A Metaheuristic Algorithm to Face the Graph Coloring Problem

Chapter · November 2020

DOI: 10.1007/978-3-030-61705-9_17

5 authors, including:

Angélica Guzmán-Ponce
Universitat Jaume I
10 PUBLICATIONS **66** CITATIONS
SEE PROFILE

J. Raymundo Marcial-Romero
Universidad Autónoma del Estado de México (UAEM)
72 PUBLICATIONS **240** CITATIONS
SEE PROFILE

Rosa María Valdovinos
Universidad Autónoma del Estado de México (UAEM)
78 PUBLICATIONS **883** CITATIONS
SEE PROFILE

Roberto Alejo
Tecnológico Nacional de Mexico / IT Toluca
71 PUBLICATIONS **882** CITATIONS
SEE PROFILE

A Metaheuristic Algorithm to Face the Graph Coloring Problem

A. Guzmán-Ponce[1(✉)], J. R. Marcial-Romero[1], R. M. Valdovinos[1], R. Alejo[2], and E. E. Granda-Gutiérrez[3]

[1] Facultad de Ingeniería, Universidad Autónoma del Estado de México, Toluca, Mexico
aguzmanp643@alumno.uaemex.mx, {jrmarcialr,rvaldovinosr}@uaemex.mx
[2] Tecnológico Nacional de México/IT Toluca, Mexico City, Mexico
ralejoe@toluca.tecnm.com
[3] Centro Universitario UAEM Atlacomulco, Universidad Autónoma del Estado de México, Mexico City, Mexico
eegrandag@uaemex.mx

Abstract. Let $G = (V, E)$ be a graph with a vertex set V and set of edges E. The Graph Coloring Problem consists of splitting the set V into k independent sets (color classes); if two vertices are adjacent (i.e. vertices which share an edge), then they cannot have the same color. In order to address this problem, a plethora of techniques have been proposed in literature. Those techniques are especially based on heuristic algorithms, because the execution time noticeably increases if exact solutions are applied to graphs with more than 100 vertices. In this research, a metaheuristic approach that combines a deterministic algorithm and a heuristic algorithm is proposed, in order to approximate the chromatic number of a graph. This method was experimentally validated by using a collection of graphs from the literature in which the chromatic number is well-known. Obtained results show the feasibility of the metaheuristic proposal in terms of the chromatic number obtained. Moreover, when the proposed methodology is compared against robust techniques, this procedure increases the quality of the residual graph and improves the Tabu search that solves conflicts involved in a path as coloring phase.

Keywords: Heuristic algorithm · Tabu search · Graph coloring · Maximal Independent Set

1 Introduction

The graph coloring problem (GCP) consists on assign colors to elements named vertex of a graph, such that no two adjacent vertices had the same color, so the smallest number of colors needed to color a graph is called the chromatic number. This is a NP-complete problem [7,11].

GCP allows describing and addressing combinatorial problems; thus, many real-problems can be solved using graphs, for instance: assignment of radio frequencies [15], chemical combinations [8], assignment of schedules [4], memory

E. A. de la Cal et al. (Eds.): HAIS 2020, LNAI 12344, pp. 195–208, 2020.
https://doi.org/10.1007/978-3-030-61705-9_17

allocation [16], among others. The approach versatility of GCP makes it very significant in several areas and, consequently, there are several approximate solutions.

Most of the researches about GCP have been divided into two solve directions: a) the use of exact algorithms and b) heuristic algorithms. In the first case, it is always an optimal solution to an optimization problem. The exact algorithms that have been developed to solve the graph coloring problem have been characterized by their exponential complexity, either in time or space, with respect to the number of vertices or edges. For instance, Christofides [5] proposed an algorithm with complexity $n!N^{O(1)}$. In contrast, Bodlaender and Kratsch [2] determined the chromatic number of a graph with complexity $O(5.283^n)$ in time and memory used.

Given the exponential complexity of the exact algorithms, heuristic strategies have been developed. These strategies do not guarantee an exact answer, but approximate the solution and provide high performance. Moreover, metaheuristic methods are used because exact algorithms can solve graph problems up to 100 vertices, while bigger graphs require heuristic techniques [13].

Several effective metaheuristic methods have been proposed for GCP. They are based on Ant Colony Optimization, Genetic algorithms, Tabu search, Simulated annealing, among other metaheuristic algorithms. The Greedy Algorithm [1] is noticeable because it can approximate the chromatic number by assigning to a vertex the lowest-numbered color that has not been used on any previously colored vertices adjacent to it. This algorithm is used by JGraphT [14], which is an open-access Java library useful to graph theory methods.

Guzmán et al. [9] proposed an approximate graph coloring algorithm based on getting the Maximal Independent Set (MIS) by the principle of *"reduce-and-solve"*. A MIS is built from non-articulated vertices with the highest degree and which are not adjacent to each other; this process is done until the graph became a bipartite graph since they can be colored with two colors.

Another proposal with heuristic algorithms intended to large graphs (more than 1000 vertices) was done by Qinghua Wu et al. [17]. The algorithm named $E2COL$ pre-processes the original graph to obtain sets of independent sets with the same size until it has a residual graph (extraction phase), which is coloring by a Tabu search. If the coloring is not valid, an expansion phase is applied which returns an independent set from the extraction phase and reapply Tabu search.

Marappan and Sethumadhavan [12] proposed three methods based on genetic algorithms and Tabu searching. Their proposals are focused on modifying a Single Parent Conflict Gene Crossover (SPCGX) with the Advanced Local Guided Search (ALGS), which decrease the search area through choosing a gene before the crossover operation, and a Tabu search as mutation process. In the first method, SPCGX and Conflict Gene Mutation are used with ALGS. The Conflict Gene Removal process is added in the second method to increase the percentage of successful runs of the genetic algorithm. Finally, multipoint SPCGX and

Multipoint Conflict Gene Mutation are used in the third method with ALGS and the Conflict Gene Removal.

Metaheuristic proposals are good alternatives to approximate the graph coloring problem over graphs with up to 400 vertices and up to 12,000 edges. Thus, a metaheuristic method named $TAMISG$ is proposed in this work. The latter is based on two strategies: the heuristic algorithm proposed by Wu et al. [17] and a deterministic algorithm (an algorithm that gets always the same solution, by following a defined series of steps) proposed by Guzmán et al. [9], which improves the construction of MIS. To improve the approximate chromatic number of a graph, the advantages of both methods are considered in $TAMISG$; additionally, a modified tabu search strategy that improves or equals results reported in the literature was also included.

2 Theoretical Framework

2.1 Graph Theory

Let $G = (V, E)$ be an undirected simple graph (i.e., loop-less and without parallel edges) with a set of vertices $V(G)$ and a set of edges $E(G)$, where an edge is a non-ordered pair of vertex $\{v, w\}$. Two vertices v and w are *adjacent* if there is an edge $\{v, w\} \in E$.

Given a graph $G = (V, E)$, the *degree* of a vertex $v \in V$, denoted by $\delta(v)$, is $|N(v)|$ (i.e. the number of edges in which v is incident). The *neighbourhood* of a vertex v in a graph $G = (V, E)$ is $N(v) = \{\forall u \in V \mid \{v, u\} \in E\}$, i.e $N(v)$ is the set of all vertices adjacent to v excluding itself.

For a subset of vertices $S \subseteq V(G)$, the subgraph of G denoted by $G|S$ has vertex set S and a set of edges $E(G|S) = \{\{u, v\} \in E : u, v \in S\}$. $G|S$ is called the *subgraph of G induced by S*. $G - S$ denotes the graph $G|(V - S)$. The subgraph induced by $N(v)$ is denoted as $H(v) = G|N(v)$ which has to $N(v)$ as the set of vertices and all edges upon them.

Given a subgraph $H \subseteq G$, for each vertex $x \in V(H)$, let $\delta_H(x)$ be the degree of x in the induced subgraph H of G, if $H = G$ then $\delta_G(x) = \delta(x)$ and $E_H(x) = \{\{x, u\} \in E(G) : u \in H\}$. Similarly, $N_H(x)$ denotes the set of vertices from H adjacent to x. For any subgraph $H \subseteq G$, $\delta_G(H) = \sum_{x \in H} \delta_G(x)$. If H is an independent set of G then $\delta_G(H)$ is the number of edges of G incident to any vertex of H.

A path from a vertex v to a vertex w in a graph is a sequence of edges: $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$ such that $v = v_0$, $v_n = w$,v_k is adjacent to v_{k+1} and the length of the path is n. A simple path is a path such that $v_0, v_1, \ldots, v_{n-1}, v_n$ are all distinct. A cycle is just a nonempty path such that the first and last vertices are identical.

$S \subseteq V$ is an independent set in G if for whatever two vertices v_1, v_2 in S, $\{v_1, v_2\} \notin E$. Let $I(G)$ be the set of all independent sets of G. An independent set $S \in I(G)$ is *maximal*, abbreviated as MIS, if it is not a subset of any larger

independent set and, it is *maximum* if it has the largest size among all independent sets in $I(G)$. The *independence number* $\alpha(G)$ is the cardinality of the maximum independent set of G.

Let $G = (V, E)$ be a graph, G is a *bipartite graph* if V can be partitioned into two subsets U_1 and U_2, such that every edge of G joins a vertex of U_1 and a vertex of U_2.

On the other hand, a graph is *bipartite* if its set of vertices can be divided into two subsets X and Y, such that, each edge has an end point in X and another point final in Y [3], denotes a bipartite G with partition (X, Y) as $G[X, Y]$. If $G[X, Y]$ is simple and each vertex of X is joined by a vertex in Y, then G is a *bipartite graph.*

Another useful graph used is a *tree* [3], which is a graph without cycles, i.e. a graph G such that, for any pair of vertices in G there is a single path that unites them. An *expansion tree* T contains all the vertices of the original graph without edges that form cycles [3].

A *co-tree* is the graph complement of the expansion tree denoted as $\bar{T}$, in other words, it is a subgraph of G which has the vertices and edges that form cycles in G and that are not in the expansion tree.

A cutting vertex in a connected graph G, is a vertex v in which G/v results in a disconnected graph; in contrast, a *non-articulation vertex* is the vertex v that G/v results as a connected graph [3].

The GCP is formally defined as a graph $G = (V, E)$ if there is a mapping $\alpha : V \rightarrow K$, where K is a finite set of colors $\{1, 2, 3, ..., k\}$, a coloring of G is a pair (G, α) such that, $\alpha(V') = k$, with $k \in K$, if $v, w \in V$, $\alpha(v) \neq \alpha(w)$, if $v, w \in E$ and $V' \subseteq V$. A coloring is feasible, if the vertices in each edge in E have different color assigned. A vertex coloring of a graph G with k colours and $k \geq 1$ is called $k - colouring$.

2.2 Tabu Search

Tabu search is a local searching algorithm that restricts elements that can improve a solution [6]. The main idea of using Tabu search is giving a feasible solution s and their neighborhood $N(s)$. The algorithm chooses the best neighbor s^* generated so far; thus, s^* is moved whether $f(s^*)$ is better than $f(s)$ or not, where f is a fitness function which assesses a possible solution. Additionally, a tabu list is used to decide whether a move can be allowed or not; this list contains moves that are not allowed at the present iteration, where a move remains into the tabu list only during a certain number of iterations until a move can be used.

Tabucol is a Tabu search for graph coloring proposed by Hertz et al. [10]. This algorithm generates a random coloring with k-color classes (Algorithm 1); this will not be legal coloring in most cases. In *Tabucol*, the aim of the fitness function is trying to minimize the number of conflicts in a possible solution, and it is defined as follows:

$$f(s) = \sum_{i=1}^{k} |\forall\{u, v\} \in E : u \in C_i, v \in C_i| \quad (1)$$

If $f(s) = 0$ is known as a legal coloring.

Once an initial coloring has been generated, Tabucol (Algorithm 1) works until a legal solution is found or when certain number of iterations are completed. For each iteration, a next move is taken to remain in the tabu list a certain number of times. The next move will depend on taking the list members with the highest improvement on $f(s)$, except those moves in the tabu list. After a move is chosen, it is applied and it prohibits, i.e. the move that was just applied should not be applied for a number of iterations.

2.3 E2COL

E2COL [17] is a heuristic algorithm for graph coloring based on an extraction-and-expansion approach. The algorithm extracts independent sets to reduce the initial graph before expanding the approximate coloring solution by progressively adding back the extracted independent sets. Essentially, the *E2COL* algorithm [17] consists of three phases, which can be summarized as follows:

- Extraction Phase: Obtains sets of independent sets until the graph has only q-vertices. This phase uses a Tabu Search algorithm to generate the independent sets.
- Initial coloring: An initial coloring from the residual graph (i.e. the graph with q-vertices) is obtained by applying a Tabu search strategy. If the coloring is legal, i.e. the possible solution has no collisions, an independent set obtained in the extraction phase is added to the final solution and stops the procedure; otherwise, the next phase is done.

Algorithm 1. *Tabu search*

Require: A non directed graph $G = (V, E)$; a solution s_0
Ensure: The best colouring s^*
1: Initialize random configuration s_0
2: **while** $conflicts \neq 0$ and not reached iteration or time limit **do**
3: $s_0' \leftarrow$ Find non-tabu neighbor with largest improvement for the fitness fuction.
4: Make the move
5: $s_0 \leftarrow s_0'$
6: $tl = \lceil Random(0, 9) + \alpha * f(s_0) \rceil$
7: Add the move into Tabu list on tl-iterations
8: Update the Tabu list by reduce tl
9: **if** $f(s_0) < f(s^*)$ **then**
10: $s^* \leftarrow s_0$
11: **end if**
12: **end while**
13: Return s^*

– Expansion Phase: The sets of independent sets obtained from the first phase are returned, i.e. the independent set that was making a disturbance to the solution, by taking t-classes to empty them, through moving vertices around the rest of classes.

2.4 Maximal Independent Set (MIS)

Guzmán et al. [9] proposed a deterministic algorithm consisting of building a MIS from non-articulated vertices upon the condition that the vertex which will be added into the set has the highest degree and it is not adjacent with another vertex of such a set. In a general way, this proposal can be divided into two phases described as follows:

– Debugging phase: It removes trees or paths from the graph before the iterative construction of a MIS, because trees or paths can be colored by two colors.
– Iterative construction of MIS: Once the graph has no trees or paths, the iterative construction of MIS (Algorithm 2) is done by selecting non-articulated vertex x, this will be the first vertex which is added into the MIS. Subsequently, the non-neighbors of x are obtained. This set contains the candidates to be part of the MIS. The non-neighbors of x are sorted in ascending order according to each vertex degree. Afterwards, each non-neighbor vertex ve is verified that no cut the graph; thus, ve can be adding into the MIS, while neighbors of ve are removed from the candidates. The algorithm builds and saves a set of MIS until the remaining graph became a bipartite graph because it can be colored by two colors.

Algorithm 2. *GetMIS*

Require: A non directed graph $G = (V, E)$
Ensure: A set of disjoint MIS to equal cardinality
1: $K = [\,]$
2: Choose a non-articulation vertex $ve \in G$
3: Add ve into K
4: $noNeighbors \leftarrow$Sort non-neighbors of ve
5: **while** $noNeighbors$ **do**
6: $ve' \leftarrow$ Get a vertex from $noNeighbors$
7: **if** ve' is non-articulation vertex **then**
8: Add ve' into K
9: Delete neighbours of ve' in $noNeighbors$.
10: **end if**
11: **end while**
12: **return** K

3 Proposed Method

Due to the exponential complexity of the exact algorithms, heuristic strategies have been chosen to approximate solutions on graphs with more than 100 vertex [13]. In this paper, a metaheuristic method named as $TAMISG$ is proposed.

The latter combines two methods: a) $E2COL$ [17] and b) MIS [9]. This proposal takes advantage of the $E2COL$ methodology with several changes, allowing $TAMISG$ to improve the approximate chromatic number by modifying the Tabu search strategy described in Sect. 2.2; then, MIS is applied.

In $TAMISG$, the $E2COL$ [17] phases are preserved; nonetheless, a variant over the algorithm was made. Modifications to $E2COL$ included in $TAMISG$ proposal are summarized as follows:

1. In the extraction phase, the process to obtain independent sets was changed and a deterministic algorithm [9], which is described in Sect. 2.4, was used.
2. The Tabu search method was modified to solve conflicts involved in a path and it is described in Sect. 3.2.
3. In the extended phase, if a legal coloring is not found, an empty class is added until a solution with zero conflicts is reached.

3.1 Maximal Independent Set Process

In the extraction phase, given an initial size of a MIS, it is necessary to find as many sets of MIS as possible, so that in the end only the MIS with big size remains. The proposal exhibited here takes advantage of the Algorithm 2 described in Sect. 2.4, because it can build a MIS and ensure a residual connected graph. Thus, $TAMISG$ proposal takes the idea of building a MIS with non-articulation vertices and remove these vertices from the graph G, until the graph has q-vertices, which is computed as below:

$$q = |V| - ((|V| * 60)/100) \tag{2}$$

where, $|V|$ is the number of vertices. The proposal considers the 60% of vertices that must be colored in the Extraction phase from the original graph, while 40% of vertices remain as part of the residual graph.

3.2 Tabu Search Proposal

Tabucol is the first Tabu search to graph coloring, as it can be seen in Sect. 2.2. However, the present proposal takes it up and seeks to resolve conflicts involved in a path.

Algorithm 1 receives an initial solution s_0, considered as the best solution (s^*) in the begining. The time limit is given by L-iterations, where, $L = |E| * 0.25$; that is, the 25% density of the graph is considered to iterate the Tabu search. To compute tl, $\alpha = 0.6$ is used and $f(s)$ is given by the Eq. 1.

After selecting a move, it is included a process that handles collisions in the path involved in the movement to be performed. In each iteration, the Tabu search moves a conflict vertex. In order to save these conflict vertices, the initial solution is evaluated to obtain the vertices that cause conflict. The process to obtain a candidate vertex to move has been performed as follows:

1. A list of conflict vertices is generated. For each conflict vertex, a subset is created with vertices that have not been visited and that has an edge between themselves.
2. A recursive process is carried out to obtain a path of conflict vertices. The latter tries to solve the conflict in this path with the vertices in the middle of the conflict vertices.

Once the candidate vertex is obtained, the move must be performed into the class with fewer conflicts. In case of a vertex has all classes in the Tabu list, i.e. the vertex can not be moved to any other class solution, the neighbors are analyzed; thus, a decision will be made whether the tabu search process continues or not. If a vertex collapses with all classes that until the actual solution s_0 has, then s_0 requires a new class.

The decision process tries to find the neighboring vertices of the vertex that collapse in a complete sub-graph. If the number of vertices that participate in the complete sub-graph is greater than the number of classes in the solution, the tabu process stops and returns the best found solution, decreasing the runtime.

3.3 *TAMISG* Algorithm

TAMISG proposal is shown in the Algorithm 3, where the first step consist on build a MIS with the deterministic Algorithm 2 until the residual graph has q-vertices. The next step, an Initial Coloring is performed by a Tabu Search algorithm (lines 7–12), which is explained in the Algorithm 1. This process returns a solution with zero collisions. The Tabu search solution must be added into the MIS extracted from the first phase and return it as a valid solution; otherwise, the Extraction Phase is done.

Because most solutions are based on a k value of color, in this proposal, the GCP is given by approximating the coloring based on a k value, which has been taken from the literature and represents the best approximation obtained until now. However, if in n iterations the coloring is not determined, then the graph is not k colorable, and the nearest integer to k for which it can be determined as colorable is looked.

Algorithm 3. *TAMISG* proposal for graph coloring

Require: A non directed graph $G_0 = (V_0, E_0)$; an integer k
Ensure: A legal $k - coloring$ of G_0 or a fail
{Extraction Phase}
$i = 0$
while $|V_i| > q$ **do**
  G_{i+1}= ***GetMIS***(G_i, I_i) ****This is a modification**
  $i = i + 1$
end while
{ Initial coloring Phase}
s_i = ***Tabu_Search***$(G_i, k - i)$ ****This is a modification, Section 3.2** *\
if s_i = is an $(k - i)$ coloring, this is $f(s_i) = 0$ **then**
  Use s_i to build a k coloring $s_0 = \{s_i, I_{i-1}, \cdots, I_0\}$ of G_0
  Return s_0 and stop the process
end if
{Extent Phase }
while $i > 0$ **do**
  $i = i - 1$
  $s_i = \{s_{i+1}, I_i\}$
  s_i = ***Tabu_Search***(s_i, G_i)
  if s_i is a $(k - i)$ coloring, i.e $f(s_i) = 0$ **then**
    Use s_i to build a legal k-coloring $s_0 = \{s_i, I_{i-1}, \cdots, I_0\}$ of G_0
    Return s_0 and stop the process
  end if
end while
{This is a modification }
while $f(s_i) \neq 0$ ***do***
  $s_i = s_i \bigcup \{\}$ * ***Add an empty class to the solution*** *\
  s_i =***Tabu_Search***(s_i, G_i)
end while

4 Experimental Set-Up

In order to evaluate the *TAMISG* performance, an experimental set was designed, It consist of comparing the results obtained with JGraphT [14], and the proposal by Guzmán et al. [9]. These methods were elected to verify the performance of the proposal depicted in this paper because their simplicity and availability as open-access resources. The results are compared in terms of approximation of the chromatic number. A MacBook Pro with Intel Core i5 microprocessor at 2.6 GHz. 8 GB DDR3 RAM 1600 MHz under OS X Version 10.11.6 was used to perform the experiments.

24 graphs (Table 1) were used in this research, which were taken from the DIMACS repository[1]. All of them are part of three groups that are described as follows:

- REG: Represents problems based on the assignment of records for variables in real codes, by Gary Lewandowski[2].
- SGB: These are graphs from the Donald Knuth's data set at Stanford, these are divided into:
  - *Book Graphs*: Based on literary works, a graph is created by representing each vertex as a character. Two vertices share an edge if the corresponding character meets another. Knuth created graphs from 5 literary works: Tolstoy's Anna Karenina (anna), Dicken's David Copperfield

[1] http://mat.gsia.cmu.edu/COLOR04/.
[2] gary@cs.wisc.edu.

(david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck) and Hugo's Les Misérables (jean).
  - *Queen Graphs*: It is a graph with mn vertices in which each vertex represents a square in a board $m \times n$, two vertices are connected by an edge if the corresponding squares are in the same row, column or diagonal.
- CAR: Graphs *k-insertion and full-insertion* are a generalization of graphs *mycel* with vertices inserted to increase the size of the graph but not the density (i.e., how a graph is connected).

Table 1. Description of the benchmarking graphs. Per each instance (first column), the number of vertices and the number of edges are reported in the next columns respectively.

	Instance	Vertices	Edges		Instance	Vertices	Edges
1	1-Insertions_4.col	67	232	13	queen8_8.col	64	1456
2	2-Insertions_3.col	37	72	14	queen9_9.col	81	2112
3	2-Insertions_4.col	149	541	15	queen8_12	96	1368
4	3-Insertions_3.col	56	110	16	queen11_11.col	121	3960
5	anna	138	986	17	queen12_12.col	144	5192
6	david	87	812	18	queen13_13.col	169	6656
7	fpsol2_i_1	269	11654	19	queen14_14.col	196	8372
8	fpsol2_i_2	363	8691	20	queen15_15.col	225	10360
9	fpsol2_i_3	363	8688	21	queen16_16.col	256	12640
10	queen5_5	25	320	22	zeroin_i_1	126	4100
11	queen6_6.col	36	580	23	zeroin_i_2	157	3541
12	queen7_7.col	49	952	24	zeroin_i_3	157	3540

5 Results and Discussion

Results obtained from the experimental study are presented here, explained in two parts: a) First, the impact of removing MIS to better understand the behavior of the proposal, b) Second, to evaluate the effectiveness of the proposed method, results with the well-known chromatic number per each graph are compared. This information has been extracted from[3].

5.1 Remove MIS

The impact of use of a deterministic algorithm as MIS removing procedure to reduce a graph as an improvement to approximate the chromatic number of a graph is discussed through Table 2, where the remaining graph configuration obtained after applying the Extraction phase is reported.

As it can be noted in Table 2, the remaining graph is small; thus, this becomes an ideal graph to be processed with a Tabu search. Taking the 60% of vertices in each graph to be part of a set of MIS and leaving the rest as a member of a connected graph is the advantage of using [9] to approximate efficiently the chromatic number in the next phase, because most number of vertices covered have been labeled through a MIS.

[3] http://mat.gsia.cmu.edu/COLOR04/.

Table 2. Summary of the extraction phase per each instance (first column). The number of vertices and edges obtained is reported in the second and third columns, respectively. The number of extracted MIS is shown in the fourth column. In the last column, the q-vertices obtained are reported.

Instance	Residual Vertices	Residual Edges	Extracted MIS	Remain vertices	Instance	Residual Vertices	Residual Edges	Extracted MIS	Remain vertices
1-Insertions_4.col	32	61	1	27	queen8_8.col	25	200	5	26
2-Insertions_3.col	13	13	1	15	queen9_9.col	21	118	7	32
2-Insertions_4.col	68	135	1	60	queen8_12	24	73	9	38
3-Insertions_3.col	17	17	1	22	queen11_11.col	49	580	7	48
anna	59	484	1	55	queen12_12.col	45	448	9	58
david	51	812	1	35	queen13_13.col	48	466	10	68
fpsol2_i_1	103	3344	7	108	queen14_14.col	78	1244	9	78
fpsol2_i_2	137	2978	2	145	queen15_15.col	82	1246	10	90
fpsol2_i_3	363	8688	2	145	queen16_16.col	91	1490	11	102
queen5_5	10	32	3	10	zeroin_i_1	44	654	17	50
queen6_6.col	12	84	4	14	zeroin_i_2	60	968	3	63
queen7_7.col	14	44	5	20	zeroin_i_3	60	970	3	63

5.2 Graph Coloring

The $TAMISG$ performance is evaluated through approximate the well-know chromatic number. Table 3 reports the experimental results of such a evaluation. The proposal of this research was compared with JGraphT (a greedy algorithm) and Guzmán et al. [9] (a deterministic algorithm). Finally, 20 executions were carried out and the results obtained from the $TAMISG$ proposal show their standard deviation in parentheses. The computational effort given in seconds is showing in square brackets.

Table 3. Graph coloring performance, the well-know chromatic number is included as a reference value, with the exception on some *queen* graphs in which value has not yet been registered.

Instance	Chromatic Number	JGraphT [14]	Guzmán [9]	TAMISG	Instance	Chromatic Number	JGraphT [14]	Guzmán [9]	TAMISG
1-Insertions_4.col	4	5 [0.1]	5 [0.0]	5 (0.366) [1.8]	queen8_8.col	9	15 [0.2]	12 [0.2]	11 (0.394) [12.2]
2-Insertions_3.col	4	4 [0.1]	4 [0.0]	4 (0.000) [0.0]	queen9_9.col	10	15 [0.3]	13 [0.3]	12 (0.324) [22.9]
2-Insertions_4.col	4	5 [0.3]	5 [0.1]	5 (0.244) [20.7]	queen8_12	12	17 [0.5]	15 [0.4]	14 (0.224) [12.5]
3-Insertions_3.col	4	5 [0.1]	4 [0.0]	4 (0.000) [0.0]	queen11_11.col	11	18 [0.3]	16 [0.9]	15 (0.394) [58.6]
anna	11	11 [0.1]	11 [0.2]	11 (0.366) [1.2]	queen12_12.col	*	20 [0.3]	17 [1.3]	16 (0.598) [138.0]
david	11	11 [0.2]	11 [0.1]	12 (0.489) [2.7]	queen13_13.col	13	22 [0.4]	18 [2.2]	18 (0.513) [160.2]
fpsol2_i_1	65	65 [1.2]	65 [28.3]	67(1.317) [69.4]	queen14_14.col	*	24 [0.6]	21 [3.6]	19 (0.224) [306.3]
fpsol2_i_2	30	30 [0.9]	30 [11.0]	33(0.761) [311.0]	queen15_15.col	*	24 [0.6]	21 [5.3]	19 (0.447) [448.9]
fpsol2_i_3	30	30 [1.2]	30 [11.1]	32 (1.040) [185.9]	queen16_16.col	*	27 [0.7]	22 [7.7]	21 (0.308) [633.3]
queen5_5	5	7 [0.1]	6 [0.0]	5 (0.000) [0.1]	zeroin_i_1	49	49 [1.1]	49 [3.6]	50 (0.510) [3.8]
queen6_6.col	7	10 [0.1]	9 [0.0]	9 (0.324) [2.6]	zeroin_i_2	30	30 [0.7]	30 [2.1]	32 (0.945) [12.2]
queen7_7.col	7	12 [0.1]	11 [0.1]	9 (1.309) [0.6]	zeroin_i_3	30	30 [0.9]	30 [2.1]	31 (0.745) [13.5]

According to these results, the algorithm from Guzmán et al. [9] improves the results obtained from JGraphT in 54% of instances (i.e. 13 graphs), while in the remain graphs, both get similar results. In contrast, when Guzmán et al. [9] is compared against $TAMISG$, the latter obtains a better approximation in 42% of the instances; however, both proposals get similar approximations in the

remain results. Note that all proposals are competitive when they are compared against previously well-known results reported in the literature. Moreover, with exception of some instances, the proposal of this research gets the best results.

In $TAMISG$, most of the successful cases are in *queen* graphs, where the standard deviation indicates that there is not significant dispersion on the results. This is due to the use of the deterministic algorithm in the extraction phase; it obtains the same results after each execution, and the number of MIS extracted from each graph is large enough to get a residual graph, which can be easily colored with Tabu search. In addition, the standard deviation obtained from $TAMISG$ indicates that in 62.5% of graphs a uniform coloring is obtained (because standard deviation is less than 0.5).

It can be seen that the algorithm that obtains results more quickly is JGraphT; however, it does not obtain the best approximation to graph coloring compared to the other proposals. Although $TAMISG$ takes more time for instances with $|E| > 1000$, our proposal gets a better approximation.

It can be concluded that the methodology followed by $TAMISG$ not only improves the JGraphT results, but it is competitive with Guzmán et al. [9]. The latter is because in $TAMISG$ the extraction phase get sets of MIS that are enough to color most vertices, thus increasing the quality of the residual graph, where their q vertices can be easily colored by Tabu Search. Also, Tabu Search improves the graph coloring by resolving conflicts involved in a given path.

6 Conclusions and Future Research

This paper presents a metaheuristic algorithm named $TAMISG$ to handle the Graph Coloring Problem. The proposal in this research approximates the chromatic number of a graph by combining a deterministic algorithm and a heuristic algorithm. Taking advantage of the $E2COL$ methodology (with specific modifications) and introducing a deterministic algorithm as the extraction phase (proposed in [9]), this procedure increases the quality of the residual graph and improves the performance of the Tabu search that solves conflicts involved in a path as coloring phase.

It can be concluded upon the experimental study that, in terms of chromatic number, both: the method proposed by Guzmán et al. [9] and the $TAMISG$ proposal presented in this article, are competitive since the approximate graph coloring is similar. Despite getting similar approximations, in some graphs (for example *queen*), the $TAMISG$ proposal gets better performance than the deterministic proposal. The latter is sustained by means of the estimation of the standard deviation obtained from 20 executions, which indicates that the results are not significantly dispersed. Additionally, $TAMISG$ has better performance compared against the *greedy* method used by JGraphT.

The proposed method can be used to obtain better approximations in graphs with a greater number of edges, because the extraction phase ensured a residual graph with a smaller number of edges; thus, it can let get a better approximation. A potential flaw could be in a forest (graphs composed of graphs) where each

graph must be processed separately in the extraction phase, since the method ensures a connected graph, leaving an open line of study.

TAMISG proposes to solve conflicts in paths by the Tabu search, which leaves interesting future research on the free parameters, like the time in which a vertex remains in the Tabu list. Another open research line is to assign conflict vertices to a new class, where it is important to consider strategies to move from the optimal local. Furthermore, the use of other heuristic algorithms such as *Hill Climbing, Simulated annealing* could be interesting.

Acknowledgment. This work has been partially supported by the 5046/2020CIC UAEM project and the Mexican Science and Technology Council (CONACYT) under scholarship [702275].

References

1. Arumugam, S., Brandstädt, A., Nishizeki, T., et al.: Handbook of Graph Theory, Combinatorial Optimization, and Algorithms. Chapman and Hall/CRC (2016)
2. Bodlaender, H.L., Kratsch, D.: An exact algorithm for graph coloring with polynomial memory. Technical report, Department of Information and Computing Sciences, Utrecht University (2006)
3. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications, vol. 290. Macmillan, London (1976)
4. Burke, E.K., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper heuristic for timetabling problems. Computer Science Technical Report No. NOTTCS-TR-2004-9, University of Nottingham (2004)
5. Christofides, N.: An algorithm for the chromatic number of a graph. Comput. J. **14**(1), 38–39 (1971)
6. Edelkamp, S., Schrödl, S. (eds.): Heuristic Search. Morgan Kaufmann, San Francisco (2012)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
8. Gross, J.L., Yellen, J.: Graph Theory and its Applications. CRC Press (2005)
9. Guzmán-Ponce, A., Marcial-Romero, J.R., Ita, G.D., Hernández, J.A.: Approximate the chromatic number of a graph using maximal independent sets. In: 2016 International Conference on Electronics, Communications and Computers (CONIELECOMP), pp. 19–24. IEEE (2016)
10. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39**(4), 345–351 (1987)
11. Johnson, D.S., Trick, M.A.: Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 11–13 October 1993, vol. 26. American Mathematical Society (1996)
12. Marappan, R., Sethumadhavan, G.: Solution to graph coloring using genetic and tabu search procedures. Arabian J. Sci. Eng. **43**(2), 525–542 (2018)
13. Mostafaie, T., Khiyabani, F.M., Navimipour, N.J.: A systematic study on metaheuristic approaches for solving the graph coloring problem. Comput. Oper. Res. 104850 (2019)
14. Naveh, B.: Contributors: Jgrapht. From: http://jgrapht.org/. Accessed 5 Dec 2019

15. Smith, D., Hurley, S., Thiel, S.: Improving heuristics for the frequency assignment problem. Eur. J. Oper. Res. **107**(1), 76–86 (1998)
16. de Werra, D., Eisenbeis, C., Lelait, S., Marmol, B.: On a graph-theoretical model for cyclic register allocation. Discrete Appl. Math. **93**(2–3), 191–203 (1999)
17. Wu, Q., Hao, J.K.: An extraction and expansion approach for graph coloring. Asia-Pacific J. Oper. Res. **30**(05), 1350018 (2013)