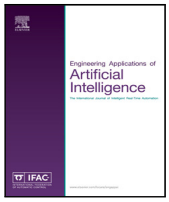
Engineering Applications of Artificial Intelligence 102 (2021) 104282

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Memetic Teaching–Learning-Based Optimization algorithms for large graph coloring problems

Tansel Dokeroglu *, Ender Sevinc

Ankara Science University, Computer Engineering Department, Ankara, Turkey

ARTICLE INFO

Keywords:
Graph coloring
Optimization
Memetic
TLBO
TabuCol
Parallel

ABSTRACT

The Graph Coloring Problem (GCP) can be simply defined as partitioning the vertices of a graph into independent sets while minimizing the number of colors used. So far, many approaches have been implemented to solve the GCP. However, researchers are still trying to solve this important NP-Hard problem much faster and with better results for large graphs. The Teaching-Learning-Based Optimization (TLBO) metaheuristic is a recent approach that has attracted the attention of many researchers due to its algorithm-specific parameterless concept and high performance. In this study, we propose a new memetic TLBO algorithm (TLBO-Color) combined with a robust tabu search algorithm to solve the GCP. A scalable parallel version of TLBO-Color is also developed for painting 43 benchmark DIMACS graphs with thousands of vertices and millions of edges. The optimization times of the TLBO-Color algorithm are very practical and the best results (for 33 of the graphs) or solutions with a few more colors are reported. On average, there are only 1.77% more colors compared to the best solutions. The obtained results confirm that the proposed algorithm is competitive with the state-of-the-art algorithms in the literature.

1. Introduction

Given an undirected graph $G = (V, E)$ where V and E are the sets of vertices and edges, respectively, the Graph Coloring Problem (GCP) can be defined as assigning the minimum number of colors to the set of vertices such that no two adjacent vertices have the same color (Jensen and Toft, 2011; Matula et al., 1972). The exact solution of the GCP is intractable, and it takes long periods for the solution of large instances of graphs with thousands of vertices and millions of edges (Hertz and de Werra, 1987; Garey and Johnson, 1976, 1979). There are many applications of the GCP in our daily lives such as timetabling (Burke et al., 2007), register allocation (de Werra et al., 1999; Briggs et al., 1994), scheduling (Gamache et al., 2007; Leighton, 1979), and cluster analysis (Hansen and Delattre, 1978),

The metaheuristic algorithms are elegant tools to obtain (near)-optimal solutions to large NP-Hard problems when it is impossible to find the solutions with exact algorithms (BoussaïD et al., 2013; Dokeroglu et al., 2019; Nayyar et al., 2018). According to No Free Lunch (NFL) theorem, there is a strong correlation between heuristic algorithms and the problems they solve (Wolpert and Macready, 1997). Any increased performance of a metaheuristic over one class of problems is offset by its performance over another class. These theorems are proved true in a geometric way and they are important to our study because we sought to experimentally determine the performance of the metaheuristic Teaching-Learning-Based Optimization (TLBO) on the GCP. According to the NFL theorem, there is always a possibility of finding a better metaheuristic. The TLBO is a recent metaheuristic inspired by the knowledge sharing practices of teachers and learners (Rao et al., 2011; Venkata Rao and Patel, 2012; Rao and Patel, 2012, 2013). Individuals in a classroom are trained by a teacher (the best individual in the population) and his/her classmates through interactions. This allows the individuals to improve their knowledge level. The TLBO is a parameterless metaheuristic that does not require specific values to be set during optimization. To obtain the best solutions, the general control parameters, population size and number of generations should be tuned either statically or dynamically at (runtime). However, apart from these, most metaheuristic algorithms have several parameters that need to be used (and well tuned) during the optimization process. The inertia weight, speed, social and cognitive parameters (Particle Swarm Optimization), crossover and mutation rate (Genetic Algorithms), number of scout/worker bees (Artificial Bee Colony), rate of pitch adaptation, rate of memory observation and number of improvisations (Harmony Search) are some of these values. Working with the best parameter setting is essential for metaheuristic algorithms to obtain (near)-optimal solutions (Nayyar and Nguyen, 2018). The TLBO does not have algorithm-specific parameters and therefore, it is not necessary to tune any parameter other than global ones.

* Corresponding author.
E-mail address: tansel@ceng.metu.edu.tr (T. Dokeroglu).

https://doi.org/10.1016/j.engappai.2021.104282
Received 17 September 2020; Received in revised form 24 April 2021; Accepted 25 April 2021
Available online 4 May 2021

The TLBO has reported impressive solutions to many classical combinatorial optimization problems. This achievement of the TLBO has attracted the attention of many researchers (Črepinšek et al., 2012). In this study, we have developed memetic TLBO algorithms for solving the GCP. Tabu search, a well-known local search algorithm from Fred Glover, is combined with TLBO (Glover and Laguna, 1998). The proposed memetic TLBO algorithm (TLBO-Color) trains individuals using crossover operators that do not require parameter settings. There is also a random selection mechanism to match the individuals, unlike classical techniques such as roulette wheel, tournament or truncation. After the recombination operators are applied and the candidate solutions are prepared, they are sent to the Tabu Search engine to minimize the number of colors so that the graphs can be painted with a minimum number of colors.

The execution time of the algorithms takes an enormous amount of time when computing the fitness values of large graphs. Some studies report days of computation time to paint large graphs with minimum number of colors (Lü and Hao, 2010). This is one of the main drawbacks of population-based algorithms, which prevents them from finding the optimal solutions in reasonable time. To reduce the execution time of the sequential TLBO-Color algorithm for large graph problem instances and evaluate more fitness values in less time, we also develop a parallel version of the TLBO-Color algorithm. The parallel TLBO-Color runs the Tabu Search on different processors and provides diversity with a scalable approach.

Developing parallel graph algorithms is a challenging task because it involves scalability and speed-up concerns. We can get our results in less time than its sequential versions and better results are observed by exploring more alternative solutions with parallel TLBO-Color algorithm. 43 DIMACS benchmark graph problem instances are optimized with the proposed TLBO-Color algorithms. The results are compared with those of state-of-the-art algorithms, and the TLBO-Color algorithm is found to be competitive. The best results are reported for 33 problems studied. For the remaining ten problems for which the TLBO-Color algorithm did not find the best solutions, the number of colors of the solutions is only a few more than the reported best solutions in the literature. Detailed results can be found in the experiments section of our paper.

Our contributions can be listed as follows: The TLBO is used for the first time in the literature to optimize the GCP. The efforts of tuning the parameters of a memetic algorithm is minimized. A master and slave MPI version of the TLBO-Color is developed for large graph problem instances. The execution times of the P-TLBO-Color are significantly reduced compared to other recent metaheuristic algorithms. It is observed that the parallel TLBO-Color algorithm is competitive with state-of-the-art algorithms. A mechanism is provided to prevent stagnation by searching the alternative regions of the search space with different initial solution points. This is developed using a parallel version of the proposed algorithm. We also show the potential of parallel metaheuristics over single-threaded metaheuristic algorithms in terms of solution quality and execution times.

Related studies are listed in the second section. The proposed TLBO-Color algorithm is presented in Section 3. Section 4 gives the details of the experimental results and compares the TLBO-Color algorithm with state-of-the-art algorithms. Concluding remarks and future work are given in the last section.

2. Related work

In this part of our study, we give information about memetic, tabu search and parallel algorithms proposed for solving the GCP in the literature. Leighton (1979) proposes the well-known algorithms Largest Saturation Degree heuristic (DSATUR) and the Recursive Largest First heuristic (RLF). These algorithms are the first greedy heuristics, and they are still used by many metaheuristic algorithms as first solution generators. San Segundo (2012) describes an exact algorithm based on the DSATUR algorithm. At each step of the algorithm, the DSATUR maximizes the saturation level to select a new vertex to color. Archetti et al. (2014) propose a branch-and-price algorithm. The pricing problem tries to find a set with minimum total weight.

Wu and Hao (2012) present EXTRACOL algorithm for coloring large graphs. The EXTRACOL preprocesses the graphs to extract large independent sets, and then uses a memetic algorithm to color the rest of the graph. Lü and Hao (2010) propose a memetic algorithm (MACOL) that integrates an adaptive multi-parent crossover operator and a distance and quality based replacement criterion for updating the population. The MACOL achieves competitive results with state-of-the-art algorithms. Markid et al. (2015) propose a Artificial Bee Colony (ABC) algorithm using tabu search. Experimental results verify the competitiveness of the algorithm on DIMACS benchmark problems. Brélaz (1979) describes new heuristic methods using degree and graph structure comparison. Zhou et al. (2016) deals with a universal solution method for grouping problems. The method combines reinforcement learning techniques with local search. Laguna and Martí (2001) provide a survey of heuristics and metaheuristics for the GCP. The authors also present a Greedy Randomized Adaptive Search Procedure (GRASP) for sparse graphs. The GRASP is competitive with Simulated Annealing and Tabu Search.

Mahmoudi and Lotfi (2015) propose a new approach based on Cuckoo Optimization Algorithm (COA). The operators of COA (addition, subtraction and multiplication) are redefined, and COA is provided with a means to solve the discrete nature of non-permutation. Blöchliger and Zufferey (2008) introduce feasible partial solutions and increase the size of the solution. The authors present a reactive tabu tenure that improves the performance in their algorithm FOO-PARTIALCOL. The proposed algorithm is an efficient and straightforward method for local coloring search.

Avanthay et al. (2003) propose an adaptation of the variable neighborhood search method. Porumbel et al. (2009) present a hybrid evolutionary algorithm (Evocol). An enhanced crossover operator collects the best color classes from multiple parents. Porumbel et al. (2010) introduce a clustering hypothesis according to which the high-quality solutions are not randomly distributed, but clustered within spheres of a given diameter. Chams et al. (1987) use thermodynamic simulations combined with other techniques. Cornaz et al. (2017) solve different versions of coloring problems formulated as Maximum Weight Stable Set Problems. Zhou et al. (2014) propose an exact algorithm that exploits the implicit constraints using propositional logic. Experimental results show that the algorithm outperforms other algorithms on many instances.

The tabu search (Tabucol) is proposed by Hertz and de Werra (1987) for the GCP. The TabuCol moves towards the minimal value of a function by visiting its neighbors. A tabu list is used to avoid being trapped in local optima. TabuCol decides whether vertices can be colored with a certain number of colors (Galinier and Hertz, 2006). Hybrid algorithms use TabuCol as a local search (and memetic algorithms). Researchers are still developing new versions of the Tabu search algorithm (Dorne and Hao, 1999; Fleurent and Ferland, 1996; Galinier and Hao, 1999). Wang et al. (2018) present a hybrid tabu search for the equitable coloring problem, which is a variant of GCP. The algorithm uses feasible and infeasible local search methods interchangeably. Recent applications of optimization techniques of other engineering problems can be found in studies of Khatir et al. (2020), Khatir and Wahab (2019), Tiachacht et al. (2018), Tran-Ngoc et al. (2020) and Tran-Ngoc et al. (2018).

There are also some parallel implementations developed for the solution of the GCP. The parallel metaheuristics are more powerful tools than their sequential versions (Harada and Alba, 2020; Dokeroglu and Cosar, 2016). Çatalyürek et al. (2012) present two different types of multi-threaded heuristic algorithms. The first algorithm is designed for shared memory systems. The second algorithm is for the non-conventional massively multi-threaded systems. Dorne and Hao (1998)

Fig. 1. The chromosome structure of the TLBO-Color algorithm for the example graph with ten vertices given in Fig. 2.

present a genetic local search algorithm that uses a tabu search algorithm. The algorithm improves the best-known results on some DIMACS benchmark problems. Jones and Plassmann (1993) present an asynchronous graph coloring heuristic that is well suited for parallel computers with distributed memory. The heuristic is scalable and produces colorings within three or four colors of known linear-time heuristics. Gebremedhin and Manne (2000) present a heuristic for parallel graph coloring with distributed memory. The algorithm yields an almost linear speed-up. The authors also present some heuristics implemented with OpenMP. The experiments are performed on a supercomputer with large graphs. Osama et al. (2019) implement parallel GPU graph coloring algorithms. Li et al. (2016) present a greedy algorithm on GPGPUs. Deveci et al. (2016) propose a novel edge-based algorithm suitable for GPUs.

As a result of our review of recent work, it is shown that the TLBO algorithm has successfully solved many combinatorial NP-Hard problems, but it has not yet been applied to the GCP. According to the NFL theorem, there is always a new opportunity to obtain better results by applying new metaheuristics to NP-Hard problems. Therefore, we decided to take this opportunity to apply TLBO and Tabu search algorithms to the GCP as a memetic computing optimization solution. We concluded that the success of parallel metaheuristic algorithms in recent years should also be studied on this problem. We concluded that it would be a good study to investigate metaheuristics that do not use any parameters and observe their performance.

3. Proposed algorithms

In this section, we present the details of our proposed algorithm, TLBO-Color, its parallel version (P-TLBO-Color) developed using Message Passing Interfaces (MPI), and TabuCol (the local search technique combined with TLBO).

3.1. TLBO-Color algorithm

The TLBO is a population-based algorithm that moves towards the global solution over generations/iterations. The population is a group of individuals (learners) who want to improve their knowledge level. The TLBO improves the knowledge level of individuals with the influence of a teacher or classmates. The TLBO has two main phases, Teacher and Learner. During the Teacher Phase, individuals use the teacher's knowledge state. In the Learner's phase, interactions take place between learners to share their information (Rao et al., 2011, 2012; Dokeroglu, 2015; Yu et al., 2016).

There is no specific parameter used in the phases of the TLBO-Color algorithm. The optimization continues until the number of generations is reached or the desired k coloring solution is obtained. The population size and the number of generations are global parameters of the metaheuristic algorithms, which should be well tuned to achieve the best results in practical times.

The chromosome structure of the TLBO-Color algorithm is given in Fig. 1. Each gene of the chromosome represents one vertex. Three colors are used in this chromosome to paint the graph given in Fig. 2. The graph has ten vertices, and there is no conflict between the colors of the vertices.

The crossover operator is used to diversify the search space. The operator creates new individuals from existing chromosomes. Mainly, there are two typical crossover operators used by the population-based algorithms for solving the GCP, namely assignment crossover and partition crossover (Dorne and Hao, 1998). The assignment crossover cannot pass the information of the parent chromosomes to the next generations. Therefore, the performance of the assignment crossover is worse than that of the partition crossover. In our study, we use the partition crossover operator because of its high performance. The operator operates on partitions of vertices and effectively transfers sets of colors from one chromosome to another.

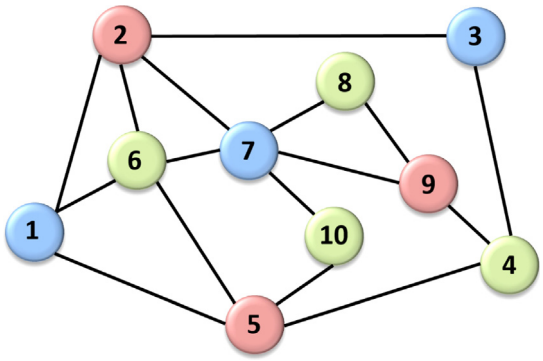


Fig. 2. An example graph with ten vertices (painted with three colors, red, blue, and green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The initial population of the TLBO-Color algorithm is generated with randomly selected partitions for each vertex in the graph. When the chromatic number k is chosen, each vertex is given a number up to k. When the optimization starts, the selected partitions for the vertices are updated and a better solution with minimum error value is searched through the generations.

After generating the new individuals, the TabuCol algorithm sends the candidate solutions to the TabuCol local search algorithm. This algorithm can be repeated many times. This is because restarting the TabuCol algorithm with a new individual and visiting different neighbors with different orders can provide better solutions (the TabuCol algorithm is not deterministic). Fig. 3 and Algorithm 1 show the flowchart and pseudocode of TLBO-Color, respectively.

3.2. Tabu search algorithm (TabuCol)

The TabuCol algorithm improves its solution starting from an initial solution (s) towards feasible solutions constructed with k colors. New solutions (neighbors of s) are denoted by $N(s)$. The search procedure generates candidate solutions in $N(s)$. The best neighbor (s^*) is compared to the solution s and the search process is directed to s^* if $f(s^*)$ is better than $f(s)$. A valid solution is a partition $s = (V_1, V_2, \ldots, V_k)$ of the set V of nodes into a number k of subsets. If $E(V_i)$ is the set of edges in G with endpoints in V_i, then the fitness function f is the number of edges and the endpoints are in the same V_i.

$$f(s) = \sum(|E(V_i)| \; : \; i = 1, 2, \ldots, k)$$

s is the coloring of the vertices with k colors if and only if the fitness function $f(s) = 0$. The best value of $f(s)$ is when $f^* = 0$. This is the stopping condition of TabuCol. The neighbor of the solution s is s' (another partition of the current solution into k subsets of nodes). A random node x is chosen among all neighbors in $E(V_1) \cup, \ldots, \cup E(V_k)$. Later, assuming $x \in V_i$, a random color $j \neq i$ is chosen and s' is obtained from $s = (V_1, \ldots, V_k)$ by:

$$V_j' = V_j \cup \{x\}; V_i' = V_i \backslash \{x\}; V_r' = V_r \; for \; r = l, \ldots, k; r \neq j, r$$

The most important feature of TabuCol is the set of forbidden moves, called the tabu list (T). The moves in T are not allowed to be recalled. This policy keeps the algorithm from returning to previously

Algorithm 1: The pseudocode of the Memetic TLBO-Color Algorithm.

Input: G is a graph, k is the number of colors to be used, P is the population.
Output: The best solution, s^*, obtained by the algorithm.
$P = S_1, ..., s_p \leftarrow$ Generate_initial_population();
for *(i=1,...,p)* **do**
  $s_i \leftarrow$ TabuCol(s_i); //execute TabuCol on individuals
$i = 0$;
while *($i++ <$ number_of_generations)* **do**
  // Choose two individuals as parents from the population
  $p_1, p_2 \leftarrow P$;
  // learning from either a teacher or classmates
  $s_0 \leftarrow$ Crossover (p_1, p_2);
  $s_0 \leftarrow$ TabuCol(s_0);
  if *($f(s_0) < f(s^*)$)* **then**
    $s^* = s_0$;
  if *(s^* is the k coloring solution)* **then**
    return; // finish the algorithm.
  Insert_into_population($S_0, S_1, ..., S_{|P|}$)
return the_best_solution(s^*);

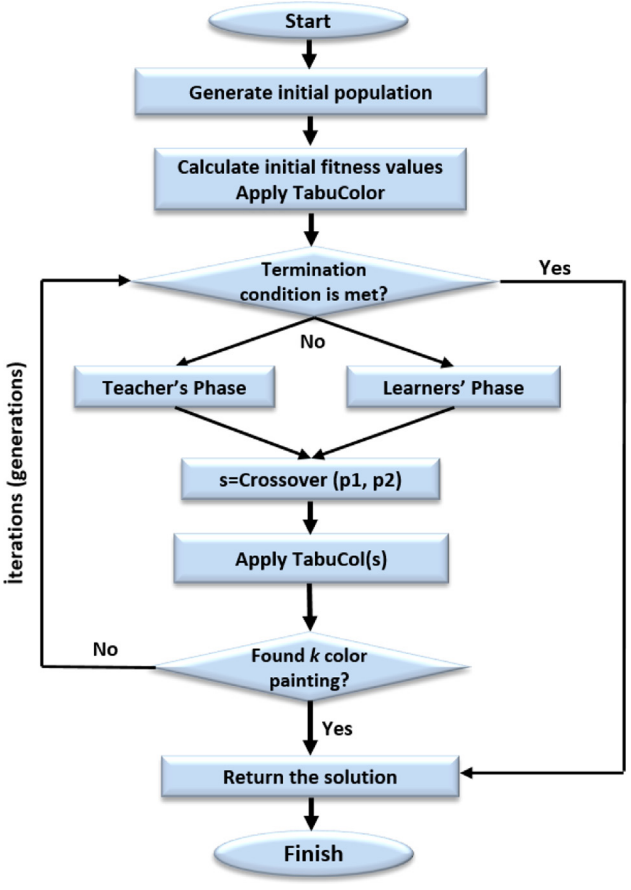


Fig. 3. The flowchart of the TLBO-Color algorithm.

visited neighbor solutions. A move is considered tabu for a certain number of iterations. Therefore, there is a list, T, and at each move ($s \rightarrow s^*$), the opposite move ($s^* \rightarrow s$) is added to the end of T, and the oldest move in T is removed. After generating a set of possible moves from s excluding the tabu moves, the best generated move is realized, and the tabu list T is consequently updated. The TabuCol algorithm stops its iterations after the maximum number of *nbmax* trials, or in the case that it finds a way to color the graph with a given number of k colors. After generating *rep* neighbors of s, we choose the best one and move towards that neighbor. When a node x is moved from V_i to V_j to obtain a new solution, the pair *(x, i)* becomes tabu: the node x cannot return to V_i for a given number of iterations.

The termination condition of the algorithm is the defined maximum number of iterations (*nbmax*) or reaching a feasible solution s such that $f(s) = f(s^*)$. If no valid solution is found, then the k-coloring of the solution $f(s) is O$.

An aspiration function *A(z)* is used to track the aspiration level of the current solutions, $z = f(s)$. If a move to a neighbor s' is tabu, but $f(s') yields \leq A(z)$, then the tabu status of that move is dropped and considered a valid move. $A(z) = (z - 1)$ is set for all values of z. Whenever an s' is generated with $f(s') \leq A(f(s))$, $A(f(s))$ is set to $f(s') - 1$. During the process of generating neighbors s' of s, a stage like s' (not in the tabu list) can be obtained with $f(s') < f(s)$. Instead of looking for as many neighbors as are generated, the process goes directly from s to s'. An important parameter of the TabuCol algorithm is the size of the tabu list ($|T|$) and is given as seven by Glover in his study. For smaller values of $|T|$, cycling may occur during optimization, and computation time may increase for larger values of $|T|$. The pseudocode of TabuCol is shown in Algorithm 2.

Algorithm 2: The TabuCol algorithm (Hertz and de Werra, 1987).

Input: G=(V, E), k is the number of colors to paint the graph G, T is the tabu list, *rep* is the number of neighboring solutions of s, *nbmax* is the maximum number of iterations.
Output: If $f(s)$=0, G is painted with k colors and $V_1, ..., V_k$ are the set of colors. Otherwise no valis solution is found with k colors.
Generate an initial random solution s= ($V_1, ..., V_k$);
count = 0;
Choose an arbitrary tabu list T.
while *($f(s) > 0$ and count < nbmax)* **do**
  Generate *rep* many neighbors s_i of s with move s $\rightarrow s_i \notin T$ or f(s_i) $\leq$ A (f(s));
  // As we get an s_i where f(s_i) < f(s) stop the generation.
  Let s' be the best neighbor produced;
  Update tabu list T;
  Present move $s \leftarrow s_i$ and remove the oldest move in the tabu list;
  $s = s'$;
  count ++ ;
report the result;

3.3. Parallel TLBO-Color algorithm, P-TLBO-color

The parallel TLBO-Color algorithm (P-TLBO-Color) was developed using the Message Passing Interface (MPI) libraries. The P-TLBO-Color algorithm uses a master–slave communication style between processors. After new individuals are generated, they are sent to slave processors to run the TabuCol algorithm. The TabuCol function of the TLBO-Color algorithm is the most time-consuming part of the algorithm, depending on the number of iterations. In the TLBO-Color algorithm, the TabuCol function is repeated many times while searching for the best/optimal solution. The P-TLBO-Color algorithm runs the TabuCol phase of the algorithm on slave nodes for speed-up and scalability.

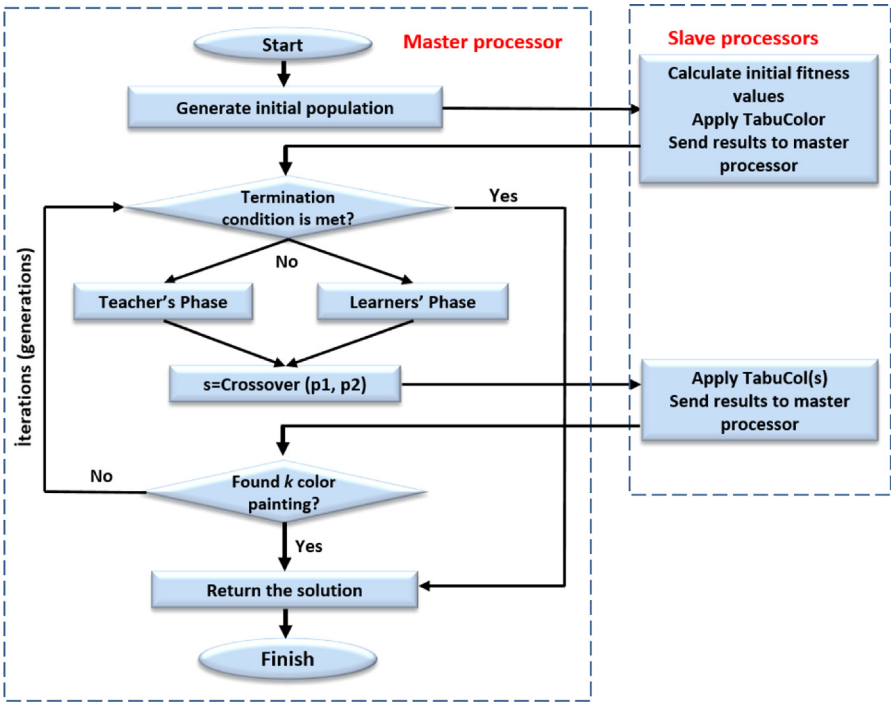


Fig. 4. The flowchart of the master–slave parallel TLBO-Color (P-TLBO-Color) algorithm.

Table 1
The results of the TLBO-Color algorithm on small size DIMACS problem instances. n_{edge} is the number of edges, $density$ is the ratio of the number of edges for the maximum possible edges, k^* is the best-known result in the literature.

Instance	n	n_{edge}	$density$	k^*	k-color	#hits/runs	#iter.	sec.
DSJC125.1	125	736	0.09	5	5	20/20	23 654	0.029
DSJC125.5	125	3 891	0.50	17	17	20/20	54 433	0.143
DSJC125.9	125	6 961	0.89	44	44	20/20	2 556	0.013
DSJC250.1	250	3 218	0.10	8	8	20/20	6 367	0.017
DSJC250.5	250	15 668	0.50	28	28	20/20	1 814 859	11.095
DSJC250.9	250	27 897	0.90	72	72	20/20	298 803	1.945
R125.1	125	209	0.03	5	5	20/20	31	0.001
R125.1c	125	7 501	0.97	46	46	20/20	2 200 132	5.183
R125.5	125	3 838	0.50	36	36	20/20	54 433	0.143
R250.1	250	867	0.03	8	8	20/20	69	0.002
DSJR500.1	500	3 555	0.03	12	12	20/20	383	0.025
R1000.1	1000	14 348	0.03	20	20	20/20	2 343	0.041
le450_15a	450	8 168	0.08	15	15	20/20	69 391	0.197
le450_15b	450	8 169	0.08	15	15	20/20	29 270	0.104
le450_25a	450	8 260	0.08	25	25	20/20	417	0.008
le450_25b	450	8 263	0.08	25	25	20/20	271	0.009
school1	385	19 095	0.26	14	14	20/20	15 327	0.102
school1_nsh	352	14 612	0.24	14	14	20/20	10 431	0.074
flat300_20_0	300	21 375	0.48	20	20	20/20	5 256	0.024

One of the biggest concerns of the P-TLBO-Color algorithm is to explore diverse search spaces of the problem instances and not revisit the same landscapes during the optimization process. The random number generators of each processor are assigned different seeding numbers depending on the Id number of the slave processors and the clock time. Therefore, each slave processor could search a diverse space.

After generating the initial population of the TLBO-Color algorithm at the master node, the individuals are sent to the slave processors, and the results are returned after the TabuCol function is completed. If any of the slaves finds the given k-color painting solution, then the algorithm terminates. Otherwise, it continues to minimize the number of colors in the candidate solutions until the number of generations is terminated or a complete solution is found. Comprehensive experiments are conducted to verify the performance of the P-TLBO-Color algorithm with large and dense graph problem instances. Fig. 4 shows the flowchart of the master–slave P-TLBO-Color algorithm.

4. Experimental setup and results

In this part, we present our experimental setup, benchmark graph instances, and the results of our experiments with TLBO-Color algorithms. We evaluate the TLBO-Color algorithm on 43 challenging DIMACS problem instances by comparing its performance with the best-known results of state-of-the-art algorithms in the literature.

The benchmark problem instances of DIMACS, flat graphs, Leighton graphs, random geometric graphs, giant random graphs (C2000.5 and C4000.5) and a Latin square graph (latin_square_10)) are included in our problem set to evaluate the performance of the proposed algorithms. State-of-the-art algorithms give the best solutions (optimal values known so far) for these problem instances. We try to obtain these results during our experiments, and when this is not possible, we report our closest results. In Tables 1 and 2, the details of the selected problem instances are presented. Most of the recent algorithms report

their results for these problem instances. Therefore, they provide a fair environment to evaluate the performance of the proposed algorithms.

In our experiments, we use an AMD Opteron Processor 6376 board. Its architecture is a Non-Uniform Memory Access (NUMA) design used in multiprocessing, where the memory access time depends on the memory location relative to the processor. It has four sockets, and each socket has eight cores. Two threads can run on one core. Therefore, we have 64 cpus to run 64 threads simultaneously. Each node has 8 cpus and one memory bank. These 8 cpus share 6 MB Last Level Cache (LLC). The system has 64 GB of RAM divided into 8 NUMA nodes. The thread ID starts from 0 to 7 for node_0, starts from 8 to 15 for node_1 and so on. The clock speed is 1400 MHz.

The population size and the number of generations are essential values for the success of population-based optimization algorithms. Tuning these parameters becomes an important issue to provide diversity while keeping the algorithm performance as high as possible. In our study, the TLBO-Color algorithm uses 20 individuals in its population. This is a reasonable population size recommended by previous studies (Dokeroglu, 2015). The number of generations is chosen to be 1000. However, due to the nature of our algorithm, the TLBO-Color algorithm terminates its optimization process whenever the TabuCol obtains a valid k-coloring. Basically, we run the TabuCol algorithm 100 times at each iteration for the sequential and parallel versions of the algorithm. The C++ programming language is used in the code development. Each problem instance is solved 20 times. The size of the tabu list $|T|$ has a significant impact on the performance of the TabuCol algorithm. We set the size of ($|T|$) to seven in our experiments, as recommended by Glover. The depth of the TabuCol algorithm is set to 100,000.

4.1. Experiments with small problem instances

First, we observe the performance of our sequential TLBO-Color algorithm on a single processor with DIMACS problem instances of small size. Detailed information about 19 different DIMACS problem instances and the results of the experiments are given in Table 1. n is the number of vertices (nodes), n_{edge} is the number of edges, $density$ is the ratio of the number of edges to the maximum possible edges, k^* is the best known result in the literature, k-color is the minimum number of colors obtained by the TLBO-Color algorithm, #hits/20 is the number of times the algorithm obtained the best solution in 20 executions, #total iter. is the average number of iterations to reach the given color k, and time (s) is the average execution time of the algorithm in 20 executions. Sixteen of the given problems are solved under one second. DSJC250.5 has the longest execution time. It is observed that as the density of graphs increases, the optimization time (execution time) also increases. The TLBO-Color has been shown to be very successful on relatively small graph instances with up to 1000 nodes (up to 30000 edges at low density).

4.2. The effect of increasing the number of processors for the P-TLBO-Color algorithm

One of the main concerns of our study is that if we add more processors and perform more fitness evaluations while increasing speed-up and scalability, can we get better solutions? When we add new computational power (processors) to the computational environment, more TabuCol iterations are executed. With a parallel machine with 64 cores, we can run 64 TabuCol algorithms simultaneously. Therefore, the possibility of getting better results increases significantly. To show the performance improvement in the number of hits, we perform some experiments on the DSJC500.1 graph problem instance by increasing the number of threads (processors).

The results are shown in Fig. 5 with increasing the number of threads from 2 to 128. The y-axis shows the number of best coloring results for the number of threads given in the x-axis. We run the experiments 20 times for each number of threads. The reader can easily observe how the number of hits improves as the number of threads is increased. With two threads it is possible to get only one solution in 20 trials, while with 128 threads we get 20 hits in 20 trials (runs). Fig. 5 indicates how often the P-TLBO-Color algorithm finds the best coloring results with increasing number of threads.

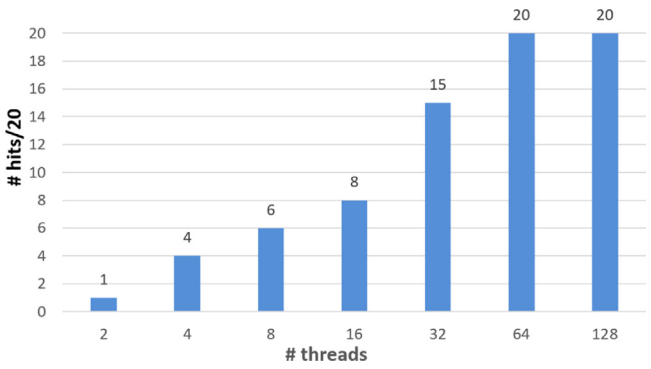


Fig. 5. The number of hits/runs with respect to the increasing number of threads.

4.3. Comparison with state-of-the-art algorithms on large graphs

To compare our solution quality with the state-of-the-art algorithms in the literature, we perform experiments on 24 challenging (large) graph problem instances given in Table 2. The problem instances have a higher number of vertices and edges than the graph instances given in Table 1. The density values of the given graphs are very high. Therefore, these properties of the graphs make them more difficult to solve. Total minimum number of colors for all these instances is 2002 with the best solutions. Our algorithm reports 2045 colors (2.14% more). For all problem instances our deviation is 1.77% in the average.

The state-of-the-art methods with which we compare the P-TLBO-Color algorithm are the local search methodology Variable Space Search (VSS) (Hertz et al., 2008), Generic Tabu Search (GenTS) algorithm (Dorne and Hao, 1999), Iterated Local Search (ILS) (Chiarandini et al., 2002), Foopar (Blöchliger and Zufferey, 2008), Genetic and Tabu Search algorithm (GTS) (Ferland and Fleurent, 1994), Parallel Coloration Neighborhood Search algorithm (PCNS) (Morgenstern, 1996), Hybrid Evolutionary Algorithms (HEA) (Galinier and Hao, 1999), Adaptive Memory Algorithm (AMACOL) (Galinier et al., 2008), Heuristic Algorithm (MMT) (Malaguti et al., 2008), MInimal-State Processing Search algorithm (MIPS _CLR) (Funabiki and Higashino, 2000), a hybrid evolutionary algorithm for the GCP (Evocol) (Porumbel et al., 2009), and the memetic algorithm (MACOL) (Lü and Hao, 2010). The results of the comparisons are shown in Table 3. The best three algorithms for each instance are shown in bold in Table 2. The performance of P-TLBO-Color is remarkable for large and dense graph instances such as C2000.5, C4000.5 and latin_sqr_10. It is possible to obtain (near)-optimal solutions in a few minutes, while the other population-based algorithms required hours of computation.

Six of the 24 problems in Table 2 are obtained by the P-TLBO-Color algorithm in less than sixty seconds. The optimization time of the remaining 24 problems is not more than thirty minutes. According to the NFL theory (Wolpert and Macready, 1997), metaheuristic algorithms may lose some performance on some problems while performing better on others. Therefore, the P-TLBO-Color algorithm shows good performance on most of the given problem instances and misses some of the best solutions of the benchmark problems. Apart from these problems, we can easily observe that the parallel TLBO-Color outperforms its sequential version.

Table 2
Detailed information about the large graphs used in the experiments and the obtained results by the P-TLBO-Color algorithm. Sixty-four processors are used. k^* is the best-known result in the literature. k is the result obtained by the P-TLBO-Color algorithm using 64 processors. The best results in the literature that are obtained by the P-TLBO-Color algorithm are given in bold letters.

Instances	n	n_{edge}	$density$	k^*	k	#hits/runs	#iter.	sec.
DSJC250.5	250	15,668	0.50	28	**28**	20/20	215 324	10.9
DSJC500.1	500	12,458	0.10	12	**12**	20/20	25 737 477	720.3
DSJC500.5	500	62,624	0.50	48	49	20/20	418 761	20.2
DSJC500.9	500	112,437	0.90	126	**126**	20/20	2 385 017	327.1
DSJC1000.1	1000	49,629	0.10	20	21	20/20	56 554	2.1
DSJC1000.5	1000	249,826	0.50	83	86	20/20	325 124	85.9
DSJC1000.9	1000	449,449	0.90	224	226	20/20	2 180 792	1060.1
DSJR500.1c	500	121,275	0.97	85	**85**	20/20	18 813 354	1428.1
DSJR500.5	500	58,862	0.47	122	124	20/20	3 951 974	295.5
R250.5	250	14,849	0.48	65	66	20/20	908 021	35.3
R1000.1c	1000	485,090	0.97	98	**98**	20/20	25 146	8.8
R1000.5	1000	238,267	0.48	234	242	20/20	6 031 381	1160.0
le450_15c	450	16,680	0.17	15	**15**	20/20	2 041 900	138.7
le450_15d	450	16,750	0.17	15	**15**	20/20	20 213 483	983.3
le450_25c	450	17,343	0.17	25	**25**	20/20	9 597 674	230.7
le450_25d	450	17,425	0.17	25	**25**	20/20	12 981 797	398.5
flat300_26_0	300	21,633	0.48	26	**26**	20/20	89 758	21.8
flat300_28_0	300	21,695	0.48	28	**28**	20/20	392 982	754.1
flat1000_50_0	1000	245,000	0.49	50	50	20/20	392 982	754.2
flat1000_60_0	1000	245,830	0.49	60	**60**	20/20	638 320	1866.5
flat1000_76_0	1000	246,708	0.49	82	86	20/20	948 204	170.2
C2000.5	2000	999,836	0.50	153	**153**	20/20	762 760	149.7
C4000.5	4000	4,000,268	0.50	280	301	20/20	98 326	380.8
latin_sqr_10	900	307,350	0.76	98	**98**	20/20	6 631 454	789.0

Table 3
Comparison of the P-TLBO-Color with state-of-the-art algorithms. k^* is the best-known result in the literature. The results of the algorithms that produce the best three results are given in bold letters.

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

4.4. The scalability and speed-up performance of the P-TLBO-Color algorithm

The most important criteria to evaluate the success of parallel P-TLBO-Color are its speed-up and scalability performances. In this part, we summarize its performance from these perspectives. As we have already explained, the parallel TLBO-Color performs as many local TabuCol searches as the number of threads running concurrently.

When investigating previous studies, it was observed that population-based algorithms that use the TabuCol algorithm as a local search technique require hours of optimization time. The parallel TLBO-Color can reduce this computation time and achieve better results due to its scalable parallel property and better exploration and exploitation properties.

The execution times of the solutions are much faster than the sequential population based algorithms like MACOL. The MACOL algorithm has better results in some instances, but the time required for the algorithm is more than ten hours, while it is not more than a few minutes for the P-TLBO-Color algorithm. Due to the excellent nature of fitness evaluation of individual chromosomes, TLBO algorithm is considered suitable for parallelization of fitness evaluation tasks.

4.5. Discussion

The GCP has many real-life applications (Ahmed, 2012). The coloring of countries in a map, where no two adjacent cities can have the same color, is one of the most famous applications of the GCP (Jensen and Toft, 2011). Another application of GCP is flight scheduling, where

the same aircraft cannot be assigned to both flights. In this problem, the graph corresponds to the flights. Assigning multiprocessors to different tasks can be designed using the GCP. Exam planning of a university (Malkawi et al., 2008), Sudoku, compiler optimization, register assignment, mobile radio frequency assignment, cluster analysis, scheduling (Sabar et al., 2012), testing printed circuit boards (Garey et al., 1976), Field Programmable Gate Array (FPGA) mapping (Wan and Perkowski, 1992), and Air Traffic Flow Management (Barnier and Brisset, 2004) are some of the interesting problems where the GCP is used to solve. Our proposed algorithm can be easily applied to these current problems.

5. Conclusions and future work

In this study, we proposed a novel robust memetic Teaching–Learning-Based Optimization (TLBO) algorithm for solving the GCP. The TLBO-Color, is an algorithm-specific parameterless metaheuristic. A scalable parallel master and slave version of the algorithm was also developed for solving challenging large graphs. The results of 33 problem instances are the best solutions in the literature. The parameters of TabuCol, tabu list size, aspiration value, number of iterations and search depth are tuned after extensive experiments to improve the solution quality of TLBO-Color.

For large graph coloring problems, the metaheuristics require a lot of computation time, which is a major drawback of the algorithms. Parallel metaheuristics are very efficient tools under such conditions. In this study, we verify this property of parallel metaheuristics by developing scalable algorithms. We experimentally show that the results of the P-TLBO-Color algorithm are better than those of its sequential version and competitive with state-of-the-art algorithms. Our algorithm produces either the best results or solutions with a few more colors than the best solutions.

In the future, we intend to develop a memetic Artificial Bee Colony (ABC) algorithm for solving the GCP. The TabuCol algorithm can be integrated with the ABC algorithm. There are also new metaheuristics such as Social Spider Web Optimization, Firefly, Dragonfly and Harris Hawk algorithms that can be applied to the GCP. There is still potential to improve the solution quality of the problems by adding more powerful parallel computations.

CRediT authorship contribution statement

Tansel Dokeroglu: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization/Data presentation, Supervision, Project administration. **Ender Sevinc:** Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Writing - review & editing, Visualization/Data presentation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmed, S., 2012. Applications of graph coloring in modern computer science. Int. J. Comput. Inf. Technol. 3 (2), 1–7.

Archetti, C., Bianchessi, N., Hertz, A., 2014. A branch-and-price algorithm for the robust graph coloring problem. Discrete Appl. Math. 165, 49–59.

Avanthay, C., Hertz, A., Zufferey, N., 2003. A variable neighborhood search for graph coloring. European J. Oper. Res. 151 (2), 379–388.

Barnier, N., Brisset, P., 2004. Graph coloring for air traffic flow management. Ann. Oper. Res. 130 (1), 163–178.

Blöchliger, I., Zufferey, N., 2008. A graph coloring heuristic using partial solutions and a reactive tabu scheme. Comput. Oper. Res. 35 (3), 960–975.

BoussaïD, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Inform. Sci. 237, 82–117.

Brélaz, D., 1979. New methods to color the vertices of a graph. Commun. ACM 22 (4), 251–256.

Briggs, P., Cooper, K.D., Torczon, L., 1994. Improvements to graph coloring register allocation. ACM Trans. Program. Lang. Syst. (TOPLAS) 16 (3), 428–455.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European J. Oper. Res. 176 (1), 177–192.

Çatalyürek, Ü.V., Feo, J., Gebremedhin, A.H., Halappanavar, M., Pothen, A., 2012. Graph coloring algorithms for multi-core and massively multithreaded architectures. Parallel Comput. 38 (10–11), 576–594.

Chams, M., Hertz, A., De Werra, D., 1987. Some experiments with simulated annealing for coloring graphs. European J. Oper. Res. 32 (2), 260–266.

Chiarandini, M., Stützle, T., et al., 2002. An application of iterated local search to graph coloring problem. In: Proceedings of the Computational Symposium on Graph Coloring and Its Generalizations. pp. 112–125.

Cornaz, D., Furini, F., Malaguti, E., 2017. Solving vertex coloring problems as maximum weight stable set problems. Discrete Appl. Math. 217, 151–162.

Črepinšek, M., Liu, S.-H., Mernik, L., 2012. A note on teaching–learning-based optimization algorithm. Inform. Sci. 212, 79–93.

de Werra, D., Eisenbeis, C., Lelait, S., Marmol, B., 1999. On a graph-theoretical model for cyclic register allocation. Discrete Appl. Math. 93 (2–3), 191–203.

Deveci, M., Boman, E.G., Devine, K.D., Rajamanickam, S., 2016. Parallel graph coloring for manycore architectures. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, pp. 892–901.

Dokeroglu, T., 2015. Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem. Comput. Ind. Eng. 85, 86–101.

Dokeroglu, T., Cosar, A., 2016. A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. Eng. Appl. Artif. Intell. 52, 10–25.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A., 2019. A survey on new generation metaheuristic algorithms. Comput. Ind. Eng. 137, 106040.

Dorne, R., Hao, J.-K., 1998. A new genetic local search algorithm for graph coloring. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 745–754.

Dorne, R., Hao, J.-K., 1999. Tabu search for graph coloring, T-colorings and set T-colorings. In: Meta-Heuristics. Springer, pp. 77–92.

Ferland, J., Fleurent, C., 1994. Object-Oriented Implementation of Heuristic Search Methods for Graph Coloring, Maximum Clique and Satisfiability. Tech. Rep., Univ. of Michigan, Ann Arbor, MI (United States).

Fleurent, C., Ferland, J.A., 1996. Genetic and hybrid algorithms for graph coloring. Ann. Oper. Res. 63 (3), 437–461.

Funabiki, N., Higashino, T., 2000. A minimal-state processing search algorithm for graph coloring problems. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 83 (7), 1420–1430.

Galinier, P., Hao, J.-K., 1999. Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. 3 (4), 379–397.

Galinier, P., Hertz, A., 2006. A survey of local search methods for graph coloring. Comput. Oper. Res. 33 (9), 2547–2562.

Galinier, P., Hertz, A., Zufferey, N., 2008. An adaptive memory algorithm for the k-coloring problem. Discrete Appl. Math. 156 (2), 267–279.

Gamache, M., Hertz, A., Ouellet, J.O., 2007. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. Comput. Oper. Res. 34 (8), 2384–2395.

Garey, M.R., Johnson, D.S., 1976. The complexity of near-optimal graph coloring. J. ACM 23 (1), 43–49.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability, Vol. 174. freeman San Francisco.

Garey, M., Johnson, D., So, H., 1976. An application of graph coloring to printed circuit testing. IEEE Trans. Circuits Syst. 23 (10), 591–599.

Gebremedhin, A.H., Manne, F., 2000. Scalable parallel graph coloring algorithms. Concurrency, Pract. Exp. 12 (12), 1131–1146.

Glover, F., Laguna, M., 1998. Tabu search. In: Handbook of Combinatorial Optimization. Springer, pp. 2093–2229.

Hansen, P., Delattre, M., 1978. Complete-link cluster analysis by graph coloring. J. Amer. Statist. Assoc. 73 (362), 397–403.

Harada, T., Alba, E., 2020. Parallel genetic algorithms: A useful survey. ACM Comput. Surv. 53 (4), 1–39.

Hertz, A., Plumettaz, M., Zufferey, N., 2008. Variable space search for graph coloring. Discrete Appl. Math. 156 (13), 2551–2560.

Hertz, A., de Werra, D., 1987. Using tabu search techniques for graph coloring. Computing 39 (4), 345–351.

Jensen, T.R., Toft, B., 2011. Graph Coloring Problems, Vol. 39. John Wiley & Sons.

Jones, M.T., Plassmann, P.E., 1993. A parallel graph coloring heuristic. SIAM J. Sci. Comput. 14 (3), 654–669.

Khatir, S., Boutchicha, D., Le Thanh, C., Tran-Ngoc, H., Nguyen, T., Abdel-Wahab, M., 2020. Improved ANN technique combined with jaya algorithm for crack identification in plates using XIGA and experimental analysis. Theor. Appl. Fract. Mech. 107, 102554.

Khatir, S., Wahab, M.A., 2019. Fast simulations for solving fracture mechanics inverse problems using POD-RBF XIGA and jaya algorithm. Eng. Fract. Mech. 205, 285–300.

Laguna, M., Martí, R., 2001. A GRASP for coloring sparse graphs. Comput. Optim. Appl. 19 (2), 165–178.
Leighton, F.T., 1979. A graph coloring algorithm for large scheduling problems. J. Res. Natl. Bur. Stand. 84 (6), 489–506.
Li, P., Chen, X., Quan, Z., Fang, J., Su, H., Tang, T., Yang, C., 2016. High performance parallel graph coloring on gpgpus. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, pp. 845–854.
Lü, Z., Hao, J.-K., 2010. A memetic algorithm for graph coloring. European J. Oper. Res. 203 (1), 241–250.
Mahmoudi, S., Lotfi, S., 2015. Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem. Appl. Soft Comput. 33, 48–64.
Malaguti, E., Monaci, M., Toth, P., 2008. A metaheuristic approach for the vertex coloring problem. INFORMS J. Comput. 20 (2), 302–316.
Malkawi, M., Hassan, M.A.-H., Hassan, O.A.-H., 2008. A new exam scheduling algorithm using graph coloring.. Int. Arab J. Inf. Technol. (IAJIT) 5 (1).
Markid, H.Y., Dadaneh, B.Z., Moghaddam, M.E., 2015. A new tabucol embedded artificial bee colony based algorithm for graph coloring. In: 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE). IEEE, pp. 112–117.
Matula, D.W., Marble, G., Isaacson, J.D., 1972. Graph coloring algorithms. In: Graph Theory and Computing. Elsevier, pp. 109–122.
Morgenstern, C., 1996. Distributed coloration neighborhood search. Discrete Math. Theor. Comput. Sci. 26, 335–358.
Nayyar, A., Le, D.-N., Nguyen, N.G., 2018. Advances in Swarm Intelligence for Optimizing Problems in Computer Science. CRC Press.
Nayyar, A., Nguyen, N.G., 2018. Introduction to swarm intelligence. Adv. Swarm Intell. Optim. Probl. Comput. Sci. 53–78.
Osama, M., Truong, M., Yang, C., Buluç, A., Owens, J., 2019. Graph coloring on the GPU. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, pp. 231–240.
Porumbel, D.C., Hao, J.-K., Kuntz, P., 2009. Diversity control and multi-parent recombination for evolutionary graph coloring algorithms. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 121–132.
Porumbel, D.C., Hao, J.-K., Kuntz, P., 2010. A search space "cartography" for guiding graph coloring heuristics. Comput. Oper. Res. 37 (4), 769–778.
Rao, R., Patel, V., 2012. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. Int. J. Ind. Eng. Comput. 3 (4), 535–560.
Rao, R.V., Patel, V., 2013. Multi-objective optimization of two stage thermoelectric cooler using a modified teaching–learning-based optimization algorithm. Eng. Appl. Artif. Intell. 26 (1), 430–445.
Rao, R.V., Savsani, V., Balic, J., 2012. Teaching–learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems. Eng. Optim. 44 (12), 1447–1462.
Rao, R.V., Savsani, V.J., Vakharia, D., 2011. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aided Des. 43 (3), 303–315.
Sabar, N.R., Ayob, M., Qu, R., Kendall, G., 2012. A graph coloring constructive hyper-heuristic for examination timetabling problems. Appl. Intell. 37 (1), 1–11.
San Segundo, P., 2012. A new DSATUR-based algorithm for exact vertex coloring. Comput. Oper. Res. 39 (7), 1724–1733.
Tiachacht, S., Bouazzouni, A., Khatir, S., Wahab, M.A., Behtani, A., Capozucca, R., 2018. Damage assessment in structures using combination of a modified cornwell indicator and genetic algorithm. Eng. Struct. 177, 421–430.
Tran-Ngoc, H., He, L., Reynders, E., Khatir, S., Le-Xuan, T., De Roeck, G., Bui-Tien, T., Wahab, M.A., 2020. An efficient approach to model updating for a multispan railway bridge using orthogonal diagonalization combined with improved particle swarm optimization. J. Sound Vib. 476, 115315.
Tran-Ngoc, H., Khatir, S., De Roeck, G., Bui-Tien, T., Nguyen-Ngoc, L., Abdel Wahab, M., 2018. Model updating for Nam O bridge using particle swarm optimization algorithm and genetic algorithm. Sensors 18 (12), 4131.
Venkata Rao, R., Patel, V., 2012. Multi-objective optimization of combined brayton and inverse brayton cycles using advanced optimization algorithms. Eng. Optim. 44 (8), 965–983.
Wan, W., Perkowski, M.A., 1992. A new approach to the decomposition of incompletely specified multi-output functions based on graph coloring and local transformations and its application to FPGA mapping. In: Proceedings of the Conference on European Design Automation. pp. 230–235.
Wang, W., Hao, J.-K., Wu, Q., 2018. Tabu search with feasible and infeasible searches for equitable coloring. Eng. Appl. Artif. Intell. 71, 1–14.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Wu, Q., Hao, J.-K., 2012. Coloring large graphs based on independent set extraction. Comput. Oper. Res. 39 (2), 283–290.
Yu, K., Wang, X., Wang, Z., 2016. An improved teaching-learning-based optimization algorithm for numerical and engineering optimization problems. J. Intell. Manuf. 27 (4), 831–843.
Zhou, Y., Hao, J.-K., Duval, B., 2016. Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Syst. Appl. 64, 412–422.
Zhou, Z., Li, C.-M., Huang, C., Xu, R., 2014. An exact algorithm with learning for the graph coloring problem. Comput. Oper. Res. 51, 282–301.