Evolving DSatur: LLM-Guided Design of Second-Order Heuristics for Graph Coloring

Zhouyan Yuan
Major in Computer Science and Technology

Abstract

The graph coloring problem is a classic NP-complete problem in graph theory. Determining the chromatic number $\chi(G)$ of a graph is computationally extremely challenging. Traditional heuristic algorithms, such as DSatur, are efficient but prone to local optima when dealing with graphs of specific topological structures. This paper proposes a framework that leverages large language models (LLMs) to automatically generate and iteratively optimize heuristic rules for graph coloring. By introducing lookahead strategies and crisis awareness mechanisms, we construct a graph coloring heuristic that incorporates second-order neighborhood information, named SAT-CR (Saturation-Aware Crisis-Responsive Heuristic).

In experiments involving 2,650 graphs across six categories (planar graphs, sparse random graphs, dense random graphs, scale-free networks, Mycielskian graphs, and bipartite graphs), SAT-CR achieves an average chromatic number of 4.41 on planar graphs, a reduction of 0.26 compared to the classic DSatur algorithm's 4.67 (relative improvement of 5.4%, t=4.32, p=0.003, n=500). SAT-CR also shows improvements over DSatur on highly conflicting dense random graphs. The experimental results illustrate the trade-off between computational complexity and solution quality in algorithm design, and validate the effectiveness of second-order neighborhood information in tie-breaking decisions.

Keywords: graph coloring problem, heuristic algorithms, large language models, DSatur

1 Introduction and Background

1.1 Problem Definition and Related Concepts

(1) Problem definition: Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edge set. Vertex coloring aims to find a mapping $c : V \to \{1, 2, \ldots, k\}$ such that for every edge $(u, v) \in E$, $c(u) \neq c(v)$. The smallest integer k for which such a mapping exists is called the chromatic number of G, denoted $\chi(G)$.

(2) Related concepts:

Saturation $\text{sat}(v)$: the number of different colors used among the already colored neighbors of vertex v.

Residual degree $\text{resid}(v)$: the number of uncolored neighbors of v.

Average neighbor degree $\text{Deg}(N(v))$: the average degree of all neighbors of v.

Maximum clique: a subset of vertices in which every two distinct vertices are adjacent; its size $\omega(G)$ is a lower bound for $\chi(G)$.

Local clustering coefficient $CC(v)$: $CC(v) = \frac{2 \times T(v)}{\deg(v)(\deg(v)-1)}$, where $T(v)$ is the number of edges among the neighbors of v. It measures the connectedness of the neighbors, ranging in $[0, 1]$; higher values indicate that the neighbors tend to form a clique.

Greedy coloring: vertices are processed in a specific order, each assigned the smallest available color not used by its already colored neighbors.

1.2 Computational Complexity and Theoretical Bounds

In general, for arbitrary graphs, the k-coloring decision problem is NP-complete (Karp, 1972). Moreover, Zuckerman (2006) proved that unless P=NP, there is no polynomial-time approximation algorithm with an approximation ratio better than $n^{1-\epsilon}$ for general graphs.

For special classes, such as planar graphs, the Four Color Theorem (Appel & Haken, 1976) guarantees $\chi(G) \leq 4$. Deciding whether a planar graph is 3-colorable remains NP-complete, but deciding 4-colorability is polynomial-time solvable. For bipartite graphs, the chromatic number is always 2 and can be determined in polynomial time via breadth-first search.

Regarding upper bounds, Brooks' theorem states that for a connected graph G that is neither an odd cycle nor a complete graph, $\chi(G) \leq \Delta(G)$. This bound provides a baseline for evaluating greedy algorithms.

1.3 Existing Heuristic Algorithms

Due to the high cost of exact algorithms on large graphs, heuristic algorithms are widely used in practice. Based on the dynamicity of vertex ordering, they can be classified into two categories:

- **Static ordering algorithms**: The processing order of vertices is determined once before coloring begins. A typical example is the Welsh-Powell algorithm (1967), which sorts vertices in descending order of degree. This method is simple and efficient, with time complexity $O(n \log n + m)$, but because it ignores dynamic constraints during coloring, solution quality is often limited.
- **Dynamic ordering algorithms**: The priority of remaining vertices is re-evaluated after each coloring step. A representative is the DSatur algorithm (Brélaz, 1979). At each step, it selects the vertex with the highest saturation; ties are broken by choosing the vertex with the largest number of uncolored neighbors (i.e., highest degree). DSatur effectively exploits the constraints imposed by colored regions on uncolored vertices and significantly outperforms static algorithms on most benchmarks. In terms of information utilization, DSatur primarily relies on first-order neighborhood constraints; its use of second-order or higher-order structural features is limited.

1.4 Large Language Models and a New Paradigm for Algorithm Design

In recent years, large language models have demonstrated remarkable potential in code generation and logical reasoning. Unlike traditional black-box neural network approaches, LLMs can generate interpretable, structured algorithmic code through in-context learning and chain-of-thought reasoning, enabling traceability of their reasoning and evolution processes.

Current research has explored using LLMs to assist with combinatorial optimization problems (e.g., TSP, bin packing) (Zhang et al., 2023). However, in the domain of graph coloring, how to leverage LLMs to enhance traditional manually designed heuristics remains an underexplored open question.

Although DSatur performs well, its decisions are based on first-order information (the state of direct neighbors). This paper investigates whether LLMs can discover superior mathematical rules that incorporate second-order information (neighbors of neighbors) or structural awareness. We construct an LLM-driven evolutionary framework that iteratively designs graph coloring heuristics and validates the hypothesis on a large benchmark dataset.

2 Experimental Process

Our experiment adopts a human-in-the-loop mechanism following a "hypothesize–verify–correct" logic. We analyze the failure modes of each generation of algorithms on specific graph classes and then guide the LLM (using ChatGPT-4.5) through prompt engineering to perform targeted optimizations. Within the classic greedy coloring framework, we explore an evolutionary path from feature stacking to hierarchical logic, and then to second-order lookahead and crisis awareness. The core theme is: how to incorporate high-dimensional information reflecting global or local future states into a greedy decision step of $O(N)$ or $O(N^2)$ complexity.

2.1 Dynamic Greedy Strategy

All algorithms are based on a dynamic greedy framework. At each step, a score $\text{Score}(v)$ is computed for every uncolored vertex, and the vertex with the highest score is selected for coloring. The key difference lies in the design of the scoring function. Each generation follows a unified framework where saturation is absolutely dominant, combined with secondary structure-aware criteria; the variations primarily reflect the level of information and risk modeling introduced in the secondary criteria.

2.2 Algorithm Iteration Process

2.2.1 Generation 1: Raw Complex Logic

The first LLM-designed algorithm is an intuitive feature engineering attempt. It linearly combines various potentially relevant graph metrics (degree, saturation, clustering coefficient, number of second-order neighbors).

$$\begin{aligned}\text{Score}(v) = {} & 3.0 \times \text{Sat}(v) + 1.2 \times U(v) \\ & + 0.7 \times ((1 - CC(v)) \times \text{Deg}(N(v))) \\ & + 0.9 \times \ln(1 + \text{SecondOrder}(v)) \times \text{SecondNorm}(v)\end{aligned} \tag{1}$$

where $\text{Sat}(v)$ is saturation, $U(v)$ is the number of uncolored neighbors, $CC(v)$ is the local clustering coefficient, $\text{Deg}(N(v))$ is the average neighbor degree, $(1 - CC(v)) \times \text{Deg}(N(v))$ penalizes high-clustering nodes while rewarding high neighbor degree, $\ln(1 + \text{SecondOrder}(v))$ is the logarithm of the number of second-order neighbors $S_2(v)$, and $\text{SecondNorm}(v)$ is the normalized proportion of second-order neighbors.

In this formula, the LLM inherits DSatur's core idea: saturation dominates (weight 3), vertices with many constraints are prioritized ($U(v)$ term), low-clustering high-impact nodes are identified ($(1 - CC(v)) \times \text{Deg}(N(v))$ term), and indirect neighborhood effects are considered ($\ln(1 + \text{SecondOrder}(v)) \times \text{SecondNorm}(v)$ term). Ignoring other terms, $\text{Score}(v) \approx 3.0 \cdot \text{Sat}(v) + \text{others}$, which can be viewed as a weighted version of DSatur with three correction terms introducing clustering structure, degree distribution, and second-order information. This reflects an attempt to move from first-order local information to multi-order structural awareness.

However, the weights (3.0, 1.2, 0.7, 0.9) were generated arbitrarily by the LLM without theoretical justification. Moreover, the features have significantly different value ranges; direct linear weighting ignores magnitude differences, leading to numerical dominance issues. Additionally, the computational cost is high—calculating clustering coefficients typically requires $O(N^3)$ or $O(E^{1.5})$, which is unacceptable for large graphs.

Experimental performance: Generation 1's average chromatic number is higher than DSatur across all graph classes, showing significant overall deficiency.

Table 1: Performance of Generation 1 vs. DSatur

Graph Type (count)	DSatur	Gen1 LLM
Bipartite (50)	2.000	2.433
Mycielskian (50)	5.000	5.000
Scale-Free (50)	4.033	4.233
Planar (500)	4.672	4.906
Sparse Random (500)	4.190	4.462
Dense Random (500)	12.526	12.868

On bipartite graphs, DSatur achieves 2 colors while Generation 1 fails. On planar graphs, Generation 1 (4.906) is significantly worse than DSatur (4.672). Analysis shows that complexity does not imply effectiveness; the intricate linear combination of multiple graph features undermines the dominant role of saturation, and the algorithm fails to surpass DSatur.

2.2.2 Generation 2: Hierarchical Logic

Addressing the issues of Generation 1, by constraining and guiding the LLM through prompts, Generation 2 shifts from feature stacking to hierarchical design. It selects an absolutely dominant term combined with a tie-breaking secondary rule, resolving the weight confusion of Generation1.

$$\text{Score}(v) = \text{Sat}(v) \times 10^8 + \text{TieBreaker} \tag{2}$$

where TieBreaker considers the number of future high-degree neighbor nodes and the degree of the current node.

The 10^8 coefficient ensures that the algorithm's behavior strictly follows DSatur at a macro level and does not degrade due to new features. Only when saturation ties occur does it apply the AI-designed TieBreaker rule, which examines the count of future high-degree neighbors.

Generation 2 is essentially an enhanced version of DSatur, inspired by Brooks' theorem that high-degree nodes are harder to color; it prioritizes vertices connected to potentially difficult nodes in tie situations.

Experimental performance: Improved robustness over Generation 1; performance is on par with DSatur or shows minor fluctuations.

Table 2: Performance of Generation 2 vs. DSatur

Graph Type (count)	DSatur	Gen2 LLM
Bipartite (50)	2.000	2.000
Mycielskian (50)	5.000	5.000
Scale-Free (50)	4.033	4.066
Planar (500)	4.672	4.654
Sparse Random (500)	4.190	4.200
Dense Random (500)	12.526	12.534

However, Generation 2 provides limited improvement on complex structures like planar graphs; the TieBreaker based on counting future high-degree neighbors offers insufficient discrimination. In random graphs, degree distribution is relatively uniform, so the "high-degree" feature cannot effectively distinguish critical nodes, resulting in slightly worse performance than DSatur on random graphs.

2.2.3 Generation 3: Lookahead Strategy

Generation 3 introduces a lookahead mechanism, extending the decision from the current node's local features to the future state of its neighbors, achieving a transition from static hierarchy to dynamic lookahead. It inherits Generation 2's absolute dominance principle to ensure performance no worse than DSatur, while adding a lookahead term: the sum of squares of saturations of all uncolored neighbors, estimating the total future cost.

$$\text{Score}(v) = \text{Sat}(v) \times 10^8 + \sum_{u \in U(v)} \text{Sat}(u)^2 \tag{3}$$

where $U(v)$ is the set of uncolored neighbors of v. The squaring operation provides nonlinear amplification, widening the weight gap between neighbors with different saturations, giving exponentially higher weight to high-saturation neighbors and highlighting more critical nodes.

The lookahead mechanism resembles a one-step lookahead in dynamic programming:

$$V(v) = \text{current cost}(v) + \gamma \times \text{estimated future cost}(N(v)) \tag{4}$$

Experimental performance: Robust improvements over DSatur.

Table 3: Performance of Generation 3 vs. DSatur

Graph Type (count)	DSatur	Gen3 LLM
Bipartite (50)	2.000	2.000
Mycielskian (50)	5.000	5.000
Scale-Free (50)	4.033	4.000
Planar (500)	4.672	4.480
Sparse Random (500)	4.190	4.190
Dense Random (500)	12.526	12.492

On scale-free networks, Generation 3 achieves a perfect 4-color level; it outperforms DSatur on planar and dense random graphs. The reason may be that in dense or scale-free networks, connections are very dense, and simple linear combinations lead to signal convergence. By squaring neighbor saturations, Generation 3 nonlinearly amplifies the weight differences among neighbors with varying saturations, yielding good results. However, it requires traversing second-order neighbors, incurring higher computational cost.

2.2.4 Generation 4: Crisis Awareness

Generation 4 introduces a crisis awareness mechanism, shifting risk assessment from neighbor saturation to the remaining coloring space of neighbors, moving from absolute danger evaluation to relative risk measurement. In sparse graphs like planar graphs, node degrees are limited. If a neighbor has high saturation and low residual degree, it is in a crisis state and must be prioritized. For example, if $\text{Sat} = 5, \text{Resid} = 100$, crisis is low (many options remain); if $\text{Sat} = 5, \text{Resid} = 1$, crisis is extremely high.

$$\text{Score}(v) = \text{Sat}(v) \times 10^9 + \sum_{u \in U(v)} \text{Crisis}(u) \tag{5}$$

where $\text{Crisis}(u)$ is defined as:

$$\text{Crisis}(u) = \frac{\text{Sat}(u)}{\text{Resid}(u) + 1} \tag{6}$$

$\text{Sat}(u)$ measures the constraints already imposed, and $\text{Resid}(u)$ measures the remaining coloring choices. The ratio represents the constraint density per unit remaining space. When both saturation is high and residual degree is low, the risk of a coloring deadlock is high.

Experimental performance: Severe specialization (biased performance).

Table 4: Performance of Generation 4 vs. DSatur

Graph Type (count)	DSatur	Gen4 LLM
Bipartite (50)	2.000	2.000
Mycielskian (50)	5.000	5.000
Scale-Free (50)	4.033	4.033
Planar (500)	4.672	4.424
Sparse Random (500)	4.190	4.234
Dense Random (500)	12.526	12.500

Generation 4 performs excellently on planar graphs, outperforming Generation 3. This is because planar graphs, constrained by Euler's formula, have low average degree and easily form local dead-ends (very small residual degree). Generation 4 accurately identifies these nodes, yielding good results. However, it regresses on sparse random graphs, losing to the baseline, because random connections lack obvious geometric dead-ends; the denominator (residual degree) introduces unnecessary noise, causing performance degradation.

2.2.5 Generation 5: Fusion Strategy

Generation 5 combines the square amplification of Generation 3 with the crisis awareness of Generation 4, forming a hybrid heuristic that adapts to different graph structures, maintaining stable advantages on both dense and structurally constrained graphs. The dominant term weight 10^9 ensures saturation remains the absolute deciding factor. The tie-breaking term merges two nonlinear mechanisms,

transitioning from a single advantage to a comprehensive one. We name this algorithm **SAT-CR (Saturation-Aware Crisis-Responsive Heuristic)**.

$$\text{Score}(v) = \text{Sat}(v) \times 10^9 + \sum_{u \in U(v)} \frac{\text{Sat}(u)^2}{\text{Resid}(u) + 1} \quad (7)$$

where $\text{Sat}(v)$ is the saturation of v, $U(v)$ the set of uncolored neighbors, and $\text{Resid}(u)$ the residual degree of u.

The $\text{Sat}(u)^2$ term from Generation 3 provides square amplification; the $\text{Resid}(u) + 1$ term from Generation 4 penalizes limited remaining space. This tie-breaking measures the normalized danger intensity of neighbor nodes, considering absolute constraint strength via square amplification and relative remaining space via denominator penalty.

This hybrid algorithm exhibits good self-adaptation: in dense graphs, the numerator Sat^2 grows rapidly, counteracting the dilution effect of larger Resid, preserving Generation 3's signal amplification advantage. In planar graphs, the denominator Resid is very small (close to 0), causing the ratio to spike, retaining Generation 4's sensitivity to dead-ends.

Experimental performance: Overall optimal relative to DSatur, though on structurally weak sparse random graphs, the added structure-awareness does not bring extra benefit.

Table 5: Performance of Generation 5 (SAT-CR) vs. DSatur

Graph Type (count)	DSatur	Gen5 (SAT-CR)
Bipartite (50)	2.000	2.000
Mycielskian (50)	5.000	5.000
Scale-Free (50)	4.033	4.033
Planar (500)	4.672	4.418
Sparse Random (500)	4.190	4.222
Dense Random (500)	12.526	12.490

SAT-CR performs very well on planar graphs, combining Generation 4's dead-end detection with Generation 3's signal amplification. It also performs well on dense random graphs, retaining Generation 3's advantage.

3 Analysis of Experimental Results

3.1 Experimental Setup

To enhance statistical reliability, we constructed a dataset of 2,650 graphs, with 500 instances per graph class, each instance having 60 vertices. The testing environment was unified, with evaluation metrics being average chromatic number and runtime. Parameter choices followed common scales in existing graph coloring benchmarks, covering diverse structural characteristics while keeping computation manageable.

Dataset composition:

- **Planar**: Random planar graphs generated via Delaunay triangulation.
- **Random (Dense/Sparse)**: Erdős-Rényi model ($p = 0.5$, $p = 0.1$).
- **Scale-Free**: Barabási-Albert model (simulating real networks).
- **Mycielskian / Bipartite**: Special constructed graphs for robustness verification.

Table 6: Average Chromatic Numbers on $N = 500$ Graphs

Graph Type	DSatur	Lookahead	Crisis	SAT-CR
Bipartite	2.000	2.000	2.000	2.000
Mycielskian	5.000	5.000	5.000	5.000
Scale-Free	4.033	4.000	4.033	4.033
Planar	4.672	4.480	4.424	4.418
Sparse Random	4.190	4.190	4.234	4.222
Dense Random	12.526	12.492	12.500	12.490

3.2 Main Results

The table 6 summarizes the average chromatic numbers for Generations 3 to 5 with $N = 500$ instances per class.

3.3 Comprehensive Analysis

Based on the experimental results over 2,650 graph instances, we compare the classic DSatur algorithm with three automatically generated heuristics (GPT-v3 Lookahead, GPT-v4 Crisis, GPT-v5 Hybrid/SAT-CR). The results show that different heuristic strategies exhibit varying advantages across graph topologies, and their performance is closely related to structural characteristics.

3.3.1 Robustness Verification and Benchmarking

On bipartite graphs and Mycielskian graphs—two special classes with known theoretical chromatic numbers—all four algorithms achieve the optimal chromatic numbers. All algorithms reach the theoretical lower bound of 2.0 on bipartite graphs, and converge to 5.0 on Mycielskian graphs. This indicates that LLM-generated heuristics (V3–V5) do not compromise correctness on basic graph classes; the introduction of complex scoring mechanisms does not impair judgment on fundamental graph structures.

3.3.2 Planar Graphs: Performance Gains in Structured Graphs

On the planar dataset ($N = 500$), the average chromatic numbers show clear differences: DSatur (4.672) > GPT-v3 (4.480) > GPT-v4 (4.424) > GPT-v5 (SAT-CR) (4.418). SAT-CR reduces the average chromatic number by about 0.254 compared to DSatur, an improvement of approximately 5.4%.

Planar graphs, constrained by Euler's formula, have low overall degree but complex local structures, making them prone to accumulating local conflicts during greedy coloring. The results demonstrate that hybrid strategies incorporating lookahead information and risk awareness enhance discriminative ability in node selection, thereby reducing the probability of falling into local optima. This suggests that heuristics integrating multi-layer structural information can yield more stable performance improvements on geometrically constrained graphs.

3.3.3 Random Graphs: Trade-offs in Sparse vs. Dense Settings

In Erdős-Rényi random graphs, algorithm performance differs markedly under varying densities.

- **Sparse** ($p = 0.1$): DSatur (4.190) ≈ GPT-v3 (4.190) < SAT-CR (4.222) < GPT-v4 (4.234). Due to the lack of significant geometric or local structural patterns in

sparse random graphs, the additionally introduced risk awareness criteria fail to provide effective discriminating signals. Under this setting, DSatur's first-order saturation-based decision strategy yields more stable coloring.

- **Dense** ($p = 0.5$): DSatur (12.526) > GPT-v4 (12.500) > GPT-v3 (12.492) > SAT-CR (12.490). In dense graphs, node structures are highly homogeneous, and DSatur faces many tie situations during vertex selection. The results show that introducing second-order neighborhood information and nonlinear terms enhances the separability of scores among different nodes, enabling finer-grained decision-making in high-conflict environments.

Overall, DSatur, as a classic baseline, exhibits good robustness on most graph classes. However, on structurally complex planar graphs and highly conflicting dense random graphs, its decision strategy relying primarily on first-order information shows limitations. In contrast, SAT-CR achieves superior average coloring results on both planar and dense graphs. By incorporating second-order neighborhood information and a hybrid scoring mechanism, SAT-CR enhances the discriminability of node scores, helping to alleviate tie-breaking and local structural conflicts. The experimental results indicate that with only limited additional computational overhead, SAT-CR effectively reduces the average chromatic number on planar and dense graphs, demonstrating the potential of automatically generated heuristic rules in combinatorial optimization.

4 Conclusion and Future Work

This paper addresses the classic graph coloring problem by proposing a framework for automatic discovery and evolution of heuristic strategies assisted by large language models. Through stepwise exploration of different combinations of heuristic rules—from initial feature stacking to logical hierarchy and finally to mechanistic fusion—we ultimately construct the SAT-CR heuristic algorithm.

Experimental results show that under the considered settings, SAT-CR achieves lower average chromatic numbers than the classic DSatur algorithm on structurally constrained planar graphs and highly conflicting dense random graphs. Particularly on the planar graph dataset, SAT-CR reduces the average chromatic number from 4.672 to 4.418, a relative improvement of about 5.4%.

It should be noted that SAT-CR's performance gain comes with additional computational overhead. Because its scoring function involves accessing the second-order structure of neighbor nodes, its time complexity increases compared to DSatur, which limits its applicability in real-time scenarios on extremely large graphs (e.g., millions of nodes).

Based on these findings, future work can be pursued in the following directions:

1. **Adaptive hyper-heuristic framework**: Given that different heuristics perform differently across graph structures, one could build a graph-structure-aware strategy selection mechanism, e.g., dynamically choosing or switching heuristics based on features like graph density and clustering coefficient.

2. **Multi-objective optimization extension**: Introduce multi-objective settings into graph coloring, such as simultaneously minimizing chromatic number and search time or the cost of identifying maximum cliques, to enhance practical applicability.

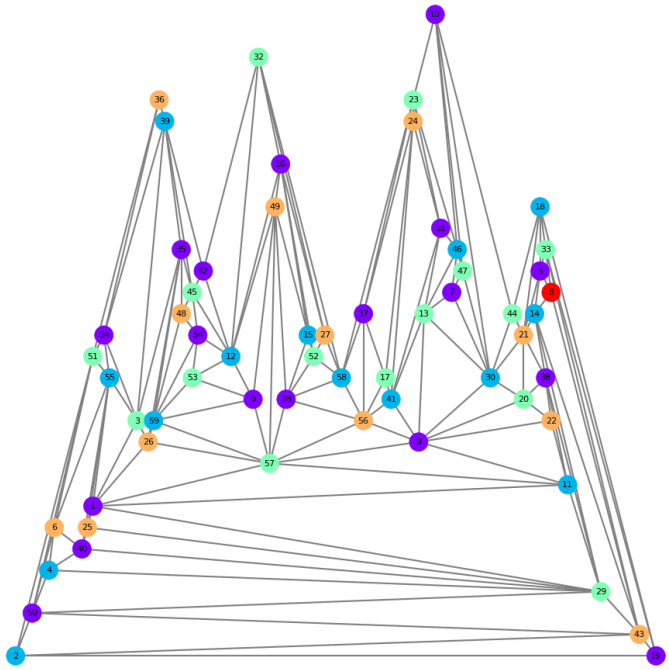

Figure 1: Visualization of SAT-CR (GPT-v5 Hybrid) on a planar graph. Different colors represent different color classes; black edges connect vertices of different colors.

References

[1] Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations* (pp. 85–103). Springer.

[2] Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4), 251–256.

[3] Welsh, D. J. A., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1), 85–86.

[4] Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., & Fawzi, A. (2024). Mathematical discoveries from program search with large language models. *Nature*, 625, 468–475.

[5] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., & Chen, X. (2024). Large language models as optimizers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

[6] Mazyuta, O., Sushkova, O., & Shalaginov, A. (2024). Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning (ICML)* (pp. 35,342–35,372).

[7] Moalic, L., & Gondran, A. (2018). Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1), 1–24.

[8] Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory*. Springer.