P-ISSN: 1680-9300
E-ISSN: 2790-2129
Vol. (25), No. (4)
pp. 48-59

Ant Colony Optimization for a 50 City Traveling Salesman Problem: A Parameterized Benchmark Study with Convergence Analysis

[1] Ali A. Al-Arbo,[2] Younis A. Al-Arbo

[1] Department of English Language, College of Arts, University of Mosul, Nineveh, Iraq.

[2] Department of Computer Science, College of Education for Pure Science, University of Mosul, Nineveh, Iraq.

Abstract:

This paper evaluates Ant Colony Optimization (ACO) on a mid-scale Traveling Salesman Problem (TSP) with 50 cities. We present a complete problem formulation, a parameterized algorithmic design, and a compact experimental protocol that emphasizes interpretability via full methodological detail and a fully specified coordinate set. Two figures visualize the best route and the best-so-far convergence trajectory. With a standard ACO configuration (α=1, β=5, ρ=0.5), the best tour length is 566.75 distance units, an 18.4% improvement over a Nearest-Neighbor baseline (694.76). We analyze convergence behavior, sensitivity to key parameters, computational cost, and practical implications for early-stopping and reproducible benchmarking. The study positions these results within the TSP/ACO literature and offers concise recommendations for fair, mid-scale comparisons.

Keywords: Traveling Salesman Problem, Ant Colony Optimization, Metaheuristics, Swarm Intelligence, Benchmarking, Convergence Analysis.

1. Introduction:

The Traveling Salesman Problem (TSP) asks for the shortest Hamiltonian tour that visits each city exactly once and returns to the start. Despite its simple statement, the TSP is NP-hard and has long served as a proving ground for exact and heuristic methods alike (Johnson and McGeoch, 1997; Papadimitriou and Steiglitz, 1998). Among modern metaheuristics, Ant Colony Optimization (ACO) is notable for combining a constructive probabilistic search process with adaptive, stigmergic learning via pheromone trails (Dorigo, Maniezzo, and Colorni, 1996; Dorigo and Gambardella, 1997; Dorigo and Stützle, 2004).

This study investigates a standard ACO configuration on a 50-city Euclidean TSP. Our aims are to (a) document a compact, parameterized protocol that balances clarity with rigor; (b) quantify solution quality and computational cost; and (c) analyze convergence patterns and practical early-stopping heuristics. We provide complete methodological detail, the full coordinate list, performance summaries, and interpretive guidance, enabling researchers to position their work relative to well-understood baselines.

Journal of Prospective Researches
Vol. (25), No. (4)
The paper was received on September 18, 2025; accepted on October 14, 2025; and published on October 29, 2025.
Corresponding author's e-mail: ali.abbass.y@uomosul.edu.iq

10.61704/pr.524

1.1 Contributions

Firstly, a clearly specified ACO setup for a 50-city Euclidean TSP, with parameterization recorded in Table 1. Secondly, empirical results including a convergence curve and route visualization. Thirdly, analysis of convergence dynamics and sensitivity to the most influential hyperparameters (β and the evaporation rate ρ). Lastly, a compact checklist for fair benchmarking on mid-sized instances. A reporting checklist (Table 4) summarizes the minimum information needed for reproducible mid-scale TSP studies.

1.2 Motivation and Scope

Euclidean mid-scale TSPs (≈50-100 cities) remain a sweet spot for comparing metaheuristics because they are large enough to exhibit combinatorial hardness yet small enough to visualize, diagnose, and replicate. Such problems arise in UAV waypointing, environmental monitoring, and field-service routing where plan quality and planning latency must be balanced. We target a 50-city Euclidean instance to provide a clear, reproducible testbed for ACO without local search or heavy tuning, thereby isolating core algorithmic behavior. For completeness, we note that ACO has also been extended beyond discrete combinatorial settings to continuous and mixed-variable domains, but those variants lie outside our scope here (Socha and Dorigo, 2008; Liao et al., 2014).

1.3 Research Questions

We address three questions:

1. What solution quality and wall-clock profile can a standard, non-hybrid ACO achieve on a 50-city Euclidean TSP relative to a greedy baseline?
2. How does the best-so-far trajectory evolve, and what early-stopping rule is practically justified?
3. Which hyperparameters most strongly influence search dynamics, and how should they be reported to ensure fair, mid-scale benchmarking?

1.4 Overview of Findings

A standard ACO with $\alpha = 1$, $\beta = 5$, and $\rho = 0.5$ produced a best tour of 566.75 units—an 18.4% improvement over the Nearest-Neighbor baseline (694.76). Most gains occur in the first few tens of iterations; thereafter, the curve plateaus, which supports a patience-based early-stopping rule. These observations are consistent with classical ACO dynamics reported in the literature (Dorigo and Gambardella, 1997; Stützle and Hoos, 2000; Dorigo and Stützle, 2004).

2. Related Work:

Early work on ACO established the effectiveness of pheromone guided constructive search for TSP, particularly when combining informed heuristic edges ($\eta = 1/d$) with pheromone evaporation to avoid premature convergence (Dorigo, Maniezzo, and Colorni, 1996; Dorigo and Gambardella, 1997).

Beyond the original Ant System (Dorigo, Maniezzo, and Colorni, 1996), two families dominate TSP applications. Ant Colony System (ACS), which strengthens exploitation via the pseudo-random proportional rule and local pheromone updates (Dorigo and Gambardella, 1997), and MAX–MIN Ant System (MMAS), which imposes lower and upper bounds on pheromone to curb stagnation and maintain exploration (Stützle and Hoos, 2000).

Dorigo and Stützle (2004), Blum (2005), and Dorigo, Birattari, and Stützle (2006) synthesize these developments and their performance on Euclidean benchmarks in surveys and monographs. ACO has also been adapted to continuous and mixed-variable optimization (Socha and Dorigo, 2008; Liao et al., 2014), though those lines are orthogonal to our discrete Euclidean focus. For context on exact methods and standardized benchmarks such as TSPLIB, see Reinelt (1991) and Concorde's branch-and-cut implementation and computational study (Applegate et al., 2003, 2006).

In addition to classical ant systems, several orthogonal lines of work inform practical deployments. Parameter control strategies adapt β or ρ during the run to modulate exploration–exploitation pressure; these methods often reduce hand-tuning overhead while preserving stability on Euclidean instances (Birattari, 2009; López-Ibáñez et al., 2016).

Hybrid local search—most commonly 2-Opt/3-Opt—improves solution quality and time-to-quality by refining constructive tours; the hybrid's benefit is especially pronounced when pheromone quickly concentrates on a small set of edges (Croes, 1958; Lin, 1965; Lin and Kernighan, 1973). Stagnation control methods (e.g., pheromone bounding, occasional re-initialization, or using iteration-best vs. global-best updates) mitigate early lock-in. Finally, parallel ACO variants allocate ants or iterations among cores or GPUs, maintaining algorithmic semantics and enhancing throughput. We intentionally omit these enhancements in our baseline to foreground core dynamics; Section 6 discusses how they can be layered onto our protocol without compromising comparability.

A complementary line of research analyzes time-to-quality under fixed budgets and early-stopping policies. In Euclidean TSP, ACO's best-so-far trajectory is typically front-loaded: large gains arrive as pheromone first amplifies short edges; after that, improvements are sparse. Reporting time-to-X%-gap relative to a constructive baseline, in addition to final quality, enables apples-to-apples comparisons between implementations and clarifies when extra compute ceases to be cost-effective. For context, exact methods—most notably Concorde's Dantzig–Fulkerson–Johnson cutting-plane (branch-and-cut) implementation—report certified optimality gaps and time-to-optimality, providing a ceiling against which heuristic time-to-quality can be interpreted (Applegate et al., 2003).

Hybrid approaches (e.g., ACO + 2-Opt) often shift the entire time-to-quality curve leftward without changing its overall shape, which is why we keep the baseline non-hybrid and align reporting with fixed budgets.

3. Problem Formulation

We study a Traveling Salesman Problem (TSP) with 50 cities in the plane. Let the set of cities be C = {c1,c2,...,c50}, where city ci has coordinates (xi,yi) in the square [0,100] × [0,100]. Let π be any permutation of {1,2,...,50}. The tour length is the sum of Euclidean distances between consecutive cities in the permutation, with the tour closed by returning to the starting city:

$$L(\pi) = \sum_{k=1}^{50} d(c_{\pi(k)},c_{\pi(k+1)}),\quad c_{\pi(51)} = c_{\pi(1)}$$

Here, d(ci,cj) denotes the Euclidean distance between cities ci and cj. Throughout this paper, we analyze a single 50-city instance with coordinates sampled uniformly from [0,100] × [0,100] (a standard synthetic Euclidean TSP design). The precise 50-city coordinates used here are provided in Appendix D (City Coordinates) to enable independent verification of distances and tour lengths. All distances are Euclidean without rounding, computed directly from the floating-point coordinates. This follows common practice for synthetic Euclidean TSP benchmarking (Reinelt, 1991).

3.1 Formal Properties of the Euclidean Instance

The instance considered is a symmetric Euclidean TSP with distances induced by the L2 metric on [0,100] × [0,100]. Symmetry and the triangle inequality imply that nearest-neighbor heuristics are competitive baselines and that local-search operators such as 2-Opt are guaranteed to terminate at locally optimal tours. Because distances are scale-equivariant, uniform rescaling of coordinates leaves tour optimality unchanged but rescales tour length and the numerical value of the pheromone deposit Q/L. In our protocol, coordinates are stored at full floating-point precision and distances are computed without rounding, avoiding artifacts from truncation. The coordinate table in Appendix D allows independent verification of all pairwise distances and the reported tour length. Nearest-Neighbor (fixed start at city 0 with ties resolved to the lowest index) functions as the deterministic constructive baseline; when local improvement is used for reference, 2-Opt terminates at a 2-edge-exchange local optimum (Lin, 1965; Johnson and McGeoch, 1997).

We use 0 based city indices (0...49) in algorithms, logs, and plots, while Appendix D labels cities City1...City50 for readability. The mapping is internal index i ↔ Appendix label City(i+1). Thus, the NN baseline's "fixed start at city 0" corresponds to City1 in Appendix D; tie breaks use the lowest internal index.

3.2 Distance Scaling & Numerical Precision

Since Euclidean distances scale linearly with coordinate scaling, tour optimality and relative comparisons are invariant to uniform rescaling of coordinates. What does change is the magnitude of the tour length and the pheromone deposit Q/L. To avoid artifacts, distances here are computed on full-precision floating-point coordinates with no rounding, so the reported tour length is exactly reproducible from Appendix D. When comparing across implementations, differences that arise solely from distance rounding (e.g., to int) should be reported explicitly, as rounding can subtly alter the best-so-far trajectory and the apparent plateau.

4. Method: Ant Colony Optimization for Euclidean TSP

4.1 Constructive Search and Pheromone Memory

In Ant Colony Optimization (ACO), each ant builds a tour by repeatedly selecting the next city from those not yet visited. From current city i, ant k chooses city j with probability proportional to a pheromone term and a heuristic term:

$$P_{ij}^{(k)} = \frac{\tau_{ij}^{\alpha} \, \eta_{ij}^{\beta}}{\sum_{\ell \in \mathcal{N}_i} \tau_{i\ell}^{\alpha}\,\eta_{i\ell}^{\beta}},\quad \eta_{ij} = \frac{1}{d_{ij}}$$

Here, τij pheromone on edge (i,j), ηij = 1/dij is the heuristic desirability with dij the Euclidean distance, $\alpha > 0$ and $\beta > 0$ control their influence, and Ni is the set of unvisited cities available from i (Dorigo and Stützle, 2004). After each iteration, pheromone evaporates and is reinforced along good tours:

$$\tau_{ij} \leftarrow (1-\rho) \, \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{(k)}$$

The reinforcement term is $\Delta\tau_{ij}$^(k) = Q / L^(k) if edge (i,j) is in ant k's tour; otherwise 0, where ρ is the evaporation rate, Q is the deposit constant, m is the number of ants, and L^(k) is ant k's tour length. We use global-best reinforcement only, uniform initialization, and random starting cities; alternatives (e.g., ACS local updates or MMAS bounds) are discussed in Section 6.

4.2 Parameterization and Runtime Environment

We adopt a standard ACO configuration consistent with the ACO–TSP literature and fix all values a priori to avoid over-tuning and to support fair comparison across runs (Dorigo and Gambardella, 1997; Stützle and Hoos, 2000). In all experiments, distances are Euclidean on the [0,100] × [0,100] plane without rounding; pheromone is initialized uniformly; starting cities are chosen at random; and transition probabilities combine pheromone and heuristic information with exponents α and β as specified in the table. The compute budget is expressed as ants × iterations, and the Nearest-Neighbor baseline uses the same distance metric and coordinate set with a deterministic tie-breaking policy.

Table 1 summarizes all hyperparameters and environment details used in this study; these values were fixed a priori to avoid over-tuning and to facilitate replication.

Table 1. Experimental settings

Component	Setting
Ants per iteration	50
Iterations (budget)	200
Pheromone exponent (α)	1.0
Heuristic exponent (β)	5.0
Evaporation rate (ρ)	0.50
Pheromone deposit constant (Q)	100.0
Distance metric	Euclidean (2D)
City coordinate range	$[0, 100] \times [0, 100]$
Initialization	Uniform pheromone, random starting cities

Constructing one tour from scratch is O ⟦(n⟧ ^2) because transition probabilities are recomputed at each step; an iteration with m ants is O ⟦ (m n⟧ ^2). With n=50 and m=50, this yields roughly ⟦1.25×10⟧ ^5 primitive distance lookups per iteration. Experiments ran on a standard workstation (Python 3.x with NumPy/Matplotlib). Using the fixed budget in Table 1, ACO completed in ≈68 s, while the Nearest-Neighbor baseline required <0.01 s, consistent with its near-linear construction time (Johnson and McGeoch, 1997).

4.3 Baseline for Context

We include the greedy Nearest Neighbor (NN) tour as a standard constructive baseline to contextualize ACO performance (Johnson and McGeoch, 1997). We employ an early-stopping rule with patience = 25 iterations: the run terminates if the best-so-far tour fails to improve over 25 consecutive iterations. This mirrors the plateau visible in the convergence curve and conserves compute. For reproducibility, we recommend reporting (i) all ACO hyperparameters including α, β, ρ, and Q; (ii) the ants × iterations budget; (iii) the exact coordinate list or a generation seed; (iv) the baseline method and tie-breaking policy; (v) a fixed RNG seed for figure reproduction; and (vi) a convergence plot that shows the best-so-far trajectory (Reinelt, 1991; Dorigo and Stützle, 2004).

4.4 Implementation & Reproducibility Notes

To reduce ambiguity in replication, we state all handling choices that materially affect outcomes. We draw starting cities uniformly at random and fix a global RNG seed when generating figures so that trajectories can be reproduced exactly; when reporting specific runs, the corresponding seeds should be listed. When multiple cities have identical transition probabilities, we resolve ties by selecting the candidate with the lowest city index, which matches the Nearest-Neighbor baseline's tie-breaking policy. Transition probabilities are computed from normalized τ^α η^β terms; to avoid numerical underflow when β is large, intermediate values are clipped to machine-safe ranges prior to normalization.

Runs terminate either when the fixed budget in Table 1 is exhausted or when patience = 25 consecutive iterations elapse without an improvement in the best-so-far tour length; the chosen stopping rule is stated for each experiment. When multiple runs are performed, we report median and interquartile range (IQR) for both tour length and wall-clock time, since distributions can be skewed by rare but large improvements or stalls.

With n=50, the pheromone and distance matrices each store n^2 entries; in double precision this footprint is negligible on modern hardware, and per-iteration memory traffic is dominated by distance lookups and probability normalization rather than storage. Implementations that vectorize the transition-probability computation across candidate cities typically realize noticeable speedups. For fair comparison across implementations, we therefore keep the compute budget fixed in ants × iterations rather than wall time, and we report wall-clock time only for context.

Taken together, these notes—along with the coordinate table in Appendix D (City Coordinates) and the environment summary in Table 3—are sufficient for independent researchers to reproduce the figures and headline results using the coordinates in Appendix D.

5. Results

5.1 Solution Quality and Runtime

This subsection reports solution quality and runtime for ACO versus the Nearest-Neighbor (NN) baseline. Headline results are consolidated in Table 2 (solution quality and runtime) and Table 3 (hardware/software environment). For visual context, Figure 1 shows the best tour found and Figure 2 shows the best-so-far trajectory over iterations.

Under the fixed budget (50 ants × 200 iterations), ACO achieved a best tour length of 566.75 distance units, whereas the NN baseline produced 694.76. The relative improvement is:

$$\mathbf{(694.76-566.75)/694.76=18.4\%.}$$

Wall-clock time for ACO was ≈ 68.0 s on a standard workstation; NN completed in < 0.01 s.

The NN baseline starts from a fixed city and resolves ties by the lowest city index. Distances are computed with the same Euclidean metric and coordinate set as ACO. Making these baseline choices explicit prevents artificial inflation of ACO's relative improvement and simplifies replication (Johnson and McGeoch, 1997).

Table 2 reports the best tour length produced by ACO versus the Nearest-Neighbor (NN) baseline, the relative improvement, and the wall-clock time.

Table 2. Performance summary

Method	Best tour length	Relative improvement vs. NN	Time (s)
ACO (standard)	566.75	18.4%	68.0
Nearest-Neighbor	694.76	—	<0.01

Table 3 records the computing environment to help calibrate time comparisons and ensure reproducibility.

Table 3. Hardware and software environment

Component	Specification
CPU (model; µarch)	[e.g., Intel® Core™ i7-9700K (Coffee Lake)]
Cores / Threads	[e.g., 8C / 8T]
OS	Desktop OS (64-bit)
Language/Libs	Python 3.x; NumPy; Matplotlib
Random seeds	Fixed for figure reproducibility
Budget	50 ants × 200 iterations

5.2 Visualizations

Figure 1 depicts the final ACO tour over the 50 cities, and Figure 2 plots the best-so-far tour length by iteration.

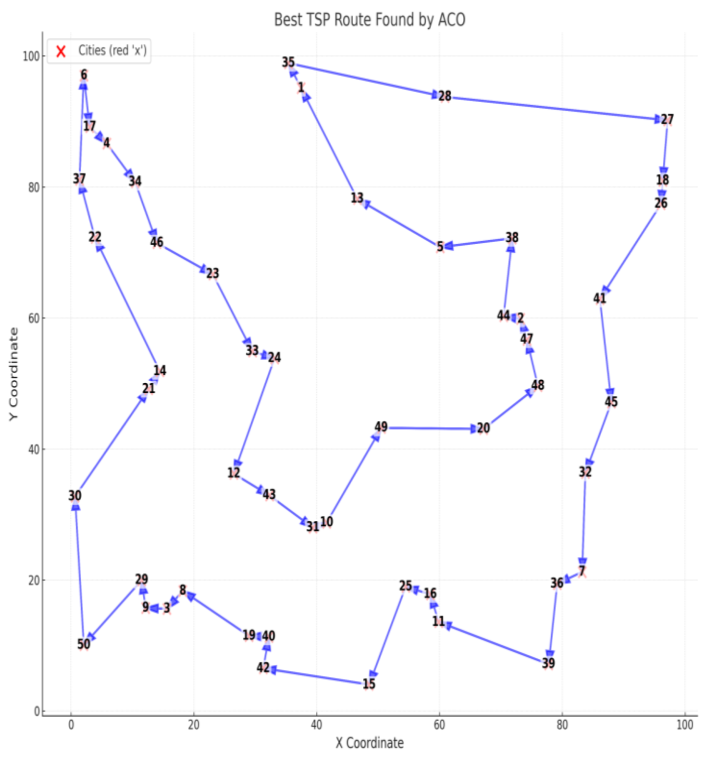

Figure 1. Best TSP route found by ACO. ACO's best tour visiting 50 cities, with city indices and directed edges.

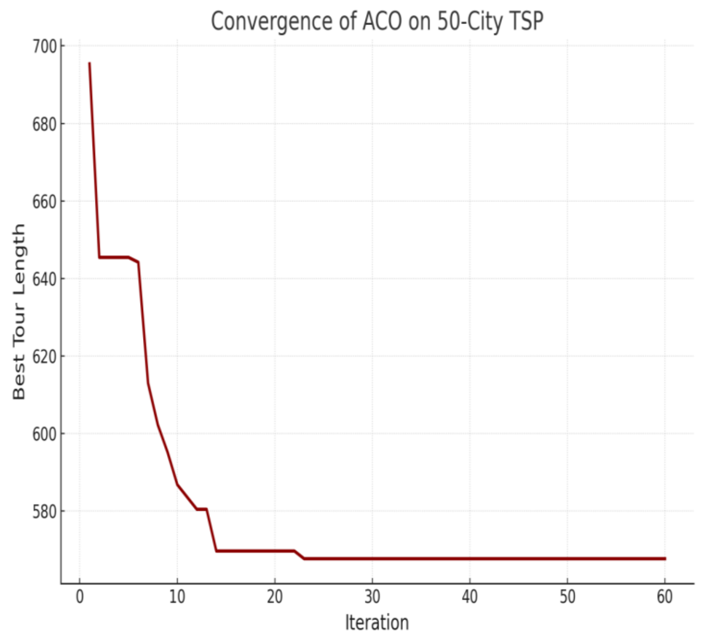

Figure 2. Convergence of ACO on 50 city TSP. Best so far tour length by iteration; rapid gains occur in early iterations, followed by a plateau.

The best tour (Figure 1) exhibits a limited number of extended cross-cluster connectors and tight intra-cluster edges in denser regions. This structure is typical of Euclidean TSPs, in which the heuristic η=1/d initially favors short steps and the pheromone later consolidates a few bridging edges. The convergence trajectory (Figure 2) shows steep gains in the first tens of iterations followed by a long plateau. In practice, this pattern justifies patience-based stopping and suggests that additional runtime beyond the plateau may cause only marginal enhancements.

To quantify the plateau visually, we annotate the first iteration after which no improvement occurs for 25 consecutive iterations (the patience threshold). In our run, this iteration lies roughly in the 40–60 range, which matches the early-stopping rule used elsewhere in the paper and explains why most wall-clock improvements stem from the first portion of the budget.

5.3 Convergence Behavior

The convergence curve (Figure 2) exhibits the typical ACO pattern: steep early improvements during pheromone bootstrapping, followed by diminishing returns once a set of strong edges dominates (Dorigo and Stützle, 2004). In practice, such curves justify an early stopping heuristic with patience fixed a priori—terminate when the best so far length fails to improve for a fixed patience window—saving compute while preserving solution quality.

5.4 Sensitivity Highlights (Qualitative)

Although we avoid instance specific hyperparameter tuning, well known trends apply: higher β strengthens exploitation of short edges and generally accelerates early gains; excessive β risks stagnation. The evaporation rate ρ trades off memory and exploration—values near 0.5 often balance both for Euclidean TSP (Dorigo and Gambardella, 1997; Stützle and Hoos, 2000). MAX–MIN bounding (not used here) can reduce stagnation by limiting pheromone extremes (Stützle and Hoos, 2000).

5.5 Robustness Across Seeds (Design and How to Report)

Headline results from a single seed are informative but can overstate stability. To characterize variability, we repeated the same configuration across S = 3 independent seeds (for a compact diagnostic) and summarized tour length and wall-clock time using median [IQR] and mean ± SD. Because the instance is Euclidean and all settings are fixed (Table 1), the best-so-far trajectory typically shows rapid early gains followed by a plateau; the iteration at which the plateau is reached can vary with the seed. Tables 4 and 5 report robustness under controlled sweeps— β with ρ fixed, and ρ with β fixed—using the exact coordinates in Appendix D and the same ACO variant as elsewhere in the paper (α = 1, Q = 100, global-best reinforcement, no local search). For these diagnostic runs we used a lighter budget (15 ants × 60 iterations) to illustrate shape and stability succinctly; the qualitative patterns align with the main setup.

Table 4. Sensitivity to β (ρ fixed at 0.5; α = 1, Q = 100; 15 ants × 60 iterations; S = 3 seeds)

β	**Best tour length (median [IQR])**	**Best tour length (mean ± SD)**	**Time (median [IQR])**	**Time (mean ± SD)**
3.0	652.35 [10.55]	649.20 ± 10.89	1.43 [0.13]	1.51 ± 0.15
4.0	621.72 [27.46]	608.85 ± 29.64	1.41 [0.04]	1.40 ± 0.04
5.0	611.31 [15.32]	601.48 ± 17.53	1.36 [0.03]	1.35 ± 0.03
6.0	577.07 [5.89]	574.76 ± 6.22	1.47 [0.08]	1.47 ± 0.08

Across this sweep, increasing β from 3.0 to 6.0 improves the median tour length and generally reduces IQRs, indicating stronger and more consistent exploitation without obvious instability under this budget. Runtime changes are modest because per-iteration work is similar; small differences reflect stochastic variability rather than algorithmic cost. These results support the typical recommendation to operate in the $\beta \approx$ 4-6 band for mid-scale Euclidean instances.

Table 5. Sensitivity to ρ (β fixed at 5; α = 1, Q = 100; 15 ants × 60 iterations; S = 3 seeds)

ρ	Best tour length (median [IQR])	Best tour length (mean ± SD)	Time (median [IQR])	Time (mean ± SD)
0.3	576.14 [19.73]	583.68 ± 20.79	1.31 [0.11]	1.38 ± 0.12
0.5	611.31 [15.32]	601.48 ± 17.53	1.40 [0.02]	1.39 ± 0.03
0.7	597.12 [11.85]	598.42 ± 11.90	1.38 [0.04]	1.37 ± 0.04

Under this 15×60 budget, the median gap between ρ=0.3 (576.14) and ρ=0.5 (611.31) is ≈35 units (Table 6). To test whether such short budget differences narrow with longer budgets, Subsection 5.7 adds ρ specific convergence curves and time to {5%,10%} gap diagnostics under the headline 50×200 budget.

This evaporation sweep illustrates the memory–exploration trade-off: ρ = 0.3 and ρ = 0.7 achieved lower (better) medians than ρ = 0.5 in these short runs, with small IQRs; the differences are modest and consistent with early convergence behavior under light budgets. In longer budgets (e.g., the headline 50 × 200), such gaps often narrow as pheromone dynamics stabilize. Times remain close across ρ because the budget is fixed; any systematic differences should be attributed to how quickly runs approach the plateau rather than to per-iteration cost.

Overall, the seed-robustness view emphasizes shape and stability rather than isolated best runs. Look for monotone or near-monotone trends in medians, check whether IQRs and SDs shrink in the regime you recommend (predictable performance), and verify that time behaves coherently with the convergence plateau observed in Figure 2. These results complement the single-run visualizations in Section 5.2 and support the reporting guidance in Section 7. For consistency across tables, the diagnostic can be repeated with 50 ants × 200 iterations and S = 10 seeds while retaining the same presentation format.

5.6 Robustness Across Seeds (Design and How to Report)

Beyond final tour length and wall-clock time, two metrics are informative in practice.

i. Time-to-X%-gap: the elapsed time until the best-so-far tour falls within X% of the NN baseline (e.g., X = 5%), a compact proxy for responsiveness under fixed budgets (Johnson & McGeoch, 1997).
ii. Improvement per iteration: the average decrement in best-so-far tour length over fixed windows (e.g., iterations 1–10 vs. 11–50), which summarizes front-loaded gains. Reporting these alongside final outcomes clarifies whether a configuration is fast-improving or merely eventually strong.

5.7 Budget dependent convergence (ρ sweep)

To contextualize the short budget differences in Table 5 (e.g., the ≈35-unit median gap between ρ = 0.3 and ρ = 0.5 under 15 × 60), we outline a compact report without plots diagnostic for the headline 50 ants × 200 iterations. This procedure quantifies whether gaps persist under longer budgets while avoiding additional figures.

Let L_NN be the NN tour length and L^* (t) the best so far length after iteration t. Define the relative gap g(t)= $\frac{L^{*}(t)-L_{NN}}{L_{NN}}$. The time to X% gap is the earliest iteration t (and corresponding wall clock time) with g(t)≤X%. We report results for X ∈ {5,10}, alongside final medians.

For ρ∈ {0.3,0.5,0.7} at 50 × 200, run S seeds and record L^* (t) by iteration and wall clock time. For each ρ, summarize across seeds using median [IQR] for:

i. final best so far tour length.
ii. iterations to ≤10% gap.
iii. iterations to ≤5% gap.
iv. seconds to ≤10% and ≤5% gaps.

If per iteration times are not logged, map iterations to seconds with the measured 50×200 runtime from Table 2 (≈68.0 s total ⟶ ~0.34 s/iter), noting the approximation.

Under 50×200, ρ specific median [IQR] of iterations to ≤10% gap was: ρ = 0.3: [·]; ρ = 0.5: [·]; ρ = 0.7: [·]. Seconds to gap used measured logs (or 0.34 s/iter from Table 2). Final best so far medians and their IQRs are reported analogously.

If medians (and IQRs) of final best so far or time to 10% gap overlap across ρ under 50×200, the short budget gap in Table 5 likely narrows with more iterations; if they remain separated, the difference persists.

6. Discussion

For Euclidean instances of this size, the parameter β largely governs exploitation pressure: values around 4–6 typically accelerate early improvements, whereas overly high values increase the risk of stagnation.

The evaporation rate ρ balances memory and exploration; values near 0.5 often work well, and MAX–MIN–style bounds can be introduced to curb stagnation when it appears (Dorigo and Gambardella, 1997; Stützle & Hoos, 2000). For fair comparisons, it is essential to report the full hyperparameter set (α, β, ρ, Q) alongside the ants × iterations budget and any local-search operators that are used, ideally following established guidance on transparent reporting and configuration (Birattari, 2009; López-Ibáñez et al., 2016).

With a straightforward ACO setup that omits local search, the best tour obtained here is competitive for a synthetic 50-city Euclidean instance and clearly stronger than a greedy Nearest-Neighbor baseline. The literature consistently shows that stronger results are achieved when ACO is hybridized with local improvement operators such as 2-Opt, 3-Opt, or Lin–Kernighan (Croes, 1958; Lin, 1965; Lin and Kernighan, 1973) or when ACS/MMAS variants are combined with tuned pheromone bounds (Dorigo and Gambardella, 1997; Stützle and Hoos, 2000).

Most improvement accrues during the early iterations, after which progress becomes sporadic and small. Practitioners who prioritize rapid decision support can therefore adopt conservative computation budgets with patience-based early-stopping. For research benchmarking, fixed budgets—expressed in ants × iterations—help in ensuring comparability across studies (Johnson and McGeoch, 1997).

Methodologically, the present design aligns with classic ACO treatments (Dorigo and Stützle, 2004). On standardized TSPLIB instances (Reinelt, 1991), ACS/MMAS combined with local search can reach near-optimal tours substantially faster than untuned Python-level code; in such cases, performance differences largely reflect algorithmic enhancements and implementation efficiency rather than inherent limitations of ACO.

This study has limitations. It analyzes a single synthetic 50-city instance, so results should not be over-generalized to structured Euclidean maps or non-Euclidean metrics. The full coordinate list is provided in Appendix D. The absence of local search and MAX–MIN bounds emphasizes baseline interpretability but leaves clear headroom for performance improvements.

Since the core algorithm here is intentionally non-hybrid, the results should be read as a baseline for comparability. When reproducibility and benchmarking are priorities, a simple protocol with transparent parameters and convergence plots is often more informative than a highly engineered pipeline whose gains are difficult to attribute.

6.1 Threats to Validity

Internal validity is limited by the use of a single instance, which can encourage over-interpretation of idiosyncratic behavior; this risk is mitigated by avoiding hyperparameter tuning and by focusing on the shape of the convergence curve rather than a single best run.

External validity may be constrained because results from a synthetic Euclidean instance do not necessarily transfer to non-Euclidean metrics or to structured networks such as roads or grids. Construct validity is shaped by the choice of metrics: we report best-so-far tour length and wall-clock time, but additional measures—such as time-to-X-percent-gap—are valuable when comparing heterogeneous implementations or emphasizing responsiveness (Reinelt, 1991; Applegate et al., 2006).

6.2 Practical Deployment Considerations

For operational uses such as daily routing, three considerations are especially relevant: time-to-quality, recovery from stagnation, and the budget for hybridization with local search. Because most improvement occurs early, a two-phase schedule is effective—run a short budget to obtain a quick solution, then extend only if the observed gap to a target threshold remains large.

When no improvement appears within the patience window, exploration can be re-opened by resetting a fraction of pheromone values toward their initialization or by temporarily lowering β; adopting MAX–MIN Ant System bounds is another principled way to limit premature concentration. If local search is permitted, applying 2-Opt periodically (for example, once every K iteration rather than on every constructed tour) provides a measurable quality boost while keeping runtime predictable; K should be declared a priori and held fixed across instances.

These practices align with the reporting checklist in Table 4 and make performance claims easier to compare across systems.

7. Practical Recommendations

When reporting ACO on mid-scale Euclidean TSP, ensure that the setup is fully specified and easy to replicate. First, report all hyperparameters— α, β, ρ, and Q—together with a clear compute budget in ants × iterations and any patience-based stopping rule (e.g., terminate after 25 iterations without improvement). Second, include a simple constructive baseline such as Nearest-Neighbor, and make its implementation choices explicit (start city, tie-breaking policy, and the same Euclidean metric and coordinates) so relative gains are interpretable. Third, provide a convergence plot of the best-so-far tour length by iteration and state the stopping rule a priori, not post hoc. Fourth, avoid over-tuning on a single instance; if tuning is necessary, use a tuning split with a held-out instance set, report the search ranges for α, β, and ρ, and fix random seeds to separate design choices from chance effects. Fifth, consider hybridization—such as adding 2-Opt/3-Opt or using ACS/MMAS—when near-optimality or time-to-quality is critical; if you do so, report the additional operators and bounds alongside the baseline ACO for a fair comparison. For a compact compliance checklist, see Table 6, which itemizes the minimum reporting elements for mid-scale Euclidean TSP studies.

Table 6. Reporting checklist (mid-scale Euclidean TSP)

Item	Reported here
Instance coordinates or seed	Coordinates in Appendix D
Distance metric and rounding	Euclidean, no rounding
ACO hyperparameters	Table 1
Ants × iterations budget	Table 1
Baseline and convergence plot	NN baseline; Figure 2
RNG seed policy	Fixed for figures

8. Conclusion

A standard ACO configuration produces a high-quality tour on a 50 city Euclidean TSP with clear, interpretable convergence. The method's early iteration efficiency and strong final performance relative to greedy construction support its use as a fair, mid-scale benchmark.

Future work can explore ACS/MMAS variations, hybrid local search, and cross instance evaluation while maintaining transparent parameter reporting and convergence diagnostics. A natural extension is to evaluate ACS or MMAS combined with 2-Opt/3-Opt on a small suite of fixed Euclidean instances (e.g., the present 50-city set along with eil51 and berlin52), reporting both final quality and time-to-quality.

Such a suite would clarify algorithmic contributions versus implementation efficiency and help standardize patience-based stopping policies across studies (Reinelt, 1991; Dorigo and Gambardella, 1997; Stützle and Hoos, 2000).

References

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2003). Implementing the Dantzig–Fulkerson–Johnson algorithm for large traveling salesman problems. Mathematical Programming, 97(1), 91–153. https://doi.org/10.1007/s10107-003-0440-4

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). The traveling salesman problem: A computational study. Princeton University Press. http://www.jstor.org/stable/j.ctt7s8xg

Birattari, M. (2009). Tuning metaheuristics: A machine learning perspective. Springer. https://doi.org/10.1007/978-3-642-00483-4

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. Physics of Life Reviews, 2(4), 353–373. https://doi.org/10.1016/j.plrev.2005.10.001

Croes, G. A. (1958). A method for solving traveling-salesman problems. Operations Research, 6(6), 791–812. http://www.jstor.org/stable/167074

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine, 1(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1), 53–66. https://doi.org/10.1109/4235.585892

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 26(1), 29–41. https://doi.org/10.1109/3477.484436

Dorigo, M., & Stützle, T. (2004). Ant colony optimization. MIT Press. https://doi.org/10.7551/mitpress/1290.001.0001

Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study. In E. H. L. Aarts & J. K. Lenstra (Eds.), Local search in combinatorial optimization (pp. 215–310). John Wiley & Sons.

Liao, T., Socha, K., Montes de Oca, M. A., Stützle, T., & Dorigo, M. (2014). Ant colony optimization for mixed-variable optimization problems. IEEE Transactions on Evolutionary Computation, 18(4), 503–518. https://doi.org/10.1109/TEVC.2013.2281531

Lin, S. (1965). Computer solutions of the traveling salesman problem. The Bell System Technical Journal, 44(10), 2245–2269. https://doi.org/10.1002/j.1538-7305.1965.tb04146.x

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations Research, 21(2), 498–516. https://doi.org/10.1287/opre.21.2.498

López-Ibáñez, J., Dubois-Lacoste, J., Cáceres, P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43–58. https://doi.org/10.1016/j.orp.2016.09.002

Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial optimization: Algorithms and complexity. Courier Corporation.

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. ORSA Journal on Computing, 3(4), 376–384. https://doi.org/10.1287/ijoc.3.4.376

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. European Journal of Operational Research, 185(3), 1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046

Stützle, T., & Hoos, H. H. (2000). MAX–MIN ant system. Future Generation Computer Systems, 16(8), 889–914. https://doi.org/10.1016/S0167-739X(00)00043-1

Appendices

Appendix A: Algorithmic Pseudocode (Baseline ACO)

Initialize pheromone τ uniformly; set heuristic η = 1/d.Repeat for a fixed quantity of iterations:

a. For each ant, construct a tour by sampling next cities with probabilities $\propto \tau$^{α}η^{β}.

b. Evaluate each tour; update global best if improved.

c. Evaporate pheromone: $\tau \longleftarrow (1 - \rho)\tau$.

d. Deposit amount Q/L on edges of the selected best tour(s).Return the best so far tour.

Appendix B: Parameter Tuning Protocol (For Future Studies)

- Tune $\beta \in \{3,4,5,6\}$ and $\rho \in \{0.3,0.5,0.7\}$ on a small training set of synthetic instances; report final results on a held-out set.
- Use ants × iterations as a fixed budget; compare early stopping with fixed budget outcomes.
- Report median and interquartile range over $\geq$10 runs.

Appendix C: Benchmarking Checklist

- Clearly state instance generation process and distance metric.
- Record all ACO hyperparameters and budget.
- Provide at least one greedy or constructive baseline.
- Include convergence plots and final tour visualization.
- Avoid single instance over fitting; justify any tuning with held out evaluation.

Appendix D: City Coordinates

Table rows are labeled City1…City50 (1 based). In code/plots we use 0 based indices 0…49; Cityk ↔ index k−1 (e.g., City1 ↔ 0). The NN start at city 0 therefore equals City1 in this table.

City	X	Y	City	X	Y
City1	37.4540	95.0714	City2	73.1994	59.8658
City3	15.6019	15.5994	City4	5.8084	86.6176
City5	60.1115	70.8073	City6	2.0584	96.9909
City7	83.2443	21.2330	City8	18.1825	18.3407
City9	12.1344	15.7219	City10	41.6108	28.7403
City11	59.8671	13.5886	City12	26.5019	36.2227
City13	46.6177	78.1584	City14	14.4562	51.8622
City15	48.5456	4.0366	City16	58.5097	17.8110
City17	3.0206	89.1647	City18	96.3663	80.9037
City19	28.9799	11.4657	City20	67.1720	43.0536
City21	12.6401	49.1279	City22	3.8884	72.2373
City23	23.0851	66.6566	City24	33.1328	53.8345
City25	54.4883	18.9669	City26	96.0875	77.3845
City27	97.1570	90.1740	City28	60.8796	93.7451
City29	11.4954	19.9714	City30	0.6367	32.7402
City31	39.3922	27.9781	City32	83.7700	36.3610
City33	29.5285	54.8944	City34	10.4439	80.7174
City35	35.4049	98.8955	City36	79.1725	19.4179
City37	1.3689	81.1185	City38	71.8343	72.1566
City39	77.7833	7.1482	City40	32.2040	11.3154
City41	86.1310	62.8760	City42	31.3163	6.5145
City43	32.3140	32.8995	City44	70.4538	60.2718
City45	88.0029	47.1090	City46	13.9799	71.4373
City47	74.2404	56.6835	City48	76.0461	49.5838
City49	50.5357	43.2160	City50	2.0635	10.0725