

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

Ant Colony Optimization for Graph Coloring Problem
Population-based Metaheuristic

Mohammed Edris Mahdy (446910613)
Mohammed Ahmed Ewida (446910614)

Under the Supervision of:
Prof. Manar Hosny

Outline

- 1 Problem Statement
- 2 Methodology: ACO P-Metaheuristic
- 3 Hyperparameter Tuning
- 4 Results and Discussion
- 5 Conclusion

Graph Coloring Problem (GCP)

Formal Definition

Given an undirected graph $G = (V, E)$, find a k -coloring function $C : V \rightarrow \{1, 2, \dots, k\}$ such that:

- $\forall (u, v) \in E : C(u) \neq C(v)$ (adjacent vertices have different colors)
- Minimize $k = \chi(G)$ (chromatic number)

Objective Function

$$\text{Minimize } k \text{ subject to } F(C) = \sum_{(u,v) \in E} \mathbb{I}(C(u) = C(v)) = 0$$

Challenge: NP-hard problem - exact methods infeasible for large graphs

Why Ant Colony Optimization?

ACO Key Advantages:

- **Constructive Approach:** Builds solutions from scratch (no initial solution dependency)
- **Collective Intelligence:** Multiple ants explore solution space in parallel
- **Adaptive Learning:** Pheromone-based memory guides search
- **Balanced Exploration-Exploitation:** Natural probabilistic mechanisms
- **Flexibility:** Adapts to different problem structures

Population-based Metaheuristic:

- Multiple solutions per iteration increase diversity
- Parallel execution leverages multi-core processors
- Collective experience accumulates over iterations

ACO Mechanism Overview

Key Components:

- ① Pheromone Matrix $\tau[v][c]$
- ② Heuristic Information $\eta[c]$
- ③ Probabilistic Selection
- ④ Pheromone Update

Probability Formula:

$$P(c) = \frac{[\tau[v][c]]^\alpha \cdot [\eta[c]]^\beta}{\sum_{c'} [\tau[v][c']]^\alpha \cdot [\eta[c']]^\beta}$$

Parameters:

- α : Pheromone importance
- β : Heuristic importance
- ρ : Evaporation rate
- m : Number of ants
- Q : Pheromone deposit

Parallel Execution:

- x ants per iteration
- Multi-threading

ACO Algorithm Pseudocode - Main Loop

Initialize: pheromone matrix $\tau[v][c] \leftarrow \tau_0$

FOR iteration = 1 to max_iterations:

 solutions $\leftarrow []$

FOR ant = 1 to m: *// Parallel execution*

 solution \leftarrow ConstructSolution()

 solutions.append(solution)

 best_solution $\leftarrow \text{argmin}(\text{solutions})$ *// Iteration best*

Evaporate: $\tau[v][c] \leftarrow (1 - \rho) \cdot \tau[v][c]$ for all v, c

Deposit: $\tau[v][c] \leftarrow \tau[v][c] + \frac{Q}{\text{num_colors}}$ for (v, c) in best_solution

IF no improvement for patience \times max_iterations:

break *// Early stopping*

RETURN best solution found

ACO Algorithm - Ant Solution Construction

ConstructSolution():

solution $\leftarrow \{\}$

vertices $\leftarrow \text{shuffle}(V)$ // Random order for diversity

FOR each v in vertices: // Sequential vertex coloring

 valid_colors $\leftarrow \{c : \text{no neighbor of } v \text{ uses } c\}$

IF valid_colors is empty:

 valid_colors $\leftarrow \{\text{new color}\}$

 // Heuristic: favor used colors (reuse)

$\eta[c] \leftarrow 2.0$ if c already used, else 1.0

 // Probabilistic selection based on pheromone and heuristic

$$P(c) \leftarrow \frac{[\tau[v][c]]^\alpha \cdot [\eta[c]]^\beta}{\sum_{c' \in \text{valid.colors}} [\tau[v][c']]^\alpha \cdot [\eta[c']]^\beta}$$

 solution[v] $\leftarrow \text{select } c \text{ with probability } P(c)$

RETURN best solution found

Solution Representation & Constraint Handling

Solution Representation

Dictionary mapping: Solution = $\{v_1 : c_1, v_2 : c_2, \dots, v_n : c_n\}$

Constraint Handling: Preserving Strategy

- Only valid colors considered (no conflicts with neighbors)
- Dynamic pheromone matrix expansion when needed
- **Guarantees:** All solutions are feasible ($F(C) = 0$)

Population Generation

- New population generated at each iteration
- Random starting nodes for diversity
- Random visitation order

Diversification & Intensification

Intensification:

- Pheromone reinforcement on best solutions
- Heuristic preference for used colors ($\eta = 2.0$)
- High α increases exploitation
- High β increases intensification

Diversification:

- Pheromone evaporation ($1 - \rho$)
- Probabilistic (not greedy) selection
- Random node ordering
- Multiple parallel ants
- Low ρ promotes exploration

Balance: Adaptive mechanism between exploration and exploitation

Hyperparameter Tuning with Optuna

Optimization Setup

- **Framework:** Optuna TPE (Tree-structured Parzen Estimator)
- **Trials:** 40 independent trials
- **Parallel Workers:** 6 (`n_jobs=6`)
- **Duration:** 46 hours wall-clock (225 hours computational)
- **Tuning Dataset:** `gc_500_9` (500 vertices, 90% density)
- **Objective:** Minimize total colors

Tuning Results

- Started at **215 colors**
- Best trial (Trial 19): **200 colors**
- **Improvement:** 15-color reduction (7%)

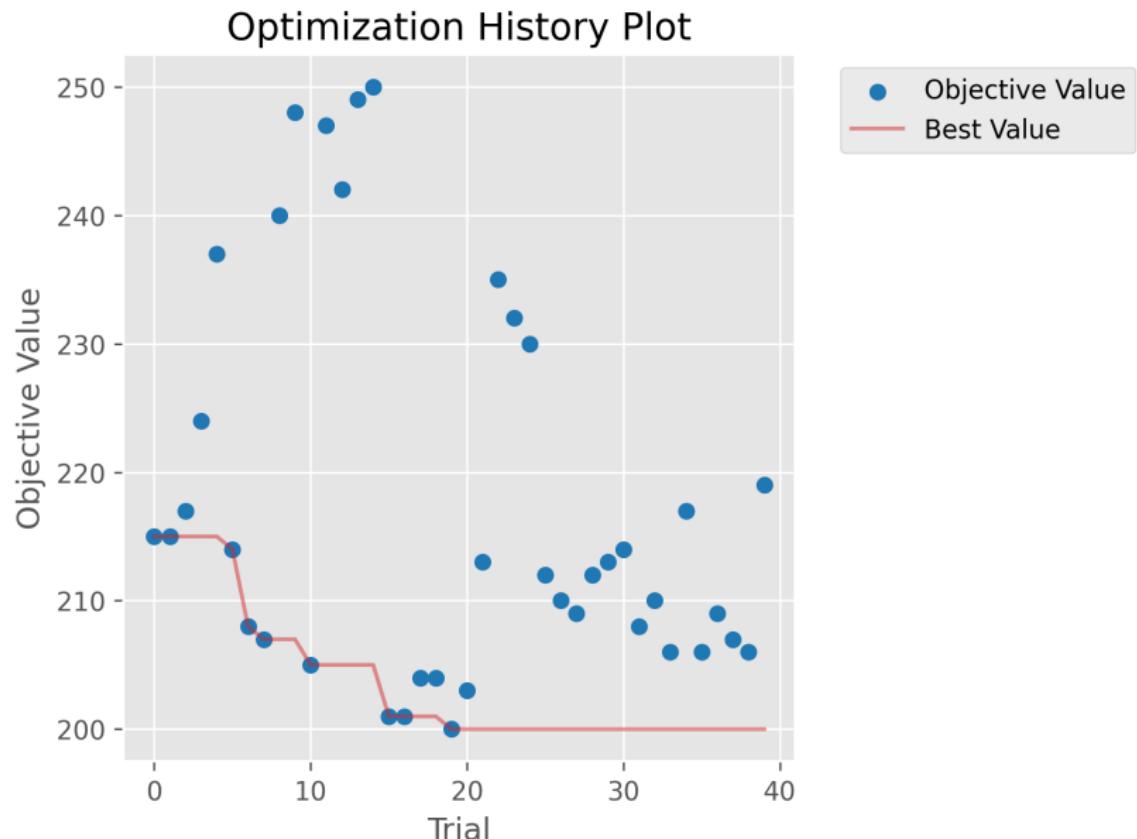
Optimal Parameters

Parameter	Search Range	Optimal Value
Iterations (T)	[200, 500]	261
Alpha (α)	[0.5, 2.0]	1.536
Beta (β)	[1.0, 10.0]	5.966
Rho (ρ)	[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

Key Findings: High β (heuristic) and low ρ (evaporation)

→ Intensification dominates over diversification

Optimization History



Optimization Timeline

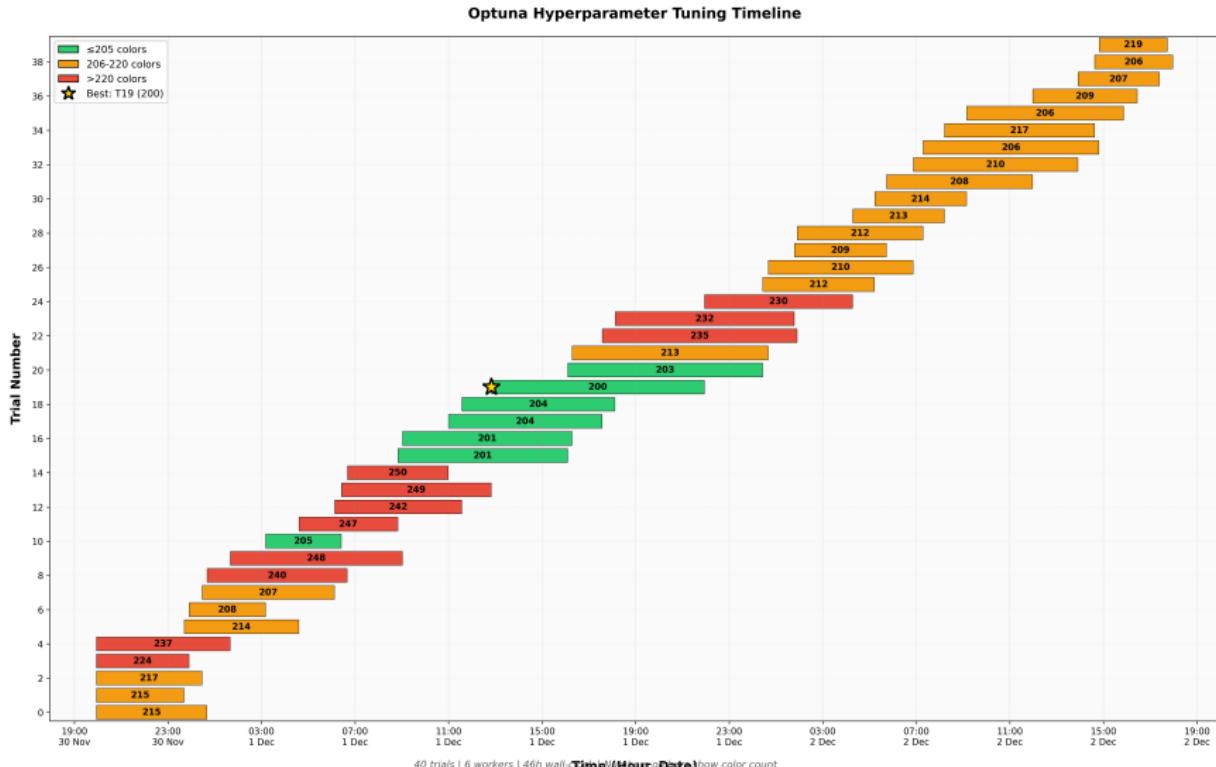


Figure: Parallel execution timeline showing 6 workers running 40 trials over 46 (KSU) ACO for Graph Coloring

Parameter Analysis

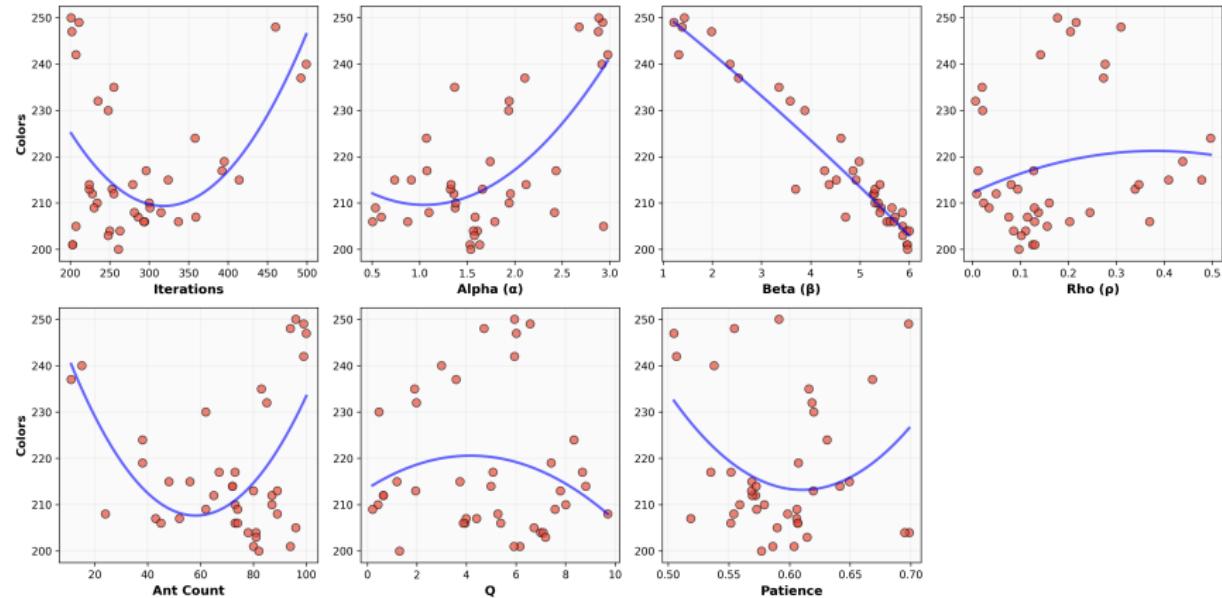


Figure: Parameter slice plots - Beta and iterations show clear trends

Parameter Importance

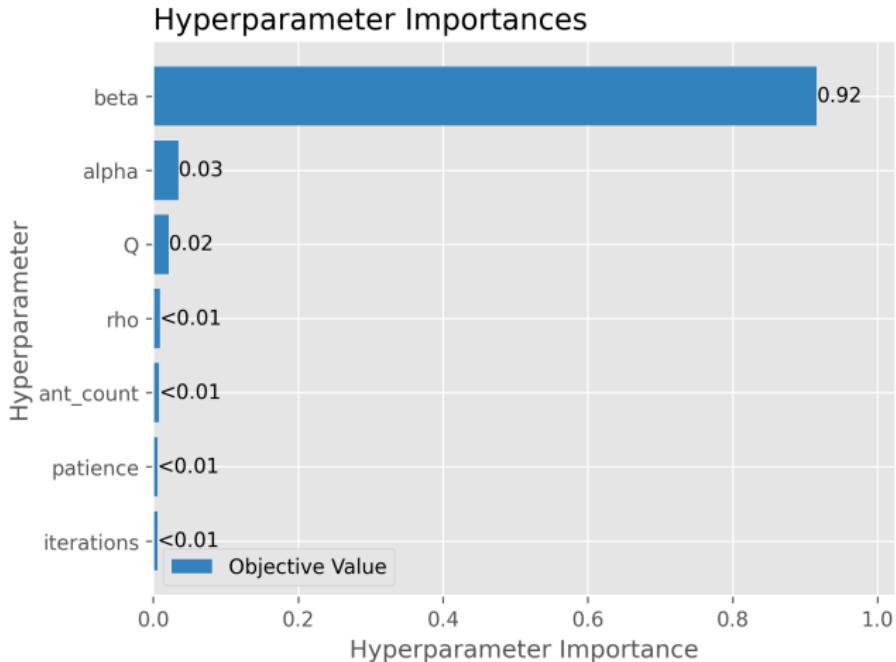
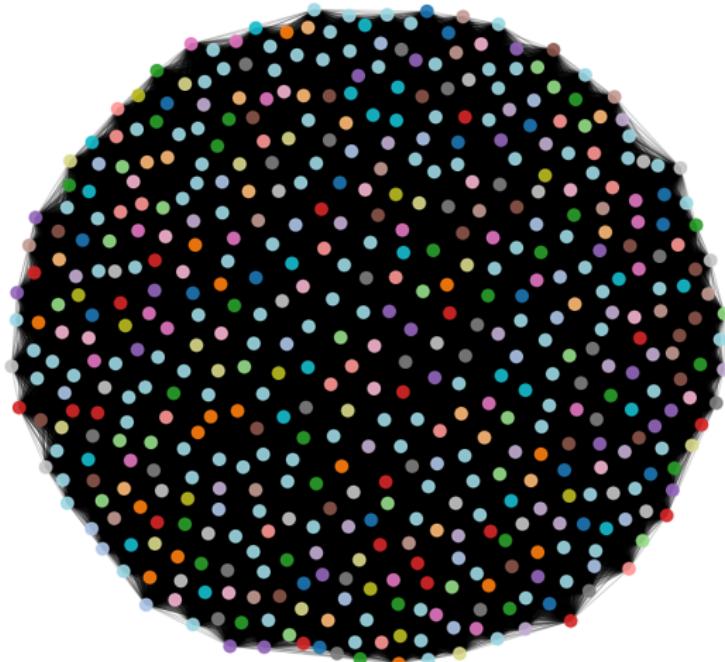


Figure: Iterations and Beta dominate performance; Rho has minimal impact

Best Trial Visualization

gc_500_9
Nodes: 500, Edges: 112224
Colors: 200



Benchmark Datasets

Instance	Vertices	Edges	Density	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

Test Protocol:

- DIMACS benchmark instances
- 3 independent runs per instance
- Statistical robustness assessment

Three-Way Performance Comparison

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

Key Finding: ACO underperforms both Greedy and Tabu Search
Deviation: 58-167% vs BKS (TS: 21-46%, Greedy: 38-50%)

Best Colors Comparison

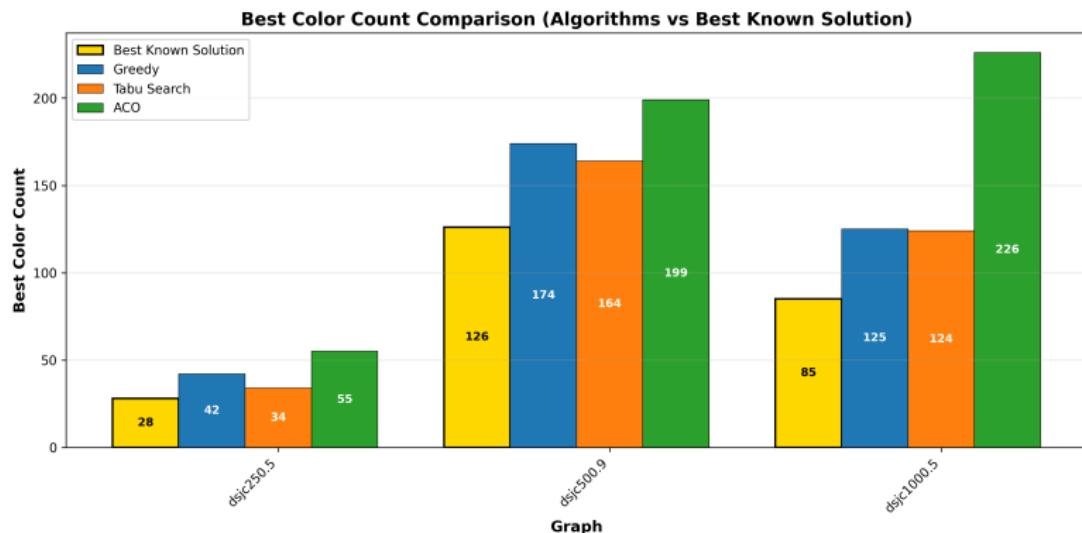


Figure: ACO produces 58-166% more colors than BKS

Average Colors Comparison

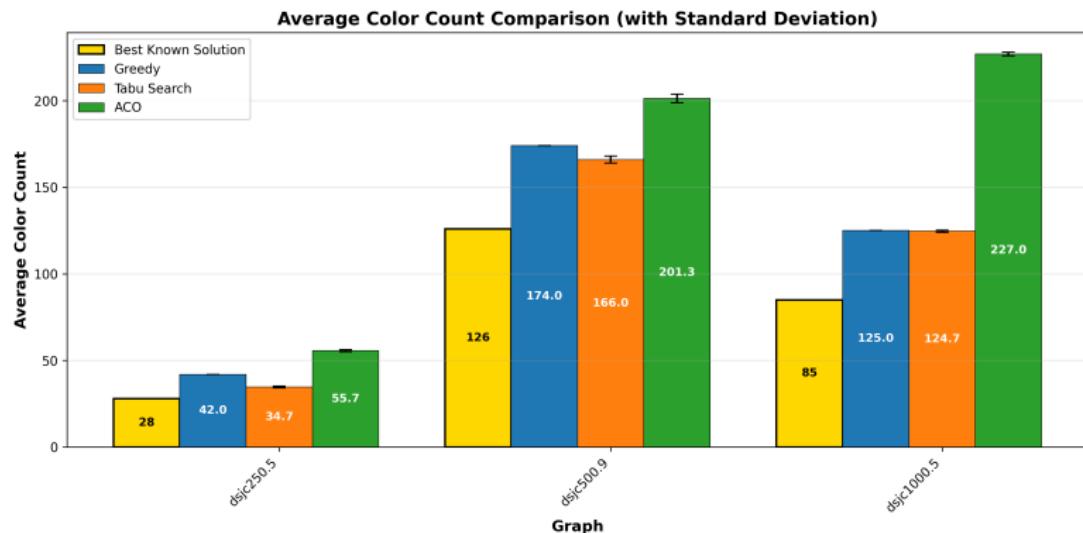


Figure: Performance gap widens with graph size: 61% (250v) → 82% (1000v)

Execution Time Comparison

Execution Time Comparison (log scale)

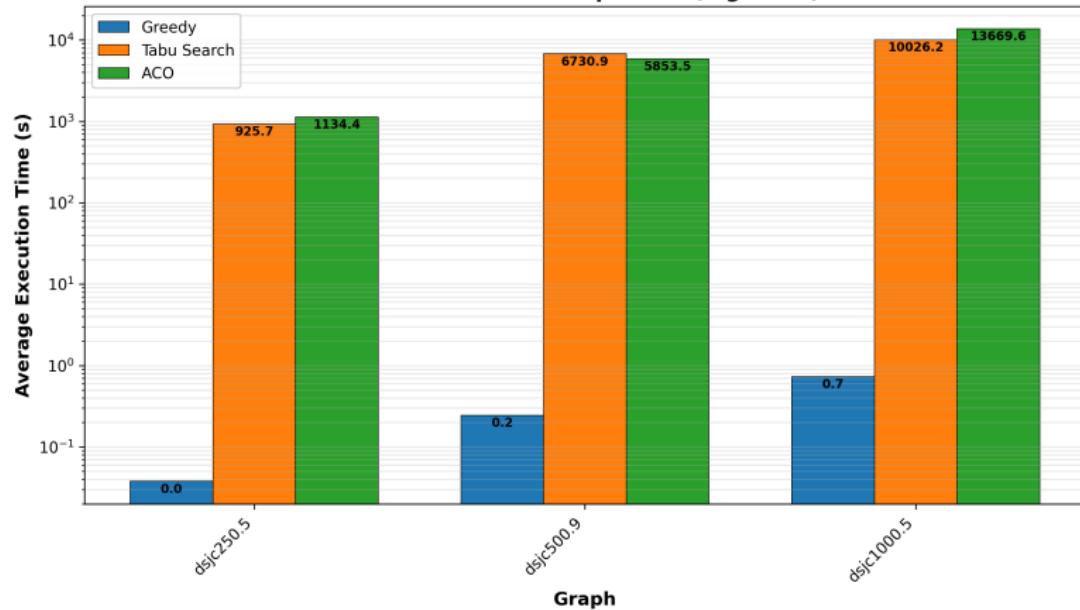


Figure: ACO time comparable to TS despite 82 parallel ants

Algorithm Comparison Summary

Greedy (Baseline)

Strength: Extremely fast (≤ 1 second)

Weakness: Moderate quality (38-50% deviation)

Use Case: Rapid prototyping, time-critical applications

Tabu Search (Assignment 2)

Strength: Best solution quality (21-46% deviation)

Weakness: High computational cost (926-10,026 seconds)

Use Case: Quality-critical applications with time budget

ACO (This Work)

Strength: Consistent, no initial dependency

Weakness: Poorest quality (58-167% deviation), comparable time to TS

Current Status: Not recommended in present form

Why ACO Underperforms

① Constructive Nature:

- Builds from scratch vs TS starts with 40 colors and improves to 35
- Must discover good assignments purely through pheromone learning

② Local Construction:

- Ants make locally optimal decisions without global view
- TS has global view of all conflicts

③ Pheromone Sparsity:

- Large color sets (50-200 colors) dilute pheromone signals
- TS's tabu list prevents specific moves without dilution

Scalability Issue: Performance degradation increases with graph size (96% deviation on 250v → 167% on 1000v)

Key Insights

Quality-Time Trade-off

- **Greedy:** Fast but moderate quality
- **Tabu Search:** Excellent quality, high cost
- **ACO:** Poor quality despite comparable cost to TS

Paradigm Suitability

- **Population-based \neq automatically better**
- Improvement-based (TS) benefits from feasible starting solutions
- Constructive methods (ACO) must learn from scratch
- Problem structure determines paradigm suitability

Tuning Insights

High β and low ρ essential \rightarrow **Intensification $\&$ Diversification**

Questions whether ACO's pheromone learning is well-suited for GCP

Thank You!

Questions?