INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2019, VOL. 65, NO. 2, PP. 235–243
Manuscript received January 30, 2019; revised April, 2019. DOI: 10.24425/ijet.2019.126306

Contemporary Methods for Graph Coloring as an Example of Discrete Optimization

Adrian Bilski

***Abstract*—The following paper provides an insight into application of the contemporary heuristic methods to graph coloring problem. Variety of algorithmic solutions for the Graph Coloring Problem (GCP) are discussed and recommendations for their implementation provided. The GCP is the NP-hard problem, aiming at finding the minimum number of colors for vertices in such a way that none of two adjacent vertices are marked with the same color. With the advent of modern processing units metaheuristic approaches to solve GCP were extended to discrete optimization here. To explain the phenomenon of these methods, a thorough survey of AI-based algorithms for GCP is provided, with the main differences between specific techniques pointed out.**

***Keywords*—Graph coloring, chromatic number, metaheuristics**

I. Introduction

THE graph coloring problems (GCP) are considered to be the most difficult combinatorial problems because of their high computational complexity. Simple graph models, consisting of a set of vertices and edges connecting them, are often insufficient to describe various dependencies between objects in practical applications. It is necessary to show that no more than k distinct colors are used. The construction of the approximation algorithm that has a polylogarithmical goodness function is NP-complete, unless P=NP [1]. Therefore the key issue is to define a boundary between computationally simple cases (Polynomial, i.e. form the P class), and those difficult, i.e. NP-complete (NPC). The question whether P=NP is currently the most pressing issue of computer science.

The purpose of this paper is to present methodologies for compression (minimization) of colors when solving GCP. The adaptation of known probabilistic algorithms to graph coloring tasks is explained. Applications of GPC for telecommunication tasks are presented as well.

The graph coloring problem started with Francis Guthries' attempt to color all countries on the map of England, which gave birth to the four color conjecture. It was initially assumed, that four colors are sufficient to process the world map, so that no two neighboring countries would be associated with the same color. This task is only one of over 200 problems collected in [2], concerning the area of chromatic graph analysis.

The paper is organized as follows. The second section introduces terminology utilized for the GCP. In section 3 bounds of the chromatic number are defined. In Section 4, different representations of graph coloring are considered. The cost function is defined there. Section 5 is devoted to circular coloring, while Section 6 describes common approximation methods utilized to solve GCP. Section 7 covers novel metaheuristic methods in GCP. These include Simulated Annealing (SA), Ant Colony Optimization (ACO) or Neural Networks (NN). Section 8 introduces the concept of on-line coloring. Finally, conclusions regarding utilization of contemporary AI algorithms in graph coloring are presented.

A. Bilski is with the University of Social Sciences, Sienkiewicza 9, 90-113 Lodz, Poland (e-mail: blindman26@o2.pl)

II. Terminology

In computer science colors are denoted by natural numbers. The proof of the 4-colored theorem for any number of countries on a world map has come with the eve of the computer technology, thanks to Appel and Haken [3] (1977). In 1981 Holyer proved that the discussed theorem is NP-complete [4], which can be expanded to the whole GCP.

The minimum positive integer k (the number of different colors) for which the graph (map) is k-colorable is the chromatic number of G denoted as $\chi(G)$ [5]. The aim of the coloring process is to minimize the number of colors utilized, equal to the chromatic number (optimal coloring). For planar graphs the chromatic number has been precisely estimated. It is easy to prove that the chromatic number for this class of graphs is not greater than 6. The most complex class, for which the polynomial algorithms exist, are the perfect graphs (in which the chromatic number of every subgraph equals the size of the largest clique of that subgraph). The simplest NP-complete class contains planar graphs.

Issues in GCP the most often examined include vertex and edge coloring. The former is the method of coloring such that no two adjacent vertices are of the same color. Similarly, edge coloring assigns a particular color to each edge of the graph, so that no two adjacent edges are of the same color. Vertex coloring can be considered as entry point of graph coloring. Other coloring problems can be transformed into a vertex variant, i.e. edge coloring of a graph is a vertex coloring of its line graph. Even though vertex and edge coloring are both considered NP-problems, usage of approximate methods make the process of edge coloring easier to execute. This is because in the case of edge coloring the Vizing theorem must be considered [6].

The task of edge coloring so all edges adjacent to any vertex are assigned different colors, has been formulated by Tait in 1880 [7]. He proved that the 4-color problem is equivalent to 3-coloring problem of edge coloring for any cubic map, thus formulating the Vizing Theorem regarding the multigraph chromatic index $\chi'(G)$ - the minimal number of colors sufficient to color the edges of G.

Theorem 1 (Vizing's Theorem): for every nonempty graph G, $\chi'(G) \leq 1 + \Delta(G)$

The value of the chromatic index of the simple graph does not exceed $\Delta(G) + 1$, where $\Delta(G)$ is the graph maximum degree. This bound is almost tight, i.e. the edge chromatic number is either $\Delta(G)$ or $\Delta(G) + 1$. When $\chi'(G) = \Delta(G)$, G is said to be of class 1. Otherwise it is of class 2.
The minimal chromatic number is a minimal number of independent sets into which $V(G)$ (a vector containing graph vertices) can be partitioned. In an independent (stable) set of vertices, in which no two vertices are adjacent, coloring can be considered as mapping $c : V(G) \to N$ (where N is the set of positive integers) such that $c(u) \neq c(v)$, if u and v are adjacent in G. A k-chromatic graph has a chromatic number k. A graph G is k-colorable only if $\chi(G) \leq k$. Typically all k colors are used in k-coloring process.

III. Bounds of the chromatic number

The lower and upper bound are defined for the chromatic number of a graph, regarding the numbers' order or independence. Some bounds are a consequence of the greedy coloring algorithm, which shows that every graph can be colored with one more color than the maximum vertex degree of a colored graph:

$$\chi(G) \leq \Delta(G) + 1 \tag{1}$$

The relation between the chromatic number and the graph size is as follows:

$$1 \leq \chi(G) \leq n \tag{2}$$

This shows that the only graph to be colored with one color is the edgeless one, while a complete graph constructed of n vertices requires n colors to be colored. Graphs that can be colored using only two colors are exactly bipartite. According to the Vizing theorem, every planar graph can be 4-colored. Fig. 1 shows the graph with all of its vertices colored using a minimum of 3 colors (Roman numerals) and all of its edges colored using a minimum of 4 colors (Arabic numerals).

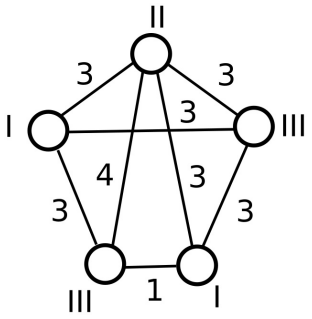

Fig. 1. An example of vertex coloring using a minimum of 3 colors and edges coloring using a minimum of 4 colors.

Even though the dependency between the large cliques of graphs and their chromatic number can be found, Mycielski's theorem states it does not work for small cliques [5].

Theorem 2 (Mycielski's theorem): For every positive integer k, there exists a triangle-free k-chromatic graph.

This theorem refers to the chromatic number 3 and larger, since graphs with chromatic numbers 1 or 2 do not contain triangles.

The clique number (i.e. the number of subsets of graphs vertices where every two vertices in the subset are connected by an edge) can be considered as a lower bound for the chromatic number of a graph [5]. The best upper bound can be found using the estimation methods of coloring graphs.

There are different criteria of graph coloring, like minimizing the number of used colors. There are non-standard methods of coloring, introducing subsequent bounds on colors, like allowing to associate a particular vertex/edge with more than one color (multicoloring) or allowing the division of colors.

The polynomial rough algorithms have a linear goodness function in the worst case or the sublime goodness function. There best heuristic is the Halldorsson algorithm, which has $O(n(loglogn)^2/(logn)^3)$ goodness function [9].

The Vizing theorem evaluates the chromatic index. Its proof shows how to color a graph with a given number of colors in the polynomial time. That is why researchers concentrate on more difficult edge coloring of multigraphs. The most complicated class of graph, for which the polynomial coloring algorithms exist are dual graphs (Fig. 2), while the simplest one, for which the coloring process is yet NP-hard are cubic graphs (Fig. 2).

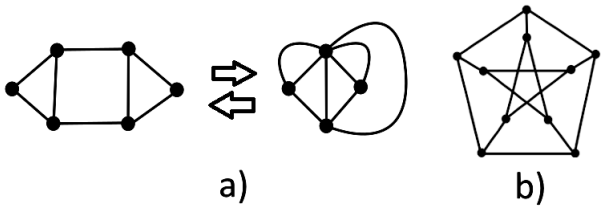

Fig. 2. a) Dual graphs b) Petersen (Cubic) graph.

Edge coloring is equivalent to vertex coloring of the edge graph. The process consists of two stages. The first one concentrates on constructing the appropriate edge graph, the second one is aimed at coloring graph vertices using any approximation method. Unfortunately, this generic approach is uneconomic. For the edge coloring problem dedicated techniques exist. More information on this topic can be found in [10].

Another aspect is total coloring of a graph, focusing on simultaneous coloring of vertices and edges. No two adjoining vertices, adjoining edges or edge incidental with a vertex can receive the same color. The smallest number of colors that allows for such coloring is the total chromatic number symbolized by $\chi''(G)$. Subsequently, $\chi''(G) \leq 2\Delta(G) + 1$. It was proven, that $\chi''(G) \leq \Delta(G) + 1026$.

Hence, all simple graphs can be divided into three types. The graph is of type 1, when $\chi''(G) = \Delta(G) + 1$, of type 2 when $\chi''(G) = \Delta(G) + 2$ and of type 3 when $\chi''(G) \geq$

$\Delta(G)+3$. Complete graphs K_n for odd values of n can be considered to be of type 1, while complete graphs K_n for even values of n are of type 2.

The third set contains such pairs of vertices and edges (v,e), for which v and e are incidental. Pairs (v,e) and (w,f) are adjoining, when $v=w$, $e=f$, edge $vw=e$ or $vw=f$.

This coloring model is applied in wireless transmission systems [11].

IV. Circular coloring

Colors are treated as natural numbers, but sometimes it is more convenient to see them as arcs on a circle. The circular coloring model exists both in vertex and edge form. the former is used to model the city traffic control on intersections with lights, where motion streams are defined by vertices. Conflicts that occur between them are represented by edges. The edge form is used for scheduling indivisible tasks [20]. The circular/star chromatic number of a graph is a natural generalization of the chromatic number of a graph [21].

Let p, q be integers and $p \geq q$. The coloring of a graph G is mapping a value c from $V(G)$ to a set $\{0,1,\ldots,p-1\}$, such that for any adjacent vertices $x,y \in G$, $q \leq |c(x)-c(y)| \leq p-q$. The circular chromatic number is then the greatest lower bound $\chi_c(G)$ of ratios p/q for which there exists a (p,q)-coloring. For any rational $p/q \geq 2$, $\chi_c(G)=p/q$. An example of a circular vertex coloring is depicted in Fig. 3.

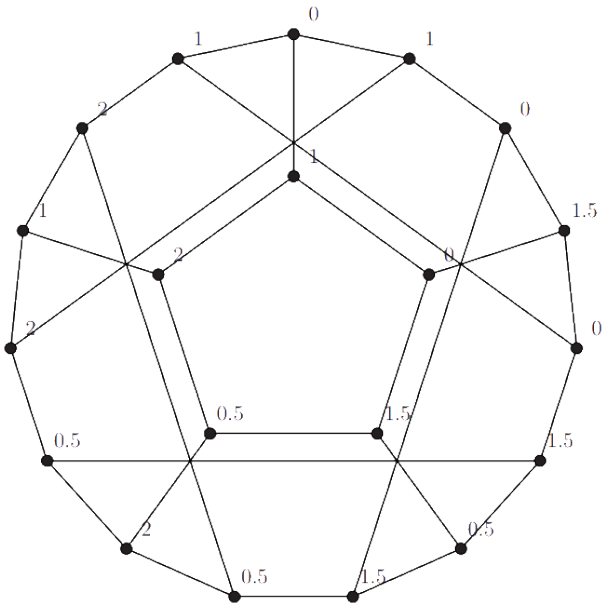

Fig. 3. Circular vertex coloring of a cycle of C_5 graph using 2.5 colors [13].

The main advantage of such a coloring model is the ability to acquire subchromatic colorings, which utilize a smaller number of colors than the $\chi(G)$. The circular chromatic number was also examined for with hypergraphs [22].

V. Coloring approximation methods

High computational complexity of the graph coloring problem forces usage of approximation methods to find suboptimal solutions in polynomial time. The problem is time consuming because it requires the overt analysis of all possible configurations of vertex-color assignments [23],[24],[25]. To color large graphs, heuristic algorithms are used. These methods color vertices in a reasonable time, but they do not guarantee optimal solutions.

The simple heuristic vertex coloring schemes take on the form of sequential algorithms. Single methods are browsed in a specific order, depending on the particular algorithm. Each visited vertex of the examined graph is associated with one of feasible colors. A color is feasible for a particular vertex, if none of its neighbors has been colored with it. All sequential algorithms presented below utilize in some way the Greedy Algorithm, which for a graph G and a certain established sequence of vertices $K=(v_1,v_2,\ldots,v_n)$ is as follows:

Algorithm ColorGreedy(G,K);
begin
for $v := v_1$ to v_n **do**
assign v the lowest possible color in a graph G;
end

A sequential coloring algorithm of a G graph is an algorithm that works as follows:

Algorithm Sequential(G)
begin
while not all vertices in G graph are colored **do**
select uncolored vertex v;
assign feasible color the v;
end while
end

Particular sequential algorithms differ from each other based on the order of vertex selection. The most popular are Largest First (LF) [26], Smallest Last (SL) [27] and Dsatur [28].

A. Random Sequential (RS) Method

The RS method, also called naive, is a sequential method, in which vertices are assigned in a random manner.

Algorithm ColorRS(G);
begin
$K :=$ random sequence of graph G vertices;
ColorGreedy(G,K);
end;

The RS method can be implemented in time that is proportional to the graph G size, that is $O(m+n)$.

B. Largest-First (LF) Method

LF method was proposed by Welsh and Powell [26]. It is one of the oldest and simplest sequential methods, which colors vertices in a non-increasing order.

Algorithm ColorLF(G)
begin

$K :=$ vetices G sorted according to nonascending ranks in a G graph;
ColorGreedy(G,K);
end

This method can be implemented in $O(m+n)$ time.

C. Smallest-Last (SL) Method

This vertex ordering proposed in [27] colors the smallest degree vertices in the following fashion. The minimum degree vertex is colored as the last one. Assume that $v_{k+1}, \ldots, v_n$ have been ordered. Choose v_k such that the degree of v_k in G induced by $V - v_{k+1}, \ldots, v_n$ is minimal.

D. Dsatur Method

In each step of the algorithm, the most saturated vertex to be colored is selected. The degree of vertex saturation denotes the number of color classes for which the vertex is adjacent to at least one other vertex in that class. If different vertices with the same degree of saturation exist, then vertex with the biggest number of edges is colored first.
Both LF and SL methods tend to color the higher degree vertices first. Even though both of them are easy to implement, they produce colorings far from optimal. The SL method optimally colors trees, unicycle graphs, cycles, complete bipartite graphs, Johnson graphs [31] and Mycielskis graphs [8]. It can be implemented in $O(m+n)$ time.
The sequential algorithms do not utilize more than $\Delta(G)+1$ colors. The SL algorithm colors planar graphs with 6 colors at most. In literature some modifications to sequential algorithms have been introduced to get better colorings while sacrificing the low computational cost. These variations are mainly based on neighbor coloring analysis [59] and color exchange within a subsection of vertices [27]. There have also been attempts to parallel graph coloring [30].
The optimal colorings with smaller amount of colors can be achieved by solving the Maximum Independent Set Problem (MISP). The vertex coloring results in division of graphs vertices into Independent Sets (IS). Each IS is colored with a different color. The essence of IS algorithms is based on successive determination of the maximal independent sets in a graph and assignment of different colors to vertices of particular IS.
Depending on the rule of vertex selection utilized for the purpose of IS construction, different algorithms can be obtained, like:

- Greedy Independent Set (GIS) [59] - method based on selection of a vertex with a minimal degree in an induced subgraph $G[X]$, containing only vertices from set X and edges connecting them.
- Recursive Largest First (RLF) [31] - method in which the vertex with the greatest degree from the subgraph $G[W]$ is selected as the first vertex from each of the newly created IS. Subsequent vertices added to the IS are the ones having the greatest number of neighbors in the $W \backslash X$ set.

The GIS rule aims at creating the largest IS in each iteration, while RLF tends to determine such IS so that the uncolored part of a graph has the least edges in relation to vertices. Computational experiments usually exhibit the superiority of RLF over GIS in producing a solution for MISP.

VI. Graph coloring representation and the cost function

Graph coloring is represented in two ways [14]:

A. Integer/direct representation

There are two methods of representing graph color by integers. If it is seen as a process of assigning colors to particular graph vertices, then a vector of natural numbers is created, in which the i-th element contains a color assigned to the i-th vertex (vector representation). On the other hand, if graph coloring is a distribution of vertices into two unassociated subsets, then coloring is represented by sets of vertices. The i-th set contains vertices colored using the i-th color (group representation) [15]. The reasoning here is as follows.
Let H be a group, while A be a subset of H such that $1 \notin A, A^{-1} \subseteq A$. For all $x, y \in H$, the relation ρ_A can be defined by $x \rho_A y$ if $xy^{-1} \in A$. Thus the group graph (H, ρ_A) is acquired.
Let $N = \{1, 2, \ldots, n\}$ be a vertex set of integers. Let $D = \{1, 2, \ldots, n\}$ be the distance set, such that $|N| \leqslant |D|$. Form the graph $G(N, D)$, whose edge set $E(G) = \{(x, y)$ if and only if $|x-y| \in D \forall x, y \in N\}$. More details on using the distance matrix to graph coloring are in [16] and [17].

B. Order-based/indirect representation

The solution is a sequence of n numbers, i.e. a permutation of n prime natural numbers. A proper decoder translates this permutation into parameters of the problem solution. In graph coloring, this permutation determines the order of graph vertex coloring. The decoder reads vertices in that order and colors the graph using the Greedy Algorithm. For a 4-vertex graph, the permutation $\{2, 4, 3, 1\}$ means that the vertices are colored in such an order. The vector representation is the most often used.
The criterion that determines the quality of the solution is the cost function $f : \Omega \to R$, where Ω describes the space of all possible colorings of a graph.

The cost function counts the number of edges with equally colored endings. In [18] the cost function for coloring p is defined as follows:

$$f(p) = \sum_{u,v \in E} q(u,v), \text{where } q(u,v) = \begin{cases} 1 & \text{when } c(u) = c(v) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The value of this function in the l-th iteration of the algorithm, is often calculated as an update of the cost function's value of the solution in $(l-1)$th iteration.

VII. Graph coloring with metaheuristics

Metaheuristics are universal calculating methods for discrete optimization that use random wandering technique to acquire the approximate result. Metaheuristics avoid being trapped in local optima by permitting moves that deteriorate the value of the objective function. Iterative algorithms like Simulated Annealing (SA) [32], Tabu Search (TS) [33], Genetic Algorithm (GA) [34] or Artificial Neural Networks (ANN) [35] differ from successive augmentation algorithms (like LF or SL) in that colors of individual vertices in a graph may change several times over the course of execution. The coloring techniques based on metaheuristics are used when the combinatorial optimization algorithms fail.
The iterative algorithm can be roughly described as follows:

begin
generate initial solution x;
while stop criterion was not achieved **do**
begin
generate new initial solution x created from the previous one;
end
end

A. Simulated Annealing (SA)

This technique is suitable for large-scale problems optimization, where a desired global extremum is hidden among many local extrema [36]. The solution of SA used in the GCP is a certain coloring of a graph, which does not have to be optimal. To use this algorithm, a neighborhood function $N : \Omega \to 2^{\Omega}$ is required. It defines a set of neighbors $N(x) \subseteq \Omega$ for each candidate solution x. Generating x (choosing a particular element from a set of neighbors) provides a vertex with a new color or color replacement between two vertices of a graph (atomic move). Unfortunately, finding a new solution in such a way is ineffective, because there is a high probability that after such a replacement the coloring will not fulfill all requirements or that it will be rejected after calculating the cost function. Therefore it is necessary to generate a solution that will instantaneously give good results.
Three different SA implementations in graph coloring are known: The Penalty Function Approach, the Kemp Chain Approach and the Fixed-K approach. Since the first method is the most popular, it will be presented in detail. For further information on the implementation of other methods, see [59]. The coloring solution is any partition of V into nonempty disjoint sets $C_1, C_2, \ldots, C_k$, $1 \leq k \leq |V|$, where the C_i are color classes. Two such solutions belong to the same neighborhood if one can be transformed to the other by moving a vertex from one color to another. A nonempty color class is denoted by C_{OLD}, with vertices $v \in C_{OLD}$, while k is the current number of color classes. The integer i meets condition $1 \leq i \leq k+1$. The neighbor is obtained by moving v to the color class C_i. If $i = k+1$, v is moved to a new, previously empty class. If v is already in class C_i, it is necessary to try again. The most important aspect of this implementation of SA to GCA is the cost function.

Let $\Pi = (C_1, \ldots, C_k)$ be a solution, while E_i, $1 \leq i \leq k$ be the set of edges from E, both of whose endpoints are in C_i. Then

$$cost(\Pi) = -\sum_{i=1}^{k} |C_i|^2 + \sum_{i=1}^{k} 2|C_i| \cdot |E_i| \tag{4}$$

There are different ways to generate the initial solution. One would be to start with all the vertices in a single class.
The frequency assignment problem has been the subject of extensive studies due to its application in cellular network technologies. The particular interest was placed on the trade-offs between bandwidth usage and system interference. A Simulated Annealing type heuristic has been utilized for the purpose of channel assignment for cellular radio in [49]. It has also been applied to the broadcast scheduling problem in [50] and was found to be effective.

B. Ant Colony Optimization (ACO) Algorithm

This algorithm is a swarm intelligence probabilistic technique designed specifically to find good paths in graphs. ACO can be used in GCP, considered as a task of dividing of a set of vertices into k independent sets $(V = V_1, \ldots, V_k)$, such that $k = \chi(G)$. The task of each agent (ant) is to color a graph vertices one by one using a sequential algorithm, like LF, SLF, SL or RLF. Information of the best coloring is kept in a square matrix A, updated periodically. For two adjacent vertices v_i and v_j, the value A_{ij} is proportional to the quality of coloring both vertices using the same color. Therefore the value A_{ij} signifies the amount of pheromone left on a path between vertices v_i and v_j. The substance collected in A evaporates with time, according to the parameter σ. After all agents have colored the graph for two adjacent vertices v_i and v_j, the value of A_{ij} is updated using the following formula:

$$A_{ij} = \rho A_{ij} + \Delta A_{ij}, \tag{5}$$

where

$$A_{ij} = \sum_{m_l \in M_{ij}} \frac{1}{k_l} \tag{6}$$

M_{ij} is a set of ants, which have colored the vertices v_i and v_j with the same color, while k_l is a number of colors used by an ant l.

Algorithm ACO(G);
begin
for each $v_i, v_j \notin E$ **do**
$A_{ij} := 1$
$f := \infty$;
for CycleNumber := 1 to MaxCycleNumber do begin
for each $v_i, v_j \notin E$ **do**
$\Delta A_{ij} := 0$;
for $a := 1$ to AntsNumber **do begin**
$s := V_1, \ldots, V_k$:=ColorGraph();
if $k < f$ then **begin**
$s* := V_1, \ldots, V_k$;

$f := k$;
end
for each $v_i, v_j \notin E$ and $v_i, v_j \subseteq Vj$ **do**
$\Delta A_{ij} := \Delta A_{ij} + 1/k$;
end;
for each $v_i, v_j \notin E$ **do**
$A_{ij} := \rho A_{ij} + \Delta A_{ij}$
end

ColorGraph() represents a sequential algorithm. Costa and Hertz presented in [37] two algorithms of such type based on SLF and RLF methods.
In [51] ACO method has been used to solve the communication network routing problem (CNRP). This solution is built upon the research conducted by [52].

C. Tabu Search (TS)

It is technique belonging to local methods of solution space search. TS was designed to guide other methods (or their component processes) to escape the trap of local optimality [33]. A given graph $G = (V, E)$ is partitioned into a fixed number of subsets, where obtainable solution is a partition $s = (V_1, V_2, \ldots, V_k)$ of the vertex V. The objective function f is defined as the number of edges of a graph $E(V_i)$ for which endpoints are in the same V_i, thus have the same color.

$$f(s) = \sum(|E(V_i)| : i = 1, ..., k) \tag{7}$$

A tabu list holds the candidate solutions observed so far. When the tabu list is too large, the oldest candidate solution is removed. In the simplest form of this algorithm, tabu list is of established length l (being the number of recently accepted moves). Subsequently, if the new solution substantially increases the value of the objective function, its tabu status is removed and it becomes the new best solution.

The stopping condition of TS is when the estimation f^* of the minimum value of the objective function $f(s)$ equals 0. From s the neighbourhood s' is generated as follows:

- a vertex x is randomly selected from all those, which are adjacent to an edge in $E(V_1) \cup \ldots \cup E(V_k)$
- vertex x is being assigned a random color $j \neq i$
- s' from $s = (V_1, \ldots, V_k$ is obtained by setting $V'_j = V_j \neq x; V'_i = V_i \backslash x; V'_r = V_r$ for $r = 1, \ldots, k; r \neq i, j$

The TS algorithm has the following form [38]:

Algorithm TS(G);
begin
generate the initial solution x;
$x* := x$;
while stop criterion has not been achieved **do begin**
find:
$x \in N(x)$ **such that** $f(x) < A(f(x*))$;
or
$x \in N(x)\ L(x)$ **such that** $f(x) < f(x)$;
if x does not exist **then**
find x **such that** $f(x) = minf(z) : z \in N(x)\ L(x)$;
if $f(x) < f(x*)$ **then** $x* := x$;
$x := x$;
UpdatingTabuList($x \rightarrow x$);
end
end;

The TS approach was utilized in [53] and [54] for optimizing the link capacities in a dynamic routing telecommunications network. Traffic between pair of vertices in a network varies with the time of day. Thus alternate routing paths are being creating from hour to hour. The TS heuristic solves the problem by utilizing probabilistic move selection and coordinated solution recovery strategies.

D. Genetic Algorithm (GA)

This method is based on the controlled evolution of a certain population of individuals. The basis is the recombination and propagation of the best individuals to next generations. The main features that differentiate GA from other metaheuristics are:

- utilization of genetic operators, that are adapted to the specific form of solution
- processing the whole population of solutions, which leads to parallel search of the solution space from different points

Initially GA randomly generates the initial population, in which each individual represents the initial graph coloring. For each solution, the distance between its encoded coloring and a legal coloring (where no vertices connected by an edge have the same color) is calculated. This value is denoted by the fitness function. For the given edges $i, j \in E$, let the function $\delta(i, j) = 1$ if $C_i \neq C_j$ and $\delta(i, j) = 0$ if $C_i = C_j$. The natural fitness function has the form:

$$F(C(G)) = \frac{\sum_{i,j \in E} \delta(i, j)}{m} \tag{8}$$

which is a fraction of edges which endpoints do not share the same color, divided by the total number of edges.
For every member of the population, the value of fitness function is calculated. In each iteration, until the stopping criterion is reached, the selection of individuals (parents) is carried out, based on which the crossover operation is performed. For each offspring, the value of fitness function is calculated.
The explored environment in GCP is the graph structure, while the individuals are particular graph colorings. The fitness function determines how good is a particular coloring.
The chromosomes of the individual are represented by an array of the length equal to the number of graph's vertices. Each element contains the number of the color assigned to this vertex. Connections between *n* vertices are represented by an *nxn* adjacency matrix. GA is terminated when a solution is found with no two adjacent vertices having the same color (zero conflicts) or the algorithm exceeds the predefined number of generations.

There are many methods of offspring population selection, the most popular being the roulette and the ranking methods [39]. There are specific crossover operators for graph coloring. The analysis of their efficiency in correlation with graph coloring is in [12] and [13].

E. Neural Networks (NN)

The subject of mapping the GCP into the neural networks was approached in [40]. Utilization of Hopfield of Hopfield-like networks for that purpose can be found in [41], [42], [43]. The problem of mapping the GCP into a Hopfield network is a reduction to MISP followed by a mapping of MISP into a Hopfield network as in Fig. 4 [44].

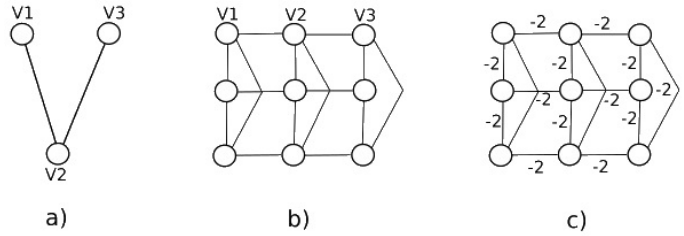

Fig. 4. a) Graph G, b) its 3-color reduction to MISP, c) Hopfield network instance [44].

This method is the function-based approach to GCP, where the ANN consists of n binary units (neurons) S_{ij}, where $j \in 1, \ldots, n$ is the vertex index in graph G, while $i \in 1, \ldots, k$ is the color number [40]. For a given graph $G(V, E)$ and k colors $1, \ldots, k$ let us assume that G is planar, which means that $k \leq 4$. For the unit (i, j) let $S_{ij} = +1$ denote that the i-th vertex has been assigned the j-th color, while $S_{ij} = 0$ denotes the opposite. This configuration allows for a vertex to have more than one color assigned to it. Between every pair of units (i, j), (k, j) in the same column j, $j = 1, \ldots, n$, set the weight $w_{i,j,k,j} = -2$. Between every pair of units (i, j), (i, k) in the same row i, $i = 1, \ldots, k$, set the weight $w_{i,j,i,k} = -2$ if v_i, v_k is an edge of graph G. The remaining weights (corresponding to units not connected by an edge) are zero. The ANN is composed of vertices with each having an output corresponding to the particular state S. As an input, each vertex receives the state of all its neighboring vertices. The algorithm minimizes a particular energy function $E(S)$, reducing the GCP to energy minimization. Various energy functions can be assumed, based on the specific problem. In [40], the following function has been designed to minimize the number of improperly colored edges in the spanning subgraph k-coloring problem:

$$E(\vec{S}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} \delta(S_i, S_j) \tag{9}$$

where $\vec{S} \in \{1, \ldots, k\}^N$ denotes the assignment of colors from 1 to k to all vertices of graph G, A_{ij} is the adjacency matrix with elements $A_{ij} = 1$ if $\{v_i, v_j\} \in E$ (and 0 otherwise), while $\delta(a, b)$ is the Kronecker delta function, providing output value 1 if a=b (a,b being integers) or 0 otherwise. $E(\vec{S}) = 0$ corresponds to a properly colored graph. There is one-to-one correspondence between a set of maximal independent sets of G in Fig. 4 (b) and the family of maximal sets of vertices of G colored properly (without violating constraints), which in turn corresponds with the discrete local minima of the Hopfield network [44]. With $k = \Delta(G) + 1$ in this mapping, every local minimum of the Hopfield network represents a proper coloring of the entire graph [45]. Every local minimum has the same energy value, despite representing a proper coloring of the entire graph. This prevents the ANN from optimizing its performance through the means of energy minimization. This can be corrected by adding bias value I_{ij} to force the network to use a smaller number of colors in a feasible solution. In Fig. 4(c) these units are equal to 1. Their pairs connected by a joint edge have weight equal to -2, while these not connected by an edge have weight equal to 0.
ANN forecaster has been used to predict voice traffic demand over an ATM network in [55]. A two-phased algorithm based on ANN and Genetic Algorithm (GA) was utilized to maximize slot utilization for the purpose of solving the broadcast scheduling problem in [56]. The length of the schedule is minimized in phase one by the means of Hopfield neural networks, while GA is used to maximize slot utilization.

VIII. On-Line Coloring

In the algorithm analysis theory, it is assumed that before beginning its operation, it has access to the whole set of data representing a certain instance (for example a graph) of the problem to be solved. This means that the algorithm operates in an off-line mode. For considerable amount of problems, due to their nature, this assumption can not be accepted. Solution is built based on incomplete information about the instance. A proper algorithm that works in an on-line mode receives input data as a set of requests $\sigma = (\sigma_1, \ldots, \sigma_n)$ and does handle each request σ_i instantaneously after receiving it. It is also assumed that handling a certain request is run without the knowledge of future requests and always leads to a partial solution. Once generated, the solution R_i cannot be modified after handling the request σ_i ends. Off-line algorithms know the "future", while on-line algorithms do not. They can be applied to solve issues concerning computer operating systems, telecommunication or task scheduling [46].

Let the graph G and a given arrangement π of the vertices set be denoted as the on-line presentation of a graph G. The number of colors used by the on-line algorithm A to color a graph G considering π is denoted as $A(G, \pi)$. The biggest number of colors used by A to color G denotes $\chi_c(G)$ and it is called the on-line chromatic number of G graph for algorithm A.

$$\chi_A(G) = max_\pi A(G, \pi) \tag{10}$$

Among algorithms intended to color any graph, two are known: First-Fit and LST.
In [57], the authors reduced the traffic load of high capacity cellular networks, by mapping the problem of caching popular files to a graph which is divided into four new subgraphs serving as the foundation of graph coloring in the presented

method. In [58], the authors presented a scheme to trackback DoS attacks based on packet marking. The trackback involves color balanced star coloring to assign color to routers. This coloring scheme minimizes the bit space required to color the packets.

A. First-Fit (FF) Algorithm

It is an iterative algorithm, making locally optimal decisions that do not have to lead to solutions that are globally optimal. The FF algorithm used to color vertices of a graph uses a greedy strategy, assigning each vertex the lowest possible color not used already on a neighbor.

Algorithm FF(G);
begin
for $i = 1$ to n **do**;
assign smallest legal color to v_i
end
end

The FF algorithm is the easiest and fastest of all greedy coloring heuristics, but it does not perform well in the worst-case [47].

B. LST Algorithm

This algorithm sequentially processes vertices and assigns each of them to a certain class of colors $D_1, \ldots, D_d$, called greedy sets [48]. This way the partial coloring is obtained. The algorithm then divides the uncolored vertices into subsets $C_1, \ldots, C_r$, which are called residual sets. Each edge that was not assigned to any greedy set is instantaneously assigned to one of residual sets Cj. The algorithm recursively tries to assign the same vertex to one of the greedy sets of subgraph H. In case of further failure, this vertex is assigned to one of the residual sets of H and so on, until it receives a certain color.

IX. Summary

The analysis provided in this paper shows there are no methods for comparing all graph coloring algorithms. There are no formal proofs supporting superiority of any method over the other corresponding to GCP. The efficiency of each algorithm depends on different parameters, specific for its nature. The size of the population and probability of execution of recombination operation in GA, the size of tabu list in TS or cooling schema in SA are a few examples. Selection of proper values influences the method's efficiency. Sequential implementations can take several hours to produce suboptimal results for relatively small graphs of a thousand vertices or less. The time required to reduce the number of colors by one increases significantly as better colorings are found.

It is difficult to design the algorithm providing good solutions to GCP on a wide range of inputs. None of current approaches dominate the others. The SA is a more preferable method in the channel assignment for cellular radio and broadcast scheduling problem. The communication network routing problem is usually solved by ACO or TS methods. For voice traffic demand over an ATM network prediction, ANN algorithms are utilized, usually with the support of discrete optimization methods, like GA.

References

[1] D. Zuckerman, "Linear degree extractors and the inapproximability of Max Clique and Chromatic Number", in *Proc. STOC06*, Seattle, 2006, pp. 681-690.

[2] T.R. Jensen and B. Toft ,"Graph Coloring Problems", New York, Wiley, 1995.

[3] K. Appel and W. Haken, "Every planar graph is four colorable. Pt. I: Discharging", *Illinois Journal of Mathematics*, vol. 21, pp. 429-490, 1977.

[4] I. Holyer, "The NP-completeness of edge-coloring", *SIAM Journal on Computing*, vol. 10, no. 4, pp. 718720, 1981

[5] G. Chartrand and P. Zhang, "Chromatic Graph Theory", Chapman and Hall, 2009

[6] V.G. Vizing, "On an estimate of the chromatic class of a p-graph", *Diskret. Analiz*, no. 3, pp. 25-30, 1964

[7] P.G. Tait, "On the coloring of maps", in *Proc. Royal Soc. Edinburgh Sec. A*, vol. 10, pp. 501-503, 1878-1880.

[8] J. Mycielski, "On the coloring of graphs", *Coll. Math*, vol. 3, pp. 161-162, 1955.

[9] M.M. Halldorsson, "A still better performance guarantee for approximate graph coloring", *Inf. Process. Lett.*, vol. 45, pp. 19-23, 1993.

[10] D. Brelaz, "New methods to color the vertices of a graph", *Communications of the ACM*, vol. 22, pp. 251-256, 1979.

[11] W. Arbaugh, S. Banerjee and A. Mishra, "Weighted Coloring Based Channel Assignment for WLAN's", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, issue 3, pp. 19-31, 2005.

[12] M. Kubale, "Optymalizacja dyskretna. Modele i metody kolorwania grafow", (in polish), Warszawa, WNT, 2002.

[13] P. Furmanowicz and K. Tanas, "A survey of graph coloring - its types, methods and applications", *Foundation of Computing and Decision Sciences*, vol. 37, no. 3, pp. 223-238, 2012.

[14] V. Yegnanarayanan, "Graph colourings and partitions", *Theoretical Computer Science*, vol. 263, pp. 59-79, 2001.

[15] K. Schnabel, "Representation of graphs by integers", in *Topics in Combinatorics and Graph Theory*, Physica-Verlag HD, Heidelberg, 1990.

[16] R.C. Entringer, D.E. Jackson and D.A. Smyder, "Distance in graphs", *Czechoslovak Math. J.*, vol. 26, no. 101, pp. 283296, 1976.

[17] I. Tomescu and A.M. Robert, "On distances in chromatic graphs", *Quart. J. Math.*, vol. 40, issue 4, pp. 475480, 1 Dec. 1989

[18] R. Dorne and J.K. Hao, "Tabu search for graph coloring, T-coloring and set T-coloring", in *Metaheuristics: Theory and Applications*, Kluwer Academic Publishers, 1997.

[19] D. Szyfelbein, "Genetic algorithms for graph coloring", in *Proc. 4th Conference Neural Networks and Their Applications*, pp. 605-610, 1999.

[20] X. Zhu, "Star Chromatic number and products of graphs", *Journal of Graph Theory*, vol. 16, issue 6, pp. 557-569, Dec. 1992.

[21] X. Zhu, "Circular chromatic number and graph minors", *Taiwanese Journal of Mathematics*, vol. 4, no. 4, pp. 643-660, Dec. 2000

[22] X. Zhu, "On the chromatic number of the products of hypergraphs", *Ars Combinatoria*, vol. 34, pp. 25-31, Jan. 1992.

[23] F. Hermann and A. Hertz, "Finding the chromatic number by means of critical graphs", *ACM Journal of Experimental Algorithmics*, vol. 7, pp. 1-9, 2002.

[24] I. Mendez-Diaz and P. Zabala, "A branch-and-cut algorithm for graph coloring", *Discrete Applied Mathematics*, vol. 154, pp. 826-847, 2006.

[25] D. de Werra and D. Kobler, "Graph coloring problems", in *Paradigms of Combinatorial Optimization*, ISTE-Wiley, pp. 265-310, 2010.

[26] D.J. Welsh and M.B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problem", *Comp. J.*, vol. 10, no. 1, pp. 85-86, 1967.

[27] D.W. Matula, G. Marble and J.D. Isaacson, "Graph coloring algorithms", in *Graph Theory of and Computing*, Academic Press, New York, pp. 109-122, 1972.

[28] D. Brelaz, "New methods to color the vertices of a graph", *Communications of the ACM*, vol. 22, pp. 251-256, 1979.

[29] D.S. Johnson, "Worst case behavior of graph coloring algorithms", in *Proc. 5th Conf. on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica Publishing, Winnipeg, pp. 513-527, 1974.

[30] G. Lewandowski and A. Condon, "Experiments with Parallel Graph Coloring Heuristics and Applications of Graph Coloring", Technical Report, 1994
[31] F.T. Leighton, "A graph coloring algorithm for large scheduling problems", *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489-506, 1979.
[32] S. Kirkpatrick, Jr.C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, pp. 671-680, 1983.
[33] F. Glover, "Tabu Search: A Tutorial", *Interfaces*, vol. 20, no. 4, pp. 74-94, 1990.
[34] J.H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, 1975.
[35] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, vol. 52, issue 3, pp. 141-152, July 1985.
[36] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, "Numerical Recipes in C", in *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, 2007
[37] D. Costa and A. Hertz , "Ants can color graphs", *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 295-305, March 1997.
[38] F. Glover, "Tabu Search - Part 2", *ORSA Journal on Computing*, vol.2, no. 1, pp.432, 1990.
[39] D.E. Goldberg, "Genetic Algorithms", Dorling Kindersley Pvt Ltd, 2008.
[40] A. DiBlas, A. Jagota and R. Hughey, "Energy function-based approaches to graph coloring", *Transactions on Neural Networks*, vol 13, no. 1, pp. 81-91, 2002.
[41] D.W. Gassen and J.D. Carothers, "Graph color minimization using neural networks", in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1541-1544, 1993.
[42] M.O. Berger, "k-coloring vertices using a neural network with convergence to valid solutions", in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 7, pp. 4515-4517, 1994.
[43] A. DiBlas, A. Jagota and R. Hughey, "Optimization neural networks on SIMD parallel computers", *Parallel Computing*, vol. 31, no. 1, pp. 97-115, 2005.
[44] A. Jagota, "An adaptive, multiple restarts neural network algorithm for graph coloring", *European Journal of Operational Research*, vol. 91, pp. 257-270, 1996.
[45] A. Jagota, "Scheduling problems in radio networks using Hopfield networks", in *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, Lawrence Erlbaun Associates, Hillsdale, NY, pp. 67-76, 1993.
[46] A. Fiat and G.J. Woeginger, "Online Algorithms The State of the Art", *Lecture Notes in Computer Science*, Berlin, Springer, 1998.
[47] M.M. Hall, "Frugal Methods for the Independent Set and Graph Coloring Problems", PhD. thesis, The State University of New Jersey, New Brunswick, New Jersey, October 1991.
[48] A. Gyarfas and J. Lehel, "First-Fit and on-line chromatic number of families of graphs", *Ars Combinatoria*, pp. 168-176, 1990.
[49] M. Duque-Anton, D. Kunz and B. Ruber, "Channel assignment for cellular radio using simulated annealing", in *IEEE Trans. Vehicular Technology*, vol. 42, pp. 1421, 1993.
[50] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing", in *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 250260, 1997.
[51] D. Zhaoa, L. Luo and K. Zhang, "An improved ant colony optimization for the communication network routing problem", *Mathematical and Computer Modelling*, vol. 52, pp. 19761981, 2010.
[52] C.H. Chen and C.J. Ting, "An improved ant system algorithm for routing problem", *Journal of the Chinese Institute of Industrial Engineers*, vol. 23, no. 2, pp. 115126, 2006.
[53] J. Xu, S.Y. Chiu and F. Glover, "Tabu search for dynamic routing communications network design", *Telecommunication Systems*, vol. 8, issue 1, pp. 55-77, 1997.
[54] J. Xu, S.Y. Chiu and F. Glover, "Probabilistic tabu search for telecommunications network design", *Combinatorial Optimization: Theory and Practice*, vol. 1, no. 1, pp. 69-94, 1996
[55] T. Edwards, D.S.W. Tansley, R.J. Frank and N. Davey, "Traffic Trends Analysis Using Neural Networks", in *Proceedings of International Workshop on Applications of Neural Networks to Telecommunications*, pp. 157-164, 1997
[56] J. Yeo, H. Lee, and S. Kim, "An efficient broadcast scheduling algorithm for TDMA ad hoc networks", *Computers and Operations Research*, vol. 29, pp. 17931806, 2002.
[57] M. Javedankherad and Z. Zeinalpour-Yazdi, "Content Placement in Cache Networks Using Graph-Coloring", 2017.
[58] A.S. Sairam, S. Roy and R. Sahay, Coloring networks for attacker identification and response, *Security Comm. Networks*, vol. 8, 751-768, 2015.
[59] D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning", *Operations Research*, Vol. 39, No. 3, May-June 1991, pp. 378-406