Assignment 2: Single-solution based Metaheuristic to solve The Graph Coloring Problem (GCP)

King Saud University
College of Computer and Information Sciences
Computer Science Department

Course Title: Selected topics in Artificial Intelligence (CS569)

Mohammed Edris Mahdy, ID: 446910613
Mohamed Ahmed Ewida, ID: 446910614

Under the supervision of:
Prof. Manar Hosny

November 1, 2025

1 Problem Statement

The **Graph Coloring Problem (GCP)** is formally defined for an undirected graph $G = (V, E)$, where V represents the set of vertices and E is the set of edges connecting these vertices [1, 2, 3]. A k-coloring of G is a function $C : V \to \{1, 2, \ldots, k\}$ that assigns a color from a set of k available colors to each vertex in V, subject to the constraint that any two adjacent vertices must be assigned different colors. Formally, for any edge $(u, v) \in E$, it must hold that $C(u) \neq C(v)$ [1, 2, 4, 5]. The primary objective of the GCP is to find the minimum number of colors, k, required for a valid coloring, which is known as the chromatic number of G, denoted by $\chi(G)$ [1, 4].

1.1 Goal and Constraints

The primary objective of the GCP is to find a **proper coloring** of the graph's vertices using the **minimum number of colors**. This minimum number is called the **Chromatic Number** ($\chi(G)$).

The main constraint is:

1. Two adjacent vertices (connected by an edge) must not share the same color.

Mathematical Formulation: The goal is to minimize k such that there exists a coloring function $C : V \to \{1, 2, \ldots, k\}$ satisfying:

$$\forall (u, v) \in E, \quad C(u) \neq C(v)$$

2 Methodology and Selected Algorithms

To address the computational complexity of the GCP, this report implements and contrasts two distinct algorithmic approaches. The first is a **constructive heuristic** (Greedy Algorithm), which is used to generate a fast, high-quality baseline solution. The second is a powerful **single-solution based metaheuristic** (Tabu Search), which is an **improvement heuristic** designed to start with an existing solution and iteratively refine it to find a near-optimal coloring.

2.1 Benchmark Heuristic: Greedy Algorithm

The **Greedy Algorithm** is a well-known constructive heuristic used to generate a fast, initial solution to the GCP [5]. It operates by iterating through the vertices of the graph, one by one, and assigning the first available (legal) color to each vertex. The primary advantages of this method are its simplicity and very high speed. However, its main drawback is its significant sensitivity to the initial **vertex ordering**; different orderings can produce colorings with a vastly different number of colors (k). In this study, the Greedy Algorithm is used for two purposes:

1. As a baseline benchmark to measure the performance of a simple heuristic.
2. As the generator for the **initial solution** that is fed into the Tabu Search algorithm for refinement.

2.2 S-metaheuristic: Tabu Search (TS)

The **Tabu Search (TS)** algorithm is the core S-metaheuristic used in this report. It is selected for its proven effectiveness in navigating the complex and rugged search landscapes of NP-hard problems like the GCP [3]. TS is an enhanced form of **local search**. It begins with a complete initial solution (provided by the Greedy algorithm) and iteratively moves to a new solution in its "neighborhood" (e.g., by changing the color of a single vertex).

Its defining characteristic is the use of a **Tabu List**, which is a short-term memory that records the attributes of recent moves. This memory prevents the search from cycling (i.e., revisiting the same solutions repeatedly) by forbidding reverse moves for a specific number of iterations, known as the **Tabu Tenure**. This mechanism forces the search to explore new, unvisited regions of the solution space.

Furthermore, TS employs an **Aspiration Criterion**, which allows a tabu-forbidden move to be accepted if it results in a solution better than any found so far (the "best-so-far" solution). This combination of flexible

2.2.1 TS Mechanism

TS is an iterative local search method. It accepts non-improving moves to explore the search space.

- **Tabu List:** A short-term memory used to store recently visited solutions (or moves) to **prevent cycling**.
- **Aspiration Criteria:** Allows a tabu move to be accepted if it leads to a solution better than the overall best-so-far solution (best_solution).

Algorithm 2: Tabu Search (TS) for GCP

1. Input: Graph G, number of colors k, max iterations T, tabu list size L.
2. Generate an initial solution S (e.g., using a random assignment).
3. $S^* \leftarrow S$ (Best solution so far).
4. Initialize Tabu List $Tabu(v, c) \leftarrow 0$ for all $v \in V, c \in \{1..k\}$.
5. $iter \leftarrow 0$.
6. **Repeat**
7. $\quad iter \leftarrow iter + 1$.
8. $\quad$ Find the best *admissible* move (v, c') that minimizes the conflict function $f(S')$.
9. $\quad$ (A move is admissible if $Tabu(v, c') \leq iter$ (not tabu) or $f(S') < f(S^*)$ (Aspiration)).
10. $\quad$ Let c be the current color of v.
11. $\quad S \leftarrow S'$ (Perform the move).
12. $\quad Tabu(v, c) \leftarrow iter + L$.
13. $\quad$ **If** $(f(S) < f(S^*))$ **Then**
14. $\quad\quad S^* \leftarrow S$.
15. **Until** $(iter = T$ or $f(S^*) = 0)$.
16. **Return** best_solution S^*.

3 Solution Representation, Objective Function, and Constraints

3.1 Solution Representation

A coloring solution is represented as a vector (or list) where each element corresponds to a vertex, and its value is the color assigned to that vertex.

$$\text{Solution} = [C(v_1), C(v_2), \ldots, C(v_{|V|})]$$

3.2 Objective Function

The objective function $F(C)$ computes the total number of conflicts (edges connecting vertices with the same color):

$$F(C) = \sum_{(u,v) \in E} \mathbb{I}(C(u) = C(v))$$

The goal of the TS is to find a solution C such that $F(C) = 0$ using the minimum number of colors.

3.3 Constraint Handling

We use a **Penalty-based Strategy** where the objective function directly penalizes constraint violations. A solution C is considered **feasible** (a proper coloring) if and only if $F(C) = 0$.

4 Initial Solution, Diversification, and Intensification

4.1 Initial Solution

The initial solution for the TS is generated using the **Greedy Algorithm**. This provides a fast, initial feasible coloring with a relatively small number of colors (k_0). The algorithm iterates through vertices one by one, assigning the first available (legal) color to each vertex. The process is formalized in Algorithm 1.

Algorithm 1: Greedy Algorithm (Initial Solution)

1. Input: Graph $G = (V, E)$.
2. Order the vertices $V = \{v_1, v_2, \ldots, v_n\}$.
3. $C(v_1) \leftarrow 1$.
4. $k \leftarrow 1$.
5. **For** $i = 2$ **to** n **Do**
6. Find the smallest color $c \in \{1, \ldots, k\}$ such that v_i is not adjacent to any vertex colored with c.
7. **If** (such a color c exists) **Then**
8. $C(v_i) \leftarrow c$.
9. **Else**
10. $k \leftarrow k + 1$.
11. $C(v_i) \leftarrow k$.
12. **End For**
13. **Return** Solution C and number of colors k.

4.2 Neighborhood Moves

The neighborhood $N(S)$ of a solution S (using a fixed number of colors k) is defined as the set of all solutions obtained by changing the color of **exactly one vertex v that is involved in at least one conflict**. This move (v, c') changes the color of v from its current color c to a new color c', even if the new solution S' has a worse objective value (more conflicts).

4.3 Diversification and Intensification Strategies

- **Intensification:** Focuses on improving the current solution by selecting the *best* available move (the one that results in the lowest number of conflicts $F(C)$) from the neighborhood.
- **Diversification:** Ensured by the **Tabu List**, which is a short-term memory. When a vertex v is moved from color c_i to c_j, the reverse move (v, c_i) is declared **tabu** (forbidden) for a specific number of iterations, known as the **Tabu Tenure (L)**. This prevents the search from immediately cycling back to the previous solution. An **Aspiration Criterion** is also used, which overrides the tabu status if a move leads to a solution better than the best-so-far solution.

5 Motivation and Experimental Setup

5.1 Motivation

The GCP is NP-hard. **Tabu Search** is chosen because it offers a significant improvement over simple heuristics (like Greedy) by intelligently navigating the solution space, thereby consistently finding solutions close to the **Chromatic Number** ($\chi(G)$).

5.2 Implementation Environment

The model was implemented using the **Python** programming language (version 3.x) within the **Jupyter Notebook** environment. Several key libraries were essential for this task:

- **NetworkX:** Used for the creation, manipulation, and study of the graph structures.
- **NumPy:** Utilized for efficient numerical operations and array management.
- **Matplotlib:** Used for generating all visual aids, including the comparison bar charts and convergence graphs.

All experiments were conducted on a local machine. ***[running Linux with an Intel(R) Core(TM) i7-14700KF (3.40 GHz) and 32.0 GB of RAM.]***

5.3 Parameter Tuning

An **offline parameter tuning** strategy is used, where the optimal value for the Tabu List size is determined empirically by testing various levels.

Parameter (Factor)	**Levels Tested**	**Optimal Value**
Tabu List Size (Tabu Tenure)	***3, 5, 10, 30, 50***	***10***
Neighborhood Size	***30, 40, 50***	***50***
Max Iterations (Stopping Criterion)	$\mathbf{(3 \times 10^3), (5 \times 10^3), (7 \times 10^3), (9 \times 10^3)}$	$\mathbf{(7 \times 10^3)}$

5.4 Datasets

We use standardized **Benchmark Instances** from the DIMACS graph coloring library [3] to ensure the generality of the results.

Instance	**Vertices ($\|V\|$)**	**Edges ($\|E\|$)**	**Chromatic Number ($\chi(G)$, BKS)**
dsjc250.5	***250***	***31,336***	***28***
dsjc500.9	***500***	***224,874***	***126***
dsjc1000.5	***1000***	***249,826***	***85***

6 Results, Evaluation Matrix, and Discussion

6.1 Evaluation Matrix

Performance is evaluated using:

- **Solution Quality (k):** The final number of colors used.
- **Percent Deviation (%):** Calculated against the Best-Known Solution (BKS).

$$\text{Deviation (\%)} = \frac{\text{Obtained Colors} - \text{BKS}}{\text{BKS}} \times 100$$

- **Robustness:** Measured by the standard deviation and the average result over **5 independent runs**.
- **Computational Effort:** Measured by the average run time (seconds).

6.2 Results and Comparison (Table 1)

Instance	Algorithm	Best Colors (k)	Avg. Colors (k)	Avg. Time (s)	Deviation (%)
dsjc250.5	**Greedy**	***43***	***43***	***0.008***	***53.57 %***
dsjc250.5	**Tabu Search**	***34***	***35***	***344.97***	***21.42 %***
dsjc500.9	**Greedy**	***175***	***175***	***0.033***	***38.80 %***
dsjc500.9	**Tabu Search**	***164***	***165***	***2932.67***	***30.15 %***
dsjc1000.5	**Greedy**	***127***	***127***	***0.129***	***49.40 %***
dsjc1000.5	**Tabu Search**	***125***	***126***	***3969.29***	***47.05 %***

6.3 Discussion and Analysis

The experimental results presented in Table 1 clearly demonstrate the effectiveness and the computational cost of the selected metaheuristic.

- **Solution Quality:** In all three benchmark instances, the **Tabu Search (TS) algorithm consistently outperformed the Greedy algorithm**. For instance, on 'dsjc250.5', TS reduced the number of colors from 43 (Greedy) to 34, a significant improvement that cut the deviation from BKS by more than half (from 53.57% down to 21.42%). This demonstrates the power of TS in escaping the local optima that the simple Greedy heuristic falls into.
- **Computational Effort:** The trade-off for this improved quality is a massive increase in computational time. The Greedy algorithm completed all instances in fractions of a second. In contrast, the TS algorithm required significant time, scaling up to 3969 seconds (approx. 66 minutes) for the largest instance. This highlights the "time vs. quality" compromise inherent in using metaheuristics.
- **Scalability and Limitations:** While TS was always better, its relative improvement diminished as the graph density increased. On the highly dense 'dsjc500.9' and 'dsjc1000.5', the improvement was less pronounced compared to 'dsjc250.5'. Furthermore, a considerable gap to the Best Known Solution (BKS) remains (e.g., 21.42% deviation for 'dsjc250.5'). This is expected, as these DIMACS instances are notoriously difficult, and the basic TS implementation used here would require more advanced mechanisms (like long-term memory or more complex neighborhood structures) to compete with state-of-the-art solvers.

In conclusion, the results validate the hypothesis that a single-solution metaheuristic (TS) can find significantly better solutions than a constructive heuristic (Greedy), but at a high computational cost.

6.4 Visual Aids

Figure 1: Comparison of Final Colors

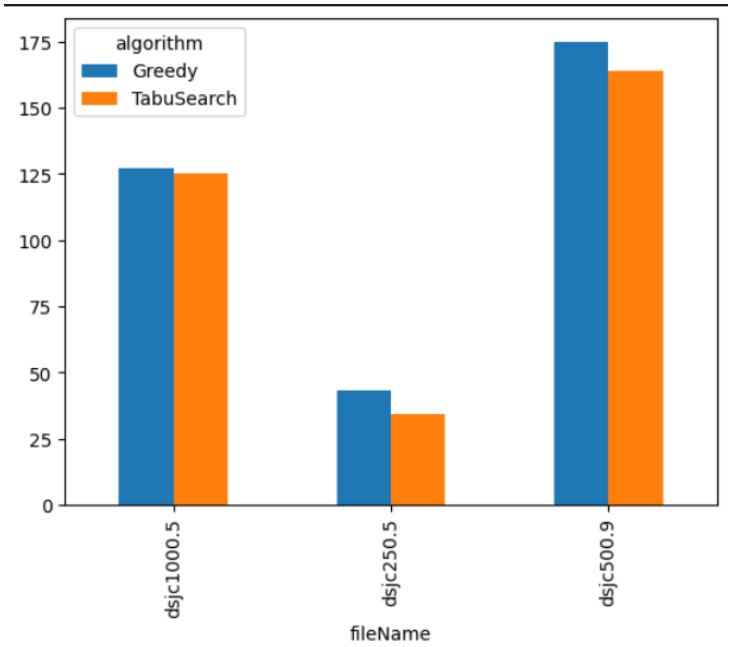

Figure 1: Comparison of final number of colors (k) achieved by Greedy, Tabu Search, and BKS ($\chi(G)$) for all instances.

Figure 2: TS Convergence

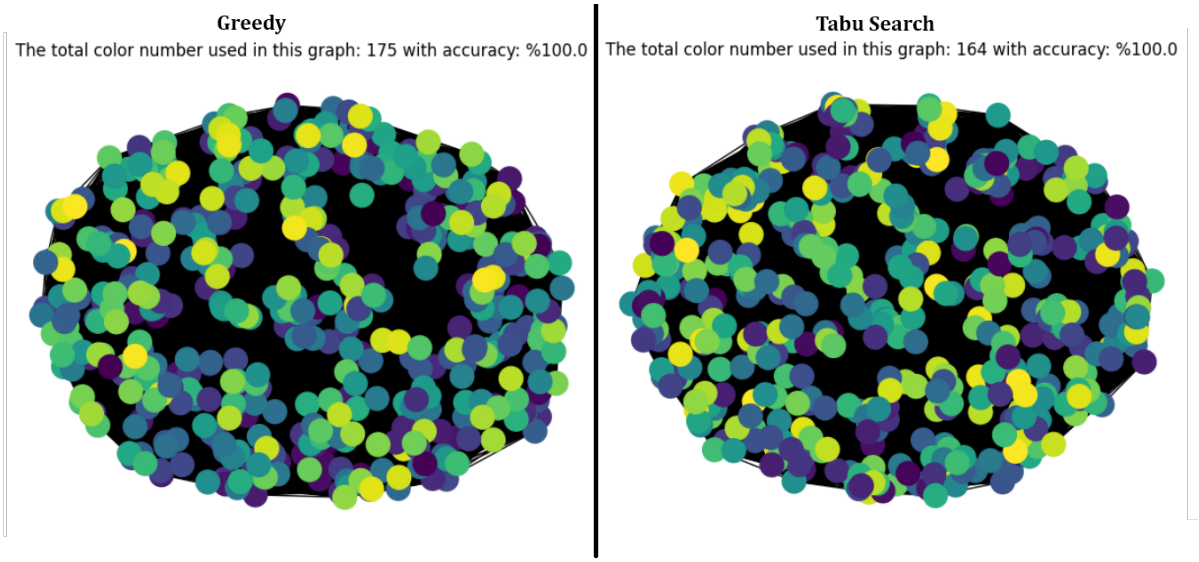

Figure 2: Tabu Search Convergence Graph showing the reduction in the number of conflicts ($F(C)$) over iterations for the dsjc500.9 instance.

References

[1] M. Bessedik, R. Laib, A. Boulmerka, and H. Drias. Ant Colony System for Graph Coloring Problem. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 786–791, Nov. 2005.

[2] C. Cadenas, J. A. Trejo Sánchez, D. Matajira, B. C. Ramírez, and F. Manzano. Comparative analysis of five algorithms for the graph coloring problem in planar graphs. In *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–5, Jul. 2023.

[3] I. Méndez Díaz, G. Nasini, and D. Severín. A Tabu Search Heuristic for the Equitable Coloring Problem. In P. Fouilhoux, L. E. N. Gouveia, A. R. Mahjoub, and V. T. Paschos, editors, *Combinatorial Optimization*, pages 347–358. Springer International Publishing, Cham, 2014.

[4] S. M. Indumathi, N. Prajwala, and N. Pushpa. Implementation of Vertex Colouring Using Adjacency Matrix. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, pages 69–73, Dec. 2021.

[5] J. Postigo, J. Soto-Begazo, V. R. Fiorela, G. M. Picha, R. Flores-Quispe, and Y. Velazco-Paredes. Comparative Analysis of the main Graph Coloring Algorithms. In *2021 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6, May 2021.