Balancing Efficiency and Accuracy: A Comparative Framework of Heuristic and Exact Methods for Graph Coloring in Complex Networks

Dr. Mahaboob Ali

Assistant Professor
Govt. First Grade College for Women, Raichur
Karnataka-India

Abstract:
An old classic in combinatorial optimisation, the graph colouring problem (GCP) seeks to determine the fewest colours that can be given to vertices in a graph without causing any two neighbouring vertices to have the same colour. Numerous scholars from other disciplines, including mathematics, computer science, and biology, have delved deeply into GCP. Several heuristic algorithms have been suggested as solutions to GCP due to its NP-hardness. Current GCP techniques, on the other hand, are only applicable to sparse networks of high size (up to 10^7 vertices) or to tiny, complex graphs. Solution quality, computational efficiency, scalability, and durability across various graph topologies are the focal points of this paper's comparison approach, which evaluates heuristic and precise techniques for graph coloring—statistical techniques, including heuristics and Integer Linear Programming (ILP). The framework proposes hybrid solutions that combine heuristic limits with accurate refinement to achieve balanced performance in complex network environments.

Keywords: Graph Coloring, independent set, heuristic, exact algorithm, approximate algorithm, sequential algorithm, and parallel algorithms.

I.INTRODUCTION

The goal of graph colouring is to ensure that adjacent nodes in a graph have distinct colours when they are assigned to them. Numerous interesting applications exist for the fundamental NP-hard issue of colouring a graph with the shortest feasible number of colours, including scheduling, code optimisation, and seating layouts [1]. There is a lack of cohesion in the literature about various methods for this and similar computational issues. There is a critical role for GCP in theory and practice. The optimal control problem (GCP) is a standard NP-hard problem. "Graph colouring is the subject of several well-known mathematical conjectures, including the four-color conjecture [2], Hajós' graph-coloring conjecture [3], and Kneser's conjecture [4]." Many real-world issues in domains such as operations research, computational biology, power systems, communication networks, machine learning, scheduling, and routing and wavelength assignment can be modeled using GCP in practice. Taking the simplest scheduling issue as an example, it aims to find the smallest number of time slots to execute n activities, with the caveat that no two processes may utilise the same time slot if they share common resources, such as an exam classroom. Two tasks are considered neighbouring if they share similar resources; this allows us to describe the issue as a graph G, where the vertex set represents the set of jobs. Since it is possible to execute vertices (jobs) of the same colour in the same time slot, the chromatic number of G is the smallest possible number of time slots [12]. There is a direct correspondence between this issue and the resource-constrained project scheduling problem [13]. It is crucial to develop practical algorithms for addressing GCP due to its both theoretical and practical importance. "A lawful k-coloring of G is a mapping $c : V \rightarrow \{1,..., k\}$, such that $c(i) \neq c(j)$ for all edges (i, j) in E, given a simple undirected graph $G = (V, E)$ with vertex set $V = \{1, 2,..., n\}$ and edge set $E \subset V \times V$." Finding out if there is a valid k-coloring of G for a certain k is known as the graph k-coloring problem (k-GCP). "A lawful k-coloring of G exists for the minimal integer k, denoted as the chromatic number $\chi(G)$, and this is known as the classical graph colouring problem (GCP)." The optimisation issue GCP is known to be NP-hard, while k-GCP is NP-complete [13].

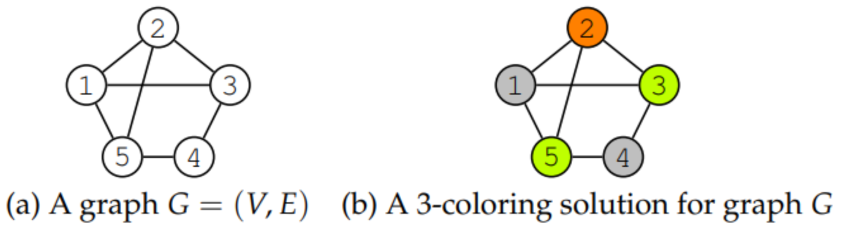

Figure 1.1 – A graph and its 3-coloring solution

A lawful three-coloring solution is illustrated on the right side of Figure 1.1, while the left side displays an undirected graph G with five vertices V = {1, 2,..., 5}. Graph colouring involves assigning a distinct colour to each vertex. Vertices 1 and 4 are grey, vertices 3 and 5 are green, and vertex 2 is orange in this case. With a chromatic number of 3, this 3-coloring is G's ideal coloring. As an alternative way of looking at the GCP, consider it a grouping issue. "In this case, the set of vertices is divided into k groups, and no two neighbouring vertices i and j can be in the same group". The solution is a k-coloring. An example of this is the grouping of the three colours seen in the top right corner of Figure 1.1, which would appear as {1, 4}, {2}, and {3, 5}.

Scholars have paid much attention to k-GCP, an NP-complete topic in graph theory that is quite popular [14]. "Graph colouring problem (GCP) is one of the three main topics of various international competitions, including the famous Second DIMACS Implementation Challenge on Maximum Clique, Graph Colouring, and Satisfiability". GCP is also found in many practical applications, including scheduling, timetabling, frequency assignment, and register allocation. You may find in-depth analyses of graph colouring techniques in [19].

II.RELATED WORK

Hertz and de Werra presented Tabucol, a regular tabu search technique, to solve GCP in 1987 [20]. In the first algorithms, the goal was to find a k-coloring that had conflict edges (edges with the same colour at both ends) and then remove them by exchanging colours in a particular order, with the tabu represented by the colour c for a vertex v. You cannot use c′ to colour v on later iterations if you replace c with it [21]. "To improve the performance of the proposed heuristic and the Tabucol heuristic, Blöhliger and Zufferey (2008) [22] introduced Foo-Partialcol, a colouring of a graph G that is constructed by extending partial solutions, and a reactive tabu tenure". On the other hand, the reactive banned search necessitates the time-consuming updating of the forbidden list. Hence, massive graphs are inefficient for Foo-Partialcol. To colour massive graphs, Wu and Hao (2012) [23] suggested the Extracol technique, which incorporates an adaptive tabu search approach to discover several pairwise disjoint independent sets and a memetic algorithm [24] to colour the remaining graph after removing these sets. Finding a better solution took Extracol five hours, which is slow and unsuitable for industrial applications, even if it outperformed prior methods on eleven big networks (with 1000 to 4000 vertices). The hybrid heuristic, first proposed by Marappan et al. [25] and Moalic et al. [26] in 2018, combines a genetic algorithm with a tabu search to solve the GCP. "The goal is to use the genetic algorithm to identify better solutions while the tabu search accelerates convergence."
Nevertheless, the algorithm's ability to execute the evolutionary process depends on the parameter values, which impact its applicability to large networks, and it necessitates a considerable amount of computation. To solve GCP, Dokeroglu et al. (2021) [27] suggested Tlbo-color, an improved version of the modulo teaching optimisation method that uses parallel metaheuristics and a robust prohibited search. Tlbo-Color outperformed its predecessors and performed well on complex, big graphs. However, getting a better answer with Tlbo-Color usually takes over 30 minutes, and many factors need to be adjusted. Restarting the local search from a fresh point established by the crossover operator does not always result in capturing a promising area with the aforementioned hybrid heuristic methods. This problem was addressed by Goudet et al. (2022) [28], who proposed a tabu search technique to find a better solution by utilizing feature information extracted from graph structures using a deep learning neural network. Because training the deep learning model requires a lot of computational resources, the technique converges slowly and has hardware limitations in terms of performance. A hybrid method called GC-SLIM was recently proposed by Schidler and Szeider [29] to address GCP. This technique combines tabu search with SLIM, a local improvement

approach based on SAT [30]. By choosing small subgraphs and ideally coloring them using SAT solvers like Glucose [31] and Cadical [32], GC-SLIM repeatedly refines the existing coloring. The approach reliably produces high-quality colouring solutions and performs well on thick graphs. To handle large-scale graphs, it encounters significant obstacles, especially in memory-constrained contexts, caused by the need to build an adjacency matrix for the graph.

A local search technique called Recolour was proposed by Rossi et al. (2014) [323] to enhance the quality of solutions obtained through greedy colourings using various vertex-ordering strategies. Due to its lack of reduction rules and poor performance on complex graphs, the method struggles with massive graphs. To determine the lower and upper limits on the chromatic number, Verma et al. (2015) [34] employed a variety of greedy heuristics. They presented reduction methods for GCP based on the k-core and k-community concepts. This approach was practical on sparse, basic networks but failed miserably on challenging graphs. To colour large-scale sparse graphs, Lin et al. (2017) [35] developed the FastColor method using an iterative greedy colouring approach. They suggested a reduction mechanism based on a lower constraint on the chromatic number. One hybrid colouring method, LS+I-DSatur, was presented by Hebrard et al. (2019) [36]. This approach uses graph degeneracy to calculate upper limits and cliques to derive lower bounds. First, it employs several reduction strategies to simplify the graph. "Then, it employs DSatur and Tabu search techniques to improve the upper limit, and last, it uses iterated DSatur (I-DSatur) to do the precise colouring. Since it uses tabu search, it may also be called a Tabu-based heuristic."

III.EXACT ALGORITHM

Typically, branch-and-cut or branch-and-price techniques with linear programming relaxations form the basis of exact algorithms for graph colouring. Many different methods exist for graph colouring. The eight graph colouring encodings and their respective integer programming formulations are summarised in Table 1.1. In particular, a formulation similar to natural assignment was employed by Coll et al. [37] and Zabala et al. [38]. A binary encoded formulation was investigated by Lee [29] and Lee and Margot [40]. For their formulations, [41] and [42] have relied on separate sets. Analysed a formulation that was subject to limits on precedence [43]. In their 2004 study, Barbosa et al. investigated acyclic orientation-based encodings. One such formulation was put forward by Campêlo et al. [44] using asymmetric representations. In [45], a novel method based on "supernodes" was presented. A precise method was suggested in [46] using the problem's well-known set covering formulation. Using propositional logic to leverage implicit restrictions, Zhou et al. [47] studied an learning exact method and demonstrated excellent computing performance on DIMACS benchmark examples. Some research on graph colouring has also focused on studying graph structure or graph decomposition. To study the GCP, for example, [48] employed a split decomposition tree, which recursively divides a graph into smaller subgraphs until they can no longer be divided. Solutions are progressively merged to get the graph's solutions after colouring the prime graphs that cannot be separated. Given that graph colouring and clique partitioning are conceptually similar, [49] investigated clique partitioning for graphs and offered two strategies for clique partitioning that outperform specific fast graph colouring algorithms. [50] introduced a linear decomposition-based exact graph colouring technique that outperforms competing exact algorithms with short linear widths. Additionally, the chromatic polynomial and the count of optimal solutions have been the subject of several research efforts. The chromatic polynomial was obtained by using the time-honoured technique of "deletion contraction" in [51], which makes use of the graph-altering properties of chromatic polynomials. [52] studied a time-complexity-optimal approximation technique for calculating the chromatic polynomial given its upper and lower bounds. The TexaCol method, as suggested in reference [53], is capable of obtaining both the chromatic polynomial and all solutions for graph coloring. The TexaCol method finds all possible colouring solutions by first breaking the network into maximum cliques and then analysing the relationships between them. The two precise graph colouring techniques proposed by [54] were used to enhance TexaCol. Partial best solutions are achieved by two algorithms: the Exact Graph Colouring algorithm (PexaCol) and the All Best Solutions Exact Graph Colouring algorithm (AexaCol). These two algorithms draw inspiration from TexaCol and employ the backtracking technique to colour sequentially until either all or a portion of the best solutions have been found.

Table 1.1 – Integer programming formulations of graph coloring

BASED ON	VARIABLES	CONSTRAINTS	REFERENCES

Vertices (Standard)	k|V|	|V| + k|E|	Méndez-Díaz and Zabala [37] Zabala and Méndez-Díaz [38] Coll et al. [37]
Binary encoding	[$\log_2$k] |V|	Exp. many	Lee [39]
Max. independent sets	Exp. many	|V| + 1	Mehrotra and Trick [41]
Any independent sets	Exp. many	|V| + 1	Hansen et al. [55]
Precedencies	O(|V|2)	|E|	Williams and Yan [43]
Acyclic orientations	|E|	Exp. many	Barbosa et al. [56]
Asymmetric representation.	O(|E|)	O(|V||E|)	Campêlo et al. [44]
Supernodes	k|Q|	|Q| + k|E|	Edmund K. Burke et al.[16]

A. Integer Linear Programming

Linear programs are used in optimisation problems when a linear function and a set of linear constraints are provided. In a linear minimisation program, finding the function's minimum value is the goal.

Represent the arrays c and b as columns in Qm, with n and m belonging to N∏ and A belonging to Qn×m. The transposed array c is represented by cT. What follows is a definition of both the canonical and standard linear program formulations.

First Definition: *A linear program with parameters A, b, and c is traditionally expressed as*
Minimize $z = c^T x$
subject to $Ax \geq b$
$x \geq 0$

Second Definition: *A linear program is often described by the variables b, c, and A, and expressed as*
Minimize $z = c^T x$
subject to $Ax = b$
$x \geq 0$

Adding slack variables to a canonical formulation makes it a standard formulation. For more information, refer to [57].

A linear program is considered integer linear if and only if all of the values of x in the program are integers. An integrality gap exists when the value of a linear program's non-integer optimum solution is less than the value of its integer optimal solution. One of the polynomial problems is solving linear programs, and the Simplex solution is the most well-known solution for this kind of issue [58]. "However, a Branch-and-Bound method coupled with the Simplex is required to optimally solve an integer linear program, making it an NP-Hard problem [59]."

ILP formulations for the graph coloring problem

Equations (2a), (2b), (2c), (2d), and (2e) provide Formulation (2), which provides a first formulation for the graph colouring issue as an ILP, for a graph G where n = |V(G)| and m = |E(G)|.

$$\begin{aligned}
&\text{Minimize} \sum_{j=1}^{n} x_j && && \text{(2a)} \\
&\text{subject to:} \sum_{j=1}^{n} y_{vj} = 1, && \text{for all } v \in V(G) && \\
& && \text{and } j \in \{1,\ldots,n\} && \text{(2b)} \\
&y_{vj} + y_{uj} \leq x_j, && \text{for all } \{v,u\} \in E(G) && \\
& && \text{and } j \in \{1,\ldots,n\} && \text{(2c)} \\
&y_{vj} \in \{0,1\}, && \text{for all } v \in V(G) && \\
& && \text{and } j \in \{1,\ldots,n\} && \text{(2d)} \\
&x_j \in \{0,1\}, && \text{for all } j \in \{1,\ldots,n\} && \\
& && && \text{(2e)}
\end{aligned}$$

For every colour $_j$ in this formulation, there is a single binary variable x_j that indicates whether or not that colour is in a solution. Each variable yvj indicates the assignment of colour j to vertex v. Since the ideal solution allocates the fewest number of vertex colours, the goal function represents a minimisation integer linear program. Vertices can only have one colour allocated to them according to the set of criteria in (2b). Two of the conditions in equation (2c) state that no two vertices on the same edge may have the same colour.

Even though there are a polynomial number of variables and constraints in (2), it produces a factorial number of symmetric solutions. When many values may be used to express the same answer, we say that the solution is symmetric. As Lewis [1] reveals, it causes the Branch-and-Bound algorithm's state space search tree to become too enormous.

According to Mehrotra and Trick's suggested Formulation (3), which they refer to as equations (3a), (3b), and (3c) [60].

$$\text{Minimize } \sum_{S \in \mathcal{S}} x_S \qquad (3a)$$

$$\text{subject to: } \sum_{\{S: v \in S\}} x_S \geq 1, \qquad \text{for all } v \in V(G) \qquad (3b)$$

$$x_S \in \{0,1\}, \qquad \text{for all } S \in \mathbf{S} \qquad (3c)$$

The set of all possible independent sets of a graph is denoted by S in Formulation (3), and the binary variable xS shows whether a set S ∈ S is a solution or not. The chromatic number of a graph is equal to the smallest set cover, according to the objective function in (3a). For every requirement in (3b), there must be a maximum independent set S that contains every vertex v. The issue of symmetry is sidestepped despite the exponential number of variables in this approach.

IV.HEURISTIC ALGORITHM

Heuristic and metaheuristic approaches are essential since exact algorithms may fail in several contexts. What follows is a discussion of many typical colouring methods that rely on heuristics. Both DSATUR [61] and RLF [62], two famous greedy algorithms, use revised rules to decide which vertex to colour next dynamically. Speed is often not an issue for these greedy heuristic methods. That is why hybrid algorithms often use them as initialisation techniques. Tabu search, one of the most widely used local search methods for the GCP, was proposed in [63]. The tabu search works on the premise of enhancing the colouring by performing local alterations, starting with an initial solution. One major drawback of local search algorithms is their inability to compete with hybrid population-based algorithms due to their limited exploitation of global information. Graph colouring local search techniques were summarised in [64]. This is why hybrid algorithms, including hybrid evolutionary algorithms, often use the tabu search as a subroutine [65]. Among the most successful methods for graph colouring, population-based hybrid algorithms [66] have shown the best results on the majority of the challenging DIMACS cases. By employing operators such as recombination, crossover, or mutation, population-based hybrid techniques can generate a large number of solutions.

Furthermore, population-based algorithms often incorporate specialized diversity preservation mechanisms that compute an appropriate distance metric between solutions [67] to preserve population variety, which is crucial for avoiding premature convergence. Combining a relevant recombination operator with an efficient local optimisation technique and a mechanism for sustaining population variety is crucial for the success of hybrid algorithms. Outlined methods for "Reduce and solve" problems [68]. In most cases, the "Reduce and solve" paradigm will have both a preprocessing and colouring stage. To obtain a reduced graph, the pre-processing step identifies and removes specific nodes (typically independent sets) from the original graph. Next, we use the colouring phase to find the right colouring for the reduced graph. On large and very massive graphs, the "reduce and solve" methods work exceptionally well, according to empirical data. While "Reduce and solve" methods excel at tackling big graphs, they struggle with smaller and medium-sized graphs. Furthermore, the vertex extraction procedure and the colouring algorithm in use are crucial to the effectiveness of such approaches. "Different methods exist for encoding the GCP; they include a modified cuckoo algorithm [70], a grouping hyper-heuristic algorithm [71], a multi-agent based distributed algorithm [72], a learning-based heuristic algorithm [73]."

A. Randomized Adaptive Search heuristic for LCP (GRASH_LCP)

Initially, every vertex is added to a candidate list (CL). Equation (2) is used to create an evaluation function, eval, for all vertices in CL. An RCL is created for vertices whose degree is equal to or higher than eval. In order to get a diversified answer, several values of α are used to construct more than one limited candidate list. We take the union of all RCL and randomly assign one vertex (let us call it v^1), the lowest accessible colour (let us call it c^1) from the set of colours (L^{v1}) associated with that vertex. Obtaining the set of neighbouring vertices N (v^1) of v1 is the first step in assigning a colour to that vertex. Then, after updating the colour lists of all the vertices in N (v^1), we remove the colour c1 from their lists. Additionally, the candidate list is refreshed by deleting vertex $v^{1.}$ To colour the next vertex, the procedure is repeated. When vertices are coloured, the colour lists of the vertices immediately nearby are reduced. A dead end occurs when the colour list for any vertex becomes empty during the operation. To get out of this jam, the vertex undergoes an enhancement step that gives it a workable color. Once all the vertices have been allocated colours, the procedure is repeated. Last but not least, we save the biggest vertex colour in k and use it as an LCP solution,

$\text{eval} = P1 - \alpha\,(P1 - P2)$

where, $P1 = \max_{v \in CL} d(v)$, $P2 = \min_{v \in CL} d(v)$ and $\alpha \in (0,1)$

B. Simple Greedy Heuristic for LCP

This section details the design and implementation of a greedy heuristic for LCP. Assume that C is the set of vertex colours and UC is the set of vertex colours that are not present. Initially, there are no vertices in C, and all vertices are in UC. Beginning with the following formula, the method determines the feedback value fb for every vertex in UC:

$fb(v) = d(v) +$ number of colored neighboring vertices of v.

We start by colouring the vertex with the highest feedback value once we have found all of the feedback values. The algorithm will select the vertex with the smallest colour from its palette in the event that multiple vertices have the maximum feedback value. If there is a tie among several vertices with the smallest colour, any of them could be chosen at random. The vertex is given the colour that is the furthest from the bottom of its colour list as soon as it is chosen. Afterwards, the colour is deleted from the colour lists of every vertex that is neighbouring the one you choose. When a vertex is coloured, it is immediately removed from the UC list of uncoloured vertices. All the vertices are coloured once this procedure is repeated. An enhancement phase is performed on each vertex whose colour list is empty at any point in time to address this issue. Finally, when every vertex has been coloured, the solution of the LCP is considered as the biggest colour given to that vertex and stored in k.

V.COMPARATIVE ANALYSIS OF HEURISTIC AND EXACT METHODS FOR GRAPH COLORING

Solution Quality

The optimum solutions are guaranteed by accurate techniques, such as Integer Linear Programming (ILP) and Linear Decomposition, which determine the exact chromatic number and a valid coloring for a graph. For example, by ensuring that no two neighboring vertices share colors, ILP can reduce the number of colors required for a 32-vertex network from 9 to 8. Linear decomposition also works very well with graphs that have a limited linear width, and it gives demonstrably optimal answers for benchmark cases like COLOR02. While chromatic number is often used to achieve near-optimal solutions, heuristic approaches such as HyColor, Squeaky Wheel Optimisation with Tabu Search (SWO+TS), and DSATUR commonly use one or two extra colours. While DSATUR performs well on dense graphs, it may deviate somewhat from the ideal. In contrast, HyColor achieves optimality in more than 60% of sparse graph benchmarks. While heuristics are not a replacement for accurate approaches, they are effective in situations where near-optimal answers suffice.

Computational Efficiency

Proprietary approaches can only be applied to small graphs (e.g., $n < 100$) due to their exponential time complexity, which is typically O(2^n) for ILP or linear width for Linear Decomposition, thereby limiting their computational efficiency. With no particular structural restrictions, solving a 32-vertex graph using ILP could take seconds to hours, and bigger cases are intractable. On the other hand, heuristic approaches work in polynomial time; for sparse graphs, HyColor achieves O(n^2) complexity, while tiny cases are solved in milliseconds by DSATUR. SWO+TS takes minutes to complete large graph colourings, which is slower than accurate techniques but quicker than pure greedy algorithms due to its iterative refinement. Because of their speed, heuristics are ideal for complicated networks with time-sensitive applications, whereas precise approaches perform better when validation or analysis is done offline.

Scalability to Complex Networks

Due to the sparse topologies and millions of vertices often found in complex networks, scalability is a significant difference. In such cases, exact approaches fail; for graphs with more than 1000 vertices, ILP is no longer feasible, and even linear decomposition requires a small linear width to be practical. In this case, heuristic approaches are effective; for example, HyColor utilizes techniques such as k-core decomposition and graph reduction to handle sparse graphs with up to 10^7 vertices efficiently. While SWO+TS can accommodate generalisations such as multicoloring in geometric graphs, DSATUR performs well when dealing with dense graphs. Heuristics are essential for real-world applications because they can handle large-scale networks, whereas precise approaches can only handle smaller subgraphs or highly structured cases.

Robustness across Graph Types

Structured graphs with specific qualities, such as low linear width or high regularity, enable exact algorithms to perform robustly, as these restrictions narrow the search area. Their processing costs, however, make them fail on random or huge sparse networks. Heuristic approaches demonstrate greater adaptability to various types of graphs. An example of such is DSATUR, which uses saturation degree to lessen the colour saturation of dense graphs. Another is HyColor, which combines clique-finding and greedy colouring and has found considerable success in scale-free or community structured sparse networks. Adapted to complex generalisations, SWO+TS further augment resilience; for instance, it addresses multicoloring in geometric or weighted graphs. This diversity permits the heuristics to contend with complex network topologies, from highly clustered to entirely random, in contrast to exact methods which are limited to specific graph characteristics for maximum efficacy.

Practical Use Cases

Two practical examples for which accurate approaches excel are optimizing important subgraphs in network architecture and validating benchmark findings. These cases require guaranteed optimality. For small wireless networks, for example, ILP can provide a guarantee for minimum channel assignments and for structured subgraphs, Linear Decomposition can ascertain a solution. Real-time, large-scale applications such as scheduling, resource allocation, and frequency assignment in social, transportation, and power networks are instances where heuristic methods predominate. HyColor is well-suited for large social networks due to its sparse graph performance; however, SWO+TS can handle more complicated cases, such as geometric graph colouring. To address various needs, a hybrid framework that utilizes heuristics to establish boundaries and employs accurate techniques to refine reduced subgraphs offers a well-rounded solution for complex networks.

VI.CONCLUSION

In complex networks, heuristic and accurate approaches to graph colouring have complementary merits, as demonstrated by the comparative framework. For the varied needs of complicated network colouring, a mixed strategy that uses heuristics for initial boundaries and accurate approaches for refining is a good bet. Adaptive algorithms that choose techniques on the fly according to graph characteristics and application restrictions should be the subject of future research.

REFERENCES:

[1].LEWIS, R. A Guide to Graph Colouring: Algorithms and Applications. 1. ed. Berlin: Springer, 2016. v. 1.

[2].O. Ore, The Four-color problem. Academic Press, New York, 1969.

[3].C. P. A, “Hajós’ graph-coloring conjecture: variations and counterexamples,” Journal of Combinatorial Theory Series B, vol. 26, pp. 268–274, 1979.

[4].J. E. Greene, “A new short proof of kneser’s conjecture,” The American mathematical monthly, vol. 109, no. 10, pp. 918–920, 2002.

[5].A. Gondran and L. Moalic, “Optimality clue for graph coloring problem,” in International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer, 2019, pp. 337–354.
[6].H. Xue, V. Rajan, and Y. Lin, “Graph coloring via neural networks for haplotype assembly and viral quasispecies reconstruction,” Advances in Neural Information Processing Systems, NeurIPS 2022, vol. 35, pp. 30 898–30 910, 2022.
[7].D. Ma, X. Hu, H. Zhang, Q. Sun, and X. Xie, “A hierarchical event detection method based on spectral theory of multidimensional matrix for power system,” IEEE Transactions on Systems, Man, and Cybernetcis-Systems, vol. 51, no. 4, pp. 2173–2186, 2021.
[8].A. Sadavare and R. Kulkarni, “A review of application of graph theory for network,” International Journal of Computer Science and Information Technologies, vol. 3, no. 6, pp. 5296–5300, 2012.
[9].S. Gupta and S. Amin, “Scalable design of error-correcting output codes using discrete optimization with graph coloring,” in Advances in Neural Information Processing Systems, NeurIPS 2022, 2022.
[10].Y. Li, X. Li, and L. Gao, “An effective solution space clipping-based algorithm for large-scale permutation flow shop scheduling problem,” IEEE Transactions on Systems, Man, and Cybernetcis-Systems, vol. 53, no. 1, pp. 635–646, 2023.
[11].E. Zhu, F. Jiang, C. Liu, and J. Xu, “Partition independent set and reduction-based approach for partition coloring problem,” IEEE Transactions on Cybernetics, vol. 52, no. 6, pp. 4960–4969, 2022.
[12].M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, San Francisco, CA, USA, 1979.
[13].B. F. Almeida, I. Correia, and F. Saldanha-da Gama, “Priority-based heuristics for the multi-skill resource constrained project scheduling problem,” Expert Systems with Applications, vol. 57, pp. 91–103, 2016.
[14].Michael R. Garey and David S. Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979. San Francisco, LA: Freeman, 58, 1979. 12, 23
[15].Gregory J. Chaitin. Register allocation & spilling via graph coloring. In ACM Sigplan Notices, volume 17, pages 98–105. ACM, 1982. 12
[16].Burke Edmund K., Elliman David, and Weare Rupert. A university timetabling system based on graph colouring and constraint manipulation. Journal of research on computing in education, 27(1):1–18, 1994
[17].Andreas Gamst. Some lower bounds for a class of frequency assignment problems. IEEE transactions on vehicular technology, 35(1):8–14, 1986. 12
[18].Frank Thomson Leighton. A graph coloring algorithm for large scheduling problems. Journal of research of the national bureau of standards, 84(6):489–506, 1979. 12, 13
[19].Philippe Galinier, Jean-Philippe Hamiez, Jin-Kao Hao, and Daniel Porumbel. Recent advances in graph vertex coloring. In Handbook of optimization, pages 505–528. Springer, 2013. 12, 19, 23, 33, 43
[20].A. Hertz and D. d. Werra, “Using tabu search techniques for graph coloring,” Computing, vol. 39, no. 4, pp. 345–351, 1987.
[21].C. Avanthay, A. Hertz, and N. Zufferey, “A variable neighborhood search for graph coloring,” European Journal of Operational Research, vol. 151, no. 2, pp. 379–388, 2003.
[22].I. Blöchliger and N. Zufferey, “A graph coloring heuristic using partial solutions and a reactive tabu scheme,” Computers & Operations Research, vol. 35, no. 3, pp. 960–975, 2008.
[23].Wu and J.-K. Hao, “Coloring large graphs based on independent set extraction,” Computers & Operations Research, vol. 39, no. 2, pp. 283–290, 2012.
[24].Z. Lü and J.-K. Hao, “A memetic algorithm for graph coloring,” European Journal of Operational Research, vol. 203, no. 1, pp. 241–250, 2010.
[25].R. Marappan and G. Sethumadhavan, “Solution to graph coloring using genetic and tabu search procedures,” Arabian Journal for Science and Engineering, vol. 43, no. 2, pp. 525–542, 2018.
[26].L. Moalic and A. Gondran, “Variations on memetic algorithms for graph coloring problems,” Journal of Heuristics, vol. 24, no. 1, pp. 1–24, 2018.

[27].T. Dokeroglu and E. Sevinc, “Memetic teaching–learning-based optimization algorithms for large graph coloring problems,” Engineering Applications of Artificial Intelligence, vol. 102, p. 104282, 2021
[28].O. Goudet, C. Grelier, and J.-K. Hao, “A deep learning guided memetic framework for graph coloring problems,” Knowledge-Based Systems, p. 109986, 2022.
[29].A. Schidler and S. Szeider, “Sat-boosted tabu search for coloring massive graphs,” ACM Journal of Experimental Algorithmics, vol. 28, pp. 1–19, 2023.
[30].F.-X. Reichl, F. Slivovsky, and S. Szeider, “Circuit minimization with qbf-based exact synthesis,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 4, 2023, pp. 4087–4094.
[31].G. Audemard and L. Simon, “Predicting learnt clauses quality in modern sat solvers,” in Twenty-first international joint conference on artificial intelligence. Citeseer, 2009.
[32].A. Fleury and M. Heisinger, “Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020,” SAT COMPETITION, vol. 2020, p. 50, 2020.
[33].R. A. Rossi and N. K. Ahmed, “Coloring large complex networks,” Social Network Analysis and Mining, vol. 4, no. 1, pp. 1–37, 2014.
[34].A. Verma, A. Buchanan, and S. Butenko, “Solving the maximum clique and vertex coloring problems on very large sparse networks,” INFORMS Journal on computing, vol. 27, no. 1, pp. 164–177, 2015.
[35].J. Lin, S. Cai, C. Luo, and K. Su, “A reduction based method for coloring very large graphs,” in IJCAI, 2017, pp. 517–523.
[36].E. Hebrard and G. Katsirelos, “A hybrid approach for exact coloring of massive graphs,” in International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer, 2019, pp. 374–390.
[37].Pablo Coll, Javier Marenco, Isabel Méndez Díaz, and Paula Zabala. Facets of the graph coloring polytope. Annals of Operations Research, 116(1-4):79–90, 2002. 12, 13
[38].Isabel Méndez-Díaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. Discrete Applied Mathematics, 154(5):826–847, 2006. 12, 13
[39].Jon Lee. All-different polytopes. Journal of Combinatorial Optimization, 6(3):335–352, 2002. 12, 13
[40].Jon Lee and François Margot. On a binary-encoded ilp coloring formulation. INFORMS Journal on Computing, 19(3):406–415, 2007. 12
[41].Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. informs Journal on Computing, 8(4):344–354, 1996. 12, 13
[42].David Schindl. Some combinatorial optimization problems in graphs with applications in telecommunications and tomography. 2004. 12
[43].H. Paul Williams and Hong Yan. Representations of the all_different predicate of constraint satisfaction in integer programming. INFORMS Journal on Computing, 13(2):96–103, 2001. 12, 13
[44].Manoel Campêlo, Victor A. Campos, and Ricardo C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. Discrete Applied Mathematics, 156(7):1097–1111, 2008. 13
[45].Edmund K. Burke, Jakub Mareˇcek, Andrew J. Parkes, and Hana Rudová. A supernodal formulation of vertex colouring with applications in course timetabling. Annals of Operations Research, 179(1):105–130, 2010. 13
[46].Enrico Malaguti, Michele Monaci, and Paolo Toth. An exact approach for the vertex coloring problem. Discrete Optimization, 8(2):174–190, 2011. 13, 14
[47].Zhaoyang Zhou, Chu-Min Li, Chong Huang, and Ruchu Xu. An exact algorithm with learning for the graph coloring problem. Computers & Operations Research, 51:282– 301, 2014. 13, 14, 83
[48].Michaël Rao. Coloring a graph using split decomposition. In International Workshop on Graph-Theoretic Concepts in Computer Science, pages 129–141. Springer, 2004. 13
[49].J. Bhasker and Tariq Samad. The clique-partitioning problem. Computers & Mathematics with Applications, 22(6):1–11, 1991. 13
[50].Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. Computers & operations research, 33(8):2189–2207, 2006. 13
[51].Ronald C. Read. An introduction to chromatic polynomials. Journal of Combinatorial Theory, 4(1):52–71, 1968. 13

[52].Nai-Wei Lin. Approximating the chromatic polynomial of a graph. In International Workshop on Graph-Theoretic Concepts in Computer Science, pages 200–210. Springer, 1993. 13
[53].Jean-Noel Martin. No Free Lunch et recherche de solutions structurantes en coloration. PhD thesis, Université de Technologie de Belfort-Montbeliard, 2010. 13
[54].Jianding Guo, Laurent Moalic, Jean-Noel Martin, and Alexandre Caminada. Exact graph coloring algorithms of getting partial and all best solutions. In ISAIM, 2018. 13
[55].Pierre Hansen, Martine Labbé, and David Schindl. Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. Discrete Optimization, 6(2):135–147, 2009. 12, 13
[56].Valmir C. Barbosa, Carlos A.G. Assis, and Josina O. Do Nascimento. Two novel evolutionary formulations of the graph coloring problem. Journal of Combinatorial Optimization, 8(1):41–63, 2004. 12, 13
[57].MATOUŠEK, J.; GÄRTNER. Understanding and using linear programming. 1. ed. Berlin, Germany: Springer, 2007. v. 1. (Universitext, v. 1)
[58].DANTZIG, G. Linear programming and extensions. 1. ed. Princeton, NJ: Princeton Univ. Press, 1963. v. 1. (Rand Corporation Research Study, v. 1)
[59].PAPADIMITRIOU, C. H.; STEIGLITZ, K. Combinatorial Optimization: Algorithms and Complexity. 1. ed. New Jersey, USA: Prentice-Hall, Inc., 1998. v. 1.
[60].MEHROTRA, A.; TRICK, M. A. A column generation approach for graph coloring. INFORMS J. Comput., v. 8, n. 4, p. 344–354, 1995.
[61].Daniel Brélaz. New methods to color the vertices of a graph. Communications of the ACM, 22(4):251–256, 1979. 13, 33
[62].Frank Thomson Leighton. A graph coloring algorithm for large scheduling problems. Journal of research of the national bureau of standards, 84(6):489–506, 1979. 12, 13
[63].Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. Computing, 39(4):345–351, 1987. 14, 16, 57, 76
[64].Philippe Galinier and Alain Hertz. A survey of local search methods for graph coloring. Computers & Operations Research, 33(9):2547–2562, 2006. 12, 14, 19, 23, 57
[65].Charles Fleurent and Jacques A. Ferland. Genetic and hybrid algorithms for graph coloring. Annals of Operations Research, 63(3):437–461, 1996. 14, 26
[66].Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. Journal of combinatorial optimization, 3(4):379–397, 1999. 14, 16, 23, 24, 25, 26, 31, 34, 35, 57, 76
[67].Daniel Cosmin Porumbel, Jin-Kao Hao, and Pascale Kuntz. An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers & Operations Research, 37(10):1822–1832, 2010. 14, 23, 34,
[68].Jin-Kao Hao and Qinghua Wu. Improving the extraction and expansion method for large graph coloring. Discrete Applied Mathematics, 160(16-17):2397–2407, 2012. 14
[69].Noureddine Bouhmala and Ole-Christoffer Granmo. Solving graph coloring problems using learning automata. In European Conference on Evolutionary Computation in Combinatorial Optimization, pages 277–288. Springer, 2008. 14
[70].Shadi Mahmoudi and Shahriar Lotfi. Modified cuckoo optimization algorithm (mcoa) to solve graph coloring problem. Applied soft computing, 33:48–64, 2015. 14
[71].Anas Elhag and Ender Özcan. A grouping hyper-heuristic framework: Application on graph colouring. Expert Systems with Applications, 42(13):5491–5507, 2015. 14
[72].Ines Sghir, Jin-Kao Hao, Ines Ben Jaafar, and Khaled Ghédira. A multi-agent based optimization method applied to the quadratic assignment problem. Expert Systems with Applications, 42(23):9252–9262, 2015. 14
[73].Yangming Zhou, Béatrice Duval, and Jin-Kao Hao. Improving probability learning based local search for graph coloring. Applied Soft Computing, 65:542–553, 2018. 14, 34,