

The Art of Color Theory in Graphs: Coloring Algorithms and Their Diverse Applications

Wisam Rafid Dawood^{1*}, Zeyad Mohammed Abed²

^{1,2} Department of Computer Sciences, College of Sciences, Al-Nahrain University, Jadriya, Baghdad, Iraq.

*Corresponding author: wisam.rafid@nahrainuniv.edu.iq

Received: 20-06-2026	Accepted: 23-09-2026	Published: 08-10-2026
	Copyright: © 2026 by the authors. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).	

Abstract:

Graph vertex coloring assigns labels to vertices so that adjacent vertices receive different labels; its optimization objective is to minimize the number of labels. This manuscript develops a mathematically grounded review of the problem, its algorithmic families, and its applications in timetabling, frequency assignment, register allocation, and resource scheduling. It defines the principal graph parameters and proves elementary bounds, corrects the hypotheses and interpretation of Brooks' theorem, and distinguishes the decision problem from the optimization problem. It describes fixed-order greedy and Welsh–Powell procedures, DSATUR, exact integer programming and branch-and-bound approaches, tabu search, and evolutionary methods. Complexity statements are qualified by data structure and implementation assumptions. The source manuscript reported numerical comparisons for graphs described as having 500 vertices and density 0.5, alongside broader runtime and optimality claims, but supplied no graph files, random seeds, code, hardware description, trial-level outcomes, or certified optimum/lower bounds. Those numbers therefore cannot be independently validated and are not retained as empirical findings. Instead, the paper provides a complete evaluation protocol using public benchmark provenance, independent runs, coloring validation, bound-aware gaps, and transparent reporting. The verified contribution is a coherent theoretical synthesis and a reproducible design for future empirical comparison; it does not establish that any one heuristic is universally superior.

Keywords: graph coloring; chromatic number; DSATUR; greedy algorithms; integer programming; tabu search; reproducibility; scheduling.

فن نظرية التلوين في الرسوم البيانية: خوارزميات التلوين وتطبيقاتها المتنوعة

وسام رافد داود^{1*}، زياد محمد عبد²

^{1,2} قسم علوم الحاسوب، كلية العلوم، جامعة النهرين، الجادرية، بغداد، العراق

الملخص

يتمثل تلوين رؤوس الرسم البياني (Graph Vertex Coloring) في إسناد ألوان إلى الرؤوس بحيث تحصل الرؤوس المتجاورة على ألوان مختلفة، بينما يتمثل الهدف التحسيني للمسألة في تقليل عدد الألوان المستخدمة إلى أدنى حد ممكن. تقدم

هذه المخطوطة مراجعة قائمة على أسس رياضية للمسألة، وعائلات خوارزمياتها، وتطبيقاتها في إعداد الجداول الزمنية، وتخصيص الترددات، وتخصيص السجلات، وجدولة الموارد.

وتعرف الدراسة المعلومات الأساسية للرسوم البيانية، وتبرهن الحدود الرياضية الأولية، وتصحح شروط ميرهنه بروكس (Brooks' Theorem) وتفسرها، كما تميز بين مسألة القرار ومسألة التحسين. وتتناول إجراءات التلوين الجشع ذات الترتيب الثابت، وخوارزمية ويلش-باول (Welsh-Powell)، وخوارزمية DSATUR، وأساليب البرمجة الصحيحة الدقيقة والفرع والتحديد (Branch-and-Bound)، والبحث المحظور (Tabu Search)، والأساليب التطورية.

كما تُعرض تعقيدات الخوارزميات مع مراعاة بنية البيانات وافتراضات التنفيذ. وقد تضمنت المسودة الأصلية مقارنات رقمية لخوارزميات طبقت على رسوم بيانية وُصفت بأنها تتكون من 500 رأس وبكثافة مقدارها 0.5، إلى جانب ادعاءات أوسع بشأن زمن التنفيذ والأمنية. إلا أن المسودة لم توفر ملفات الرسوم البيانية، أو البنود العشوائية، أو الشيفرات البرمجية، أو وصف الأجهزة المستخدمة، أو النتائج التفصيلية للتجارب، أو القيم المثلى المعتمدة والحدود الدنيا المثبتة. ولذلك، لا يمكن التحقق من تلك الأرقام بصورة مستقلة، ولم تُعتمد بوصفها نتائج تجريبية في هذه النسخة.

وبدلاً من ذلك، تقدم الورقة بروتوكولاً متكاملًا للتقييم، يعتمد على توثيق مصادر مجموعات الاختبار العامة، وإجراء تشغيلات مستقلة، والتحقق من صحة التلوين، وحساب فجوات الحلول مع مراعاة الحدود المثبتة، والإبلاغ عن النتائج بشفافية.

وتتمثل المساهمة التي تدعمها الأدلة في تقديم مراجعة نظرية متماسكة، وتصميم منهجي قابل لإعادة الإنتاج للمقارنات التجريبية المستقبلية؛ دون الادعاء بأن أي خوارزمية استدلالية تتفوق على جميع الخوارزميات الأخرى في مختلف الحالات.

الكلمات المفتاحية: تلوين الرسوم البيانية؛ العدد اللوني؛ خوارزمية DSATUR؛ الخوارزميات الجشعة؛ البرمجة الصحيحة؛ البحث المحظور؛ قابلية إعادة الإنتاج؛ الجدولة.

1. Introduction

Graph theory represents pairwise relationships with vertices and edges. A graph-coloring model is useful when a collection of entities must be assigned to categories while pairwise conflicts prohibit simultaneous use of the same category. For vertex coloring, the entities are vertices, conflicts are edges, and colors are resource labels. The model is simple, but the optimization problem can be computationally difficult: finding the minimum number of colors is NP-hard for general graphs, while deciding whether a graph has a proper coloring with a fixed number k of colors is NP-complete for each fixed $k \geq 3$ [6], [7].

The classical map-coloring question supplied an early motivation for coloring theory. The Four Color Theorem states that every planar graph can be colored with at most four colors; it is a structural theorem about planar graphs, not a general algorithm for arbitrary conflict networks [1], [2]. Modern applications extend well beyond maps. In examination timetabling, vertices can represent examinations and edges indicate shared students; in register allocation, vertices represent live ranges and edges indicate interference; in frequency assignment, vertices represent transmitters and edges encode incompatibility constraints. In each setting, an acceptable model depends on whether the chosen graph captures all relevant constraints. [9], [10], [12]

The research problem addressed here is twofold. First, introductory accounts often combine elementary mathematical guarantees, algorithm descriptions, and performance claims without clarifying which statements are theorems and which depend on an implementation or dataset. Second, empirical comparisons are difficult to reproduce when graphs, seeds, stopping rules, software versions, and raw outcomes are absent. A technically correct discussion should make both distinctions explicit.

The objectives are to (i) establish consistent notation and validate central bounds and theorem statements; (ii) compare constructive heuristics, exact optimization, and metaheuristics by mechanism and evidence requirements; (iii) explain application modeling and algorithmic trade-offs; and (iv) specify a benchmark protocol that would support a defensible future performance study. The guiding questions are: Which mathematical bounds can be certified without computation? What does each algorithm guarantee, and what does it merely attempt? What evidence is required before one may claim that an algorithm is faster or yields better colorings?

The contribution is a corrected, source-grounded synthesis rather than a new benchmark experiment. The original manuscript's authorship and topic are retained. Its unsupported numerical table and performance conclusions are treated as unverified draft material, not as evidence. A small hand-worked example is included solely to demonstrate calculations; it is explicitly not an experiment. Section 2 reviews foundations and applications, Section 3 presents bounds and theorems, Section 4 describes algorithms, Section 5 sets out an experimental protocol, Section 6 audits the source results and provides only verifiable analytical demonstrations, Section 7 discusses applications and trade-offs, Section 8 states limitations and future work, and Section 9 concludes.

2. Literature Review and Mathematical Foundations

2.1 Graph model and notation

Unless stated otherwise, let $G = (V, E)$ be a finite, simple, undirected graph with nonempty vertex set V and edge set E consisting of unordered pairs of distinct vertices. Write $n = |V|$ and $m = |E|$. The open neighborhood of v is $N(v) = \{u \in V : \{u, v\} \in E\}$; the degree is $d(v) = |N(v)|$; and the maximum degree is $\Delta(G) = \max d(v)$. A clique is a set of pairwise adjacent vertices, and $\omega(G)$ is the largest clique size. An independent set is a set with no internal edges; $\alpha(G)$ is the largest independent-set size.

A proper k -coloring is a function $c: V \rightarrow \{1, \dots, k\}$ satisfying $c(u) \neq c(v)$ for every edge $\{u, v\} \in E$. Color labels have no intrinsic meaning: relabeling colors leaves the partition into color classes unchanged. Every color class is an independent set. The chromatic number $\chi(G)$ is the minimum number of colors in any proper coloring. These definitions concern vertex coloring. Edge coloring assigns colors to edges so that incident edges differ, and has a different chromatic parameter $\chi'(G)$; the two problems must not be conflated.

$$\chi(G) = \min \{ k \in \mathbb{N} : G \text{ has a proper } k\text{-coloring} \} \quad (1)$$

Two elementary lower bounds follow immediately. Each clique requires distinct colors, so $\omega(G) \leq \chi(G)$. Since each color class has at most $\alpha(G)$ vertices, $n \leq \chi(G)\alpha(G)$, giving $\chi(G) \geq \lceil n/\alpha(G) \rceil$. The latter may be weak when $\alpha(G)$ is unknown or difficult to compute, but it is valid for every finite graph.

$$\max \{ \omega(G), \lceil n/\alpha(G) \rceil \} \leq \chi(G) \leq \Delta(G) + 1 \quad (2)$$

Equation (2) combines general lower and upper bounds; it need not be tight.

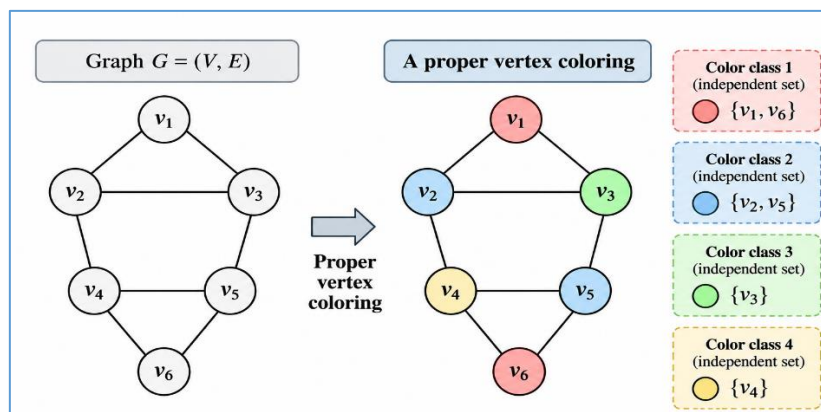


Figure 1. A proper vertex coloring. Each color class is an independent set; the labels are illustrative.

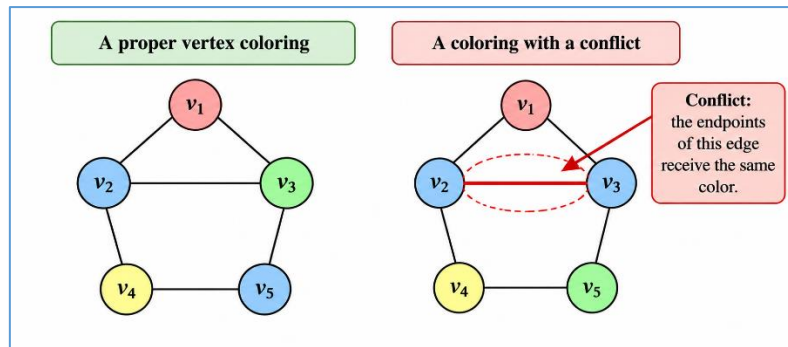


Figure 2. A conflict occurs when the endpoints of an edge receive the same color.

The chromatic number is a property of the graph, whereas the number of colors used by a particular heuristic is an outcome of an algorithm, ordering, parameters, and often randomness. A reported coloring with q colors certifies $\chi(G) \leq q$, because a reader can check every edge. It does not prove $\chi(G) = q$ unless a matching lower bound or an exact optimality certificate is established. This distinction is essential when interpreting “optimality gaps.”

2.2 Established results and their scope

The greedy bound states that $\chi(G) \leq \Delta(G) + 1$. Consider any vertex ordering and color vertices sequentially with the smallest color absent from already colored neighbors. When a vertex is processed, it has at most $\Delta(G)$ neighbors, so at most $\Delta(G)$ colors are forbidden. One of $\Delta(G) + 1$ colors remains. This proof establishes a universal upper bound; it does not say that every ordering uses exactly $\Delta + 1$ colors, nor that greedy is close to optimal. The ordering can strongly affect the result [11], [13].

Brooks’ theorem sharpens the degree bound for connected graphs. If a connected graph G is neither a complete graph nor an odd cycle, then $\chi(G) \leq \Delta(G)$. For complete graphs K_r , $\chi(K_r) = r = \Delta + 1$. For odd cycles C_{2t+1} , $\chi = 3 = \Delta + 1$. In the connected $\Delta \leq 2$ cases, paths and even cycles are 2-colorable, while odd cycles are the exception. The theorem is an upper bound; it does not imply $\chi(G) \geq \Delta(G)$, and it is not a universal stopping rule for a heuristic unless the graph meets the theorem’s hypotheses and an appropriate constructive procedure is used [3], [4].

The Four Color Theorem says $\chi(G) \leq 4$ for planar G . Appel and Haken’s computer-assisted proof used reducibility and an unavoidable set of configurations; subsequent work simplified and formalized aspects of the proof. It is incorrect to present the theorem as a general bound for all graphs, or to suggest that a typical coloring experiment proves it [1], [2].

The decision problem k -COLOR asks whether $\chi(G) \leq k$. It belongs to NP because a proposed assignment can be verified by scanning all edges in $O(n+m)$ time. For every fixed $k \geq 3$, the problem is NP-complete; $k=1$ and $k=2$ are decidable in polynomial time, with the latter equivalent to bipartiteness. The optimization problem of finding $\chi(G)$ is NP-hard. A short reduction sketch is not a complete proof unless the gadgets and both directions of the equivalence are rigorously specified; therefore, this manuscript cites the established complexity result rather than retaining the incomplete clause-gadget sketch from the source [6], [7].

Inapproximability results require especially careful wording. Zuckerman established strong hardness of approximation for chromatic number under standard complexity assumptions. Such results have precise approximation factors, reductions, and assumptions; they should be quoted directly from the source rather than represented by an unqualified formula. The broad lesson is that general-purpose polynomial-time approximation with a strong guarantee is unlikely, not that every heuristic is ineffective on every structured instance [8].

For edge coloring, Vizing’s theorem states that every simple graph satisfies $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$. This result applies to edge coloring, not vertex coloring. It is useful as a

contrast: the model and parameter must be identified before importing a theorem. In K_4 , $\Delta=3$ and a 3-edge-coloring exists by partitioning its six edges into three perfect matchings, so $\chi'(K_4)=3$. The original draft's statement " $\chi'(K_4)=\{3,4\}$ " is not a numerical conclusion; the exact value is 3. Vizing's theorem gives the general Δ to $\Delta+1$ bound for simple-graph edge coloring [21].

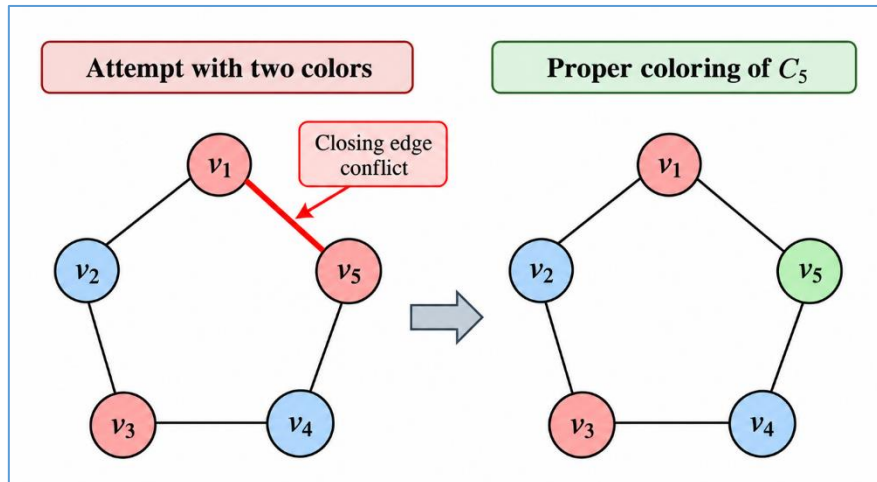


Figure 3. C_5 requires three colors: alternating two labels leaves the closing edge in conflict.

2.3 Algorithmic literature and problem families

Classical constructive methods include sequential greedy coloring, degree-ordered Welsh–Powell coloring, and Brélaz's DSATUR strategy. Welsh–Powell sorts vertices by nonincreasing degree before applying a sequential coloring procedure. DSATUR chooses vertices dynamically using the number of distinct colors already present among each uncolored vertex's neighbors; ties are commonly broken by degree, although tie-breaking details must be reported. Brélaz introduced DSATUR both as a heuristic and within an exact approach [5], [13].

Exact methods seek a proof of optimality. They include branch-and-bound procedures, integer programming, constraint programming, and specialized algorithms exploiting graph structure. In branch-and-bound, a feasible coloring supplies an upper bound, while cliques, relaxations, or other valid inequalities supply lower bounds. Search branches on assignments or color-class membership and prunes nodes that cannot improve the incumbent. Their worst-case time is exponential, but practical performance varies greatly with structure and bounds; a single vertex-count cutoff such as "exact methods are infeasible above 100 vertices" is not a theorem and should not be generalized.

Local-search methods permit intermediate infeasible assignments. Tabucol, a tabu-search approach, fixes a candidate number k and searches among assignments of vertices to k labeled classes while minimizing the number of monochromatic edges. A move typically relocates a conflicted vertex. A tabu list discourages cycling; aspiration criteria can allow a tabu move if it improves the best known solution. Reaching zero conflicts proves the graph is k -colorable, but failure to reach zero before a budget expires does not prove that k colors are insufficient [14], [15].

Evolutionary and genetic approaches maintain a population of candidate partitions or color assignments. Selection favors useful candidates; recombination combines parent structures; mutation introduces variation; and local search may refine offspring. Encoding matters: fixed- k encodings may maintain a color count but allow conflicts, while partition encodings represent color classes directly. There is no single universal "genetic algorithm" or complexity expression; the cost depends on representation, population size, fitness evaluation, operators,

local refinement, and stopping rule. Hybrid methods that combine recombination with tabu search have been studied as a distinct algorithm family [16].

Surveys of vertex-coloring algorithms emphasize that no method dominates independently of graph class, objective, and computational budget. Benchmark results are instance-set dependent. A high-performing algorithm on random graphs may perform differently on scheduling graphs, sparse structured graphs, or register-interference graphs. Comparisons should therefore describe both the algorithm and the distribution or provenance of the instances [9], [10].

3. Mathematical Formulation, Bounds, and Proofs

3.1 Basic bounds and elementary proofs

The clique bound $\omega(G) \leq \chi(G)$ follows because vertices in a clique are pairwise adjacent and therefore must all receive distinct colors. For the independence bound, suppose a proper coloring uses k nonempty color classes $C_1, \dots, C_k$. Each class is independent, so $|C_i| \leq \alpha(G)$. As the classes partition V , $n = \sum_i |C_i| \leq k\alpha(G)$; dividing and using integrality yields $k \geq \lceil n/\alpha(G) \rceil$. Minimizing over all proper colorings gives the stated lower bound.

For the greedy upper bound, order vertices $v_1, \dots, v_n$. At the moment v_i is colored, the set of already colored neighbors has size at most $d(v_i) \leq \Delta$. They can forbid no more than Δ distinct colors. From a palette of $\Delta+1$ colors, at least one is available. Applying this argument to every vertex constructs a proper coloring using at most $\Delta+1$ colors. The proof does not rely on a favorable ordering.

For a cycle C_n , if n is even, alternating colors 1 and 2 around the cycle is proper, and an edge requires at least two colors; hence $\chi(C_n) = 2$. If n is odd, alternation with two colors fails at the closing edge. Every cycle contains an edge so one color is insufficient; the assignment 1,2,1,2,...,1,3 gives a proper 3-coloring, hence $\chi(C_n) = 3$. For K_n , every pair is adjacent, so all n vertices require distinct colors and $\chi(K_n) = n$.

These derivations provide elementary certificates. A lower bound and a coloring upper bound that coincide establish an exact value. For example, K_n has a clique of size n and a construction using n colors, so the lower and upper bounds match. For many graphs, clique and degree bounds do not coincide; algorithms can narrow the interval without proving exactness.

3.2 Brooks' theorem: statement and interpretation

The correct connected-graph form is: if G is connected and is neither a complete graph nor an odd cycle, then $\chi(G) \leq \Delta(G)$. The exceptional families show that the general $\Delta+1$ bound cannot always be reduced. An equivalent contrapositive characterization says that if $\chi(G) = \Delta(G) + 1$ for a connected graph, then G is complete or an odd cycle. Neither form gives a lower bound of Δ on $\chi(G)$; for example, every nontrivial bipartite graph has $\chi = 2$ even when its maximum degree is much larger than 2.

A proof of Brooks' theorem is beyond the scope of a short algorithm comparison because its regular cases require nontrivial structural arguments. The source draft presented a sketch with an incomplete BFS argument and omitted the difficult regular case. This revision labels the theorem as an established result, explains its hypotheses and consequences, and directs readers to standard proofs [3], [4]. That treatment is more accurate than presenting an incomplete argument as a proof.

3.3 Integer linear programming model

Let K be a valid upper bound on $\chi(G)$, such as $\Delta+1$ or the number of colors in a heuristic coloring. Introduce binary x_{vi} indicating whether vertex v uses color i and binary y_i indicating whether color i is used. The following assignment model minimizes the number of active colors:

$$\begin{aligned}
& \text{minimize } \sum_{i=1}^K y_i & (3) \\
& \text{subject to } \sum_{i=1}^K x_{vi} = 1 & \text{for every } v \in V & (4) \\
& x_{ui} + x_{vi} \leq y_i & \text{for every } \{u,v\} \in E \text{ and } i=1,\dots,K & (5) \\
& x_{vi} \leq y_i & \text{for every } v \in V \text{ and } i=1,\dots,K & (6) \\
& & x_{vi}, y_i \in \{0,1\} & (7)
\end{aligned}$$

The first constraint assigns exactly one color to each vertex. For an edge $\{u,v\}$, the second constraint prevents both endpoints from using color i ; it also forces $y_i=1$ if either endpoint uses it. The third constraint links every vertex assignment to an active color. Some formulations omit the third constraint because edge constraints do not activate colors for isolated vertices; that omission can invalidate the objective's interpretation on graphs with isolated vertices. The displayed model is valid for all finite simple graphs.

There are $O(nK)$ assignment variables, K usage variables, n assignment constraints, mK edge-color constraints, and nK linking constraints. The model size is polynomial in n and K , but solving the integer program remains NP-hard. Its linear relaxation provides a lower bound on the integer optimum only when solved correctly; a heuristic solution to the relaxation is not automatically a certificate. Symmetry-breaking constraints such as $y_i \geq y_{i+1}$ can reduce equivalent color-label permutations without changing the optimum.

3.4 Objective functions and solution-quality measures

For a fixed- k local-search method, define the conflict count $F(c) = \sum_{\{u,v\} \in E} 1[c(u) \neq c(v)]$. A valid k -coloring exists exactly when a search finds an assignment with $F(c)=0$. During search, F is a surrogate objective; it is not the chromatic number. A convenient move-cost update can examine only edges incident to a recolored vertex, but its complexity depends on degree and data structures.

$$F(c) = \sum_{\{u,v\} \in E} 1[c(u) \neq c(v)] \quad (8)$$

If an optimal value $\chi^*(G)$ is known, an absolute additive gap is $q - \chi^*(G)$, and a relative gap may be defined as $100(q - \chi^*)/\chi^*$. If only a certified lower bound $L \leq \chi^* \leq U$ is known, the reported coloring q establishes $q \geq \chi^*$, while the interval-based relative certificate can be reported against L as $100(q - L)/L$, explicitly called a lower-bound-relative gap rather than an optimality gap. If $L=0$, this ratio is undefined; for nonempty graphs, useful positive lower bounds are normally available. A best-known heuristic value is not necessarily an optimum and must not be substituted silently for χ^* .

$$\text{Gap to known optimum (\%)} = 100 \times (q - \chi^*) / \chi^* \quad (9)$$

For stochastic methods, let q_r denote the number of colors in independent run r and t_r the runtime under a clearly defined stopping rule. Report the number of instances, number of runs, seed policy, best/median/mean and dispersion as appropriate, and any censoring at a time limit. Since color counts are integer-valued and distributions can be skewed, the median and interquartile range may be more informative than the mean and standard deviation. Confidence intervals require a stated sampling unit and a justified method; repeated seeds on a single graph do not turn one graph into many independent graph instances.

3.5 Proof certificates and algorithm outputs

A coloring certificate consists of a vertex-to-color assignment and the graph edge list. A verifier checks that every vertex has exactly one assigned label and that no edge has equal endpoint labels. Given adjacency lists, the check takes $O(n+m)$ time, and successful verification establishes an upper bound $\chi(G) \leq q$, where q is the number of distinct labels used. A clique

certificate is a set of pairwise adjacent vertices; checking it takes $O(r^2)$ by direct pair tests or $O(r+m)$ with suitable data structures. Together, a q -coloring and a clique of size q prove $\chi(G)=q$. When the best lower bound is smaller than q , the result remains an interval rather than an exact answer.

Exact solvers can return more than a coloring. A proof of optimality requires a feasible upper bound q and a certified lower bound q , or a verifiable proof log establishing that all smaller possibilities were eliminated. Solver status strings such as “optimal” are useful but depend on correct model construction and software execution. For high-assurance work, preserve model files, solver logs, versions, parameter settings, and any available independently checkable certificates. A reported runtime without status and bound information is incomplete when the research question concerns exactness.

Heuristics should also be validated independently of their internal bookkeeping. Bugs in incremental conflict counts, color-class updates, or graph parsing can produce invalid assignments or understated objective values. The evaluator should recompute the number of colors and conflicts directly from the original graph, not trust a cached counter maintained by the algorithm. A graph parser should reject malformed edges, self-loops if the model is simple, duplicate vertices, and inconsistent n or m metadata; if preprocessing removes isolated vertices or components, the transformation and reconstruction should be documented.

The distinction between feasible evidence and optimality evidence is central to responsible reporting. A validated q -coloring is a constructive witness. A lower bound L is a proof that fewer than L colors cannot suffice. An exact answer is available when $L=q$. If $L < q$, the interval $[L, q]$ accurately describes current evidence. When q is best-known rather than optimal, use that wording consistently in tables, abstracts, and conclusions. A later improvement in the literature can change best-known values without changing the validity of a previously verified coloring.

4. Algorithms and Complexity

4.1 Sequential greedy and Welsh–Powell

Sequential greedy requires an explicit ordering σ . For each vertex in σ , it assigns the smallest positive integer not already assigned to a colored neighbor. A correct implementation may store, for each vertex, a set or mark array of forbidden colors and update it from incident edges. With adjacency lists and a suitable reusable mark array, the coloring pass can be implemented in $O(n+m)$ time up to color-management overhead; using a naïve set construction or scanning all existing colors may cost more. Space is $O(n+m)$ for the graph and color assignment.

The Welsh–Powell procedure first sorts vertices in nonincreasing degree and then colors sequentially. Comparison sorting costs $O(n \log n)$, while coloring costs depend on representation and implementation, often $O(n+m)$ under suitable adjacency structures. The order is a heuristic choice: degree sorting may help but does not guarantee optimality. A deterministic tie-breaking rule, such as vertex identifier, is needed for reproducibility.

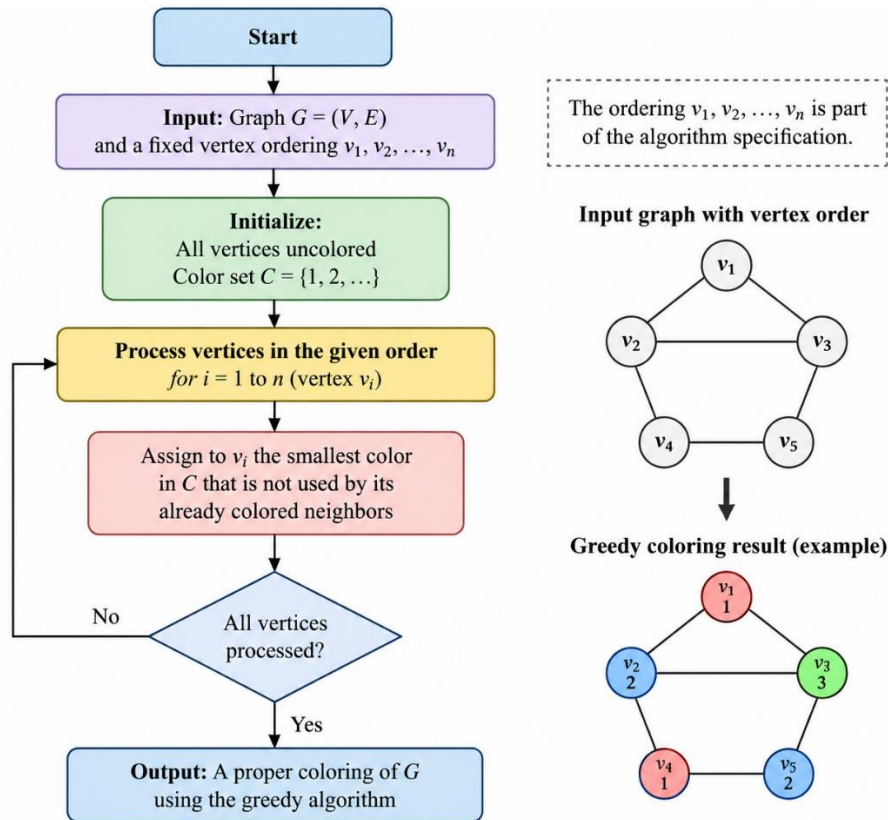


Figure 4. Greedy coloring flow. The ordering is part of the algorithm specification.

4.2 DSATUR

For an uncolored vertex v , define its saturation degree as the number of distinct colors appearing on already colored neighbors: $\text{sat}(v) = |\{c(u) : u \in N(v), u \text{ already colored}\}|$. DSATUR repeatedly selects an uncolored vertex with maximum saturation, commonly breaking ties by maximum degree and then a fixed identifier. It colors the selected vertex with the smallest feasible color and updates the saturation sets of its uncolored neighbors.

$$\text{sat}(v) = |\{c(u) : u \in N(v), u \text{ is colored}\}| \quad (10)$$

A straightforward implementation that scans all uncolored vertices at each iteration and recomputes saturation information can take $O(n^2 + m)$ or worse depending on the set operations. Priority queues and incremental color-set updates improve selection costs, but lazy invalidation and ties require careful bookkeeping. A commonly quoted $O((n+m) \log n)$ bound applies only to a specific implementation model and should not be asserted for DSATUR in general without stating data structures and update costs. In exact branch-and-bound variants, DSATUR is a branching rule, and total search time is exponential in the worst case.

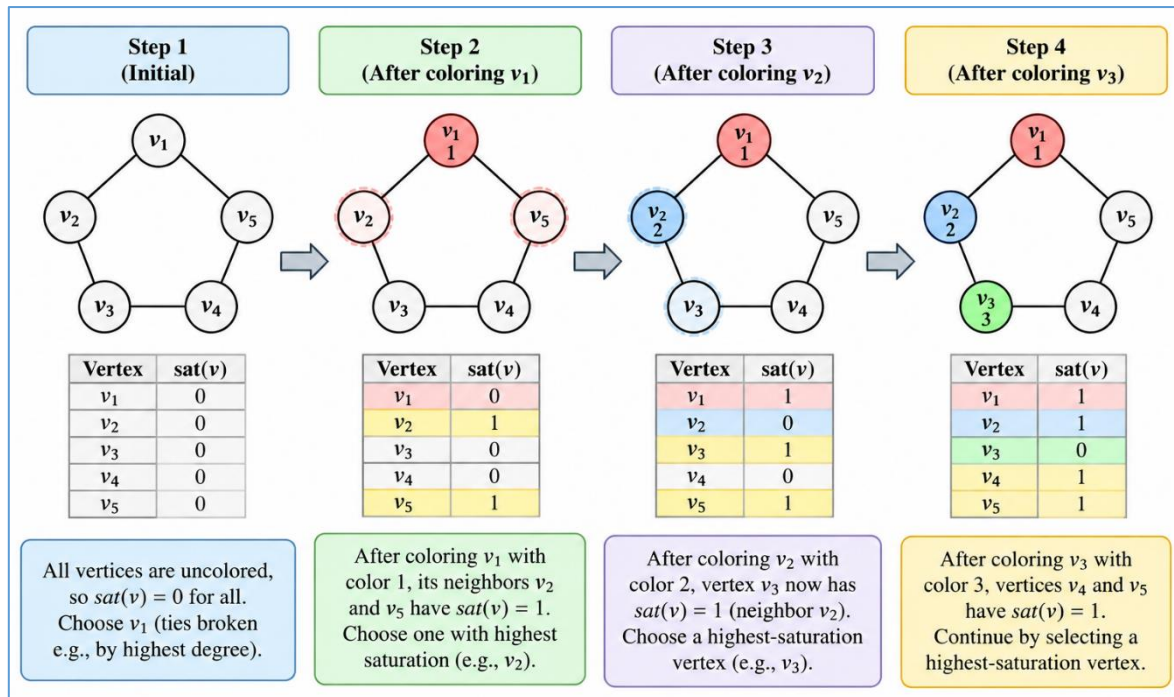


Figure 5. DSATUR selection depends on the evolving partial coloring.

DSATUR is a constructive heuristic when it returns a coloring from its greedy selection process. If combined with branch-and-bound, it can participate in an exact method that proves optimality. The same name therefore may describe different algorithmic roles. Reports should state whether backtracking occurs and whether an optimality proof was produced.

4.3 Exact optimization and certificates

Exact branch-and-bound starts from a feasible incumbent, computes lower bounds for subproblems, branches, and prunes any branch unable to improve the incumbent. A clique is an immediate lower bound. Stronger bounds may come from fractional relaxations or problem-specific structure. An exact result should include solver status and, where possible, a proof certificate or a reproducible solver log. Time-limited exact search may return a feasible upper bound and a lower bound without closing the gap; this is useful evidence but not an optimal solution.

The ILP formulation in Section 3 can be solved with general-purpose mixed-integer solvers. A rigorous experiment should report solver name and version, all options that affect search, presolve and symmetry handling, machine information, wall-clock or CPU-time definition, time and memory limits, and whether the solver terminated optimally. A statement that exact algorithms were “limited to $n \leq 100$ ” should be understood as a chosen study cutoff only if that cutoff was actually used and documented; it is not a universal complexity threshold.

4.4 Tabu search

A standard fixed- k tabu search begins with an assignment of vertices to k classes, possibly containing conflicts. It evaluates conflicting edges, considers moving a conflicted vertex to another class, and selects the best allowed move according to a deterministic or randomized tie rule. A tabu tenure prevents recently reversed assignments; aspiration can override the restriction when a move improves the best solution. If a zero-conflict state is found, it certifies k -colorability. If not, the run ends without proving infeasibility at k .

A meaningful complexity estimate must identify the neighborhood. If every conflicted vertex may move to each of $k-1$ other colors, the nominal neighborhood can be $O(nk)$, but an implementation often evaluates only feasible or conflict-relevant moves and caches conflict counts. One iteration can then cost substantially less than rescanning every edge, or more if

full recomputation is used. The source draft's fixed tenure rule $t \approx 0.6n$ is not a general theorem and should not be presented as a universal parameter; tenure is an algorithm parameter to tune or report.

4.5 Genetic and hybrid evolutionary algorithms

A genetic approach requires an explicit encoding. In a vertex-to-color encoding, the chromosome stores one label per vertex and the fitness may penalize monochromatic edges. In a partition encoding, each chromosome is a partition into color classes, which can simplify feasibility but complicate crossover. Common operators include tournament or rank selection, partition-aware recombination, mutation, and local improvement. An objective that rewards fewer conflicts alone may not distinguish assignments well when the fixed color count changes; lexicographic or two-stage objectives can handle validity and color count explicitly.

If population size is P , the number of generations is G , and one fitness evaluation scans all m edges, the evaluation component alone is $O(PGm)$. This is not the full complexity: crossover, mutation, local search, and data structures add costs. Any claimed runtime or complexity should state whether it is per generation, per run, or total. Hybrid methods can be effective because global recombination and local refinement explore different structures, but their performance must be established on specified instances rather than inferred from the algorithm label.

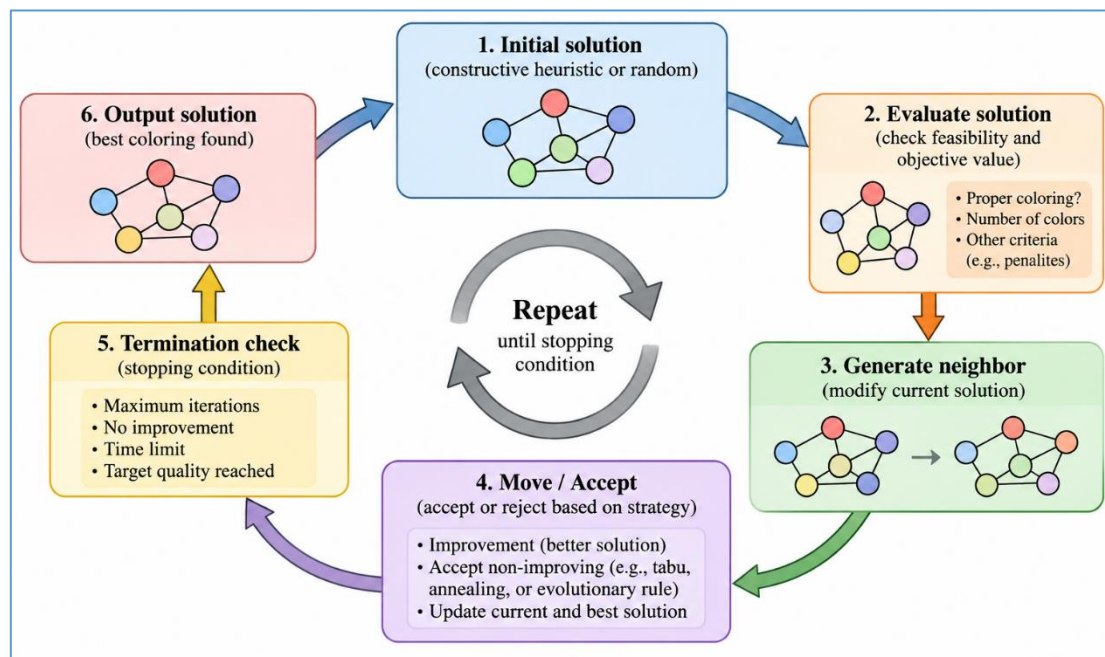


Figure 6. Conceptual search cycle. The diagram is methodological, not a claim of measured performance.

4.6 Summary comparison

Table 1. Algorithm families, guarantees, and reporting requirements

Method	Class	What it returns / guarantees	Main costs and caveats
Greedy	Constructive heuristic	Valid coloring; at most $\Delta+1$ colors	Order-sensitive; typically near-linear with adjacency lists
Welsh–Powell	Degree-ordered heuristic	Valid coloring; no optimality guarantee	Sorting plus coloring; tie rule must be fixed

DSATUR heuristic	Dynamic constructive heuristic	Valid coloring; no general optimality guarantee	Selection/update costs depend on implementation
DSATUR + branch-and-bound	Exact search	Optimality if search closes the proof gap	Exponential worst case; report bounds and termination
ILP / exact solver	Exact optimization	Optimality when solver proves it	Model size $O(nK+mK)$; solving is NP-hard
Tabu search	Local-search metaheuristic	Zero-conflict assignment certifies k-colorability	Failure under a budget proves no infeasibility
Genetic / hybrid	Population metaheuristic	Best candidate found within budget	Encoding and operators define behavior; no universal bound

5. Research Methodology and Reproducible Experimental Design

5.1 Design principles

The source manuscript does not contain enough information to reconstruct an experiment. The protocol below is therefore a design for a future study, not a report of completed tests. A comparison should separate two tasks: (a) evaluate constructive coloring quality at a fixed computational budget; and (b) evaluate exact proof capability, including lower-bound progress and optimality certification. Mixing these tasks in one ranking can obscure meaningful differences.

Use public instances with known provenance, such as graph-coloring collections associated with the DIMACS challenge, and include multiple families: random graphs, structured benchmark graphs, and application-derived instances where licensing and documentation permit. Record each instance's filename, checksum, source URL, n , m , density, directedness conversion, and any preprocessing. Benchmark sets should not be selected after observing algorithm performance. Publicly available instances support independent replication and comparison with prior studies [17], [18].

For generated graphs, specify the exact model and parameters. In an Erdős–Rényi $G(n,p)$ model, every unordered pair is independently included with probability p , with expected edge count $p \cdot n(n-1)/2$. A preferential-attachment model is generated by a different process and tends to produce a degree distribution unlike $G(n,p)$; calling one model “dense random graphs” and another “preferential attachment” does not define an equivalent density comparison. Describe the generator, its parameters, and random seed. Scheduling graphs should be generated from explicit event-conflict rules, not only labeled “structured.”

5.2 Algorithms, environment, and budgets

Implementations must be identified by source repository and commit or release version. For each algorithm, publish pseudocode or configuration, tie-breaking policy, initial solution procedure, parameters, stopping conditions, and random seed. The same hardware, compiler/interpreter, operating system, and resource limits should be used where feasible. Report whether time means CPU time or wall-clock time, whether preprocessing is included, and how parallel threads are controlled. Use an independent validation routine that checks all m edges after every returned assignment.

A fair heuristic comparison can allocate the same wall-clock budget per instance or the same number of objective evaluations, provided the choice reflects the intended use. Both metrics may be reported because algorithms differ in the cost of one move. For randomized algorithms,

run independent seeds for every instance, predeclare the number of runs, and report failures and timeouts. Exact solvers should share a time and memory limit; publish both their incumbent coloring and certified lower bound if they do not prove optimality.

5.3 Instances, metrics, and statistics

The primary quality metric is q , the number of colors in a validated proper coloring. Runtime is measured under the declared execution environment. Validity rate is the proportion of runs that return a proper coloring before the limit. For exact methods, report best upper bound U , certified lower bound L , solver status, and whether $U=L$. If χ^* is proven, report $q-\chi^*$ and the relative gap. If not, report the interval $[L,U]$ and do not call a heuristic's best-known q an optimum.

Graph density for a simple undirected graph with $n \geq 2$ is $\rho = 2m/[n(n-1)]$. It is a descriptive property, not a complete predictor of difficulty: graphs with equal density can differ in clique structure, degree distribution, degeneracy, community structure, and symmetry. Report these characteristics when they inform interpretation. For benchmark graphs with a known optimum, cite the source that certifies it; otherwise report the best known lower and upper bounds together with retrieval date.

$$\rho(G) = 2m / [n(n-1)] \quad (11)$$

For each instance, report the distribution across independent algorithm runs. Across instances, use paired comparisons because algorithms are tested on the same graph set. Consider medians, interquartile ranges, bootstrap confidence intervals over instances, and paired nonparametric tests when assumptions justify them. Adjust for multiple comparisons if making many pairwise claims. Statistical significance does not by itself imply practical importance; report effect sizes and absolute color differences. Avoid pooling all run-level observations as if they were independent when they share the same graph.

Archive raw outcomes in a machine-readable file with columns for instance, graph family, n , m , density, algorithm, version, seed, run index, runtime, number of colors, conflicts, lower bound, upper bound, status, and error message. Release scripts for parsing, validation, aggregation, and figure generation. Record all exclusions and unsuccessful runs. These measures make it possible to reproduce the analysis rather than only the headline table.

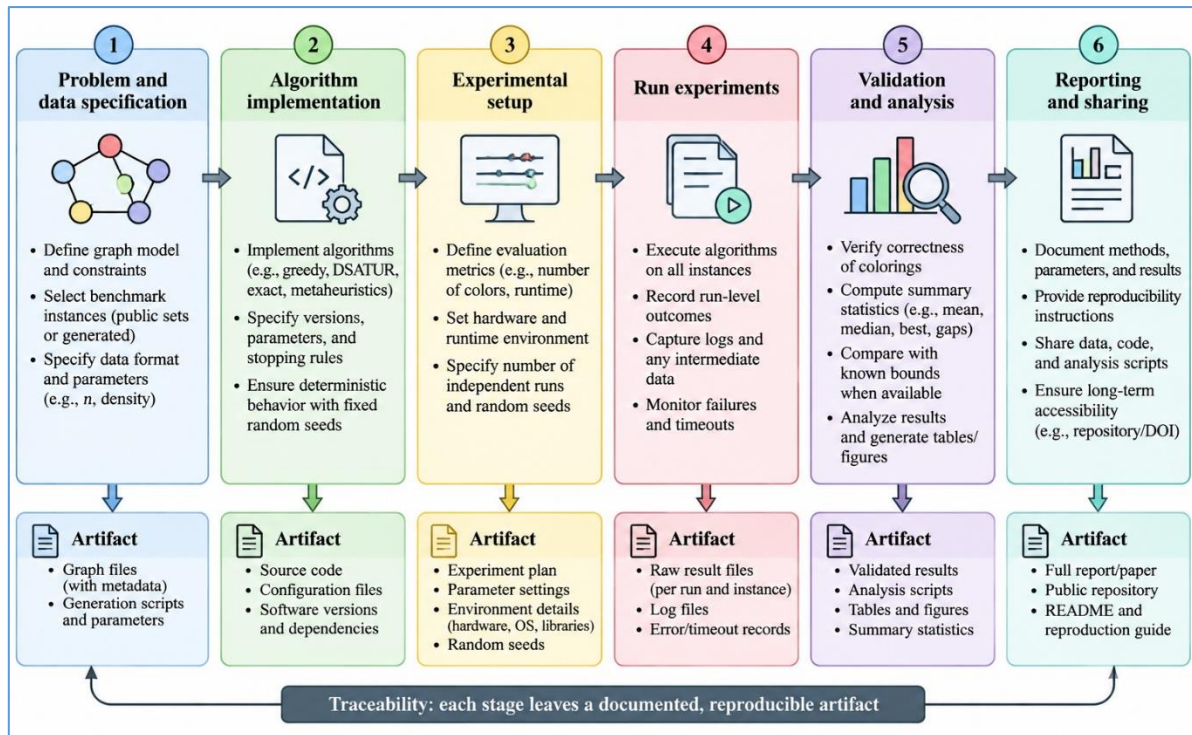


Figure 7. Reproducible evaluation workflow. Every stage should leave a traceable artifact.

6. Results and Quantitative Audit

6.1 Evidence available in the source manuscript

The source draft reports one table for a dense graph described as $n=500$ and $p=0.5$. It lists average color counts of 38, 34, 31, 29, and 28 for greedy, Welsh–Powell, DSATUR, a genetic algorithm, and tabu search, respectively, with runtimes and gaps. The file does not identify the actual edge list or generator seed, number of instances, number of independent runs, software versions, hardware, timing method, tabu or genetic parameters, or source of the optimal chromatic number. No raw trial-level observations or code are included. The table therefore cannot be independently recomputed or verified.

The reported gap values also cannot be validated from the table alone. If 28 colors is treated as a candidate best solution and a listed gap is meant to be relative to the unknown optimum, then the denominator is missing. In addition, $q-\chi^*$ divided by χ^* cannot be calculated unless χ^* is known or certified. “Average colors” and a single gap percentage require a definition of averaging and aggregation; averaging per-run relative gaps is not generally equal to computing a gap from average color counts.

The text also claims that exact algorithms determined optima for small graphs, that particular heuristics fell within percentage bands, and that algorithms were tested through $n=1000$. These claims are not supported by instance files, logs, raw outcomes, or implementation details in the provided manuscript. They are removed as findings and listed as unresolved verification requirements. The draft’s statements that Tabu Search was fastest or best, and that results “prove” Brooks’ theorem, are not supported and are mathematically inappropriate: experiments cannot prove a universal theorem.

Table 2. *Audit of the source draft's performance claims*

Source claim	Evidence supplied	Revision treatment
n=500, p=0.5 table with five methods	Summary values only; no graph, seeds, logs, or code	Excluded from empirical findings; preserve in audit log only
Optimality gaps of 22%, 14%, 6%, 3%, 1–2%	No certified χ^* or lower bound; formula formatting ambiguous	Not recalculated; gap is undefined from supplied evidence
Small-instance optimality and percentage bands	No instance list or trial outcomes	Marked unverified; requires exact solver records
Tests up to n=1000 and runtime statements	No code, hardware, timing method, or results per instance	Removed as demonstrated result; can be tested under Section 5
Tabu Search universally best / fastest	No paired results, uncertainty, or fair-budget description	Replaced by conditional methodological discussion

6.2 Analytical examples (not experiments)

To demonstrate a verifiable bound, consider the cycle C_5 . It has $n=5$, $m=5$, $\Delta=2$, $\omega=2$, and $\alpha=2$. The lower bound from independence is $\lfloor 5/2 \rfloor = 3$, while the generic greedy upper bound is $\Delta+1=3$. Since the bounds coincide, $\chi(C_5)=3$. This is a hand-derived mathematical example, not a measurement of an algorithm's runtime.

For the source draft's five-vertex example with edges $\{a,b\}$, $\{a,c\}$, $\{b,c\}$, $\{b,d\}$, $\{c,e\}$, $\{d,e\}$, the triangle $\{a,b,c\}$ gives $\chi \geq 3$. Under the order (a,b,c,d,e) , greedy assigns colors 1,2,3,1,2, so the coloring uses three colors and is optimal because the lower bound is also three. The numerical result can be verified directly from the listed graph and order; it does not support broader performance estimates.

The formula for the density of a simple undirected graph at $n=500$ and $p=0.5$ yields an expected edge count of $0.5 \times 500 \times 499 / 2 = 62,375$ under $G(500,0.5)$. This is an expectation, not the exact edge count of an unspecified sampled graph. A single random graph's realized m must be reported before its exact density can be computed. This distinction illustrates why "n=500, p=0.5" is not enough to reproduce an instance.

6.3 Theory-only scalability illustration

Figure 8 plots normalized asymptotic growth functions for $O(n)$, $O(n^2)$, and $O(n^3)$. The curves illustrate growth-rate differences only: axes are normalized for visual comparison and do not represent measured runtimes, operation counts for a particular implementation, or fitted data. No empirical performance chart is provided because the source values lack provenance and the manuscript contains no raw results. Adding a bar chart of those values would convert unverified claims into a persuasive but unsupported visual.

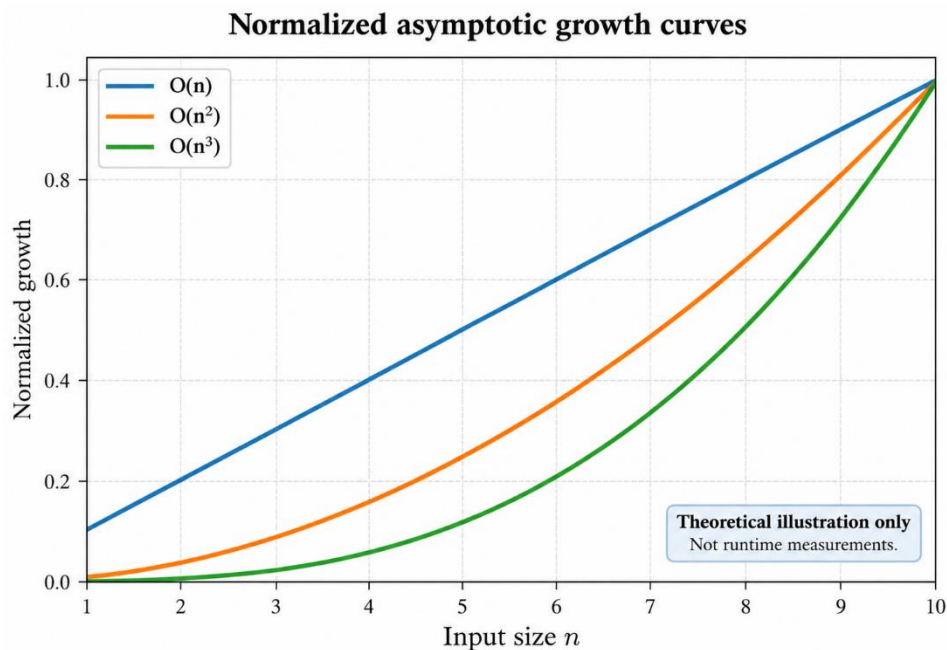


Figure 8. Normalized asymptotic growth curves. These are theoretical illustrations, not runtime measurements.

7. Discussion and Practical Applications

7.1 Interpreting algorithmic trade-offs

Constructive heuristics are attractive when a valid coloring is needed quickly. They provide a certificate of feasibility and an upper bound but usually no proof of optimality. Degree orderings and saturation-based selection use local structure to avoid some poor fixed orders; their benefit is instance dependent. Exact algorithms are appropriate when proof of optimality or infeasibility matters and the instance is tractable under available resources. Metaheuristics can search for smaller colorings under fixed budgets but do not provide general certificates when they fail to find a solution.

The appropriate method depends on the decision context. A small coloring improvement may be valuable if each color corresponds to a costly resource, while a rapid valid assignment may be sufficient in a daily scheduling process. If colors have unequal costs, capacities, or preference penalties, minimizing $\chi(G)$ alone may be the wrong model. The objective may need weighted colors, balance constraints, or soft penalties; those extensions alter both the formulation and guarantees.

Graph size alone is not a reliable indicator of difficulty. Density, clique number, degeneracy, degree distribution, symmetry, and benchmark family can influence algorithm behavior. Runtime comparisons should not attribute a difference to algorithm quality if one implementation uses a more efficient data structure or a different stopping budget. Claims about “better scalability” require measurements over stated instance families plus complexity analysis; asymptotic bounds by themselves do not establish practical speed.

7.2 Application modeling

In examination timetabling, each exam is a vertex and an edge joins two exams with at least one student in common. A proper coloring assigns conflict-free time slots. Real timetabling commonly adds room capacities, exam-duration constraints, student preferences, spacing rules, and penalties for inconvenient schedules. Those constraints are not represented by ordinary coloring and may require weighted or multiobjective models. The graph-coloring abstraction is most useful as a core subproblem or a first feasibility model [11], [12].

In frequency assignment, vertices can represent transmitters and edges indicate pairs that cannot share a frequency under the chosen interference threshold. Some systems require frequency separation that grows with interference or distance; this is a T-coloring or channel-assignment variant, not plain vertex coloring. The model must disclose how interference is measured and whether constraints are binary or distance-based [19].

In compiler register allocation, an interference graph connects variables whose live ranges overlap. Colors represent machine registers. If the graph cannot be colored with the available register count, spilling moves some values to memory, introducing execution costs. Thus the practical problem is not always to minimize the number of colors alone; spill cost, register classes, calling conventions, and instruction constraints matter [20].

Other scheduling and resource-allocation problems can use the same abstraction when incompatibilities are pairwise and resources are interchangeable. If resources differ, if constraints are higher order, or if assignments have temporal dependencies, a plain graph may omit material structure. Model validation should precede algorithm selection: a precisely solved wrong model is still a poor schedule.

7.3 Limits on conclusions

This manuscript supports conclusions about definitions, elementary bounds, theorem statements, algorithmic mechanisms, and evidence requirements. It does not support a conclusion that DSATUR is best for medium instances, that Tabu Search is the fastest method, or that any reported percentage gap holds across a graph family. Those conclusions require the experiments specified in Section 5. Published studies can inform algorithm design, but their numerical results should not be transferred to the source manuscript as if they were new measurements.

The literature itself spans variants and implementations. Comparing a modern hybrid algorithm with a basic greedy implementation on different hardware or different benchmark files would not isolate algorithmic effects. A fair study must describe the experimental unit, benchmark selection, computational budget, implementation provenance, and statistical analysis. Independent reproduction and artifact release are particularly important for randomized methods.

8. Limitations and Future Research

The principal limitation is that no underlying experimental data were supplied with the source document. The original empirical claims cannot be confirmed, recalculated, or repaired by editing prose. The revised manuscript therefore replaces empirical conclusions with a transparent protocol and with small hand-checkable examples. The absence of experiments is not evidence that the original values are false; it means their truth cannot be established from the provided materials.

A first future step is to rerun the comparison on public benchmark instances using archived code, fixed versions, and a preregistered evaluation plan. It should include several graph families and sizes, independent seeds, equalized budgets, edge-by-edge validity checks, certified lower bounds, and raw data. An accompanying artifact should contain graph checksums, environment metadata, scripts, and logs. Results should distinguish an optimum from the strongest known bound at the time of the study.

Hybrid algorithms offer a reasonable research direction because construction, population search, and local refinement can complement each other. A study should compare the hybrid not only with older baselines but also with strong modern implementations under equal budgets. Ablation experiments can isolate the contribution of recombination, tabu search, and initialization. Such a proposal is a research question, not evidence that a hybrid will outperform all alternatives.

Parallel and GPU-assisted coloring raise questions about synchronization, color conflicts, load balancing, and deterministic output. A parallel algorithm may reduce wall-clock time while

producing more colors, so quality and runtime must be reported jointly. Streaming and online settings require updates as vertices or edges arrive; static algorithms may need repair or recoloring policies. Machine-learning-assisted ordering can be evaluated for transfer across graph families, but data leakage and distribution shift require explicit controls.

Quantum approximate optimization algorithms have been applied experimentally to combinatorial optimization formulations, including graph coloring, but no general super-polynomial speedup should be claimed without a proven complexity result. Future work should compare encodings, penalty design, embedding overhead, noise, solution validity, and classical baselines. Practical quantum advantage remains an empirical and theoretical question.

Finally, model extensions deserve attention. List coloring allows a different set of permissible colors per vertex. Equitable coloring constrains class sizes. Weighted coloring assigns costs to color classes. Robust scheduling handles uncertain or changing conflicts. Each extension changes the feasible region and may invalidate guarantees from ordinary coloring; theorem statements must be rechecked in the relevant model.

9. Conclusion

This revised manuscript establishes a corrected foundation for graph vertex coloring and distinguishes mathematical guarantees from heuristic outcomes. It defines proper coloring and chromatic number, derives clique and independence lower bounds and the $\Delta+1$ greedy upper bound, states Brooks' theorem with its connectedness and exception conditions, and separates vertex coloring from edge coloring. It also explains the role and limitations of greedy, Welsh–Powell, DSATUR, exact optimization, tabu search, and evolutionary methods.

The source draft's numerical performance table, optimality gaps, runtime statements, and claims of algorithmic superiority could not be independently verified because the manuscript did not include instance files, seeds, software and hardware details, parameters, raw outcomes, or certified optimal values. They are therefore not presented as findings. The paper supplies a reproducible protocol that can support a future empirical study and keeps hand-calculated examples explicitly separate from experimental evidence.

The defensible conclusion is methodological: graph-coloring algorithms should be compared with validated solutions, stated computational budgets, traceable instances, and bound-aware metrics. No single heuristic can be declared universally best from the available evidence. Once the missing artifacts are recovered or the experiments are rerun, quantitative conclusions can be added with appropriate uncertainty and scope.

References

1. Appel, K., & Haken, W. (1977a). Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3), 429–490. <https://doi.org/10.1215/ijm/1256049011>
2. Appel, K., Haken, W., & Koch, J. (1977). Every planar map is four colorable. Part II: Reducibility. *Illinois Journal of Mathematics*, 21(3), 491–567. <https://doi.org/10.1215/ijm/1256049012>
3. Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4), 251–256. <https://doi.org/10.1145/359094.359101>
4. Brooks, R. L. (1941). On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2), 194–197. <https://doi.org/10.1017/S030500410002168X>
5. Chaitin, G. J. (1982). Register allocation and spilling via graph coloring. *ACM SIGPLAN Notices*, 17(6), 98–105. <https://doi.org/10.1145/872726.806984>
6. Cranston, D. W., & Rabern, L. (2014). *Brooks' theorem and beyond* [Preprint]. arXiv. <https://arxiv.org/abs/1403.0479>

7. DIMACS. (n.d.). *Implementation challenges: Second challenge (1992–1993): NP-hard problems: Maximum clique, graph coloring, and satisfiability*. <https://dimacs.rutgers.edu/programs/implementation-challenges/>
8. Galinier, P., & Hao, J.-K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3, 379–397. <https://doi.org/10.1023/A:1009823419804>
9. Galinier, P., & Hertz, A. (2006). A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9), 2547–2562. <https://doi.org/10.1016/j.cor.2005.07.028>
10. Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman.
11. Garey, M. R., Johnson, D. S., & Stockmeyer, L. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3), 237–267. [https://doi.org/10.1016/0304-3975\(76\)90059-1](https://doi.org/10.1016/0304-3975(76)90059-1)
12. Hale, W. K. (1980). Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12), 1497–1514. <https://doi.org/10.1109/PROC.1980.11899>
13. Hertz, A., & de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, 39, 345–351. <https://doi.org/10.1007/BF02239976>
14. Jensen, T. R., & Toft, B. (2011). *Graph coloring problems*. Wiley.
15. Johnson, D. S., & Trick, M. A. (Eds.). (1996). *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26). American Mathematical Society. <https://doi.org/10.1090/dimacs/026>
16. Lewis, R. (2016). *A guide to graph colouring: Algorithms and applications*. Springer. <https://doi.org/10.1007/978-3-319-25730-3>
17. Lewis, R. M. R. (2018). A general-purpose hill-climbing method for graph colouring. *Journal of the Operational Research Society*, 69, 810–816. <https://doi.org/10.1057/s41274-017-0239-4>
18. Malaguti, E., & Toth, P. (2010). A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1), 1–34. <https://doi.org/10.1111/j.1475-3995.2009.00696.x>
19. Vizing, V. G. (1964). On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3, 25–30.
20. 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. <https://doi.org/10.1093/comjnl/10.1.85>
21. Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6), 103–128. <https://doi.org/10.4086/toc.2007.v003a006>

Compliance with ethical standards

Disclosure of conflict of interest

The authors declare that they have no conflict of interest.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of ALBAHIT and/or the editor(s). ALBAHIT and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content