate that their proposed solution improves the solvability of the problem.

Despite the substantial progress made in the field of graph coloring, there remain several intriguing research gaps that warrant further exploration and investigation. Large instances of the GCP are still challenging and need robust approximation algorithms. Dynamic graph coloring, multi-objective graph coloring (minimizing both chromatic number and execution time), parallel and distributed graph coloring to improve the efficiency of graph coloring using modern advanced computing platforms, machine learning integration to this optimization problem, robustness and resilience, and real-world applications, bridging the gap between theoretical advancements and practical applications, are crucial issues. Research that focuses on tailoring graph coloring algorithms to specific real-world problems, such as scheduling, resource allocation, or frequency assignment, is essential. Addressing these research gaps will not only contribute to a deeper understanding of the GCP but also enhance the practical applicability of graph coloring algorithms across various domains.

To our knowledge, our proposed algorithm is the first study to develop a parallel ensemble metaheuristic algorithm for the GCP using state-of-the-art metaheuristics, HHO, ABC, and TLBO, and to demonstrate its effectiveness on large graph instances from the well-known DIMACS benchmark.

3. PEM-Color Framework

This section explains our proposed parallel ensemble metaheuristic algorithm for the GCP (PEM-Color) and provides brief information about the metaheuristics (HHO, TLBO, ABC) and the local search technique (TabuCol) used for the development of the PEM-Color algorithm. The ensemble algorithms can leverage the strengths of various metaheuristic algorithms and compensate for their weaknesses; the ensemble algorithms can provide more accurate and robust solutions for the GCP. The selected recent metaheuristics are successfully applied to the solution of the GCP. This approach, which uses high-performance parallel computation techniques, is applied for the first time in this field. The PEM-Color algorithm is an island-parallel ensemble algorithm that distributes the optimization process of metaheuristics

(HHO, ABC, and TLBO) evenly among the available processors, with one processor allocated to each metaheuristic. For example, if there are 64 processors, one processor serves as the master node, and the remaining 63 processors are allocated to the three different metaheuristics (21 processors for each metaheuristic). The algorithm also uses the TabuCol algorithm as a local search algorithm during the exploitation phase of each candidate solution in the population of the metaheuristics. The initial individuals of each population are diversified to provide a better exploration possibility, and the population size is set to 20 for each metaheuristic. The PEM-Color algorithm is implemented using MPI libraries, and communication between processors is minimized to provide scalability. For synchronization purposes, the slave nodes send their best solution in the population to the master node only at the end of the generations. The master node then reports the overall best solution. The implementation and combination of the metaheuristics are explained in the following subsections. The flowchart and pseudocode of the parallel PEM-Color algorithm are provided in Figure 3 and Algorithm 1, respectively.

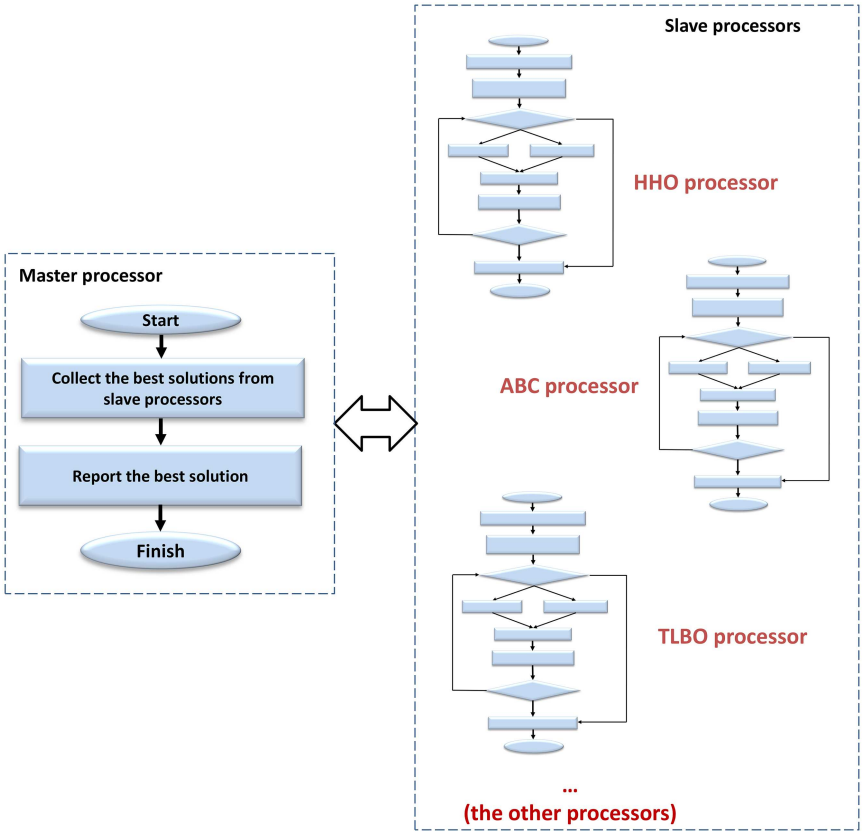
Algorithm 1: Revised Pseudocode of the Parallel Ensemble PEMColorAlgorithm

Input: Graph $G(V, E)$, number of slaves S, number of generations T
if node is **master then**
    Initialize $Best_{global} \leftarrow \emptyset$
    for $s = 1$ to S **do**
        Receive $Best_s$ from slave s
        if $Best_{global}$ is empty **or** $fitness(Best_s)$ is better **then**
            $Best_{global} \leftarrow Best_s$
        end if
    end for
    Output: $Best_{global}$
end if
if node is **slave then**
    Generate initial population P
    Evaluate fitness of all individuals in P
    for $t = 1$ to T
        Apply metaheuristic operator (HHO/ABC/TLBO)
        Update population P
        Apply local search using TabuCol to best individual
        Update best solution $Best_{local}$
    end for
    Send $Best_{local}$ to master
end if

3.1. Parameter tuning of the metaheuristics

The parameters of the metaheuristics are randomly selected within the defined ranges, making the parameter setting at each processor unique. Therefore, we provide a comprehensive method depending on the type of graphs that we paint. Various settings

Figure 3
The parallel computation framework of the PEM-Color algorithm. Master and slave nodes and how they interact are presented in this drawing



are available when parallel versions of the ABC, HHO, and TLBO are executed concurrently.

The ABC algorithm exhibits simplicity and adaptability in its parameter settings. Key parameters include the colony size, representing the number of artificial bees, and the maximum cycle count, determining the algorithm's convergence criteria. Additionally, ABC features exploration/exploitation factors, regulating the exploration–exploitation trade-off. Fine-tuning these factors influences the balance between global and local search capabilities. While the algorithm is relatively robust to parameter changes, employed bee and onlooker bee ratios are adapted to suit specific graph characteristics. The effectiveness of the ABC algorithm stems from its ability to strike a balance through minimal yet impactful parameter adjustments.

The key parameters of the HHO algorithm include the population size, representing the number of hawks in the search space, and the convergence factor, controlling the algorithm's termination criteria. Additionally, HHO employs the exploration factor to strike a balance between global and local search capabilities, enhancing adaptability across different problem landscapes. The attack strategy parameter guides the hawks' hunting behavior, influencing the optimization process. Our parallel algorithm can fine-tune these parameters to suit specific graph coloring optimization challenges, allowing HHO to efficiently converge toward optimal solutions while maintaining robustness across diverse problem domains.

TLBO is renowned for its simplicity, requiring no manual parameter tuning. This metaheuristic algorithm relies on a collaborative teaching and learning approach inspired by the classroom environment, making it inherently parameter-free. TLBO's self-adaptive nature enhances its appeal, eliminating the need for user intervention in parameter adjustments.

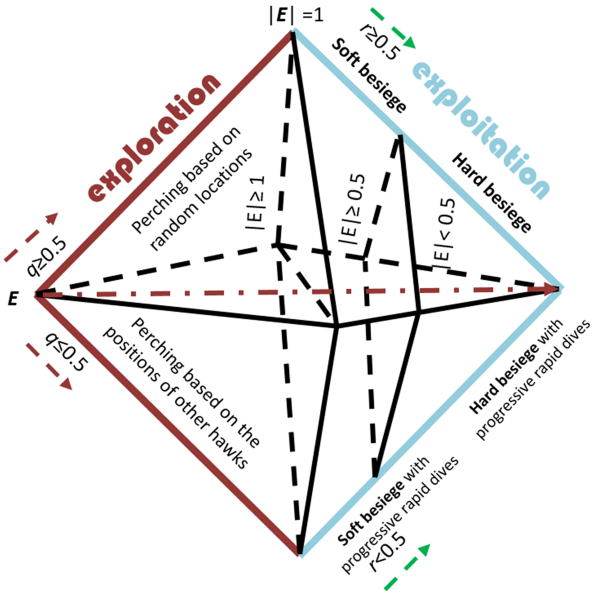
3.2. Harris Hawk Optimization metaheuristic for the GCP

HHO is one of the metaheuristics of the PEM-Color algorithm. The HHO metaheuristic simulates the hawks that attack prey from various directions synchronously [18, 40, 41]. These birds use some patterns to exhaust their prey. The HHO is gradient-free and inspired by the attacking strategies of the hawks. Figure 4 shows the steps of the HHO according to the energy level, E, and the parameter, q. In Algorithm 2, the details of the HHO algorithm that uses TabuCol as a local search algorithm are presented.

In their quest to find prey, the hawks employ a perching strategy, often perched on trees while observing the hunting field. The perching behavior is guided by the positions of other hawks and the prey, but this guidance only occurs when the value of q is less than 0.5. Otherwise, the hawks resort to random perching. This study introduces two exploration operators. The first one, exploration_1, involves selecting two hawks, hawk_1 and hawk_2, and transferring a subset of features from hawk_1 to hawk_2. The second operator, exploration_2, copies selected features from the best-performing hawk to the current hawk. Subsequently, the newly generated candidate is incorporated into the population. The determination between the exploration and exploitation phases relies on the prey's escaping energy value (E), which follows a decreasing trend as outlined below:

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{1}$$

Figure 4
Harris Hawk (HH) metaheuristic hunting strategy summary. The HH metaheuristic exhibits distinct stages that depend on the energy levels of the hawks. These stages correspond to various behaviors and strategies adopted by the hawks, which are influenced by their energy conditions



Algorithm 2: Pseudocode of the HHO algorithm

1: N: is the population of hawks
2: T: # of iterations
Initialize $X_i(i = 1, 2, \ldots, N)$
3: **while** ($i <= T$) **do**
4: Calculate the fitness values
5: Update the hawks into the population
6: Set $\boldsymbol{X_{rabbit}}$ as rabbit's location
7: **for** (each hawk (X_i)) **do**
8: Update E_0
9: Update the value of E
10: **if** ($|E| > 1$)
11: Perform Exploration
12: **end if**
13: **if** ($|E| < 1$) **then**
14: Perform Exploitation
15 **end if**
16: **end for**
17: TabuCol(hawk(X_i));
18: **end while**
19: **Return**$\boldsymbol{X_{rabbit}}$ // the best solution

In the context of the algorithm, E_0 represents the initial energy, and T denotes the total number of iterations. The initial energy E_0 lies within the range of (-1, 1) and gradually increases from 0 to 1 as the prey becomes stronger. Conversely, if E_0 decreases from 0 to -1, it indicates a decline in the prey's mobility. As the number of iterations progresses, the value of E decreases. When $E \geq 1$, the HHO algorithm leans toward exploration; otherwise, it focuses on exploitation.

The hawks employ a besieging strategy around the prey, contingent upon the rabbit's energy level. If $E \geq 0.5$, a soft besiege is executed; otherwise, a hard besiege is employed. In instances

where both $r \geq 0.5$ and $E \geq 0.5$, signifying a good energy level in the rabbit, it can execute random jumps to elude the hawks (where r is a random value). Meanwhile, the hawks can encircle the prey and execute surprise jumps. These surprise jumps are initiated by generating a random number (J), and the features of the rabbit are then copied into the current solution. Prey performs random jumps if $r \geq 0.5$ and $E < 0.5$. The position of the hawk changes as described by the following equation:

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (2)$$

At iteration t, $\Delta X(t)$ represents the disparity between the hawk and the current rabbit. In this context, a specific dimension of the prey is copied. To fend off attacks, the rabbit can nullify them when $r < 0.5$ and $E \geq 0.5$. A novel operator, characterized by a high perturbation value, has been introduced to facilitate this action. Depending on the prey's energy level, a subset of features is extracted and then copied to the most recent hawk.

In scenarios where $E < 0.5$ and $r < 0.5$, the hawks are presumed to initiate attacks to minimize the distance. To achieve this, certain features are selected from the prey and transferred to a randomly chosen hawk. The number of features chosen is kept minimal to avoid excessive perturbations. For a detailed implementation, refer to the pseudocode of the HHO metaheuristic presented in Algorithm 2.

3.3. Artificial Bee Colony Optimization metaheuristic for the GCP

The ABC metaheuristic is inspired by the behavior of bee colonies [19]. The ABC metaheuristic makes use of exploring and exploiting the resources of food (solutions). The hive has three types of bees: employed, observer, and scout. The employed search for food sources and discuss their findings by dancing. When a worker has finished collecting nectar, it transforms into a scout and searches for alternative sources. Onlookers observe how the hired bees dance and select food. Scouts look for new sources. In the first step, the initial population is produced randomly. This optimization procedure is repeated with a new group of bees. A forager begins as an unemployed bee without any knowledge. A regular bee can act as a scout, exploring the solution space or observing the other bees, looking for new food sources. The bee gets the food and collects the nectar in the hive. Scout bees are randomly deployed to search the issue space during the initial phase of each iteration and return to discuss their findings with other bees. Onlookers choose the top candidates for exploitation. The employed flies are applied to the examined solutions, and the exploitation begins according to the parameters of the ABC. After the iterations are completed, the best result is provided as the solution. 50% of the population is the number of onlooker bees, the other is the number of employed bees, and there is a single scout bee in the hive population. The pseudocode of the metaheuristic is given in Algorithm 3.

Algorithm 3: Pseudocode of the ABC algorithm

1: **While** (i++ < #max_iterations) **do** Scouts forage for food();
2: Scouts return and dance();
3: Onlookers evaluate the sources();
4: Checking visited resources();
5: Choose the best resources();
6: Employed bees go to sources();
7: Collect the food ();
8: **end while**

3.4. Teaching–Learning-Based Optimization metaheuristic for the GCP

The learning environment of the students inspires the TLBO in a classroom [20, 26]. The knowledge of individuals is improved by the teachers and classmates. Teaching and learning are the two phases of this metaheuristic. There are no specific parameters in the TLBO algorithm. The iterations of the TLBO continue until the minimum chromatic k value is obtained or the number of generations is finished. Algorithm 4 shows the pseudocode of the TLBO metaheuristic. Each gene in the chromosome is a vertex in the solution structure of the GCP. The partition crossover operator used in this algorithm generates new diversified individuals. The initial population is generated using random solutions.

Algorithm 4: Pseudocode of the TLBO algorithm

1: **Input:** G is a graph, k is the number of colors to be used,
2: P is the population.
3: **Output:** The best solution, s^*, obtained by the algorithm.
4: Generate_initial_population(); Calculate_fitness_values(P);
5: **while** $i++ < \#generations$ **do**
6: $p_1, p_2 \leftarrow P$;
7: // learning phase
8: $s_0 \leftarrow$ Crossover (p_1, p_2);
9: $s_0 \leftarrow$ TabuCol(s_0);
10: Insert_into_population($S_0, S_1, ..., S_{|P|}$)
11: **end while**
12: ***return*** the_best_solution(s^*);

3.5. Tabu Search algorithm, TabuCol

The TabuCol [42] is an efficient local search algorithm proposed in 1987. It starts with an initial solution (s) and tries to improve this solution iteratively. The neighbors of the solution s are defined as $N(s)$. Candidate solutions are generated in $N(s)$. The search is directed to s^* when $f(s^*)$ has a better value than $f(s)$. The stopping condition of TabuCol is when $f^* = 0$. A random node x is selected among the neighbors, and assuming $x \in V_i$, a random color $i \neq j$ is selected, and s' is obtained from $s = (V_1, ..., V_k)$ as given below:

$$V'_j = V_j \cup \{x\}; V'_i = V_i \setminus \{x\}; V'_r = V_r \ for \ r = l, ..., k; r \neq j, r \quad (3)$$

Tabu list (T) is the most important component of TabuCol. Repeated moves are prevented using this list. For each move ($s \rightarrow s^*$), the opposite move ($s^* \rightarrow s$) is inserted to the end of T, and the oldest one is deleted. The TabuCol algorithm stops after $nbmax$ iterations. When a vertex x is moved from V_i to V_j, the pair (x, i) becomes tabu. An aspiration value $A(z)$ is used to follow the aspiration level, $z = f(s)$. If a move is a tabu, but $f(s')produces \leq A(z)$, then the status of the move is not tabu anymore. The tabu list size ($|T|$) is another crucial parameter. It is assumed to be seven during our experiments. With small tabu list sizes, cyclings may occur, whereas larger values may increase the computation time. Detailed pseudocode of TabuCol can be seen in Algorithm 5.

Algorithm 5: Pseudocode of the TabuCol algorithm

1: G=(V, E), k is the # of colors
2: T: the tabu list
3: *rep*: the number of neighboring solutions of s
4: *nbmax*: the maximum # of iterations
5: $V_1, ..., V_k$ are the set of colors.
6: Produce an initial solution s= ($V_1, ..., V_k$);
7: count = 0;
8: Produce a tabu list T.
9: **while** (f(s) > 0 and count < *nbmax*) **do**
    Produce *rep* many neighbors s_i of s with
    move $s \rightarrow s_i \notin T$ or f(s_i) ≤ A (f(s));
    s' = best neighbor;
    Update T;
    Move $s \leftarrow s_i$ and delete the oldest move from T;
    $s = s'$;
    count ++;
10: **end while**
11: If $f(s)$=zero, G is colored with k colors
12: Otherwise no solution with k colors is obtained

4. Results and Discussion

In this section, we give details of the experimental setup, graph instances of DIMACS, the results, and the comparison of the PEM-Color algorithm to the state-of-the-art algorithms in the literature.

The experiments were conducted on an AMD Opteron Processor 6376, featuring a Multiprocessing Non-Uniform Memory Access (NUMA) architecture. The processor boasts four sockets, each equipped with eight cores, and each core can support two threads. In our setup, each node is equipped with eight processors, which collectively share a Level Cache of 6 MB. To complement the processing power, the system is equipped with 64 GB of RAM, which is distributed evenly across the eight nodes. The frequency of the clock is 1400 MHz. The DIMACS problem instances are used in our experiments. The instances are flat, Leighton, random geometric, C2000.5 and C4000.5, and latin_square_10 graphs. In the library of these benchmark problems, (near)-optimal solutions are reported. In this study, we actively pursued the potential for discovering new and improved solutions for these problem instances. Our efforts focused on maximizing the number of instances where better solutions could be attained.

The population size of the metaheuristics used in our study is 20, and the number of generations is 1000. The PEM-Color algorithm collects the best results obtained from the studies. The C++ programming language is used while developing our code. The results are the average value of 20 executions. The depth of the TabuCol is selected as 100,000, and the size of the tabu list $|T|$ is 7. The other random values in each metaheuristic are generated according to the processor ID it is executed by. The seeding mechanism of the random values is initialized with these numbers, and it is observed to provide a good diversification and exploration/exploitation mechanism during the execution of the metaheuristics. Table 1 gives the details of the parameters used in our algorithms.

Table 1
The parameters of the metaheuristics used in PEM-Color algorithm

Parameter	Value
Population size	20
Initial population generation	Random
# Cores	64
# Generations	1000
E_0, initial energy of HHO	Range of (–1, 1)
Surprise jumps of HHO, J	Random
Number of onlooker bees, ABC	50% of the population
Number of scout bees	1
Number of teachers	1
Deployment of scout bees	Random
Tabu list (T) size	7
Depth of TabuCol	100,000

4.1. The performance of the PEM-Color algorithm on small problem instances

In this part, we report the performance of the PEM-Color algorithm (using 64 processors) on small instances of the DIMACS benchmark given in Table 2. #*vertices* is the number of vertices, #*edges* is the number of edges, k^* is the best-known chromatic number, k is the number of colors reported by the PEM-Color algorithm, #hits/20 is the number of times the PEM-Color algorithm hits the best solution in 20 executions, and time (s.) is the average execution time. The best-known solutions to this set's problem instances are obtained with practical execution times in our experiments. The average execution time is 0.939 s. DSJC250.5 problem instance (having 0.50 edge density) has the longest execution time at 10.2 s. R1000.1 had the largest number of vertices in this set. However, obtaining its best solution in 0.041 s was very easy. A direct correspondence is observed between the execution time and the density of edges in the graphs. If the density is higher and the number of vertices is large, the execution time significantly increases in our proposed algorithm. Consequently, the PEM-Color has been proven to be very successful on small DIMACS problem set graph instances.

4.2. The performance of the PEM-Color algorithm on large problem instances

Table 3 gives our results with harder graph problem instances from DIMACS using 64 processors. There are 24 problem instances in this set. The average numbers of vertices and edges are 845 and 334,456, respectively, whereas 320 vertices and 9296 edges are on average in easier problem instances. k^* is also a meaningful parameter to understand the difficulty level of the problem sets. As this value increases, we observe that the difficulty of the problem set becomes harder. k^* values are 22.6 and 83.5 for easier and harder problem instances, respectively. We could not find the best solutions for six of the problems reported in the literature. For 18 problem instances, we obtained the best results. We used 2040 colors to paint all these graphs in Table 2, and the total number of best solutions was 2003. We have used 37 more colors (1.84% more) than the total sum of the best solutions in all graphs. This value is a promising result in terms of solution quality. According to the NFL theorem, there will always be new opportunities for achieving improved solutions by employing novel metaheuristics. It is observed that different metaheuristics can be better on some classes of problems, while they are not good with the rest of the combinatorial problems. This is a current research topic. However, executing many metaheuristics simultaneously in a parallel computation environment can be considered to select the best among the others.

Table 2
The results of the PEM-Color algorithm for 19 small DIMACS problem instances

Instances	#vertices	#edges	k^*	k	#hits/runs	Time(s.)
DSJC125.1	125	736	5	5	20/20	0.026
DSJC125.5	125	3891	17	17	20/20	0.135
DSJC125.9	125	6961	44	44	20/20	0.011
DSJC250.1	250	3218	8	8	20/20	0.018
DSJC250.5	250	15668	28	28	20/20	10.214
DSJC250.9	250	27897	72	72	20/20	1.822
DSJR500.1	500	3555	12	12	20/20	0.023
school1	385	19095	14	14	20/20	0.098
school1_nsh	352	14612	14	14	20/20	0.070
flat300_20_0	300	21375	20	20	20/20	0.020
le450_15a	450	8168	15	15	20/20	0.187
le450_15b	450	8169	15	15	20/20	0.101
le450_25a	450	8260	25	25	20/20	0.007
le450_25b	450	8263	25	25	20/20	0.011
R1000.1	1000	14348	20	20	20/20	0.038
R125.1	125	209	5	5	20/20	0.001
R125.1c	125	7501	46	46	20/20	4.925
R125.5	125	3838	36	36	20/20	0.143
R250.1	250	867	8	8	20/20	0.003
Average	320	9296	22.58	22.58	20/20	0.939

Table 3
The results of the PEM-Color algorithm for 24 larger (harder) DIMACS problem instances

Instances	#vertices	#edges	k^*	k	#hits/runs	Time(s.)
C2000.5	2000	999,836	153	148	20/20	187.6
C4000.5	4000	4,000,268	280	272	0/20	468.2
latin_sqr_10	900	307,350	98	98	20/20	614.1
DSJC250.5	250	15,668	28	28	20/20	8.7
DSJC500.1	500	12,458	12	12	20/20	670.5
DSJC500.5	500	62,624	49	48	20/20	22.4
DSJC500.9	500	112,437	126	126	20/20	420.2
DSJC1000.1	1000	49,629	20	20	20/20	2.7
DSJC1000.5	1000	249,826	83	83	0/20	88.2
DSJC1000.9	1000	449,449	224	223	0/20	110.8
DSJR500.1c	500	121,275	85	85	20/20	1318.8
DSJR500.5	500	58,862	122	122	0/20	325.7
R250.5	250	14,849	65	65	20/20	38.4
R1000.1c	1000	485,090	98	98	20/20	7.7
R1000.5	1000	238,267	234	240	0/20	1240.1
flat300_26_0	300	21,633	26	26	20/20	15.7
flat300_28_0	300	21,695	28	28	20/20	956.5
flat1000_50_0	1000	245,000	50	50	20/20	846.3
flat1000_60_0	1000	245,830	60	60	20/20	1705.4
flat1000_76_0	1000	246,708	82	82	0/20	198.2
le450_15c	450	16,680	15	15	20/20	121.4
le450_15d	450	16,750	15	15	20/20	840.2
le450_25c	450	17,343	25	25	20/20	190.5
le450_25d	450	17,425	25	25	20/20	445.8
Average	845.8	334,456.3	83.5	85	15/20	451.8

Table 4
Comparative performance of the PEM-Color algorithm

Instances	PEM-Color	VSS	GenTS	ILS	Foopar	GTS	PCNS	HEA	AMA-COL	MMT	MIPS_CLR	Evocol	MACOL	P-TLBO-Color
C2000.5	153	–	–	–	–	153	165	–	–	–	162	151	148	153
C4000.5	301	–	–	–	–	280	–	–	–	–	301	–	272	301
latin_sqr_10	98	–	–	99	–	106	98	–	104	101	99	–	99	98
DSJC250.5	28	–	28	28	–	29	28	28	28	28	28	28	28	28
DSJC500.1	12	12	13	12	12	–	–	–	12	12	12	12	12	12
DSJC500.5	49	48	50	49	48	49	49	48	48	48	49	48	48	49
DSJC500.9	126	126	127	126	127	–	–	–	126	127	127	126	126	126
DSJC1000.1	20	20	21	–	20	–	–	20	20	20	21	20	20	21
DSJC1000.5	84	86	90	89	89	84	89	83	84	83	88	83	83	86
DSJC1000.9	226	224	226	–	226	–	–	224	224	224	228	224	223	226
DSJR500.1c	85	85	–	–	85	85	85	–	86	85	85	–	85	85
DSJR500.5	125	125	–	124	125	130	123	–	127	122	122	124	122	124
R250.5	65	–	66	–	66	69	65	–	–	65	65	–	65	66
R1000.1c	98	–	98	–	98	99	98	–	–	98	98	98	98	98
R1000.5	240	–	242	–	248	268	241	–	–	234	237	245	245	242
flat300_26_0	26	–	26	26	–	26	26	–	26	26	26	–	26	26
flat300_28_0	28	28	31	31	28	33	31	31	31	31	31	31	29	28
flat300_50_0	50	50	50	–	50	84	50	–	50	50	50	–	50	50
flat300_60_0	60	60	60	–	60	84	60	–	60	60	60	–	60	60
flat300_76_0	86	85	89	–	87	84	89	83	84	82	87	82	82	86
le450_15c	15	15	–	15	15	16	15	15	15	15	15	–	15	15
le450_15d	15	15	–	15	15	16	15	–	15	15	15	–	15	15
le450_25c	25	25	–	26	25	–	–	26	26	25	26	25	25	25
le450_25d	25	25	–	26	25	–	–	–	26	25	26	25	25	25

Setting the parameters of the metaheuristics is a critical area for the quality of the solutions [43]. In our study, we have used previously optimized values of the HHO, ABC, and TLBO metaheuristics. There will always be good opportunities in this field to obtain better solutions using better parameter settings. With the results we have obtained after the experiments, most of the best results in the literature are obtained.

4.3. Comparisons with state-of-the-art algorithms

We compare our solutions with 13 state-of-the-art algorithms in the literature. The algorithms are local search methodology Variable Space Search (VSS) [42, 44–46], Genetic and Tabu Search (GTS) algorithm [47, 48], Parallel Coloration Neighborhood Search (PCNS) algorithm [49], Hybrid Evolutionary Algorithms (HEA), Adaptive Memory Algorithm (AMACOL) [50–53], the memetic algorithm (MACOL) [54], and P-TLBO-Color [26].

Table 4 presents the results of the comparisons. They are obtained from the goals of other studies. When the results obtained are compared with other algorithms, it can be seen that the PEM-Color algorithm is among the state-of-the-art algorithms. PEM-Color achieved the best results in the literature for 18 out of 24 problems. With large graph instances, the algorithm has a promising performance. Adding more processors to the computation environment positively affects the performance of the PEM-Color algorithm.

We note that, at first sight, Evocol and MACOL algorithms performed equally well, reporting some of the best-known results on certain datasets. However, these two studies focus on result quality as the main metric and are allowed to execute the proposed algorithms for an unlimited amount of time. The corresponding articles report execution times up to 120 h, which may be a questionable setting for larger, more practical settings. Additionally, the differing objectives of these studies prevent a direct comparison with PEM-Color. Even for the hardest problem instances, PEM-Color finds results in less than 28 min, which promotes its merits for larger and more practical applications.

The innovative PEM-Color algorithm, initially designed for efficiently solving large instances of the GCP, presents a promising avenue for addressing other NP-complete problems, including the Traveling Salesman Problem (TSP) and Maximum Clique Problem (MCP). While the paper briefly mentions the applicability of PEM-Color to diverse problem domains, further exploration of its potential impact on TSP and MCP, supported by preliminary results or theoretical discussions, would elucidate the broader significance of this research. Ensemble algorithms, known for their effectiveness in optimizing NP-complete problems, are harnessed in PEM-Color to seamlessly integrate metaheuristics such as HHO, ABC, and TLBO. Leveraging MPI parallel computation libraries with 64 processors, the algorithm demonstrates remarkable improvements in execution times when applied to large graph instances from the DIMACS benchmark. The algorithm not only exhibits enhanced efficiency but also outperforms 13 state-of-the-art algorithms, showcasing its potential as a versatile and powerful tool for addressing a spectrum of combinatorial optimization challenges beyond GCP. Further exploration of PEM-Color's adaptability and performance in TSP and MCP could unlock new avenues for advancing solutions to these notoriously complex problems.

We had the opportunity to do more fitness evaluations in a parallel computing environment, and with different parameter settings, this parallel exploration and exploitation environment became even better. The possibility of escaping from local optima is also provided more easily in parallel computing environments. Finding the best solutions in computing environments with different populations becomes much more advantageous than working with a single CPU. Using metaheuristics that work with different mechanics creates a very robust calculation environment.

4.4. The scalability and speed-up analysis

The PEM-Color algorithm integrating HHO, ABC, and TLBO in an MPI computation environment exhibits promising speed-up and scalability. The utilization of multiple optimization algorithms concurrently at each node harnesses diverse search strategies, enhancing the algorithm's ability to explore and exploit the solution space effectively. MPI facilitates communication and coordination among nodes, enabling parallelization and efficient distribution of computational tasks. Due to the communication overhead of the MPI environment, not more than 5% of the execution times are observed.

The speed-up of the algorithm is evident as it leverages the parallel processing capabilities of MPI to concurrently execute optimization tasks, leading to reduced computation time compared to a sequential implementation. This acceleration is particularly pronounced for large-scale problem instances, where the parallelization of tasks offers significant time savings. It was possible to provide a linear speed-up in the number of fitness evaluations. 20,000 fitness evaluations are executed at each node, and when the number of processors is 64, within the same execution times, 1,280,000 fitness evaluations are possible.

The PEM-Color algorithm for the GCP, incorporating HHO, ABC, and TLBO in an MPI environment, demonstrates commendable speed-up and scalability. Its ability to efficiently color graphs on a parallel scale makes it a promising approach for addressing real-world instances of the GCP.

5. Conclusions and Future Work

Handling large-scale graphs in the context of the GCP remains a challenging task. Over the past three decades, metaheuristic algorithms have emerged as highly effective methods, delivering significant advances in this domain. Their ability to produce (near-)optimal solutions within practical time limits has made them particularly appealing to researchers. Among these, island-parallel ensemble metaheuristic algorithms have demonstrated even greater potential in improving solution quality and scalability.

In this study, we propose a novel and scalable island-parallel ensemble metaheuristic algorithm, named PEM-Color, specifically designed to optimize large GCP instances. Our primary objective is to enhance the efficiency and robustness of the solution process by leveraging three state-of-the-art metaheuristics: HHO, ABC, and TLBO. The core concept of ensemble learning lies in combining the strengths of diverse algorithms while mitigating their individual limitations. This hybridization aims to produce more accurate and stable results than any single metaheuristic could achieve alone. To our knowledge, this is among the first applications of such an ensemble-based, high-performance parallel strategy to the GCP.

Several persistent challenges are addressed in our approach, including the high computational cost of fitness evaluations, the risk of stagnation in local optima, early convergence issues, and the sensitivity of algorithm performance to parameter settings. To tackle these issues, we integrate parallel computing techniques, which not only significantly reduce total execution time but also contribute to the diversification and quality of the search process. However, designing an effective parallelization strategy for

metaheuristics is non-trivial. In our implementation, care was taken to balance computational loads across processors by equalizing iteration counts and execution durations for each metaheuristic. Scalability and speed-up were central considerations in this design.

Our experimental evaluations were conducted on a suite of large undirected graph instances from the well-established DIMACS benchmark, utilizing 64 parallel processors. The results demonstrate that our algorithm achieves near-linear speed-up, highlighting its strong scalability potential. Performance-wise, PEM-Color achieved the best-known solutions in 37 out of 43 benchmark instances. For the remaining instances, the deviations from the best-known results were marginal. Notably, while the MACOL algorithm remains the strongest competitor overall—particularly on the R1000.5 problem instances—PEM-Color ranks among the top three algorithms reported in the literature.

There are several promising directions for future work. The current implementation adopts an island-parallel strategy where each computational node is assigned a specific metaheuristic. Future research could explore more sophisticated parallelization techniques, such as population-based or hybrid parallelism, to further enhance scalability and performance. Additionally, a generalized parallel metaheuristic optimization framework could be developed to facilitate algorithm extensibility. Though such approaches may be more complex and computationally demanding, they have the potential to yield even higher-quality solutions.

Finally, it is worth considering the potential synergy between GAs and hardware acceleration platforms such as FPGAs (Field-Programmable Gate Arrays). GAs, like neural networks, operate iteratively and often exhibit localized computational bottlenecks. Recent studies on neural networks have shown that Single Instruction Multiple Data (SIMD)-based architectures like FPGAs can deliver super-scalar performance gains. Integrating GAs with FPGA technology could offer substantial benefits in terms of computational efficiency, making this a highly promising avenue for future investigation.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Tansel Dokeroglu: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Deniz Canturk:** Conceptualization, Methodology, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Visualization, Supervision, Project administration.

References

[1] Lewis, R. M. R. (2021). *Guide to graph colouring: Algorithms and applications*. Springer International Publishing. https://doi.org/10.1007/978-3-030-81054-2

[2] Nandal, P., Satyawali, A., Sachdeva, D., & Tomar, A. S. (2021). Graph coloring based scheduling algorithm to automatically generate college course timetable. In *2021 11th International Conference on Cloud Computing, Data Science and Engineering*, 210–214. https://doi.org/10.1109/Confluence51648.2021.9377151

[3] Manjula, T., Rajeswari, R., & Praveenkumar, T. R. (2022). Analytical modeling on the coloring of certain graphs for applications of air traffic and air scheduling management. *Aircraft Engineering and Aerospace Technology*, *94*(4), 623–632. https://doi.org/10.1108/AEAT-04-2021-0104

[4] Bravyi, S., Kliesch, A., Koenig, R., & Tang, E. (2022). Hybrid quantum-classical algorithms for approximate graph coloring. *Quantum*, *6*, 678. https://doi.org/10.22331/q-2022-03-30-678

[5] Andreace, F., Lechat, P., Dufresne, Y., & Chikhi, R. (2023). Comparing methods for constructing and representing human pangenome graphs. *Genome Biology*, *24*(1), 274. https://doi.org/10.1186/s13059-023-03098-2

[6] Alaidi, A. H., Soong Der, C. S., & Weng Leong, Y. (2021). Systematic review of enhancement of artificial bee colony algorithm using ant colony pheromone. *International Journal of Interactive Mobile Technologies*, *15*(16), 172–180. https://doi.org/10.3991/ijim.v15i16.24171

[7] Li, H., Gao, K., Duan, P.-Y., Li, J.-Q., & Zhang, L. (2023). An improved artificial bee colony algorithm with Q-learning for solving permutation flow-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *53*(5), 2684–2693. https://doi.org/10.1109/TSMC.2022.3219380

[8] Awadallah, M. A., Makhadmeh, S. N., Al-Betar, M. A., Dalbah, L. M., Al-Redhaei, A., Kouka, S., & Enshassi, O. S. (2025). Multi-objective ant colony optimization: Review. *Archives of Computational Methods in Engineering*, *32*(2), 995–1037. https://doi.org/10.1007/s11831-024-10178-4

[9] Hou, W., Xiong, Z., Wang, C., & Chen, H. (2022). Enhanced ant colony algorithm with communication mechanism for mobile robot path planning. *Robotics and Autonomous Systems*, *148*, 103949. https://doi.org/10.1016/j.robot.2021.103949

[10] Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *IEEE Access*, *10*, 10031–10061. https://doi.org/10.1109/ACCESS.2022.3142859

[11] Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review. *Archives of Computational Methods in Engineering*, *29*(5), 2531–2561. https://doi.org/10.1007/s11831-021-09694-4

[12] Kumar, A., Kumar, V., & Kumari, S. (2021). A graph coloring based framework for views construction in multi-view ensemble learning. In *2021 2nd International Conference on Secure Cyber Computing and Communications*, 84–89. https://doi.org/10.1109/ICSCCC51823.2021.9478138

[13] Zhou, Z.-H. (2021). Ensemble learning. In Z.-H. Zhou (Ed.), *Machine learning* (pp. 181–210). Springer. https://doi.org/10.1007/978-981-15-1967-3_8

[14] Singh, P., & Kottath, R. (2021). An ensemble approach to meta-heuristic algorithms: Comparative analysis and its applications. *Computers & Industrial Engineering*, *162*, 107739. https://doi.org/10.1016/j.cie.2021.107739

[15] Azizi, M., Talatahari, S., & Gandomi, A. H. (2023). Fire Hawk Optimizer: A novel metaheuristic algorithm. *Artificial Intelligence Review*, *56*(1), 287–363. https://doi.org/10.1007/s10462-022-10173-w

[16] Rashki, M., & Faes, M. G. R. (2023). No-free-lunch theorems for reliability analysis. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, *9*(3), 04023019. https://doi.org/10.1061/AJRUA6.RUENG-1015

[17] Sun, Y., Chu, S.-C., Hu, P., Watada, J., Si, M., & Pan, J.-S. (2022). Overview of parallel computing for meta-heuristic algorithms. *Journal of Network Intelligence*, *7*(3), 656–684.

[18] Shehab, M., Mashal, I., Momani, Z., Shambour, M. K. Y., AL-Badareen, A., Al-Dabet, S., . . . , & Abualigah, L. (2022). Harris hawks optimization algorithm: Variants and applications. *Archives of Computational Methods in Engineering*, *29*(7), 5579–5603. https://doi.org/10.1007/s11831-022-09780-1

[19] Kaya, E., Gorkemli, B., Akay, B., & Karaboga, D. (2022). A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. *Engineering Applications of Artificial Intelligence*, *115*, 105311. https://doi.org/10.1016/j.engappai.2022.105311

[20] Raghav, L. P., Kumar, R. S., Raju, D. K., & Singh, A. R. (2021). Optimal energy management of microgrids using quantum teaching learning based algorithm. *IEEE Transactions on Smart Grid*, *12*(6), 4834–4842. https://doi.org/10.1109/TSG.2021.3092283

[21] Klavzar, S., Kuziak, D., Tripodoro, J. C. V., & Yero, I. G. (2025). Coloring the vertices of a graph with mutual-visibility property. *Open Mathematics*, *23*(1), 20250193. https://doi.org/10.1515/math-2025-0193

[22] Assadi, S. (2025). Faster Vizing and near-Vizing edge coloring algorithms. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms*, 4861–4898. https://doi.org/10.1137/1.9781611978322.165

[23] Gaspers, S., & Lee, E. J. (2023). Faster graph coloring in polynomial space. *Algorithmica*, *85*(2), 584–609. https://doi.org/10.1007/s00453-022-01034-7

[24] Shimizu, K., & Mori, R. (2022). Exponential-time quantum algorithms for graph coloring problems. *Algorithmica*, *84*(12), 3603–3621. https://doi.org/10.1007/s00453-022-00976-2

[25] van Hoeve, W.-J. (2022). Graph coloring with decision diagrams. *Mathematical Programming*, *192*(1), 631–674. https://doi.org/10.1007/s10107-021-01662-x

[26] Dokeroglu, T., & Sevinc, E. (2021). Memetic Teaching–Lerning-Based Optimization algorithms for large graph coloring problems. *Engineering Applications of Artificial Intelligence*, *102*, 104282. https://doi.org/10.1016/j.engappai.2021.104282

[27] Hijazi, N. M., Faris, H., & Aljarah, I. (2021). A parallel metaheuristic approach for ensemble feature selection based on multi-core architectures. *Expert Systems with Applications*, *182*, 115290. https://doi.org/10.1016/j.eswa.2021.115290

[28] Bogle, I., Slota, G. M., Boman, E. G., Devine, K. D., & Rajamanickam, S. (2022). Parallel graph coloring algorithms for distributed GPU environments. *Parallel Computing*, *110*, 102896. https://doi.org/10.1016/j.parco.2022.102896

[29] Alabandi, G., & Burtscher, M. (2022). Improving the speed and quality of parallel graph coloring. *ACM Transactions on Parallel Computing*, *9*(3), 10. https://doi.org/10.1145/3543545

[30] Giannoula, C., Peppas, A., Goumas, G., & Koziris, N. (2023). High-performance and balanced parallel graph coloring on multicore platforms. *The Journal of Supercomputing*, *79*(6), 6373–6421. https://doi.org/10.1007/s11227-022-04894-6

[31] Thadani, S., Bagora, S., & Sharma, A. (2022). Applications of graph coloring in various fields. *Materials Today: Proceedings*, *66*, 3498–3501. https://doi.org/10.1016/j.matpr.2022.06.392

[32] Kole, A., De, D., & Pal, A. J. (2022). Solving graph coloring problem using ant colony optimization, simulated annealing and quantum annealing—A comparative study. In S. Bhattacharyya, G. Das, & S. De (Eds.), *Intelligence enabled research* (pp. 1–15). Springer. https://doi.org/10.1007/978-981-19-0489-9_1

[33] Datta, D., Guha, R., Banerjee, N., Adhikary, S., & Acharya, A. (2022). Examination scheduler using a linear-time graph coloring algorithm. *ICTACT Journal on Soft Computing*, *12*(4), 2678–2684. https://doi.org/10.21917/ijsc.2022.0382

[34] Ananda, R., Indra, Z., & Nasution, H. (2022). Application of graph coloring on nurse work scheduling at H. Adam Malik Hospital Medan using the Tabu Search algorithm. *ZERO: Jurnal Sains, Matematika, dan Terapan*, *6*(1), 1–8. https://doi.org/10.30829/zero.v6i1.12451

[35] Goudet, O., Grelier, C., & Hao, J.-K. (2022). A deep learning guided memetic framework for graph coloring problems. *Knowledge-Based Systems*, *258*, 109986. https://doi.org/10.1016/j.knosys.2022.109986

[36] Barenboim, L., & Elkin, M. (2013). *Distributed graph coloring: Fundamentals and recent developments*. Springer. https://doi.org/10.1007/978-3-031-02009-4

[37] Xu, Y., Cheng, H., Xu, N., Chen, Y., & Xie, C. (2023). A distribution evolutionary algorithm for the graph coloring problem. *Swarm and Evolutionary Computation*, *80*, 101324. https://doi.org/10.1016/j.swevo.2023.101324

[38] de Freitas, R., Dias, B., Maculan, N., & Szwarcfiter, J. (2021). On distance graph coloring problems. *International Transactions in Operational Research*, *28*(3), 1213–1241. https://doi.org/10.1111/itor.12626

[39] Seker, O., Ekim, T., & Taskin, Z. C. (2021). An exact cutting plane algorithm to solve the selective graph coloring problem in perfect graphs. *European Journal of Operational Research*, *291*(1), 67–83. https://doi.org/10.1016/j.ejor.2020.09.017

[40] Dokeroglu, T., & Sevinc, E. (2022). An island parallel Harris hawks optimization algorithm. *Neural Computing and Applications*, *34*(21), 18341–18368. https://doi.org/10.1007/s00521-022-07367-2

[41] Dokeroglu, T., Kucukyilmaz, T., & Talbi, E.-G. (2024). Hyper-heuristics: A survey and taxonomy. *Computers & Industrial Engineering*, *187*, 109815. https://doi.org/10.1016/j.cie.2023.109815

[42] Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., & Khammas, B. M. (2022). Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. *Heliyon*, *8*(5), e09399. https://doi.org/10.1016/j.heliyon.2022.e09399

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

[44] Dorne, R., & Hao, J.-K. (1999). Tabu search for graph coloring, T-Colorings and set T-Colorings. In S. Voß, S. Martello, I. H. Osman, & C. Roucairol (Eds.), *Meta-heuristics: Advances and trends in local search paradigms for optimization* (pp. 77–92). Springer. https://doi.org/10.1007/978-1-4615-5775-3_6

[45] Alicastro, M., Ferone, D., Festa, P., Fugaro, S., & Pastore, T. (2021). A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers & Operations Research*, *131*, 105272. https://doi.org/10.1016/j.cor.2021.105272

[46] Blochliger, I., & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, *35*(3), 960–975. https://doi.org/10.1016/j.cor.2006.05.014

[47] Zhu, E., Zhang, Y., Sun, H., Wei, Z., Pedrycz, W., Liu, C., & Xu, J. (2026). HyColor: An efficient heuristic algorithm for graph coloring. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *56*(1), 681–695. https://doi.org/10.1109/TSMC.2025.3629990

[48] Terci, G. S., & Boz, B. (2025). Coloring dynamic graphs with a similarity and pool-based evolutionary algorithm. *IEEE Access*, *13*, 38054–38075. https://doi.org/10.1109/ACCESS.2025.3546108

[49] Morgenstern, C. (1994). *Distributed coloration neighborhood search* (Report No. CONF-9408161). University of Michigan.

[50] Galinier, P., Hertz, A., & Zufferey, N. (2008). An adaptive memory algorithm for the *k*-coloring problem. *Discrete Applied Mathematics*, *156*(2), 267–279. https://doi.org/10.1016/j.dam.2006.07.017

[51] Malaguti, E., Monaci, M., & Toth, P. (2008). A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, *20*(2), 302–316. https://doi.org/10.1287/ijoc.1070.0245

[52] Funabiki, N., & Higashino, T. (2000). A minimal-state processing search algorithm for graph coloring problems. *IEICE Transactions on Fundamentals, E83-A*(*7*), 1420–1430.

[53] Porumbel, D. C., Hao, J.-K., & Kuntz, P. (2009). Diversity control and multi-parent recombination for evolutionary graph coloring algorithms. In *Evolutionary Computation in Combinatorial Optimization: 9th European Conference*, 121–132. https://doi.org/10.1007/978-3-642-01009-5_11

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

How to Cite: Dokeroglu, T., & Canturk, D. (2026). A New Metaheuristic Algorithm for Large Graph Coloring Problems. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS62028596