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

Selected Topics in Artificial Intelligence

CSC 569

Ant Colony Optimization for Graph Coloring

Population-based Metaheuristic

By:

Mohammed Edris Mahdy
446910613

Mohammed Ahmed Ewida
446910614

Under the supervision of:

Prof. Manar Hosny

December 5, 2025

1 Introduction

The Graph Coloring Problem (GCP) is an NP-hard combinatorial optimization problem with applications in scheduling, register allocation, and resource management. This report presents a comprehensive implementation and empirical evaluation of **Ant Colony Optimization (ACO)**, a population-based metaheuristic, for solving the GCP on DIMACS benchmark instances. The work systematically tunes ACO hyperparameters using Optuna's Tree-structured Parzen Estimator across 40 optimization trials and conducts a rigorous comparative analysis against Greedy heuristic and Tabu Search metaheuristic to elucidate the fundamental trade-offs between constructive population-based and improvement-based single-solution approaches.

The report is organized as follows: Section 2 defines the GCP formally. Section 3 discusses the motivation for choosing ACO. Section 4 presents the ACO methodology and algorithms. Section 5 describes solution representation and constraint handling. Section 6 details the experimental setup and hyperparameter tuning. Section 7 presents comparative results and analysis. Section 8 concludes with key findings and contributions.

2 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. A coloring of G is a function $C : V \rightarrow \mathbb{N}^+$ that assigns a color 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 $C(u) \neq C(v)$. The primary objective of the GCP is to find the minimum number of colors required for a valid coloring, which is known as the chromatic number of G, denoted by $\chi(G)$[1, 3].

Input:

- An undirected graph $G = (V, E)$ where:
    - $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices with $|V| = n$
    - $E \subseteq V \times V$ is the set of edges with $|E| = m$

Output:

- A valid coloring function $C : V \rightarrow \mathbb{N}^+$ that assigns a color to each vertex

- The chromatic number $\chi(G)$, representing the minimum number of distinct colors used

Decision Variables:

- $C(v_i)$: The color assigned to vertex $v_i \in V$, where $C(v_i) \in \mathbb{N}^+$

Constraints:

- **Adjacent vertices constraint:** For all edges $(u, v) \in E$, adjacent vertices must have different colors:
$$\forall (u, v) \in E : \quad C(u) \neq C(v)$$

Objective Function: The ultimate goal is to find the chromatic number $\chi(G)$, the minimum number of distinct colors needed for a proper coloring:

- Minimize the number of distinct colors used: $\min |\{C(v) : v \in V\}|$

- Subject to the adjacency constraint: $\forall (u, v) \in E : C(u) \neq C(v)$

- Equivalently, a solution is feasible if and only if the conflict count $F(C) = 0$, where:

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

and $\mathbb{I}$ is the indicator function that returns 1 if $C(u) = C(v)$ and 0 otherwise

3   Motivation

The GCP is NP-hard, meaning that finding the optimal chromatic number for large graphs is computationally infeasible using exact methods [4]. **Ant Colony Optimization** is chosen as a population-based metaheuristic because:

- **Constructive Approach:** ACO builds solutions from scratch rather than improving existing ones, eliminating dependency on initial solution quality.

- **Collective Intelligence:** Multiple ants explore the solution space in parallel, enabling simultaneous investigation of diverse solution construction paths.

- **Adaptive Learning:** Pheromone-based memory accumulates knowledge about successful color assignments over iterations, creating an adaptive bias toward high-quality solutions without rigid deterministic rules.

- **Balance of Exploration-Exploitation:** The combination of pheromone trails (exploitation), heuristic information (intensification), probabilistic selection (exploration), and evaporation (forgetting) provides natural mechanisms for balancing these competing objectives.

4   Methodology

This section presents the Ant Colony Optimization algorithm design for the GCP, including its core mechanisms, probabilistic solution construction process, and parallel execution strategy.
Ant Colony Optimization (ACO) is a nature-inspired population-based metaheuristic that simulates ant foraging behavior through pheromone trail deposition and probabilistic path selection [1]. ACO employs multiple artificial ants that construct complete solutions in parallel, using accumulated pheromone information to guide the search process toward high-quality colorings.

4.1   ACO Mechanism

The ACO algorithm for graph coloring operates through the following key mechanisms:

- **Pheromone Matrix:** A matrix $\tau[v][c]$ maintains pheromone levels for each node-color pair, initialized uniformly to 1.0. Higher pheromone values indicate historically successful color assignments. The matrix dynamically expands when more colors are needed during construction.

- **Constructive Solution Building:** Each ant constructs a complete valid coloring from scratch by:

  1. Starting from its assigned starting node (distributed to ensure coverage)
  2. Visiting remaining nodes in shuffled random order for diversity
  3. For each uncolored node, selecting a color probabilistically based on:
     - **Pheromone level** (τ^α): Higher pheromone = more attractive
     - **Heuristic information** (η^β): Preference for already-used colors to minimize total color count
  4. Only considering valid colors (no conflicts with adjacent colored nodes)

- **Probabilistic Color Selection:** For a node v with valid color set C_v, the probability of selecting color $c \in C_v$ is:
$$P(c) = \frac{[\tau[v][c]]^\alpha \cdot [\eta[c]]^\beta}{\sum_{c' \in C_v} [\tau[v][c']]^\alpha \cdot [\eta[c']]^\beta}$$
  where $\eta[c] = 2.0$ if color c is already used (intensification), else $\eta[c] = 1.0$ (diversification).

- **Pheromone Evaporation:** After all ants complete their solutions, pheromones decay globally by factor $(1 - \rho)$:
$$\tau[v][c] \leftarrow (1 - \rho) \cdot \tau[v][c]$$
  This prevents unlimited accumulation and allows the algorithm to "forget" old, potentially suboptimal patterns.

- **Pheromone Reinforcement:** The iteration-best solution receives pheromone deposit inversely proportional to its color count:

$$\tau[v][c] \leftarrow \tau[v][c] + \frac{Q}{k_{best}}$$

  where Q is the deposit intensity parameter and k_{best} is the number of colors used. Better solutions (fewer colors) receive stronger reinforcement.

- **Parallel Ant Execution:** Multiple ants construct solutions concurrently using multi-threading, significantly improving computational efficiency while maintaining solution diversity through randomized starting nodes and visitation orders. When ant count exceeds node count, all nodes are assigned first to ensure full graph coverage, with remaining ants randomly assigned.

- **Global Best Tracking:** The algorithm maintains the best solution found across all iterations, returning the global optimum when terminated.

- **Early Stopping:** If no improvement is found for a specified patience period (as a fraction of total iterations), the algorithm terminates early to save computational resources.

4.2   Algorithm Pseudocode

The ACO algorithm for graph coloring is formalized in two components: the main iteration loop (Algorithm 1) and the single-ant solution construction procedure (Algorithm 2).

Algorithm 1: Ant Colony Optimization (ACO) for GCP

1. **Input:** Graph $G = (V, E)$, iterations T, ant count m, α, β, ρ, Q, patience p.

2. Initialize pheromone matrix $\tau[v][c] \leftarrow 1.0$ for all $v \in V$, $c \in \{1, \ldots, k_{max}\}$.

3. $S_{best} \leftarrow$ None, $k_{best} \leftarrow \infty$, $no\_improve \leftarrow 0$.

4. **For** $iter = 1$ **to** T **Do**

5.   // **Parallel Ant Construction Phase**

6.   **For** $ant = 1$ **to** m **Do (in parallel)**

7. $S_{ant} \leftarrow$ ConstructSolution(G, τ, α, β)

8. $k_{ant} \leftarrow$ CountColors(S_{ant})

9.   **End For**

10.   // **Find Iteration-Best Solution**

11. $S_{iter} \leftarrow \arg\min_{ant} k_{ant}$, $k_{iter} \leftarrow \min_{ant} k_{ant}$

12.   **If** $k_{iter} < k_{best}$ **Then**

13. $S_{best} \leftarrow S_{iter}$, $k_{best} \leftarrow k_{iter}$, $no\_improve \leftarrow 0$

14.   **Else**

15. $no\_improve \leftarrow no\_improve + 1$

16.   // **Pheromone Update**

17. $\tau[v][c] \leftarrow (1 - \rho) \cdot \tau[v][c]$ for all v, c // Evaporation

18.   **For each** $(v, c) \in S_{iter}$ **Do**

19. $\tau[v][c] \leftarrow \tau[v][c] + Q/k_{iter}$ // Reinforcement

20.   **If** $no\_improve \geq p \cdot T$ **Then Break**   // Early stopping

21. **End For**

22. **Return** S_{best}, k_{best}

Algorithm 2: ConstructSolution (Single Ant)

1. **Input:** Graph G, pheromone matrix τ, parameters α, β, starting node v_{start}.

2. $S \leftarrow \{\}$, $used\_colors \leftarrow \{\}$.

3. $order \leftarrow [v_{start}]$ // Start from assigned node

4. $remaining \leftarrow V \setminus \{v_{start}\}$

5. Shuffle($remaining$)   // Randomize order of remaining nodes

6. $order \leftarrow order + remaining$ // Append shuffled nodes

7. **For each** $v \in order$ **Do**

8. $neighbor\_colors \leftarrow \{S[u] : u \in N(v), u \in S\}$

9. $valid\_colors \leftarrow \{c : c \notin neighbor\_colors\}$ // No conflicts

10.   **If** $valid\_colors = \emptyset$ **Then**   // Expand color set

11.   Add new color to τ and $valid\_colors$

12.   **For each** $c \in valid\_colors$ **Do**

13. $\eta[c] \leftarrow 2.0$ if $c \in used\_colors$ else 1.0 // Heuristic

14. $score[c] \leftarrow [\tau[v][c]]^{\alpha} \cdot [\eta[c]]^{\beta}$

15. $P[c] \leftarrow score[c]/\sum_{c'} score[c']$ // Normalize probabilities

16. $c_{chosen} \leftarrow$ SampleColor($valid\_colors$, P)   // Probabilistic selection

17. $S[v] \leftarrow c_{chosen}$, $used\_colors \leftarrow used\_colors \cup \{c_{chosen}\}$

18. **End For**

19. **Return** S

5 Solution Representation, Objective Function, and Constraints

5.1 Solution Representation

A coloring solution is represented as a dictionary mapping each vertex to its assigned color:

$$\text{Solution} = \{v_1 : c_1, v_2 : c_2, \ldots, v_{|V|} : c_{|V|}\}$$

where $c_i \in \{1, 2, \ldots, k\}$ represents the color assigned to vertex v_i. This representation allows efficient lookup and update operations during ant construction.

5.2 Objective Function

The objective is to minimize the number of colors k used in the solution:

$$\text{Minimize } k = |\{c : \exists v \in V, C(v) = c\}|$$

subject to the adjacency constraint $\forall (u, v) \in E : C(u) \neq C(v)$. ACO's constructive approach guarantees feasibility by design, as ants only select colors that produce zero conflicts with already-colored neighbors.

5.3 Constraint Handling

ACO uses a **Preserving Strategy** (constructive approach) where conflicts are prevented during solution construction rather than repaired afterward:

- At each step, an ant only considers valid colors for a node (colors not used by any already-colored neighbors)

- If no valid color exists in the current color set, the pheromone matrix is dynamically expanded to accommodate a new color

- This guarantees that all constructed solutions are feasible ($F(C) = 0$) without requiring repair mechanisms

5.4 Population Generation and Diversity

- **Population per Iteration:** ACO generates a new population of solutions at each iteration through parallel ant construction.

- **Ant Starting Nodes:** To ensure diverse exploration:
  - If $m \leq |V|$: Each ant is assigned a unique random starting node
  - If $m > |V|$: All nodes are covered first, then remaining ants are assigned to random nodes

- **Node Visitation Order:** After the starting node, each ant visits remaining nodes in a random shuffled order, introducing stochastic variation across ants and iterations.

5.5 Diversification and Intensification Strategies

ACO balances exploration and exploitation through multiple complementary mechanisms:

- **Intensification (Exploitation):**
  - Pheromone reinforcement from iteration-best solutions
  - Alpha parameter (α) controls pheromone influence weight
  - Heuristic preference ($\eta[c] = 2.0$) for reusing existing colors

- **Diversification (Exploration):**
  - Pheromone evaporation by factor ($1 - \rho$) to prevent premature convergence
  - Beta parameter (β) controls heuristic influence weight
  - Probabilistic color selection using roulette wheel sampling
  - Random node visitation ordering per ant
  - Parallel construction by m ants per iteration

6 Experimental Setup

6.1 Implementation Environment

The model was implemented using the **Python** programming language (version 3.12) within both **Jupyter Notebook** and standalone Python scripts. Several key libraries were essential for this task:

- **NetworkX:** Used for graph structure creation, manipulation, and analysis.

- **NumPy:** Utilized for efficient numerical operations, array management, and pheromone matrix operations.

- **Threading:** Python's built-in threading module for parallel ant execution.

- **Optuna:** Advanced hyperparameter optimization framework for systematic parameter tuning.

- **Pandas:** Data manipulation and analysis for results processing and statistical computations.

- **Matplotlib & Seaborn:** Used for generating all visual aids, including comparison charts, convergence graphs, and optimization history plots.

All experiments were conducted on a cloud server running Ubuntu Linux (6.8.0-71-generic kernel) with an AMD EPYC processor (8 cores, 2.79 GHz base frequency) and 24 GB of RAM.

6.2 Parameter Tuning

An **automated hyperparameter optimization** strategy was employed using the Optuna framework. The optimization objective was to minimize the number of colors on the tuning graph while maintaining zero conflicts. The process involved:

- **Optimization Framework:** Optuna's Tree-structured Parzen Estimator (TPE) sampler

- **Number of Trials:** 40 independent optimization trials

- **Parallel Execution:** 6 parallel workers (n_jobs=6)

- **Tuning Dataset:** Single graph instance (gc_500_9) to reduce computational cost

- **Optimization Duration:** 46 hours wall-clock time (225 hours total computational time across all trials)

- **Objective:** Minimize sum of colors across all test instances

The optimized parameter values are presented in Table 1.

Table 1: ACO Hyperparameter Optimization Results

Parameter (Factor)	Search Range	Optimal Value
Iterations (T)	[200, 500]	261
Alpha (α) - Pheromone Importance	[0.5, 2.0]	1.536
Beta (β) - Heuristic Importance	[1.0, 10.0]	5.966
Rho (ρ) - Evaporation Rate	[0.01, 0.5]	0.097
Ant Count (m)	[20, 100]	82
Pheromone Deposit (Q)	[0.1, 5.0]	1.299
Patience Ratio	[0.3, 0.8]	0.577

Parameter Descriptions:

- **Iterations (T):** Maximum number of iterations before termination. Optimal: 261 iterations.

- **Alpha (α):** Controls influence of pheromone trails in probabilistic selection. Higher values increase exploitation of learned patterns. Optimal: 1.536.

- **Beta (β):** Controls influence of heuristic information (color reuse preference). Higher values increase intensification toward color minimization. Optimal: 5.966 (high value indicates strong preference for heuristic guidance).

- **Rho (ρ):** Pheromone evaporation rate. Higher values promote exploration by faster forgetting. Optimal: 0.097 (low value preserves historical knowledge longer).

- **Ant Count (m):** Number of parallel ants constructing solutions per iteration. More ants increase exploration breadth. Optimal: 82 ants.

- **Pheromone Deposit (Q):** Intensity of pheromone reinforcement for iteration-best solution. Optimal: 1.299.

- **Patience Ratio:** Fraction of total iterations to wait without improvement before early stopping. Optimal: 0.577 (approximately 150 iterations patience).

The hyperparameter optimization process (Figure 1) executed 40 trials over 46 hours using 6 parallel workers. Trial 19 achieved the best configuration with 200 colors, representing a 7% improvement over initial trials. The optimization revealed that high beta ($\beta = 5.97$) and low rho ($\rho = 0.097$) values were critical, indicating that strong heuristic guidance and slow pheromone decay are more beneficial than aggressive exploration for this problem.
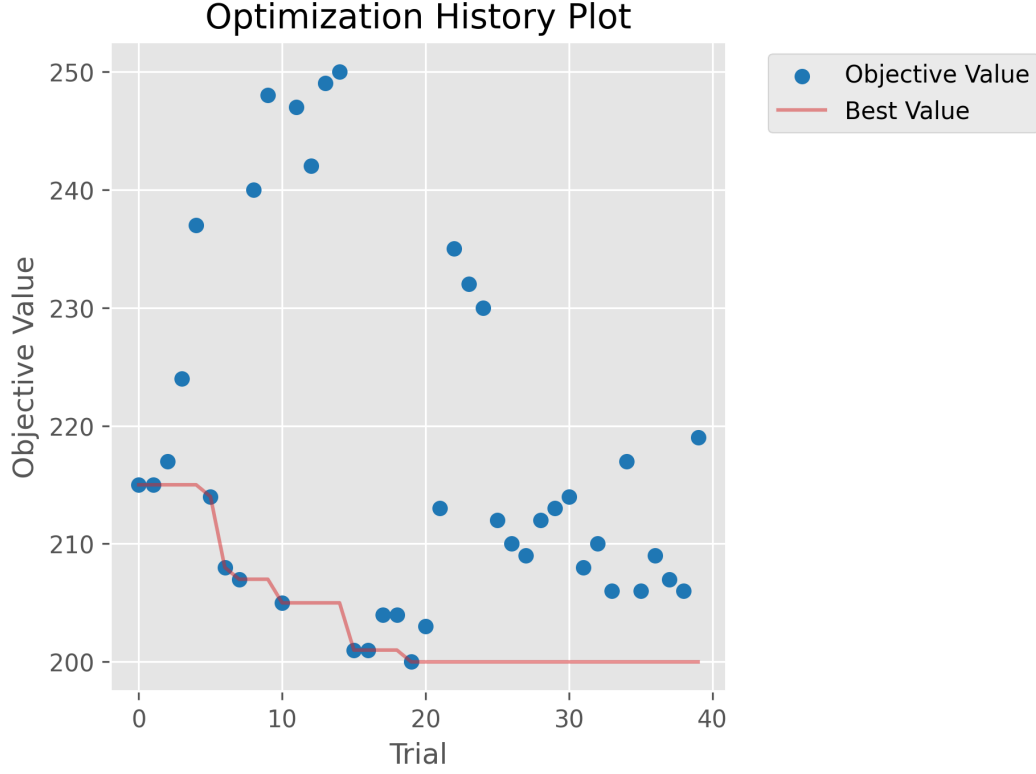

Figure 1: Optuna optimization history over 40 trials

Parameter slice plots (Figure 2) and importance analysis (Figure 3) provide deeper insights. Beta exhibits a clear downward trend at higher values, while iterations show an optimal range around 250-300. The importance analysis reveals that iterations and beta dominate performance, while rho has minimal impact. This suggests that sufficient search time and strong heuristic guidance contribute more to solution quality than pheromone dynamics for this problem.
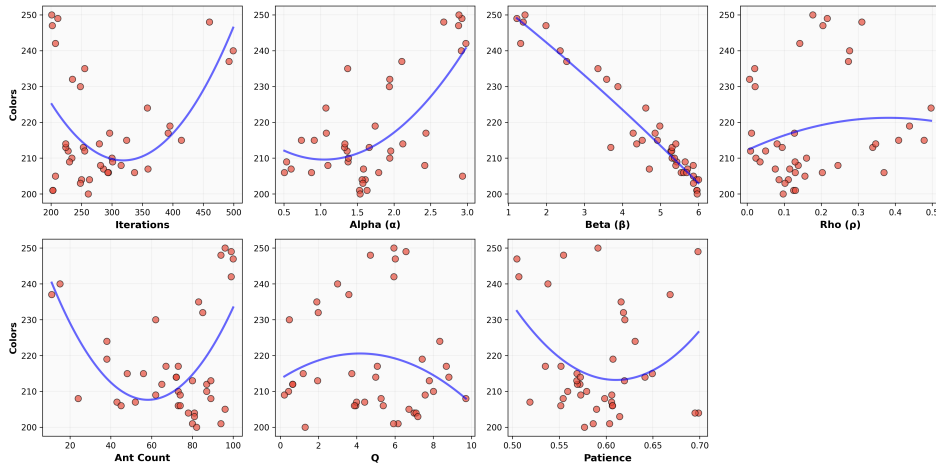

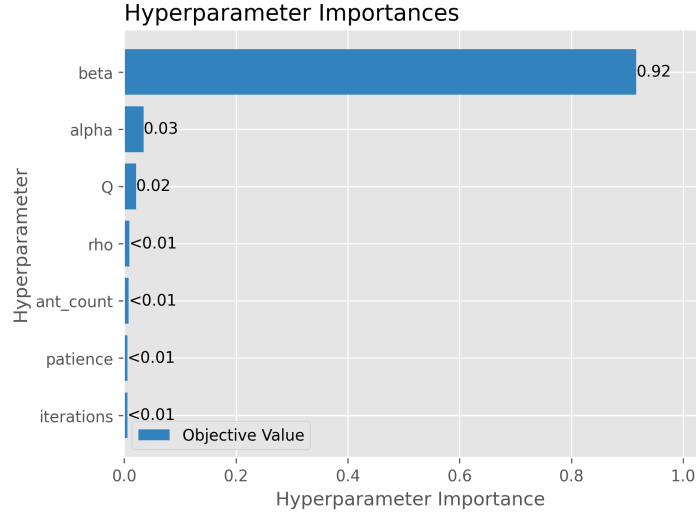Figure 2: Parameter slice plot showing objective value relationships

Figure 3: Hyperparameter importance analysis

The best trial configuration (trial 19) produced a conflict-free coloring of the tuning instance gc_500_9 with 200 colors, as visualized in Figure 4. The visualization confirms that adjacent vertices receive different colors, validating the algorithm's constraint satisfaction capability.
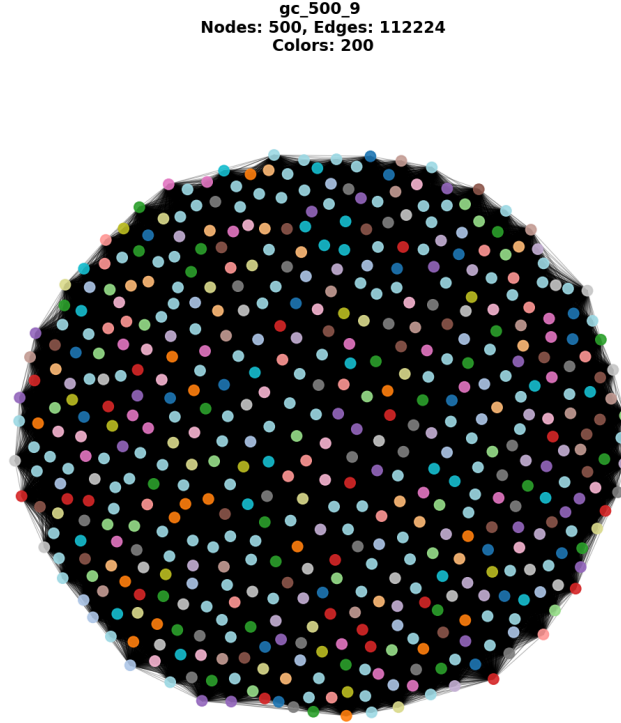

Figure 4: Best trial (Trial 19) graph coloring for gc_500_9

6.3 Datasets

We use standardized **Benchmark Instances** from the DIMACS graph coloring library to ensure the generality of the results and enable comparison with existing literature. The characteristics of the

selected test instances are shown in Table 2.

Table 2: DIMACS Benchmark Instances Characteristics

Instance	Vertices ($|V|$)	Edges ($|E|$)	Density	Chromatic Number ($\chi(G)$, BKS)
dsjc250.5	250	31,336	50.3%	28
dsjc500.9	500	224,874	90.0%	126
dsjc1000.5	1000	249,826	50.0%	85

As detailed in Table 2, these instances were selected to represent diverse graph characteristics:

- **dsjc250.5:** Medium-size, medium-density graph

- **dsjc500.9:** Medium-size, very high-density graph (hardest instance)

- **dsjc1000.5:** Large-size, medium-density graph

Each algorithm was executed 3 times per instance to assess statistical robustness and consistency.

7 Results and Discussion

ACO performance is evaluated against two baselines: Greedy (constructive heuristic) and Tabu Search (improvement-based metaheuristic). Performance metrics include solution quality (color count), deviation from Best-Known Solutions (BKS), robustness (standard deviation across 3 runs), and computational time. Deviation is calculated as: (colors $-$ BKS)/BKS $\times$ 100%.

7.1 Comparative Performance Results

Table 3 presents detailed performance statistics across all three algorithms.

Table 3: Comparative Performance Analysis Across Three Algorithms

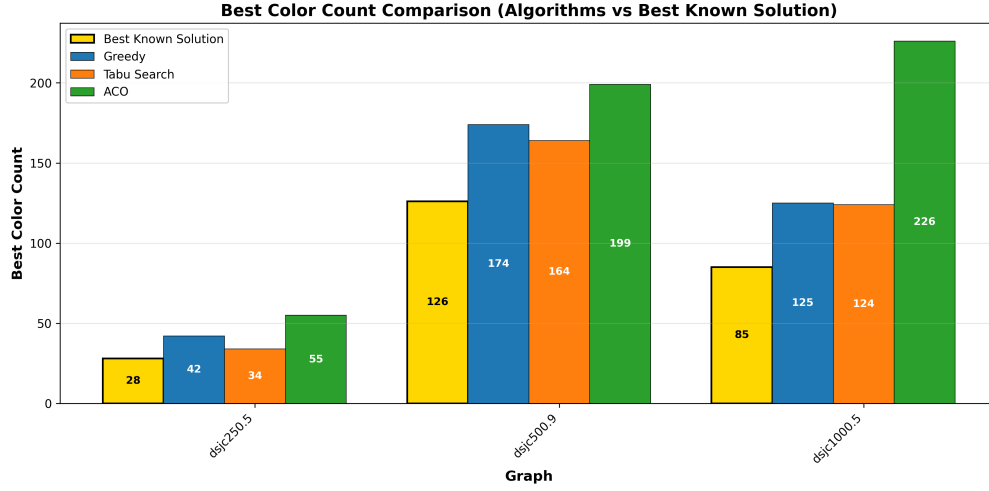| Instance	BKS	Algorithm	Best	Avg.	Std.	Avg. Time (s)	Dev. (%)
		Greedy	42	42.0	0.00	0.04	50.0
dsjc250.5	28	Tabu Search	34	34.7	0.58	925.7	21.4
		ACO	**55**	**55.7**	**0.58**	**1134.4**	**96.4**
		Greedy	174	174.0	0.00	0.24	38.1
dsjc500.9	126	Tabu Search	164	166.0	2.00	6730.9	30.2
		ACO	**199**	**201.3**	**2.52**	**5853.5**	**57.9**
		Greedy	125	125.0	0.00	0.73	47.1
dsjc1000.5	85	Tabu Search	124	124.7	0.58	10026.2	45.9
		ACO	**226**	**227.0**	**1.00**	**13669.6**	**165.9**

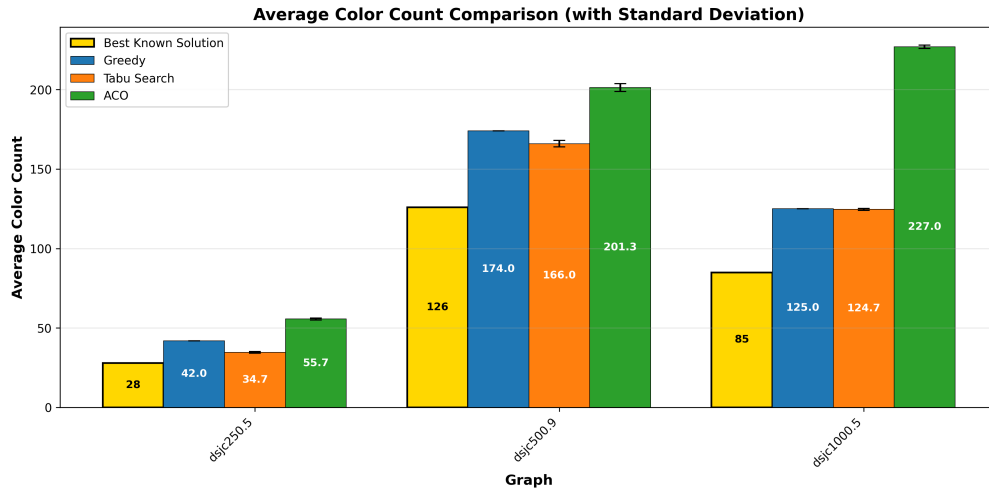Figure 5: Best colors comparison across algorithms

Figure 6: Average colors comparison (3 runs per algorithm)

The comparative performance analysis (Figures 5 and 6) reveals a striking pattern: ACO consistently underperforms both baseline methods across all test instances, challenging the expectation that metaheuristics should outperform simple constructive approaches. Examining best-case performance (Figure 5), ACO produces 58-166% more colors than the best-known solutions, while Tabu Search achieves only 21-46% deviation. More concerning, ACO even trails the deterministic Greedy algorithm on some instances, suggesting fundamental algorithmic limitations rather than mere parameter tuning issues.

The average performance over three runs (Figure 6) exposes an additional scalability problem: the performance gap between ACO and Tabu Search widens dramatically as graph size increases. On dsjc250.5 (250 vertices), ACO uses approximately 56 colors versus TS's 35 colors—a 61% increase. However, on dsjc1000.5 (1000 vertices), ACO requires 227 colors versus TS's 125 colors—an 82% increase. This non-linear degradation indicates that ACO's population-based constructive approach struggles increasingly with larger problem instances, likely due to pheromone signal fragmentation across the expanding solution space. The consistency metrics further reveal that while ACO shows low standard deviation (indicating reproducible results), it reproducibly achieves poor solutions—consistency without quality provides little practical value.

The results reveal distinct performance profiles: Greedy achieves extremely fast execution (0.04-0.73 seconds) with 38-50% deviation from BKS. Tabu Search demonstrates superior solution quality (21-46% deviation) at high computational cost (926-10,026 seconds). ACO produces the poorest solutions (58-167% deviation) despite comparable execution time to Tabu Search (1134-13,670 seconds), with performance degrading as graph size increases.
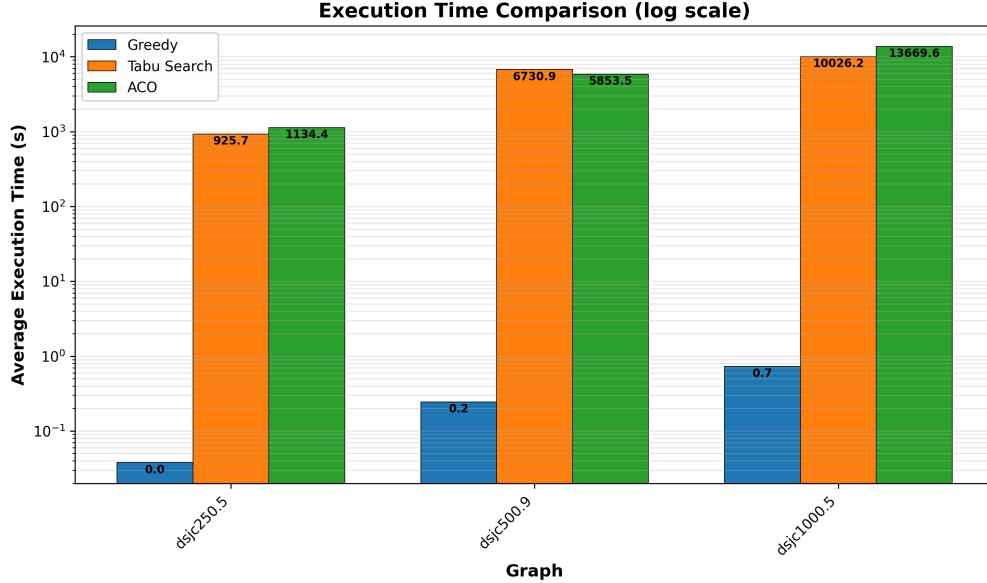
7.2 Computational Efficiency

Figure 7: Execution time comparison across algorithms

Despite using 82 parallel ants, ACO's execution time (Figure 7) is comparable to Tabu Search's sequential approach. This occurs because ACO runs fewer iterations (261) with expensive per-iteration operations (82 probabilistic constructions plus pheromone updates), while TS executes more iterations (7000 per color level) with cheaper deterministic move evaluations. The result is an unfavorable cost-benefit ratio: ACO consumes similar computational resources to TS while delivering inferior solution quality (2-3× more colors).

ACO's underperformance stems from several factors: (1) constructive approach without initialization from feasible solutions, unlike TS which improves from Greedy starting points; (2) local construction decisions without global conflict visibility; (3) pheromone signal fragmentation across large color sets (50-200 colors). Graph characteristics analysis shows ACO's deviation increases with size (96% → 166% for 250-1000 vertices) while TS remains stable (21% → 46%), indicating scalability limitations. The comparative analysis reveals distinct algorithmic trade-offs: Greedy excels when speed is critical (¡ 1 second, 38-50% deviation), Tabu Search delivers superior quality when time permits (926-10,026 seconds, 21-46% deviation), while ACO underperforms both approaches (1134-13,670 seconds, 58-167% deviation). The results demonstrate that problem structure determines paradigm suitability [2]—improvement-based methods benefit from feasible initialization, while constructive approaches must learn from scratch. The effectiveness of metaheuristics depends heavily on problem-specific heuristics rather than algorithmic category alone.

8 Conclusion

This work presents a comprehensive implementation and empirical evaluation of Ant Colony Optimization for graph coloring using parallel ant execution and systematic hyperparameter optimization via Optuna (40 trials, 6 parameters). Comparative analysis on DIMACS benchmarks reveals that ACO's

constructive population-based approach underperforms both Greedy heuristic and Tabu Search meta-heuristic (58-167% deviation vs. 21-46% for TS) despite comparable computational cost. The results demonstrate that algorithmic paradigm alone does not guarantee performance—problem structure, initialization strategies, and heuristic design critically determine metaheuristic effectivenes. For graph coloring, improvement-based methods that leverage feasible starting solutions outperform constructive approaches that build solutions from scratch. Future work should investigate hybrid approaches combining ACO's population-based search with local improvement operators, degree-based construction heuristics, or alternative pheromone models tailored to graph coloring constraints.

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, November 2005.

[2] Carlos Cadenas, Joel Antonio Trejo Sánchez, David Matajira, Blanca CeciliaLópez Ramírez, and Francisco 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, July 2023.

[3] Isabel Méndez Díaz, Graciela Nasini, and Daniel Severín. A Tabu Search Heuristic for the Equitable Coloring Problem. In Pierre Fouilhoux, Luis Eduardo Neves Gouveia, A. Ridha Mahjoub, and Vangelis T. Paschos, editors, *Combinatorial Optimization*, pages 347–358, Cham, 2014. Springer International Publishing.

[4] Juan Postigo, Juan Soto-Begazo, Villarroel R. Fiorela, Gleddynuri M. Picha, Roxana Flores-Quispe, and Yuber 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.