ISSN: 2834-7706

Research Article

Journal of Mathematical Techniques and Computational Mathematics

A Memetic Algorithm Designed for Solving Graph Coloring Problems: A Round-Robin Sports Scheduling Case Study

Babak Javadi[1*], Hamed Habibnejad-Ledari[2], Nezam Mahdavi-Amiri[3] and Mohammadreza Abdali[1]

[1]*Department of Industrial Engineering, Faculty of Engineering, College of Farabi, University of Tehran, Iran*

[2]*School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*

[3]*Faculty of Mathematical Science, Sharif University of Technology, Tehran,Iran*

[*]Corresponding Author
Babak Javadi, Department of Industrial Engineering, Faculty of Engineering, College of Farabi, University of Tehran, Iran

Submitted: 2024,Mar 01; **Accepted:** 2024, Mar 20; **Published:** 2024,Mar 29

Citation: Javadi, B., Ledari, H. H., Amiri, N. M., Abdali, M. (2024). A Memetic Algorithm Designed for Solving Graph Coloring Problems: A Round-Robin Sports Scheduling Case Study. *J Math Techniques Comput Math, 3*(3), 01-13.

Abstract

The issue of graph coloring is concerned with assigning colors to each vertex of an undirected graph so that adjacent vertices are not assigned the same color. Due to its NP-hard nature, a variety of heuristics and metaheuristics have been developed to tackle this problem. One of these approaches is the memetic algorithm, which was introduced as a solution for this problem. Another example is using a metaheuristic approach to address the round-robin sports scheduling problem. The algorithm suggested for this task comprises three primary components: (1) a randomized danger heuristic algorithm, (2) tabu search, and (3) a genetic algorithm with adaptive multi-parent crossover. To assess the performance of this algorithm, a set of test problems from different benchmark graphs in two categories were selected and compared with nine effective heuristics from existing literature. The results show that the algorithm performs exceptionally well on benchmark graphs that are known to be challenging. Additionally, a case study was conducted to demonstrate the effectiveness of the proposed algorithm.

Keywords: Graph Coloring, Memetic Algorithm, Tabu Search, Genetic Algorithm, Round-Robin Sports Scheduling.

1. Introduction

The problem of graph coloring (GCP) is a renowned challenge in the branch of graph theory. It necessitates the assignment of colors to each vertex in an undirected graph G = (V, E) while ensuring that adjacent vertices do not share the same color. The objective of graph coloring is to minimize the number of colors required for a given graph G, which is denoted by the chromatic number. GCP has numerous applications in different fields such as map coloring, scheduling, timetabling, layout problems, register allocation, storage problems, and frequency assignments [1]. An intriguing use of graph coloring is in the scheduling of round-robin sports tournaments. In round-robin sports, there are n teams that need to compete against each other many times in predetermined rounds. There are two common types of round-robin sports, single round robins (SRRs) and double round robins (DRRs). In a single round-robin (SRR), each team competes against all other teams exactly once, which is equivalent to m =1. When m=equals 2, each team plays twice with every other team. DRRs are also called home-away matches. The problem of round-robin sports scheduling can be tackled using graph representations and graph coloring techniques. In this formulation, each match is considered a vertex, and two matches are adjacent if they cannot be scheduled in the same round. Each color represents a round, and each match is assigned to one round. For instance, a DRR problem with four teams is presented in Figure 1, where the only constraint considered is that they must compete once during each round. Several constraints can be taken into account in a round-robin sports problem. For instance, two teams may be prohibited from playing against each other in a particular round, or two teams may be required to play against each other in a specific round.

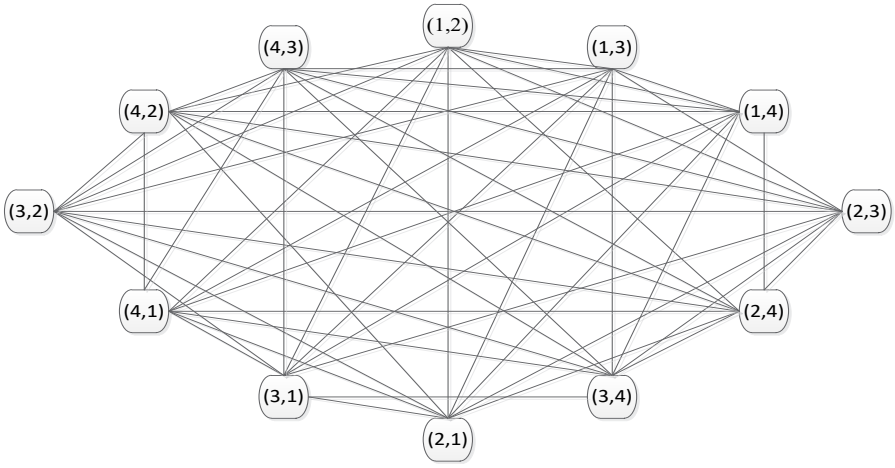

Figure 1: A graph for DRRs with several teams= 4

This constraint arises when two teams share the same stadium and cannot play home matches concurrently with other teams. To address this issue, an edge is added between the matches involving these two teams. Due to the NP-hard nature of the graph coloring problem (GCP), several heuristics and metaheuristic algorithms have been created to solve it. These include Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA), Tabu Search (TS), Neural Network Algorithm (NNA), DNA algorithm, and hybridizations of these algorithms with others. Additionally, the DSATUR algorithm is a popular algorithm that offers an upper limitation for the chromatic number in graph coloring problems. Within the case of the round-robin scheduling problem, there are several algorithms available in the literature, such as polygon and greedy algorithms. However, because of the intricacy of the problem, metaheuristic approaches such as those used in GCP, can also be applied to solve round-robin scheduling. As far as we know, a metaheuristic algorithm has not been utilized to address the round-robin sports scheduling problem. The current research suggests a memetic algorithm that comprises three primary stages. The first stage involves producing the initial population through a heuristic technique called the randomized danger algorithm, which utilizes a measure called dynamic vertex danger to assign rounds to matches. The generated individuals are then improved in the next phase using tabu search. In the third phase, new offspring are produced by a GA crossover called adaptive multi-parent, and the offspring are further optimized by tabu search. Finally, the best k-coloring is selected.

To evaluate the effectiveness of the proposed algorithm, we selected 41 test problems from two categories and compared our results with the most effective heuristics available in the literature. Additionally, we presented a real double-round-robin problem as a case study, where a football league with 16 teams and several constraints was considered. The rest of this paper is organized as follows. In Section 2, we present a literature review. Section 3 outlines our solution methodology and describes the randomized danger algorithm, tabu search, and GA crossover. Section 4 discusses the Taguchi method and other details proposed for tuning parameters. In Sections 5 and 6, we provide our experimental results and the case study, respectively. Finally, our conclusion is presented in Section 7.

2. Literature Review

A wide variety of algorithms can be found in the literature that can be used to address graph coloring problems (GCP) and related subproblems. Bui et al and Dowsland and Thompson proposed an ant colony optimization (ACO) algorithm as a solution to GCP, while Kole et al conducted a comparative analysis of three metaheuristic algorithms (ACO, simulated annealing (SA), and quantum annealing (QA)) within a single framework to solve GCP and determine the chromatic number [2,3]. The results showed that QA outperformed SA and ACO; however, it required more time than the latter two for larger graph instances. Nonetheless, all three algorithms exhibited favorable outcomes [4].

Lü and Hao introduced a memetic algorithm (MACOL) to address GCP [5]. Goudet et al proposed a memetic framework guided by deep learning for graph coloring problems and implemented it on GPU devices to solve the classical vertex k-coloring problem and the weighted vertex coloring problem [6]. In a related study, Marappan et al concentrated on developing a new Particle Swarm Optimization (PSO) model that minimizes the search space and number of generations required. Through behavioral analysis of this stochastic search model, it was discovered that premature convergence is primarily due to a decrease in particle velocity within the search space, leading to an implosion and eventual stagnation of swarm fitness [7]. To address this issue, Xu and Chen introduced a Cuckoo Quantum Evolutionary Algorithm (CQEA) that combines a cuckoo search strategy, a local search operation, and a perturbance strategy to improve the algorithm's global exploration capabilities and enhance its performance on GCP [8]. Moalic and Gondran introduced a new memetic algorithm for GCP called HEAD, which utilizes a local search algorithm (TabuCol) as an intensification operator and a crossover operator (GPX) to escape from local minima. Computational experiments on challenging DIMACS graphs showed that HEAD produced accurate results [9]. In another related study, Zhou et al proposed an improved probability learning-based local search algorithm for GCP [10]. Additionally, Marappan et al developed a new PSO model that minimizes the search

space and number of generations required for optimization. The decrease in particle velocity, leading to fitness stagnation, was identified as the primary cause of premature convergence in the stochastic search model [11]. Assi et al performed an analysis of the Genetic Algorithm (GA) approach for graph coloring in the timetable problem [12]. Meanwhile, Kusumawardani et al utilized a combination of a graph-coloring-based sequential greedy algorithm and a simulated annealing (SA) algorithm to address the Examination Timetabling Problem (ETP) using two real-world datasets [13]. Meraihi, et al presented a Chaotic Binary Salp Swarm Algorithm (CBSSA) to address GCP [14]. In a related study, Silva et al proposed a hybrid algorithm called iColourAnt, which employs ant colony optimization (ACO) and an efficient local search strategy to achieve suitable solutions for GCP. Experimental results indicated that iColourAnt outperformed its predecessor, ColourAnt [15]. Furthermore, Bandopadhyay et al generalized the Bounded Coloring Problem (BCP) and the Equitable Coloring Problem (ECP) and demonstrated that the vertex cover size can parameterize BCP and is FPT [16]. In addition to showing that the vertex cover size can parameterize the Bounded Coloring Problem (BCP) and is unlikely to have a polynomial kernel when parameterized by the deletion distance to clique, Bandopadhyay et al demonstrated that BCP is polynomial-time solvable for cluster graphs, generalizing a similar result for Equitable Coloring Problem (ECP) [16]. In general, the most popular algorithms for improving the solving of GCP are currently MA, ACO, GA, TS, SA, QA, PSO, and COA.

3. Solution Procedure

The memetic algorithm we propose is comprised of three primary elements. The first component involves generating the initial population using a heuristic known as the Randomized Danger Algorithm. Subsequently, a tabu search is applied to improve the individuals generated by the danger algorithm and reduce the number of conflicts. In the final phase, a specific GA crossover operation is utilized to generate new k-colorings. These individuals are further optimized by tabu search and combined with the population generated in phase two, and the best k-coloring is selected. A Pseudo-code of the memetic algorithm is illustrated in Figure 2.

Algorithm 1: memetic algorithm procedure.
Generate initial population by randomized danger algorithm.
Population = $\{IP_1,...,IP_{npop}\}$.
npop = number of populations;
for $i=1$: npop
$TS1_i$ = Tabu search (IP_i);
End for
for $i=1$: npop
GA_i = Crossover ($TS1_i$);
$TS2_i$ = Tabu search (GA_i);
End for
Combine TS1 with TS2;
The best coloring= min $\{TS1_1,...TS1_{npop},TS2_1,...TS2_{npop}\}$.

Figure 2 : A Pseudo-code for the memetic algorithm.

3.1 Fitness Function

In the literature, there are various strategies available to solve graph coloring problems, including legal strategy, k-fixed partial legal strategy, penalty strategy, and k-fixed penalty strategy. For the graph coloring problem, the k-fixed penalty strategy is implemented. With this strategy, the number of colors is fixed, and the search space contains both legal and illegal k-colorings. The objective function is to minimize the number of conflicts. Given an undirected graph G=(V, E) with a vertex set V and an edge set E, A= is a k-coloring where Vi is the set of vertices that color i is assigned to. If a and b are two adjacent vertices in graph G, then we say a conflict exists, and the edge (a,b) is referred to as a conflicting edge. The objective function is to minimize the total number of conflicting edges, defined as follows:

$$f(A) = \sum_{\{a,b\}\in E} \alpha_{ab}, \tag{1}$$

$$\text{where } \alpha_{ab} = \begin{cases} 1, & \text{if } a \in V_i,\ b \in V_j \text{ and } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

So, we say that a legal k-coloring is obtained if $f(A)=0$.

3.2 Randomized Danger Algorithm

The danger algorithm, introduced in 1996, is based on the concept of dynamic vertex danger measure, which prioritizes uncolored vertices. In our presented danger algorithm, we utilize a randomized approach to generate individuals for the population. The proposed heuristic comprises two phases,

namely match danger and round danger. In match danger, we select an unassigned match with the highest danger value in each iteration. Then, in round danger, we choose a round with the lowest danger value and assign it to the chosen match from match danger. To prevent becoming trapped in a local optimum, we apply a randomized approach. Specifically, we randomly select a match from the top three highest values in match danger and a round from the top three lowest values in round danger.

3.2.1 Match Danger

Algorithm 2: Match Danger

Step 1: Compute different rounds (i) for each match. (Different-round (i) represents the number of different rounds assigned to the neighbors of the match (i)).

Step 2: Compute unassigned (i) for each match. (Unassigned (i) represents the number of neighbors of the match (i) unassigned to any rounds).

Step 3: Compute share (i) for each match. (Share (i) represents the number of rounds available to match (i) and its unassigned neighbors).

Step 4: Compute avail (i) for each match. (Avail (i) represents the number of rounds available for the match (i)).

Therefore, match danger (i) can be defined as follows:

$$\text{match danger}\,(i) = F(\text{different-round}\,(i)) + k_{\text{u}}.\text{unassigned}\,(i) + k_{\text{a}}.\frac{\text{share}\,(i)}{\text{avail}\,(i)}, \tag{3}$$

where k_u and k_a are non-negative constants and F is a monotonically increasing function defined as follows [18]:

$$F(x) = \frac{M_1}{(R - x)^{k_1}}, \tag{4}$$

where M_1 is a positive constant, R is the number of rounds and k_1 is a non-negative constant. Finally, a match is selected randomly from the top three highest values match danger.

3.2.2 Round Danger

Algorithm 3: round danger.

Step 1: compute the different neighbors (r) for each round. (Different-neighbors (r) is the maximum number of different assigned neighbors, overall unassigned matches having r available as a round).

Step 2: compute the m (r) for each round. (m (r) is the match that achieves the maximum number of different assigned neighbors, overall unassigned matches having r available as a round).

Step 3: compute the no (r) for each round. (No (r) is the number of times round r is assigned to different neighbors).

So, round danger (r) can be defined as follows:

$$\text{round danger}\,(r) = G(\text{different-neighbors}\,(r)) + k_3.\text{unassigned}\,(m\ (r)) - k_4.\text{no}\,(r), \tag{5}$$

where k_3 and k_4 are non-negative constants and G is a monotonically increasing function defined as follows:

$$G(x) = \frac{M_2}{(R - x)^{k_2}}, \tag{6}$$

where M_2 is a positive constant and k_2 is a non-negative constant. Finally, a round is selected randomly from the top three lowest values in round danger and is assigned to the chosen match in match danger.

3.3 Tabu Search

In this step, we optimize the populations generated in the previous phase using tabu search. The proposed tabu search improves each individual separately until the stop condition is met. In each iteration, we select a conflict match and assign a round different from the current one. To avoid being trapped in a local optimum and increase the diversity of the algorithm, we define a tabu list. When a round is assigned to a specific match, it is forbidden to be assigned to that match for the next L iterations. Therefore, this move is added to the tabu list. L is the tabu tenure, which is defined as follows [15]:

$$L = f(\text{A}) + \text{rand}\,(10), \tag{7}$$

The tabu tenure, L, is a random number selected from the set {1, ..., 10} in our proposed tabu search. Additionally, we define an aspiration criterion to consider moves in the tabu list that result in good objective function values (OFV). The Pseudo-code of our proposed tabu search is illustrated in Figure 3.

Algorithm 4: tabu search procedure.
Set the tabu list= $\varnothing$.
Generate initial solution (x^{now}) by randomized danger algorithm.
The tabu tenure L is initialized according to the chosen scheme
Set $x^{best} = x^{now}$ and BestObj= OFV (x^{now}).
i=1
Repeat
    Compute all moves N by the 2-opt algorithm.
    Remove tabu moves from N (considering aspiration criterion).
    Choose the best move N (x^{now}) $\in N$ minimizing the objective function.
    If OFV (N (x^{now})) < BestObj then
        Update the best solution
            set $x^{best} = N$ (x^{now})
        Set the move N (x^{now}) tabu for the next L iterations.
i=i+1
Until i= maximum number of iterations.

Figure 3: A Pseudo-code for tabu search.

3.4 Genetic Algorithm Crossover Operation

In a hybrid algorithm, the crossover is an important operator to produce new individuals and improve the current solution. Two types of crossover operations are commonly used for graph coloring: assignment crossover and partition crossover. In our approach, we utilize a partition crossover known as an adaptive multi-parent crossover, which was introduced by Lü and Hao [18]. The features distinguishing this crossover from others are the number of selected parents and the way rounds are assigned to matches. Individuals improved by tabu search are assigned to different classes and each match belongs to only one class. In the presented crossover, the first m parents are chosen ($2 \leq m \leq$ npop), and the parents with maximal cardinality class are selected. Then, the first round is assigned to matches belonging to the maximal cardinality class and the matches are removed from all the m individuals. We repeat this procedure until all rounds are assigned to matches. For each remaining match, we randomly select a round. Also, a forbidden list is used to avoid focusing on a single parent. When a parent with maximal cardinality class is selected, this parent is forbidden to select until L iterations, where L is [m/2] [18]. After the adaptive multi-parent crossover is performed, the new offspring are further improved using the tabu search algorithm described in Section 3.3. The Pseudo-code of the proposed GA crossover operation is presented in Figure 4.

Algorithm 5: GA crossover operation procedure.
Set the forbidden list= $\varnothing$.
A select number of parents (m) randomly ($2 \leq m \leq$ npop).
Set r=1
Repeat
    Select a parent with maximal cardinality class if it is not on the forbidden list.
    Assign round r to matches in themaximal cardinality class and remove them from all parents.
    Include this parent on the forbidden list and update it.
Set r=r+1
Until r= maximum number of rounds.
Choose a round randomly for each unassigned match.
Improve offspring by tabu search.

Figure 4: A Pseudo-code for GA crossover operation.

The presented crossover operation is exemplified in Figure 5. In this example, there are 10 matches, 3 rounds, and 3 parents. In the first step, parent 1 had maximal cardinality, and matches $\{v_2,v_5,v_6,v_7,v_9\}$, were selected and assigned to round 1. Moreover, all parents remove these matches, and parent 1 is added to the forbidden list. With a similar procedure, matches V1, V4, and V10 are assigned to round 2. Finally, matches V3 and V8 are assigned to round 3.

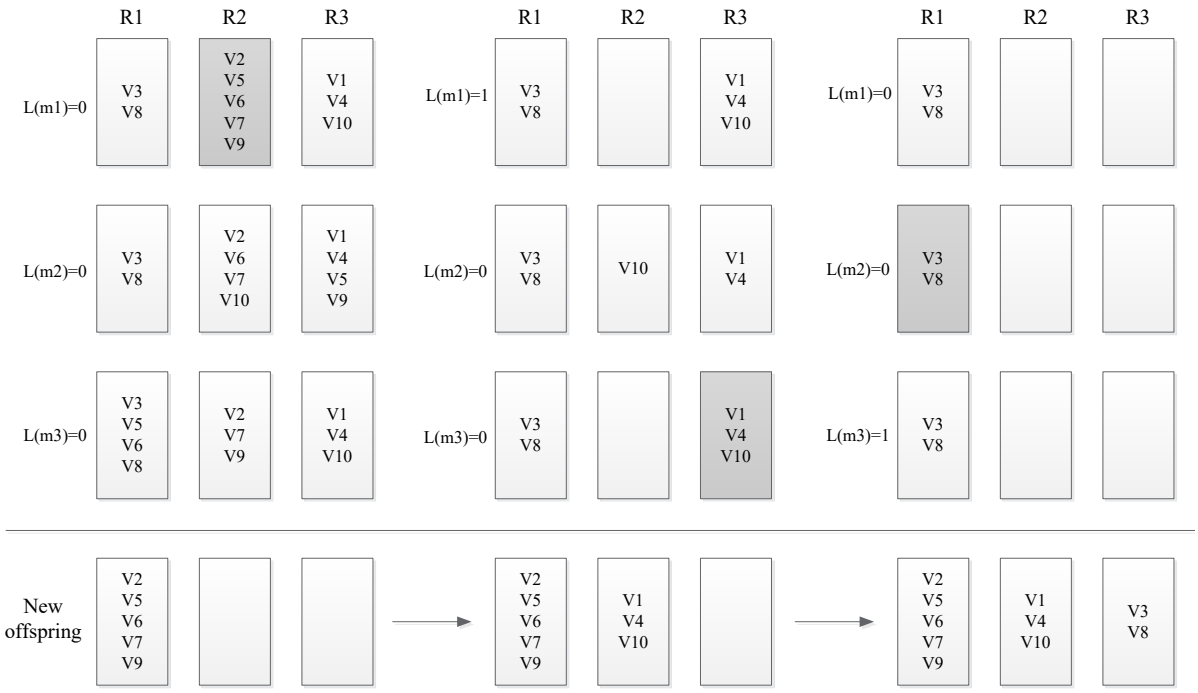

Figure 5: An instance of adaptive multi-parent crossover could be provided as an example.

4. Parameter Tuning

The effectiveness of the proposed memetic algorithm for GCP is highly dependent on the selection of appropriate parameters for each phase of the algorithm. The study explains that these parameters have a significant impact on the algorithm's performance, and a well-tuned parameter configuration can help prevent the algorithm from getting stuck in a local optimum and increase diversity in the search space. This finding is consistent with a separate study from 2009. Each parameter has a range of potential values, and the Taguchi method is adopted to determine the optimal parameter levels, which include npop, ku, ka, k3, and k4, to ensure dependable computational outcomes.

According to [19], the values k_u= 0.025, k_a= 0.33, k_3= 0.5, and k_4= 0.025 work well over a large class of graphs in practice. However, these parameters should be tuned appropriately for graphs with particular structures to get better solutions. Also, In [18], the npop parameter is set to 20. According to this information, our Taguchi method is planned and the parameters of the memetic algorithm and their levels are shown in Table 1. In the proposed Taguchi method for each problem, 16 different plans in 4 replications are considered and the proposed memetic algorithm is run under these designs. Note that parameters k_1, k2, M_1, and M_2 are considered to be constant and equal to 1 [19].

parameter \ level	1	2	3	4
npop	20	25	30	35
k_u	0.020	0.025	0.030	0.035
k_a	0.25	0.30	0.35	0.40
k_3	0.40	0.45	0.50	0.55
k_4	0.020	0.025	0.030	0.035

Table 1: Memetic parameters and level values

Figure 6 provides a visualization of how the selected parameters behave, while Table 2 displays the optimal parameter values that have been determined.

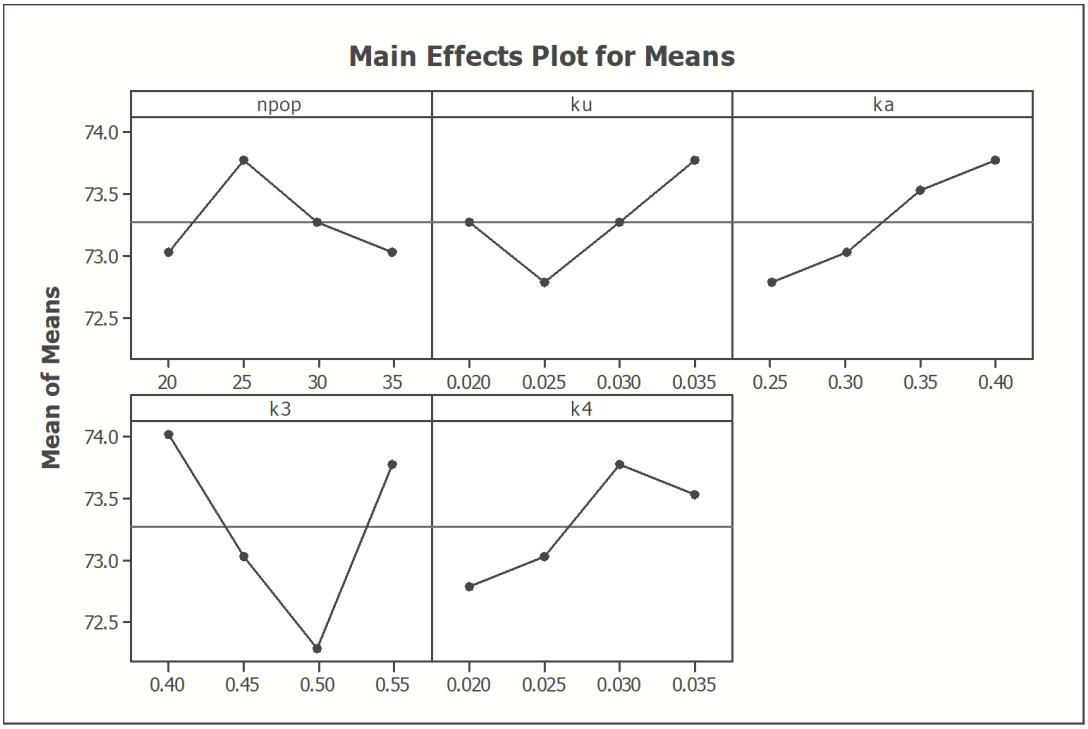

Figure 6: Tuning parameters by Taguchi.

Parameter	Best level
npop	20-35
k_u	0.025
k_a	0.25
k_3	0.50
k_4	0.020

Table 2: The optimal levels for the parameters in the memetic algorithm have been established

5. Experimental Findings

In order to showcase the effectiveness of the proposed algorithm, a total of 41 distinct test problems were selected from two categories of the widely recognized DIMACS coloring benchmarks. In order to compare the results, nine other graph coloring algorithms that were previously introduced in literature were used. The proposed memetic algorithm was implemented in Matlab 7.11 and executed on a PC with a core i5 processor running at 1.8 GHz, and with 6 GB of RAM. The instances are categorized into two classes: Easy graphs are classified as class 1, which can be solved by most heuristics, while difficult graphs fall under class 2 and require strong competitive algorithms to solve them. Tables 3 and 4 present these test problems, along with their characteristics, which belong to classes 1 and 2, respectively. For comparison, only test problems in class 2 were compared to other algorithms in the literature. Tables 3 and 4 provide information on the benchmark problems used in the study. Columns 2 and 3 provide the number of vertices and edges, respectively. Columns 4 and 5 present the density and ID class of these benchmark problems, which include random graphs (DSJC), random geometric graphs (DSJCR and R), Leighton graphs (LEI), class scheduling graphs (SCH), flat graphs (FLAT), and Latin square graphs (LAT). The last column reports the best-known results in the literature and Kbest, which represents the minimum number of colors obtained.

Graph	V	E	Density	ID class	K_{best}
DSJC125.1	125	736	0.09	DSJC	5
DSJC125.5	125	3891	0.50	DSJC	17
DSJC125.9	125	6961	0.89	DSJC	44
DSJC250.1	250	3218	0.10	DSJC	8
DSJC250.9	250	27897	0.90	DSJC	72
R125.1	125	209	0.03	R	5
R125.1C	125	7501	0.97	R	46
R125.5	125	3838	0.5	R	36
R250.1	250	867	0.03	R	8
R250.1C	250	30227	0.97	R	64
DSJR500.1	500	3555	0.03	DSJR	12
R1000.1	1000	14348	0.03	R	20
le450_15a	450	8168	0.08	LEI	15
le450_15b	450	8169	0.08	LEI	15
le450_25a	450	8260	0.08	LEI	25
le450_25b	450	8263	0.08	LEI	25
school1	385	19095	0.26	SCH	14
school1_nsh	352	14612	0.24	SCH	14
flat300_20_0	300	21375	0.48	FLAT	20

Table 3: Computational results on easy DIMACS benchmarks

No.	Graph	V	E	Density	ID class	K_{best}
1	DSJC250.5	250	15668	0.50	DSJC	28
2	DSJC500.1	500	12458	0.10	DSJC	12
3	DSJC500.5	500	62624	0.50	DSJC	48
4	DSJC500.9	500	112437	0.90	DSJC	126
5	DSJC1000.1	1000	49629	0.10	DSJC	20
6	DSJC1000.5	1000	249826	0.50	DSJC	83
7	DSJC1000.9	1000	449449	0.90	DSJC	223
8	DSJR500.1c	500	121275	0.97	DSJR	85
9	DSJR500.5	500	58862	0.47	DSJR	122
10	R250.5	250	14849	0.48	R	65
11	R1000.1c	1000	486090	0.97	R	98
12	R1000.5	1000	238267	0.48	R	234

13	le450_15c	450	16680	0.17	LEI	15
14	le450_15d	450	16750	0.17	LEI	15
15	le450_25c	450	17343	0.17	LEI	25
16	le450_25d	450	17425	0.17	LEI	25
17	flat300_26_0	300	21633	0.48	FLAT	26
18	flat300_28_0	300	21695	0.48	FLAT	28
19	flat1000_50_0	1000	245000	0.49	FLAT	50
20	flat1000_60_0	1000	245830	0.49	FLAT	60
21	flat1000_76_0	1000	246708	0.49	FLAT	82
22	latin_sqr_10	900	307350	0.76	LAT	98

Table 4: Computational results on difficult DIMACS benchmarks

Table 5 provides a summary of research findings reported in the literature for comparison and contains results for the challenging test problems. Table 6 presents the results of 22 test problems applying 9 methods and shows a comparison of these methods and our proposed memetic algorithm. The fifth row from the bottom shows each author's number of test problems. The fourth row from the bottom shows the number of best results obtained from each algorithm. The value of the percentage of best solution represents the percentage of best solutions obtained by each algorithm. This value is calculated as follows:

$$\text{Percentage of best solutions} = \frac{\text{Number of the best solutions}}{\text{Number of test problems}}. \tag{8}$$

The second row from the bottom of the table calculates the total percentage of the best solution, which represents the percentage of the best solutions obtained by each algorithm. The calculation is performed as follows:

$$\text{Total percentage of best solutions} = \frac{\text{Number of the best solutions}}{22}. \tag{9}$$

Finally, the last row shows the difference between each method and the best solution for each test problem. It is calculated only for methods with 22 test problems and is obtained as follows:

$$\text{Gap}(\%) = \sum_{n=1}^{22} \frac{\left(\text{obtained solution}\right)_n - \left(\text{best solution}\right)_n}{\left(\text{best solution}\right)_n}, \tag{10}$$

The index of test problems is denoted by n. Table 6 presents the performance of the proposed memetic algorithm compared to the most effective heuristic algorithms in the literature. Each instance is solved 30 times and is stopped when a legal k-coloring is found or when the processing time reaches its timeout limit set at six CPU hours. The results show that the proposed algorithm outperforms algorithms introduced by references 1, 2, 5, and 6 (see rows 1, 2, 5, and 6 of Table 5). The proposed algorithm yields better results than these algorithms (see the last four rows of Table 6). Additionally, the proposed algorithm performs even better than the algorithms in references 1, 2, 5, and 6 for eight, sixteen, nine, and fourteen instances, respectively.

Two algorithms proposed in references 3 and 4 obtain better solutions in two and one instance, respectively, but our algorithm obtains better results in eleven and ten instances compared to these two algorithms, respectively.

The proposed algorithms in references 7, 8, and 9 perform well and obtain the best solutions in most instances. However, our algorithm performs even better than the algorithms in references 7, 8, and 9 for four, one, and two instances, respectively. Only in one instance, algorithms in references 7 and 9 obtain better results than our memetic algorithm (see row 12 of Table 5). In addition, only our algorithm and algorithms in references 8 and 9

can achieve the nineteen best results (see the row of the number of best solutions in Table 6). Overall, our algorithm has a smaller gap compared to the above-mentioned three algorithms and has results closest to the best solutions reported in the literature, with a gap of 8.43% compared to 9.29%, 14.46%, and 15.01% associated with references 7, 8, and 9, respectively (see the last row of Table 5). Based on the obtained results, the proposed memetic algorithm is recommended for the Iranian Football League case study.

No.	Problem	Proposed algorithm	Time (m)	Reference								
				1	2	3	4	5	6	7	8	9
1	DSJC250.5	**28**	1	**28**	29	**28**	--	**28**	--	**28**	**28**	**28**
2	DSJC500.1	**12**	2	**12**	13	**--**	**12**	**12**	**12**	**12**	**12**	**12**
3	DSJC500.5	**48**	25	49	50	49	**48**	**48**	49	**48**	**48**	**48**
4	DSJC500.9	**126**	110	127	127	--	**126**	**126**	127	127	**126**	**126**
5	DSJC1000.1	**20**	114	21	21	--	**20**	**20**	21	**20**	**20**	**20**
6	DSJC1000.5	**83**	61	88	91	89	86	84	89	**83**	**83**	**83**
7	DSJC1000.9	**223**	183	228	229	--	224	224	227	224	**223**	**223**
8	DSJR500.1c	**85**	17	**85**	**85**	**85**	**85**	86	**85**	**85**	**85**	**85**
9	DSJR500.5	**122**	133	**122**	128	123	125	127	128	**122**	**122**	**122**
10	R250.5	**65**	13	**65**	--	**65**	--	--	67	**65**	**65**	**65**
11	R1000.1c	**98**	21	**98**	--	**98**	--	--	**98**	**98**	**98**	**98**
12	R1000.5	243	330	237	--	241	--	--	254	**234**	245	238
13	le450_15c	**15**	6	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
14	le450_15d	**15**	9	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
15	le450_25c	**25**	25	26	26	--	**25**	26	26	**25**	**25**	**25**
16	le450_25d	**25**	22	26	26	--	**25**	26	26	**25**	**25**	**25**
17	flat300_26_0	**26**	16	**26**	--	**26**	--	**26**	--	**26**	**26**	**26**
18	flat300_28_0	29	141	31	--	31	**28**	31	29	31	29	31
19	flat1000_50_0	**50**	18	**50**	--	**50**	**50**	**50**	73	**50**	**50**	**50**
20	flat1000_60_0	**60**	11	**60**	--	**60**	**60**	**60**	79	**60**	**60**	**60**
21	flat1000_76_0	**82**	73	87	--	89	85	84	87	**82**	**82**	**82**
22	latin_sqr_10	99	173	99	100	**98**	--	104	--	101	99	100
23	Number of test problems	22	----	22	14	16	16	19	19	22	22	22
24	Number of the best solutions	19	----	11	3	10	12	10	5	18	19	19
25	Percentage of the best solution	86.36	----	50.0	21.4	62.5	75.0	52.6	26.3	81.8	86.3	86.3
26	Total percentage o the best solution	86.36	----	50.0	13.6	45.5	54.5	45.5	22.7	81.8	86.4	86.4
27	Gap (%)	8.43	----	43.02	--	--	--	--	--	15.01	9.29	14.46

Table 5: Performance of proposed algorithm as compared to other approaches

6. Case Study: Iranian Football League

Here, a real-world double round-robin sports scheduling problem is presented as a case study. An interesting problem in DRR sports is soccer league competitions. Our considered problem is concerned with the Iranian football league called the Iran Premier League (IPL). In this league, there are 16 teams from different cities. Every team is required to compete against each of the other teams twice, once in their opponent's home venue and once in their home venue. There are 30 weeks (rounds) in the season, which means 15 weeks for each half-season. In each

round, there are 8 matches to play and in total 120 matches in a half season. Each team can play once in each round and when it plays with another team for the first time, the second play is made in the opposite half of the schedule. Also, several teams have the same stadium. So, when a team of the group plays at home, another team must play away. There is also another constraint. All teams must have good patterns and avoid breaks as much as possible. A break is said to occur when a team plays at home or away two or more times in consecutive rounds. Our goal is to minimize the number of breaks as much as possible. Table 6 displays the teams participating in the IPL along with their respective cities and stadiums.

No.	Team	City	Stadium
1	Damash	Rasht	Azodi
2	Esteghlal	Tehran	Azadi
3	Esteghlal Khuzestan	Ahvaz	Ghadir
4	Fajr Sepasi	Shiraz	Hafezieh
5	Foolad	Ahvaz	Ghadir
6	Gostaresh	Tabriz	Sahand
7	Malavan	Anzali	Takhti
8	Mes	Kerman	Shahid Bahonar
9	Naft	Tehran	Dastgerdi
10	Persepolis	Tehran	Azadi
11	Rah Ahan	Tehran	Takhti Tehran
12	Saba Qom	Qom	Yadegare Emam
13	Saipa	Karaj	Enghelab Karaj
14	Sepahan	Esfahan	Foolad Shahr
15	Tractor Sazi	Tabriz	Sahand
16	Zob ahan	Esfahan	Foolad Shahr

Table 6: List of clubs in IPL

To address the DRR scheduling problem, the proposed memetic algorithm was utilized. Initially, the first matches were randomly arranged using a randomized danger algorithm. Then, the population was enhanced using Tabu search. Finally, a genetic algorithm crossover was applied to generate new k-colorings for the problem.

The matches scheduled for the first half-season are shown in Table 7. For the second half season, The match schedule for the second half of the season closely resembles that of the first half, with the only difference being that the teams who played at home during a match in the first half season will play away during the corresponding match in the second half. The results indicate that there were 13 interruptions in the schedule.

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
11-13	15-6	11-2	5-12	15-7	5-3	15-9	5-16
12-9	5-11	12-13	8-3	11-14	11-15	12-14	11-12
14-5	8-12	16-15	9-10	12-2	8-16	16-13	8-9
10-15	9-4	10-7	14-15	16-9	9-7	10-5	14-4
1-8	16-10	1-5	2-1	10-8	14-1	1-11	2-10
4-7	2-14	4-8	7-16	1-6	2-4	4-6	1-15
6-2	7-3	6-14	13-4	4-5	13-10	7-8	13-7
3-16	13-1	3-9	6-11	3-13	6-12	3-2	6-3

Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
15-8	5-9	15-13	5-13	15-5	11-9	15-2
9-13	11-3	8-5	11-16	8-6	12-16	5-6
12-1	12-15	9-2	12-3	9-14	14-8	8-11
16-2	14-10	16-14	14-7	16-1	2-5	9-1
10-6	2-7	10-11	2-8	10-12	1-7	16-4
4-11	1-4	4-12	1-10	7-11	4-10	10-3
7-5	13-8	7-6	4-15	13-2	6-13	7-12
3-14	6-16	3-1	6-9	3-4	3-15	13-14

Table 7: Scheduling the first half of the season in IPL

7. Conclusions

The authors introduced a memetic algorithm that utilizes tabu searches and genetic algorithms to improve the efficiency of solving round-robin sports scheduling problems presented as graph coloring problems. The initial population was created using a randomized danger algorithm and then refined using tabu search. The algorithm produced new k-colorings with a GA crossover operation named adaptive multi-parent and further improved the offspring with tabu search.

The effectiveness of the proposed memetic algorithm was improved using a Taguchi plan, which involved 16 different plans in 4 replications that considered the parameters npop, ku, ka, k3, and k4. The respective values of 20, 0.025, 0.25, 0.50, and 0.020 were selected for these parameters. To demonstrate the algorithm's validity, it was evaluated on the DIMACS challenge benchmarks with 41 test problems from different benchmark graphs categorized into easy and difficult graphs. The results from 22 difficult graphs were compared to those of 9 effective heuristics in the literature, and the algorithm proved to be highly competitive on a set of benchmark graphs known to be challenging. The proposed memetic algorithm achieved the best solution and the smallest gap compared to other heuristics in the literature in 19 out of 22 test problems, reaching results closest to the best solutions in the literature. The gap was equal to 8.43% compared to 9.29%, 14.46%, and 15.01% associated with references 7, 8, and 9, respectively. The algorithm was also applied to real-world round-robin sports scheduling problems as an application of graph coloring problems, specifically in the Iran football premier league. In this league, 16 teams played against each other twice, once in each other's home and once away. The round-robin sports scheduling problem in the Iran football premier league has a specific constraint called the concurrent match constraint, which requires that when one team plays at home, another team with the same stadium must play away. Additionally, all teams must have good patterns of avoiding breaks as much as possible. This study aimed to schedule the matches in different rounds while minimizing the number of breaks and satisfying the given constraints. The results of the algorithm showed that the number of breaks was equal to 13. Round-robin sports scheduling is an intriguing area of research that currently lacks efficient algorithms for solving it. Developing a novel algorithm that can offer better solutions can be considered a potential future direction for researchers in this field.

Data availability

Data will be made available on request.

Declarations

Conflict of interest

The authors declare no conflict of interest.

References

1. Smith, Derek H., Steve Hurley, and S. U. Thiel, (1998). "Improving heuristics for the frequency assignment problem." *European Journal of Operational Research* 107.1: 76-86.
2. Bui, Thang N., et al., (2008). "An ant-based algorithm for coloring graphs." *Discrete Applied Mathematics* 156.2:

190-200.
3. Dowsland, Kathryn A., and Jonathan M. Thompson, (2008). "An improved ant colony optimization heuristic for graph coloring." *Discrete Applied Mathematics* 156.3: 313-324.
4. Kole, Arnab, Debashis De, and Anindya Jyoti Pal, (2022). "Solving Graph Coloring Problem Using Ant Colony Optimization, Simulated Annealing and Quantum Annealing—A Comparative Study." Intelligence Enabled Research: DoSIER 2021. Singapore: Springer Singapore,1-15.
5. Lü, Zhipeng, and Jin-Kao Hao, (2010). "A memetic algorithm for graph coloring." *European Journal of Operational Research* 203.1: 241-250
6. Goudet, Olivier, Cyril Grelier, and Jin-Kao Hao, (2022). "A deep learning guided memetic framework for graph coloring problems." Knowledge-Based Systems 258: 109986.
7. Marappan, Raja, and Gopalakrishnan Sethumadhavan, (2021). "Solving graph coloring problem using divide and conquer-based turbulent particle swarm optimization." *Arabian Journal for Science and Engineering*: 1-18.
8. Xu, Yongjian, and Yu Chen, (2020). "A Cuckoo Quantum Evolutionary Algorithm for the Graph Coloring Problem." Bio-Inspired Computing: Theories and Applications: 16th International Conference, BIC-TA 2021, Taiyuan, China, December 17–19, 2021, Revised Selected Papers, Part I. Singapore: Springer Singapore.
9. Moalic, Laurent, and Alexandre Gondran, (2018). "Variations on memetic algorithms for graph coloring problems." *Journal of Heuristics* 24: 1-24.
10. Zhou, Yangming, Béatrice Duval, and Jin-Kao Hao, (2018). "Improving probability learning based local search for graph coloring." Applied Soft Computing 65: 542-553.
11. Marappan, Raja, and Gopalakrishnan Sethumadhavan, (2021). "Solving graph coloring problem using divide and conquer-based turbulent particle swarm optimization." *Arabian Journal for Science and Engineering*: 1-18.
12. Assi, Maram, Bahia Halawi, and Ramzi A. Haraty, (2018). "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem." *Procedia Computer Science* 126: 899-906.
13. Kusumawardani, Dian, Ahmad Muklason, and Vicha Azthanty Supoyo, (2019). "Examination timetabling automation and optimization using greedy-simulated annealing hyper-heuristics algorithm." *2019 12th International Conference on Information & Communication Technology and Systems (ICTS)*. IEEE.
14. Meraihi, Yassine, Mohammed Mahseur, and Dalila Acheli, (2020). "A modified binary crow search algorithm for solving the graph coloring problem." *International Journal of Applied Evolutionary Computation (IJAEC)* 11.2: 28-46.
15. Silva, Anderson Faustino da, Luis Gustavo Araujo Rodriguez, and João Fabrício Filho, (2020). "The improved ColourAnt algorithm: a hybrid algorithm for solving the graph coloring problem." *International Journal of Bio-Inspired Computation* 16.1: 1-12.
16. Bandopadhyay, Susobhan, et al., (2023). "Structural parameterizations of budgeted graph coloring." Theoretical Computer Science 940: 209-221.
17. Philippe Galinier, Alain Hertz, and Nicolas Zufferey, An adaptive memory algorithm for the k-coloring problem, *Discrete Applied Mathematics* 156 (2008), no. 2, 267–279.
18. Lü, Zhipeng, and Jin-Kao Hao, (2010). "A memetic algorithm for graph coloring." *European Journal of Operational Research* 203.1: 241-250.
19. Glover, Fred W., Mark Parker, and Jennifer Ryan, (1993). "Coloring by tabu branch and bound." Cliques, Coloring, and Satisfiability 26: 285-307.