A Wide Classification of Graph Vertex Coloring

Dr A. Sri Krishna Chaitanya[1], **Dr P. Srilakshmi**[2]

[1]Associate Professor of Mathematics, Malineni Lakshmaiah Women's Engineering College, Puladigunta, Guntur, Andhra Pradesh, India
askc_7[at]yahoo.com

[2]Associate Professor of Mathematics, Malineni Lakshmaiah Women's Engineering College, Puladigunta, Guntur, Andhra Pradesh, India
sripasumarthi[at]gmail.com

Abstract: ***Graph vertex coloring is one of the most studied NP-hard combinatorial optimization problems. Given the hardness of the problem, various heuristic algorithms have been proposed for practical graph coloring, based on local search, population-based approaches and hybrid methods. The research in graph coloring heuristics is very active and improved results have been obtained recently, notably for coloring large and very large graphs. This chapter surveys and analyzes graph coloring heuristics with a focus on the most recent advances.***

Keywords: Coloring, Graphs, complete heuristic, recombination, configuration

1. Introduction

The graph coloring problem is one of the most important and the most studied problems in combinatorial optimization. The problem has applications in many domains, such as timetabling and scheduling, frequency assignment, register allocation, routing and wave length assignment, and many others.

In the following, we consider a non-oriented graph $G = (V, E)$, with a set V of n vertices and a set E of m edges . Given an integer k, a k-coloring c is a function that assigns to each vertex v of the graph aninteger $c(v)$ chosen in set $\{1, 2, ..., k\}$ (the set of colors), all vertices colored the same defining a "color class". A k-coloring c is a *proper* coloring if the endpoints of any edge are assigned different colors. A graph is k-colorable if it admits a proper k-coloring. The chromatic number □(G) of a given graph G is the smallest integer k for which G is k-colorable. A proper k-coloring such that $k =$ □(G) is named an *optimal* coloring. The *graph coloring* problem is the problem to find an optimal coloring of a given graph. For a given graph G and a given integer k, the *k-coloring*problem is the problem to determine if G is k-colorable and, if it is the case, to finda proper k-coloring of G.

Graph coloring is NP-hard and k-coloring is NP-complete for any integer $k \geq 3$ (but 2-coloring is polynomial). Therefore, no algorithm can solve graph coloring in polynomial time in the general case (assuming that N /= NP). In addition, note that the problem to find a k-coloring with no more than twice the optimal number of colors is still NP-hard. Also, finding an optimal coloring turns out to be particularly difficult in practice. As a matter of fact, there are graphs with as few as 125 vertices that cannot be solved optimally even by using the best performing exact algorithms. For larger graphs, it is therefore necessary to resort to *heuristics*, i. e., algorithmic techniques that provide sub-optimal solutions within an acceptable amount of time.

A simple and classical way to generate a sub-optimal proper coloring is to use a greedy heuristic. A greedy coloring heuristic builds a solution step by step by fixing on each step the color of a given vertex. In the *greedy sequential heuristic*, the order in which the vertices are colored is determined beforehand (i. e., statically), randomly or according to a specific criterion. On each step, the considered vertex is assigned the smallest possible color number such that no conflict is created with the already colored vertices. More efficient greedy heuristics like DSATUR [5] and Recursive Largest First (RLF) employ refined rules to dynamically determine the next vertex to color.

Greedy heuristics are generally fast, but they tend to need much more colors than the chromatic number to color a graph. Better results can be obtained by using much more powerful heuristics, such as notably *local search heuristics* and *evolutionary algorithms* – a survey of heuristics and metaheuristics is proposed in [4]. In fact, almost all of the most popular metaheuristics have also been applied and implemented on the graph coloring problem. From a historical point of view, simulated annealing and tabu search are among the first local search approaches that were successfully applied to graph coloring. In local search, a candidate solution is progressively improved by local transformations (named "moves"). Population-based approaches (e. g., memetic algorithms, quantum annealing) represent another family of heuristics that established them self as one of the most effective coloring methods. Thanks to the introduction of a pool of solutions, powerful search operators are made available like various solution recombinations and hybridization with local search. Finally, a latest approach based on independent set extraction and progressive coloring proves to be quite useful to color very large graphs.

In the following sections, we present the different coloring heuristics grouped in four categories: Local search heuristics in Sect.3, evolutionary algorithms in Sect.4, independent set extraction approaches in Sect.5, and other heuristics (such as quantum annealing, neural networks, ant colonies, etc.) in Sect.6. Then, we review ex-tensions of graph coloring problems and applications related to graph coloring in Sect.7. Finally, Sect.8 reports on standard benchmark graphs along with the results obtained by the best performing coloring heuristics on these graphs.

2. General Solution Strategies Applicable to Coloring Problems

It is important to note that the search space and the evaluation function used by a heuristic are not necessarily the same as those of the original problem. In the following, we name “strategy” a particular way to define the search space and the evaluation function. Thus, a strategy can be seen as a reformulation of the original problem. Solution strategies designed for *k*-coloring are presented below.

- In the *k-fixed penalty strategy*, a configuration is any (non necessarily proper) *k*-coloring and its cost is the number of conflicts – i. e., edges whose endpoints are assigned the same color. This strategy makes it possible to use a very simple type of move, named “1-moves”. A 1-move consists in changing the color of a single vertex.
- In the *k-fixed partial proper strategy (*named *Impasse neighborhood)*, a configuration is any partial (proper) *k*-coloring – a partial proper *k*-coloring is similar to a proper *k*-coloring except that some vertices may remain uncolored. The cost of a given configuration is the number of unassigned vertices – it can also be the sum of the degrees of unassigned vertices. A move (named “*i*-swap”) consists in assigning a color *i* to an uncolored vertex *v* and, at the same time, to deassign all neighbors of *v* whose color is *i* in the current configuration.

A *k*-coloring heuristic can also be used to solve the graph coloring problem. This can be done as follows: First, find a proper *k*-coloring by using a greedy heuristic. Then apply the *k*-coloring heuristic and reduce the value of *k* each time a proper *k*-coloring is found. Strategies designed to tackle directly the graph coloring problem are briefly presented below.

- In the *proper strategy*, a configuration is any proper *k*-coloring. The evaluation function is designed so as to promote the variance of the size of the different color classes and, as a side-effect, to reduce their number. A move (named a
- “Kempe chain”) consists in making an exchange of vertices between two color classes, while preserving the legality of the solution.
- In the *penalty strategy*, a configuration is any *k*-coloring. The evaluation function is designed so as to decrease both the number of conflicts and the number of colors. The moves used in this strategy are 1-moves (defined above).
- In the *order-based strategy*, a configuration is any ordering of the vertices, which is subsequently “decoded” (i. e., transformed into a proper coloring) by using the greedy sequential heuristic.
- In the *edge orienting strategy*, a configuration is any orientation of the edges of the input graph, and its cost is the length of a longest path in the resulting digraph. Types of move mechanisms applicable to this strategy are presented.

3. Local Search Heuristics

Local search is a simple yet very powerful approach. The most basic local search heuristic is iterative improvement. An *iterative improvement* procedure (or *descent*) consists in choosing on each iteration a move that produces a decrease in the cost function until a *local optimum* is reached. Unfortunately, iterative improvement may get trapped rapidly in a poor local optimum. Advanced local search heuristics resort to different kinds of mechanisms in order to circumvent the problem posed by local optima. For example, *simulated annealing* relies on randomness in order to allow “uphill” moves (i. e., moves that increase the cost function), thanks to the *temperature* parameter. *Tabu search* uses a short term diversification mechanism named a tabu list. *Iterated local search* and *variable neighborhood search* combine descents and perturbations (equivalent to jumps performed in the search space). Prominent local search heuristics proposed for coloring graphs will be presented and analyzed in the remaining of this section.

3.1. Local Search Heuristics Proposed for Coloring Graphs

Two heuristics based on the *k*-fixed penalty strategy were proposed in 1987: A simulated annealing algorithm in [9] and a tabu algorithm named TABUCOL. The tabu mechanism used in this latter algorithm is as follows: After performing a move (i. e., changing the color of a vertex), it is forbidden to reassign to this vertex its former color for a few iterations. In spite of its simplicity, this technique turned out to be remarkably efficient. In particular TABUCOL outperformed the simulated annealing heuristic proposed in [9].

An exhaustive experimental study is conducted in order to evaluate and compare several graph coloring heuristics, notably three simulated annealing heuristics based on three different strategies. Note that none of those strategies (the *k*-fixed penalty strategy, the proper strategy, and the penalty strategy) dominated clearly the others in these experiments.

One of the first and most efficient local search coloring heuristic is the simulated annealing algorithm proposed as its temperature parameter remains constant, this algorithm is named a Metropolis algorithm. This Metropolis algorithm is based on the *k*-fixed partial proper strategy and it combines two types of moves: *i*-swaps and *s*-chains – a *s*-chain is a generalization of Kempe chains. More recently, a tabu heuristic (named PA R T I A L C O L) also based on the *k*-fixed partial proper strategy has been developed in [3]. Experiments conducted with TABUCOL and PA R T I A L C O L indicate that, although they are based on different strategies, the two tabu algorithms obtain similar results on most graphs.

3.2 Recent Advances

Several authors have tried to improve the efficiency of a tabu coloring heuristic, notably by a more careful management of the tabu list. The original TABUCOL uses a static tabu list (i. e., the tabu tenure is constant). However, it can be noticed that the size of the neighborhood changes

throughout the search. For this reason, it is suggested to set the tabu tenure accordingly. A still more sophisticated and robust technique proposed in [3] allows tuning automatically the tabu tenure throughout the search.
A different idea makes it possible to improve the efficiency of TABUCOL by modifying its evaluation function. While any edge violation entails the same cost in the original algorithm, a specific penalty is assigned to each edge. Two new evaluation functions are proposed. In the first one, the penalty assigned to a given edge depends on the degree of its endpoints – edges whose endpoints have a larger degree (assumed to be more difficult to satisfy) are assigned a larger penalty. In the second one, the penalty of each edge is dynamically adjusted during the search.

It is proposed to guide a local search operator throughout the search space by measuring distances between solutions. Two algorithms that exploit this idea are presented. The first one uses a learning process to guide a basic tabu operator (TABUCOL) toward yet unvisited spheres. The second algorithm makes deep investigations within a bounded region by organizing it as a tree-like structure of connected spheres. Experiments show that these algorithms outperform the under-lying tabu algorithm and obtain remarkable results.

The iterated local search and variable neighborhood search metaheuristics have also been investigated for coloring graphs. The variable neighborhood search algorithm proposed in [2] uses TABUCOL as a local search operator and applies different types of perturbation operators. Also, an iterated local search algorithm is presented in [10].

A new local search coloring heuristic, named “Variable Space Search” (VSS-Col), is proposed. The VSS-Col heuristic alternates the use of three different lo-cal search heuristics that adopt different strategies: TABUCOL, PA R T I A L C O L, and a third tabu algorithm, which is not competitive by itself but complements adequately the two others. This algorithm obtains better results than each of its components and produces some of the best results obtained with local search heuristics.

Several other graph coloring local search heuristics have been proposed recently.

In particular, large-scale neighborhoods are explored in.

References

[1] Aardal, K., van Hoesel, S. P. M., Koster, A. M. C. A., Mannino, C., Sassano, A.: Models and solution techniques for frequency assignment problems. Ann. of Oper. Res.1**53** (1), 79–129 (2007)
[2] Avanthay, C., Hertz, A., Zufferey, N.: A variable neighborhood search for graph coloring. Eur. J. of Oper. Res.1**51** (2), 379–388 (2003)
[3] Blo¨echliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Comput. & Oper. Res.3**5** (3), 960–975 (2008)
[4] Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surveys, **35** (3), 268–308 (2003)
[5] Bre´laz, D.: New methods to color the vertices of a graph. Commun. of the ACM **22** (4), 251– 256 (1979)
[6] Bui, T. N., Nguyen, T. V. H., Patel C. M., Phan K.-A. T.: An ant-based algorithm for coloring graphs. Discrete Appl. Math.1**56** (2), 190–200 (2008)
[7] Chaitin, G. J.: Register allocation and spilling via graph coloring. ACM SIGPLAN Notices
[8] **17** (6), 98–105 (1982)
[9] Chalupa, D.: Population-based and learning-based metaheuristic algorithms for the graph col-oring problem. In: N. Krasnogor, P. Lanzi (eds.) Proc. of the 13th annual Genet. and Evol. Comput. Conf. (*GECCO, Dublin, Ireland, July 12–16, 2011)*, pp.465–472. ACM Press, N. Y., USA (2011)
[10] Chams, M., Hertz, A., de Werra, D.: Some experiments with simulated annealing for coloring graphs. Eur. J. of Oper. Res.3**2** (2), 260–266 (1987)
[11] Chiarandini, M., Stu¨tzle, T.: An application of iterated local search to graph coloring. In: D. Johnson, A. Mehrotra, M. Trick (eds.) Proc. of the Comput. Symp. on Graph Color. and its Gen. (*COLOR, Ithaca, N. Y., USA, Sept.7–8, 2002)*, pp.112–125 (2002)