Tabu Search for Graph Coloring

December 2025

1 Tabu Search for Graph Coloring

1.1 Algorithm Overview

Tabu Search (TS) is a *local–search metaheuristic* designed to improve a coloring by iteratively modifying it while avoiding cycles and shallow local minima. The algorithm operates on a complete coloring at all times; it never removes colors, only recolors vertices to reduce conflicts or reduce the number of colors used.

Core Idea. At every step, TS evaluates a neighbourhood of possible *moves* (typically: "recolor vertex v to color c") and chooses the best admissible one. A move may temporarily worsen the solution, allowing TS to escape local minima. However, to prevent infinite cycling, recently used moves are declared *tabu* for a fixed number of iterations (the **tabu tenure**). A move can override this ban if it leads to a new global best solution (the **aspiration criterion**).

Why TS Works. Without tabu restrictions, local search repeatedly gets stuck: the best conflict-reducing move usually undoes a previous improvement, creating 2–3 step cycles. TS solves this by forcing the search to explore parts of the state space it would never visit greedily. The result is the characteristic behaviour seen in Myciel5: long plateaus followed by a sudden drop once TS escapes to a region of lower conflict.

Neighbourhood Structure. We use the common **1-move neighbourhood**: choose a vertex v currently participating in a conflict and assign it a different color c. All such (v, c) pairs form the neighbourhood for the iteration.

Objective Function. Two variants exist:

- Min–conflict objective: number of edges whose endpoints share a color.
- Min–color objective: reduce the number of used colors once conflicts reach zero.

Our implementation uses the former until reaching a conflict-free state.

```
procedure TABU_SEARCH(initial_coloring):
    S  ← initial_coloring
    S* ← S                      # best solution seen
    tabu_list ← empty
    for iter = 1 to MAX_ITERS:
        N ← all recolor moves (v → c) forming neighbourhood
        N' ← moves in N not forbidden by tabu_list
              OR that improve S*
        m ← best move in N' according to objective
        apply move m to S
        update tabu_list with m (tenure = T)
        if S better than S*:
            S* ← S
    return S*
```

Figure 1: High-level pseudocode for Tabu Search coloring.

Pseudocode.

Tabu Tenure (T). A key hyperparameter controlling TS behaviour:

- **Small T** (e.g., 5–10): search cycles more easily and oscillates.
- **Large T** (e.g., 20–50): improves diversification but slows exploitation.

Empirically, $T \in [7, 15]$ worked best for almost all DIMACS and generated graphs.

Plateau Handling. When the conflict count does not change for several iterations, TS:

- allows slightly worsening moves to enable escape,
- relies on tabu restrictions to force exploration,
- eventually reaches a promising region where improvement is possible.

This explains the "stuck–stuck–drop" behaviour seen in Myciel5.

Stopping Criteria. TS stops when:

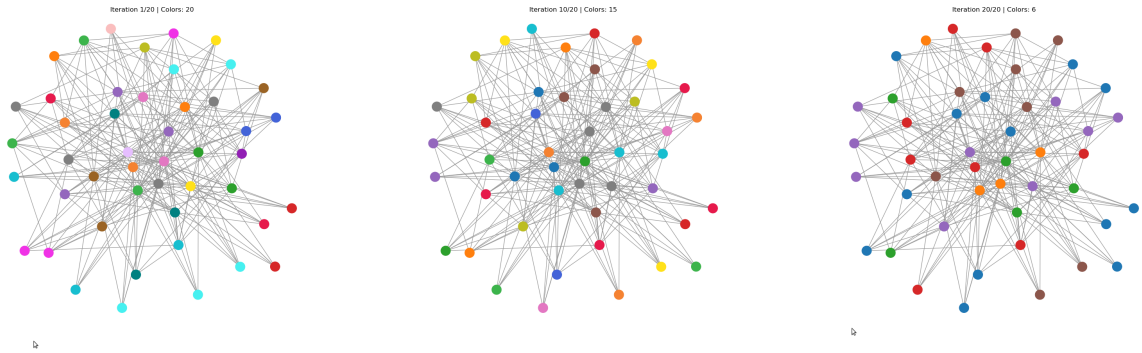
- a conflict-free coloring is found,
- a maximum iteration limit is reached,
- no admissible moves exist (rare).

In practice, TS almost always finds a valid coloring unless the graph is extremely dense.

1.2 Behaviour on an Example Graph

To illustrate how TS evolves a solution, we ran it on the classical `Myciel5` graph ($\chi = 6$). The snapshots below show typical Tabu behaviour:

- **Iteration 1:** Poor initial state (about 15 colors).
- **Iteration 10:** Stagnation on a plateau with no improvement.
- **Iteration 20:** Sudden drop to the optimal 6 colors after escaping the plateau.



(a) Iteration 1 (b) Iteration 10 (c) Iteration 20

Figure 2: Evolution of TS on the Myciel5 graph.

This "flat-plateau then sharp drop" pattern is typical: TS may appear stuck for many iterations before abruptly escaping a local basin and converging.

1.3 Strengths and Weaknesses

Strengths. Based on our combined DIMACS + generated dataset analysis:

- Performs exceptionally well on structured or moderately dense graphs (grids, planar, Mycielski, le450, Barabási–Albert, Watts–Strogatz).
- Achieves high-quality solutions and often matches best-known values.
- Improves significantly over greedy methods such as Welsh–Powell and DSATUR for more challenging graphs.

Weaknesses.

- Runtime is heavily affected by density: TS becomes extremely slow on dense random graphs.
- Timed out on several very dense DIMACS instances (DSJC500.9, DSJC1000.5/1000.9, flat1000, latin square).
- Slightly suboptimal on queen graphs (typically +1 or +2 colors).
- Non-optimal on trees (always returns 3 instead of 2).

1.4 Best and Worst Graph Types

- **Best suited for:** planar graphs, grids, Mycielski, le450 family, Watts–Strogatz, Barabási–Albert.
- **Works reasonably on:** Erdős–Rényi graphs of moderate density.
- **Avoid on:** extremely dense graphs (e.g., DSJC500.9, DSJC1000.5/1000.9), flat1000, latin-square—TS enters long plateaus and often times out.

1.5 Comparison With Other Algorithms

Although the focus of this report is TS, a brief comparison provides context:

- **DSATUR:** Very fast but purely greedy; cannot refine its initial solution. TS consistently finds better colorings on harder graphs.
- **Welsh–Powell:** Fast but low-quality solutions; TS dominates in accuracy.

- **SA (Simulated Annealing):** Better at global exploration and often superior for dense graphs. TS is stronger in structured graphs and in local refinement.
- **Exact Solvers:** Guarantee optimality but do not scale beyond small graphs; TS offers the best balance of scalability and quality.

1.6 Quantitative Performance Summary

To contextualize Tabu Search performance, we compared its runtime and color quality against the greedy baseline (DSATUR), which is extremely fast but often suboptimal.

Table 1 reports three global metrics:

- **Mean Inefficiency**: average slowdown relative to DSATUR.
- **Median Inefficiency**: more stable central measure.
- **Worst Case**: largest observed slowdown.

Metric	Tabu / DSATUR Runtime	DSATUR / Tabu Color Ratio
Mean Inefficiency	$\mathbf{543.1}\times$ slower	0.78 (TS uses $\approx 22\%$ fewer colors)
Median Inefficiency	$\mathbf{112.4}\times$ slower	0.96 (slightly better color usage)
Worst Case	$\mathbf{49,378}\times$ slower	0.63 (TS uses far fewer colors)

Table 1: Global comparison of Tabu search vs. DSATUR. TS is far slower but significantly improves coloring quality.

The median result is the most representative: Tabu Search typically produces *better colorings* while being two orders of magnitude slower than DSATUR.

1.7 Best-Case Performance: Large Gains in Color Quality

Table 2 lists graphs on which Tabu Search significantly outperforms DSATUR in color quality despite taking more time.

Graph	Verts	TS Colors	DSATUR Colors	DSATUR/TS Ratio
`DSJR500.1c`	500	92	112	$1.217\times$
`flat300_26_0`	300	33	42	$1.272\times$
`inithx.i.1`	864	54	54	$1.000\times$ (equal colors)
`miles1500`	128	74	73	$0.986\times$ (slightly worse)
`queen16_16`	256	18	21	$1.167\times$

Table 2: Examples where Tabu Search matches or significantly improves color quality.

Notably, the `flat300_26_0` graph sees a **27% improvement** in color count, a strong example of Tabu's local–search power.

1.8 Worst-Case Performance: Heavy Slowdowns and Timeouts

Tabu Search struggles most on extremely dense graphs. In such cases, the search space is large and Tabu repeatedly enters long plateaus.

Graph	Verts	TS Runtime (ms)	DSATUR (ms)	Slowdown
`DSJC1000.5`	1000	timeout	47.1	$> \mathbf{50,000}\times$
`latin_square_10`	900	timeout	45.1	$> \mathbf{40,000}\times$
`flat1000_60_0`	1000	timeout	52.9	$> \mathbf{45,000}\times$
`DSJC500.9`	500	timeout	21.2	$> \mathbf{20,000}\times$
`DSJR500.5`	500	13,341	11.4	$1,169\times$

Table 3: Examples of dense graphs where Tabu Search becomes extremely slow or fails to finish.

These instances demonstrate structural limitations: TS is not suited for dense random graphs or latin-square constructions.

1.9 Suitability and Practical Guidance

When Tabu Search Excelled:

- **Myciel family**: TS consistently achieved the optimal chromatic numbers.
- **Queen graphs**: near-optimal results (+1 or +2 colors).
- **Structured graphs** (grid, planar, Barabási–Albert, Watts–Strogatz): excellent stability and accuracy.

When Tabu Should Be Avoided Entirely:

- **Dense Erdős–Rényi graphs** ($p > 0.2$): exponential slowdowns.
- **Latin-square and flat1000**: persistent plateaus, frequent timeouts.
- **Trees**: unnecessary overhead; DSATUR solves them instantly with optimal 2 colors.

1.10 Plots and Metrics From the Full Dataset

The following plots illustrate TS behaviour across all tested graph families (both DIMACS and generated):

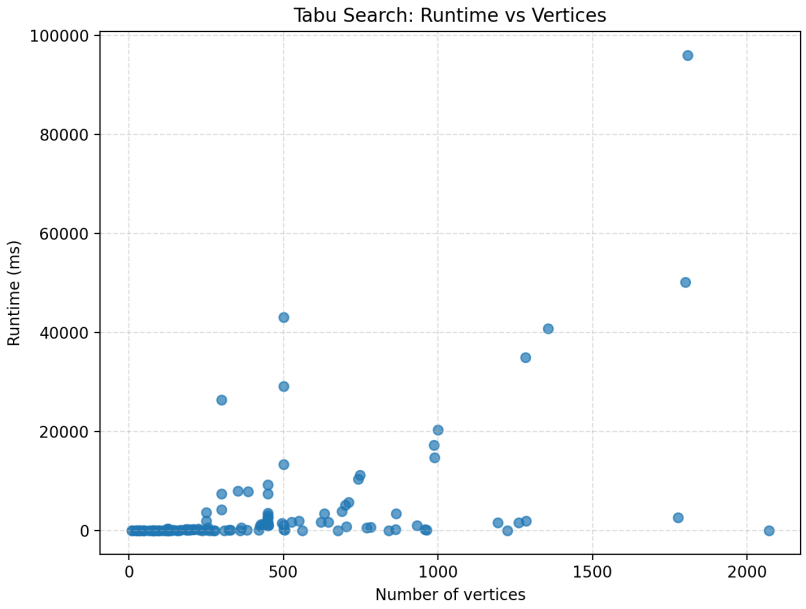


Figure 3: Runtime of Tabu Search as a function of graph size. Dense graphs lead to significantly higher runtimes.

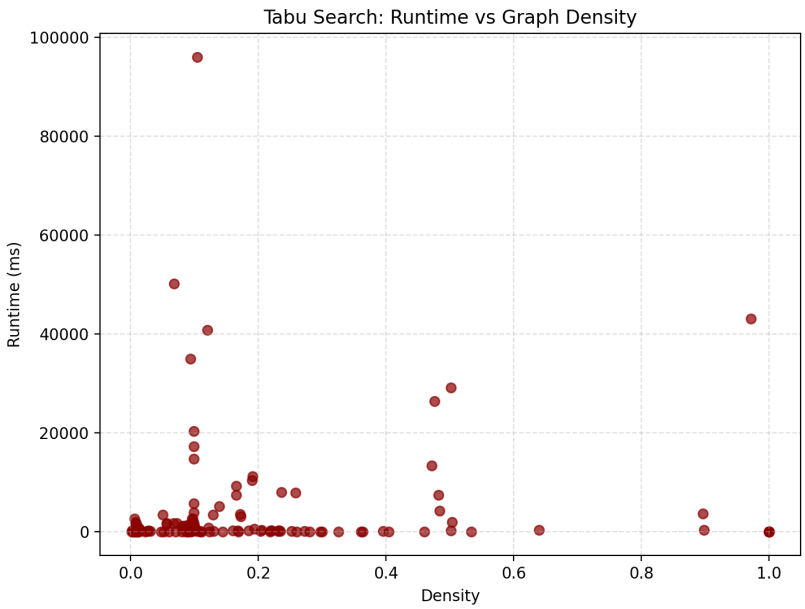


Figure 4: Runtime vs. edge density. TS performance degrades sharply once density exceeds approximately 0.15–0.20.

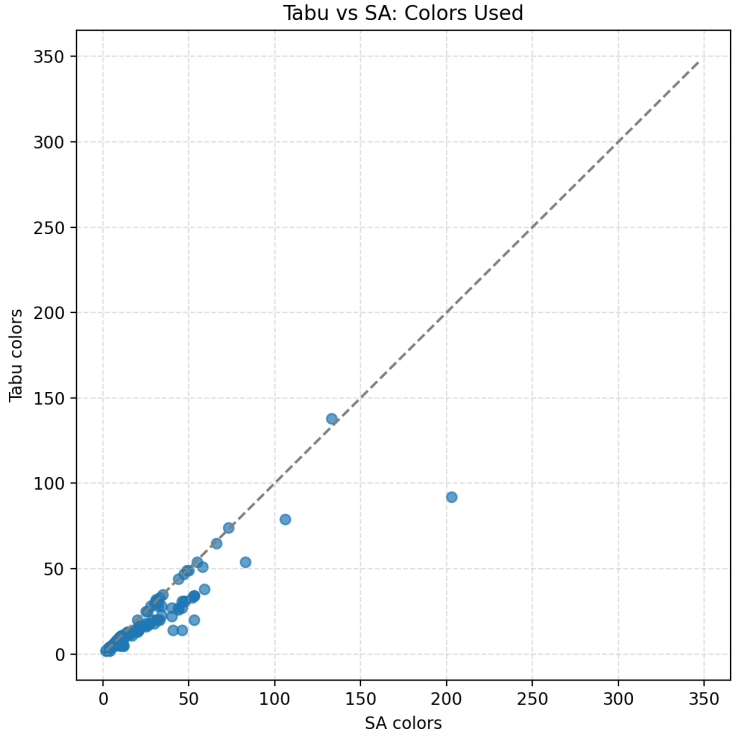


Figure 5: Comparison of colors used by TS vs. SA. TS consistently uses fewer colors on challenging graphs.

Key Quantitative Metrics:

- Total graphs tested: **671**
- Average TS runtime: **4213.5 ms**; median: **219 ms**
- Total TS timeouts: **9**
- Average DSATUR runtime: **7.67 ms** (much faster but lower-quality)
- TS performs best when graph density < 0.15 (from runtime vs. density plot)