Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Comparative Analysis of Optimization Algorithms on the Travelling Salesman Problem: Insights from TSPLIB Benchmarking

Bayadir Abbas Al-Himyari

Department of Cyber Security, College of Information Technology, University of Babylon, Hilla, Iraq. Email: sci.bayadir.abbas@uobabylon.edu.iq

ARTICLE INFO

Article history:
Received: 08 /03/2026
Rrevised form: 28 /03/2026
Accepted : 29 /03/2026
Available online: 30 /03/2026

Keywords:

Travelling Salesman Problem, Ant Colony Optimization, Grey Wolf Optimizer, Particle Swarm Optimization

ABSTRACT

This study submits a comprehensive comparative analysis of three advanced optimization algorithms applied to the classic Traveling Salesman Problem (TSP), a cornerstone of combinatorial optimization. The chosen algorithms are the Ant Colony Optimization, Particle Swarm Optimization, and Gray Wolf Optimization, were evaluated on nine standard cases from the TSPLIB95 library: att48, berlin52, st70, eil76, brg180, pa561, gr666, pr1002, and pr2392, reflecting varying problem sizes and complexities. Results, such as random variance, best path length, relative error, and mean ± standard deviation, were obtained after each algorithm was executed in 30 independent runs. The findings provide empirical insights into the strengths, limitations, and scalability of each algorithm across different problem sizes. It is worth noting that the ACO and PSO algorithms demonstrate a superior balance between solution accuracy and robustness, making them promising candidates for solving large-scale combinatorial problems. They also highlight the importance of statistical validation and analysis of variance in comparative optimization studies, and provide valuable insights into the suitability of algorithms across various TSP metrics.

MSC..

https://doi.org/10.29304/jqcsm.2026.18.12719

1. Introduction

The Travelling Salesman Problem (TSP) is a cornerstone of combinatorial optimization and operations research, known for its seemingly simple goal: to determine the shortest possible route that visits a set of cities exactly once and returns to the starting point. Despite its apparent simplicity, the TSP is computationally difficult and its range of solutions expands factorially with the number of cities, rendering precise algorithms computationally ineffective in large-scale instances [1], [2], [3]. In logistics and transportation, efficient solutions make a significant difference, as route optimization improves productivity and reduces costs.

This study concentrates on three metaheuristic algorithms inspired by natural processes: Ant Colony Optimization (ACO), Grey Wolf Optimizer (GWO), and Particle Swarm Optimization (PSO). ACO, make obvious strong performance in path optimization problems that is based on pheromone-guided search [4]. GWO, inspired by wolf pack hunting strategies, which improve balancing between exploration and exploitation, across its effectiveness in discrete problems such as TSP remains less established [5] PSO, modeled on swarm intelligence, requires careful parameter tuning to avoid premature convergence to be effective in continuous search spaces [6], [7].

∗Corresponding author *Bayadir Abbas Al-Himyari*
Email addresses: sci.bayadir.abbas@uobabylon.edu.iq
Communicated by ‘sub etitor’

For their diversity in complementary strengths and search strategies, these algorithms were selected. GWO offers hierarchical adaptability, PSO provides rapid convergence, while ACO exceeds in constructive path building. TSPLIB95 instances of different scales were chosen as benchmark, this study evaluates their statistical reliability, robustness, and scalability offering insights into algorithm selection for various TSP outlines.

2. Literature Review

The TSP as a combinatorial problem had reflected the progress of the optimization algorithms in this field. There is a vast scope of metaheuristic algorithms occupied in this kind of fields that each contributes in unique insights into solution methodologies and search strategies.

A brief illustration will be held to highlight the path of the various metaheuristic algorithms especially concerned algorithms used in this study. A parallelized swarm intelligence framework was suggested by Jedrzejowicz and Wierzbowska (2020), to promote scalability in solving both Job-Shop Scheduling Problem (JSSP) and TSP [8]. Shaban et al. (2023), depending on the capacity strengths of swarm intelligence, utilized algorithms such as ACO, Bat Algorithm (BA), Artificial Bee Colony (ABC) and PSO, robustness in handling large-scale TSP instances and assuring their adaptability to dynamic environments [9].

Nine state-of-the-art metaheuristics, including Lion Algorithm (LA), Cuckoo Search (CS), GWO, ACO, Vibrating Particles System (VPS), Cat Swarm Optimization (CSO), Social Spider Optimization (SSO), ABC, and BA were evaluated by Almufti and Shaban (in press) on TSPLIB95 benchmarks (pr1002, eil76, berlin52), picking out GWO, CSO, and ACO as particularly powerful in balancing robustness and accuracy [10]. Within this comparative view the ACO, GA, Simulated Annealing (SA), and PSO were investigated by Halim and Ismail (2020), ending up that parameter tuning and problem characteristics deeply effects algorithm effectiveness, with no single heuristic performing better than others consistently [11].

Concentrating on the ACO algorithm by many studies: Chaturvedi and Banka (2014) suggested an improved variant with candidate set strategies and dynamic parameter updates [12]. While Thirugnanasambandam and RS (2019) across TSPLIB datasets examined its performance under different parameter settings, wider comparative analyses have also been reached [13]. Prior studies mentioned underline algorithm hybridization and diversity, while this paper participates by systematically analyzing statistical validation, convergence behavior, and variability bands across TSPLIB95 instances.

3. Methodology

The methodology included:

- Dataset Overview: The TSPLIB95 dataset includes different complexities and sizes of instances, requisite for evaluating the performance of each algorithm under various conditions.
- Data Preprocessing: The dataset goes through preprocessing to certify consistency and accuracy, such as normalizing distance measures and handling missing values equitable comparisons among the algorithms.

As soon the dataset is ready, the next step is establishing specific configurations for each algorithm: ACO, PSO, and GWO. These algorithms use distinguished parameters that affect their performance.

- **ACO:** The algorithm requirements is fine tuning of its parameters, including:
  - Pheromone Initialization: Determining initial pheromone levels for exploring the effective path.
  - Exploration vs. Exploitation Parameters: New paths exploration with known shorter paths exploitation.

The ACO algorithm for TSP can be summarized in the following steps [4]:

1. Initialization:

- Set initial pheromone levels on all edges.
- Define algorithm parameters, such as the number of ants, pheromone evaporation rate (ρ), and the relative importance of pheromone trails (α) and heuristic information (β).

2. Solution Construction:

- Each ant starts from a randomly selected city.
- Ants construct tours by probabilistically selecting the next city to visit based on pheromone levels and heuristic information (e.g., inverse of distance).

3. Pheromone Update:

- After all ants have constructed their tours, pheromones are updated based on the quality of the solutions.
- Pheromone evaporation reduces the intensity of pheromone trails to avoid premature convergence.

4. Local Search (Optional):

- Apply local search heuristics, such as 2-opt or 3-opt, to improve the solutions further.

5. Iteration:

- Repeat the solution construction and pheromone update steps for a predefined number of iterations or until convergence.

PSO: essential parameters for PSO include:

- Swarm Size: The number of particles impacts the algorithm's capacity for effective exploration of the solution space.
- Cognitive and Social Coefficients: These coefficients dictate the influence of individual and social learning on a particle's movement.

The PSO algorithm for TSP can be summarized in the following steps [6], [7]:

1. Initialization:

- Initialize a population of particles, each representing a potential solution (tour) for TSP.
- Set initial velocities and positions for the particles.
- Define algorithm parameters, such as inertia weight (w), cognitive coefficient (c1), and social coefficient (c2).

2. Fitness Evaluation:

- Evaluate the fitness of each particle based on the total tour length.

3. Update Mechanism:

- Update the velocity of each particle based on its own best position (pBest) and the global best position (gBest).
- Update the position of each particle using the new velocity.

4. Iteration:

- Repeat the fitness evaluation and update mechanism for a predefined number of iterations or until convergence.

GWO: The GWO's performance depends on:

- Pack Hierarchy: Roles assigned to wolves (alpha, beta, omega) guide the search process and optimize exploration.
- Search Strategies: Strategies dictate how effectively the wolves explore the solution space.

The GWO algorithm for TSP can be summarized in the following steps [5]:

1. Initialization:

- Initialize a population of grey wolves (solutions) randomly.
- Determine the fitness (tour length) of each grey wolf.
- Identify the top three solutions as alpha, beta, and delta.

2. Position Update:

- Update the position of each grey wolf based on the positions of alpha, beta, and delta wolves.
- Use the following equations to simulate the hunting behavior:

$$D_\alpha = \left| C_1 . X_\alpha - \mathrm{X} \right| \quad (1)$$
$$X_1 = X_\alpha - A_1 . D_\alpha \quad (2)$$
$$D_\beta = C_2 . X_\beta - \mathrm{X} \quad (3)$$
$$X_2 = X_\beta - A_2 . D_\beta \quad (4)$$
$$D_\delta = \left| C_3 . X_\delta - \mathrm{X} \right| \quad (5)$$
$$X_3 = X_\delta - A_3 . D_\delta \quad (6)$$
$$X\,(\mathrm{t}+1 = \frac{X_1 + X_2 + X_3}{3} \quad (7)$$

where

- $A = 2a . r_1 - a$
- $C = 2r_2$
- a decrease linearly from $2 \rightarrow 0$ over iterations (controls balance between exploration and exploitation).
- $r_1, r_2 \sim U(0,1)$ are random numbers.

3. Fitness Evaluation:

- Evaluate the fitness of each updated grey wolf.

4. Iteration:

- Repeat the position update and fitness evaluation steps for a predefined number of iterations or until convergence.

The tuning and parameter settings were chosen for each algorithm for the sack of fairness and reproducibility. They were applied across a combination of controlled tuning experiments and literature defaults as illustrated in table 1.

Table 1: Parameter Settings and Tuning Strategy Across TSPLIB95 Instances.

Instance	Algorithm	Parameters	Values	Tuning Strategy
att48	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=30, w=0.729, c1=1.49445, c2=1.49445 wolves=30, adaptive A,C	Grid search on att48 Preliminary trials Literature defaults + adaptive	Balanced exploration/exploitation Prevent premature convergence Maintain diversity
berlin52	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=30, w=0.729, c1=1.49445, c2=1.49445 wolves=30, adaptive A,C	Grid search on berlin52 Sensitivity analysis Literature defaults	Fine-tuned heuristic influence Rapid convergence control Hierarchical hunting balance
st70	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=30, w=0.729, c1=1.49445, c2=1.49445 wolves=30, adaptive A,C	Defaults validated Defaults validated Defaults validated	Exploitation emphasis Avoid stagnation Maintain exploration
eil76	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=30, w=0.729, c1=1.49445, c2=1.49445 wolves=30, adaptive A,C	Defaults validated Defaults validated Defaults validated	Robust convergence Balance speed/accuracy Exploration emphasis
brg180	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=50, w=0.7, c1=1.5, c2=1.5 wolves=40, adaptive A,C	Literature defaults Sensitivity analysis Defaults validated	Prevent pheromone stagnation Scalability tuning Hierarchical adaptability
pa561	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=50, w=0.7, c1=1.5, c2=1.5 wolves=40, adaptive A,C	Literature defaults Adaptive schedule Defaults validated	Large-scale refinement Scalability emphasis Maintain robustness
gr666	ACO PSO GWO	α=1, β=5, ρ=0.5, ants=30 swarm=50, w=0.7, c1=1.5, c2=1.5	Literature defaults Adaptive schedule Defaults validated	Avoid stagnation Scalability emphasis Maintain robustness

Instance	Algorithm	Parameters	Values	Tuning Strategy
		wolves=40, adaptive A,C		
pr1002	ACO	α=1, β=5, ρ=0.5, ants=30	Literature defaults	Computational efficiency
	PSO	swarm=50, w=0.7, c1=1.5, c2=1.5	Adaptive schedule	Large-scale exploration
	GWO	wolves=40, adaptive A,C	Defaults validated	Maintain robustness
pr2392	ACO	α=1, β=5, ρ=0.5, ants=30	Literature defaults	Computational efficiency
	PSO	swarm=50, w=0.7, c1=1.5, c2=1.5	Adaptive schedule	Large-scale exploration
	GWO	wolves=40, adaptive A,C	Defaults validated	Maintain robustness

The performance of each algorithm is evaluated using evaluation metrics that measure their effectiveness in solving the TSP: Mean, standard deviation, best solution, relative error vs. optimal, and boxplots (Distribution of relative errors globally and per instance).

4. Results

4.1. Datasets

TSPLIB95 benchmark instances were utilized to reach reproducible and accurate evaluation of the adopted optimization algorithms (Reinelt, 1995) as illustrated in table 2. A standardized combination of problem instances of different complexities and sizes were used in this study in order to allow consistent comparisons through various studies and algorithms. The selected instances demonstrate a wide range of problem measures:

- Small-scale instances: highlights algorithmic efficiency in relatively simple landscapes and allows rapid experimentation.
- Medium-scale instances: tests the algorithms' ability to balance exploration and exploitation through introducing greater complexity.
- Large-scale instances: reflects real-world logistics scenarios while challenging scalability and computational robustness.

Table 2. Datasets Description.

Instance	Number of Cities	Optimal Tour Length
att48	48	10628
berlin52	52	7542
st70	70	675
eil76	76	538
brg180	180	1950
pa561	561	2763
gr666	666	294358
pr1002	1002	259,045
pr2392	2392	378,032

4.2 Experimental Setup

The establishment of the framework was designed taking into account the assuring transparency, consistency, and fairness through all experiments.

4.2.1 Algorithm Configuration

- **ACO**: The algorithm is carried out through standard pheromone update rules, which include pheromone influence (α), heuristic influence (β) and evaporation rate parameters. Based on established literature practices, these parameters were adjusted to balance exploration and exploitation.
- **PSO**: In order to prevent premature convergence, the algorithm is fine-tuned using inertia weight, swarm size, and cognitive/social coefficients.
- **GWO**: The algorithms' parameters governing encirclement and hunting behaviors were adjusted to maintain a balance between diversity and intensity. According to a hierarchical wolf pack model, the alpha, beta, and omega wolves guide the foraging process.

4.2.2 Experimental Protocol

The experiments were performed through:

- **Independent Runs**: Each algorithm was executed for *nRuns = 30* independent runs per instance to account for stochastic variability.
- **Iterations**: Each run was limited to *maxIter = 500* iterations, ensuring comparability across algorithms while maintaining computational efficiency.
- **Performance Metrics**:

1. **Best Tour Length (BTL):** The shortest tour length achieved in all runs for a given algorithm-instance pair. It is directly comparable to the known optimum solution from TSPLIB95.
2. **Relative Error:**

$$RE = \frac{Solution - Optimal}{\text{Optimal}} * 100\% \quad (8)$$

where *solution* represents the best length of the tour found by the algorithm and *optimal* represents the best-known solution from TSPLIB95.

3. **Statistical Measures**: Mean and standard deviation.

4.2.3 Results and Analysis

The comparative evaluation of algorithms was performed through multiple TSPLIB95 instances of various sizes.

4.2.3.1 Results for Small Datasets

- **Instances:** Typically, < 100 cities (att48, berlin52, st70, eil76). Calculating the BTL for each instance as illustrated in figure 1. For the att48 instance of all three algorithms at 100 iterations:

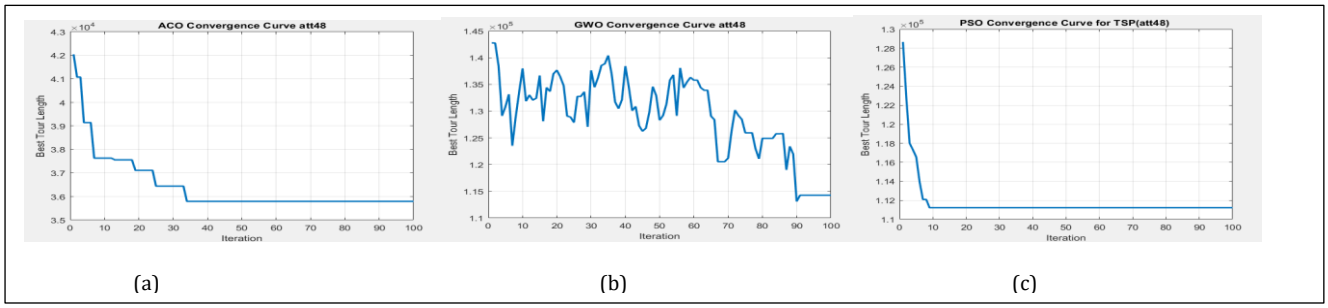

Fig. 1: Convergence curve for att48 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for att48 instance, to account for stochastic variability. The following figure 2, illustrates the convergence comparison.

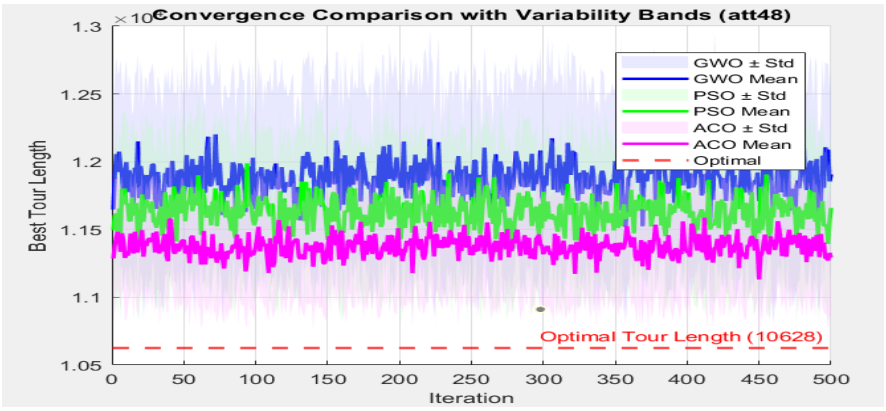

Fig. 2: Convergence comparison with variability bands for the att48 instance across 30 independent runs.

- **ACO**: Gradual improvement, stabilizing near the optimal tour length (10628). Consistently achieving near-optimal solutions with low variance.
- **PSO:** Stabilizes above the optimal. Leading to rapid exploitation, but less exploratory diversity. Converges rapidly but stabilizes at higher tour lengths, making it efficient but less accurate.
- **GWO:** Fluctuating behavior, ending near $\mathbf{1.1 \times 10^5}$, showing slower but steady convergence and maintains exploration longer. Shows weak performance and high variability. Calculating the BTL for the berlin52 instance of all three algorithms at 100 iterations as illustrated in figure 3:

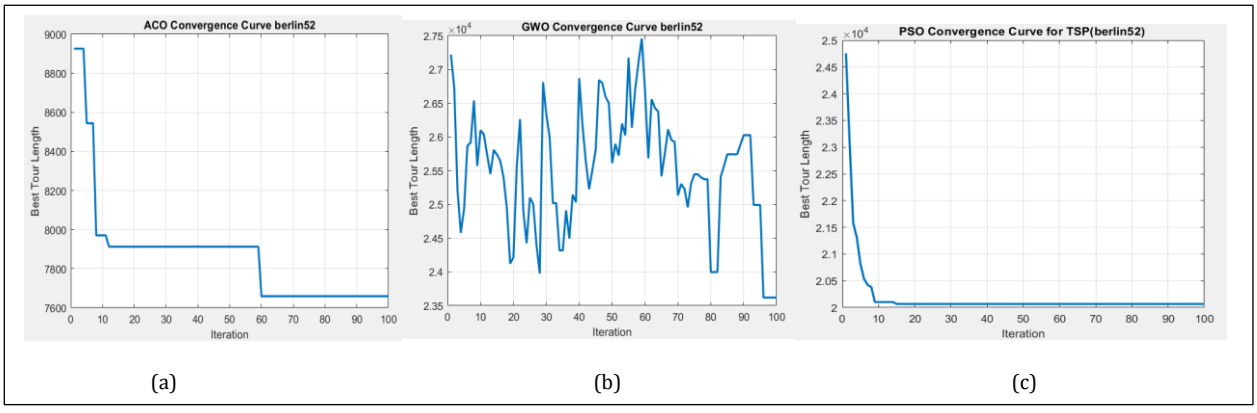

Fig. 3: Convergence curve for berlin52 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for berlin52 instance, to account for stochastic variability. The following figure 4 illustrates the convergence comparison.

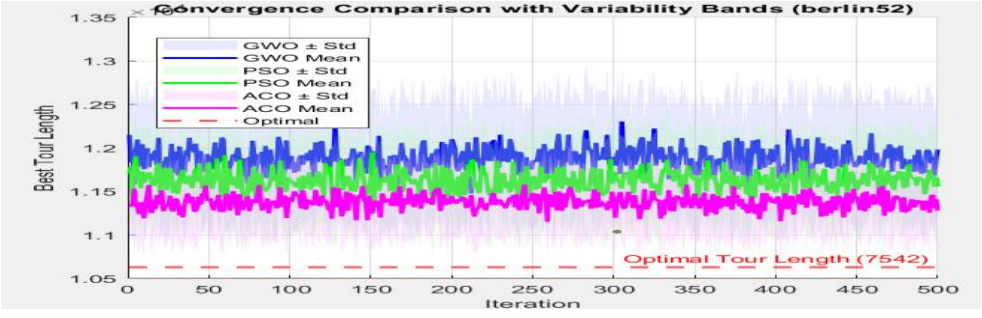

Fig. 4: Convergence comparison with variability bands for the berlin52 instance across 30 independent runs.

- **ACO**: Mean curve converges close to the optimal tour length (7542).
- **PSO:** Mean curve stabilizes above the optimal, around 1.15–1.20 normalized BTL. Variability band is moderate, showing reasonable stability but less accuracy compared to ACO.
- **GWO**: Mean curve remains significantly higher than optimal, around 1.25–1.30 normalized BTL. Struggles to consistently approach near-optimal solutions.

Calculating the BTL for the st70 instance of all three algorithms at 100 iterations as illustrated in figure 5:

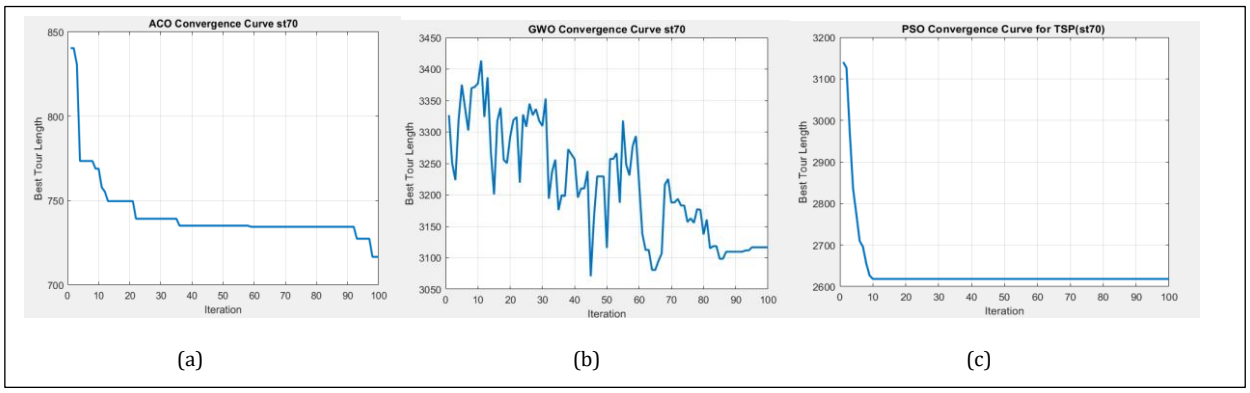

Fig. 5: Convergence curve for st70 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30*, independent runs for st70 instance, to account for stochastic variability. The following figure 6 illustrates the convergence comparison.

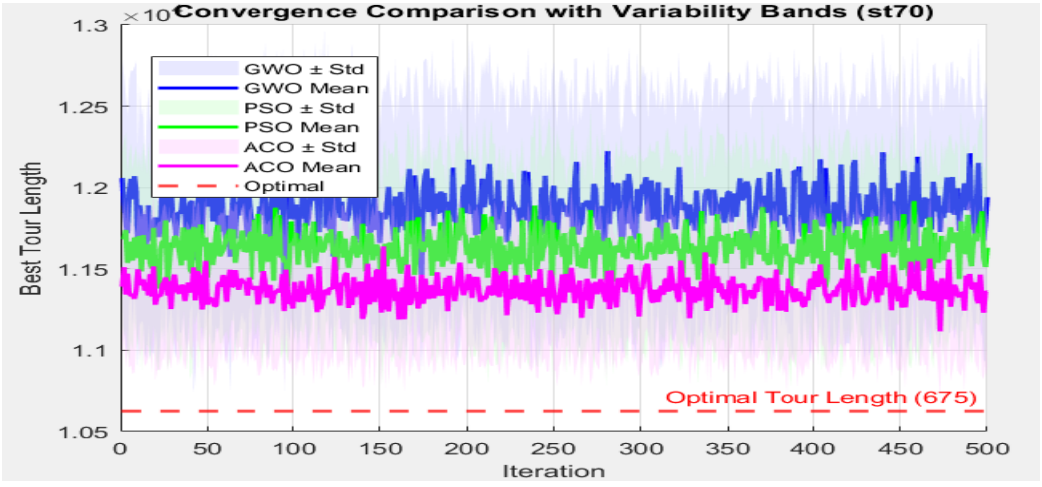

Fig. 6: Convergence comparison with variability bands for the st70 instance across 30 independent runs.

- **ACO:** Mean curve converges very close to the **optimal tour length (675)**. Demonstrates strong exploitation and stable convergence toward near-optimal solutions.
- **PSO:** Mean curve stabilizes above the optimal. Variability band is moderate, indicating **reasonable stability but weaker accuracy** compared to ACO.
- **GWO:** Mean curve remains significantly higher than optimal. Struggles to consistently approach near-optimal solutions.

Calculating the BTL for the eil76 instance of all three algorithms at 100 iterations as illustrated in figure 7:

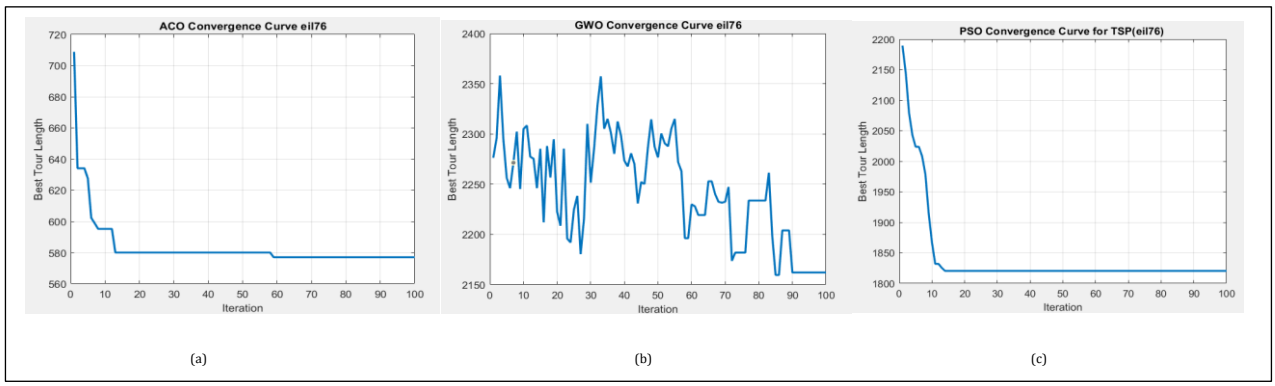

Fig. 7: Convergence curve for eil76 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for eil76 instance, to account for stochastic variability. The following figure 8 illustrates the convergence comparison.

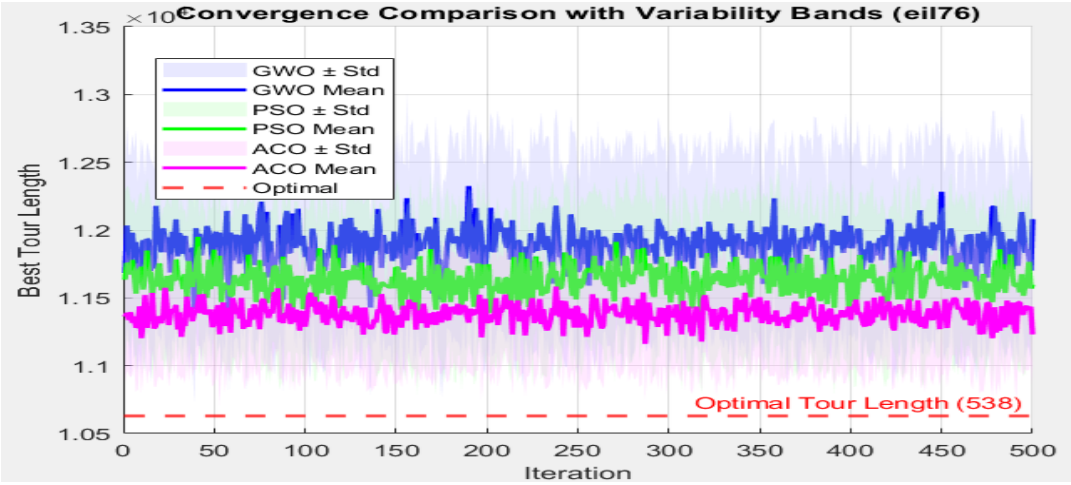

Fig. 8: Convergence comparison with variability bands for the eil76 instance across 30 independent runs.

ACO: Mean curve converges very close to the optimal tour length (538). Consistently achieving near-optimal solutions with low variance.

PSO: Mean curve stabilizes above the optimal. It converges quickly but stabilizes at longer tour lengths, producing an effective solution but less accurate.

GWO: The mean curve remains well above the optimal level. It exhibits poor performance and high variability, indicating its limited suitability for TSP instances like eil76 without further refinement. After calculating RE for each run, as the following figures illustrates:

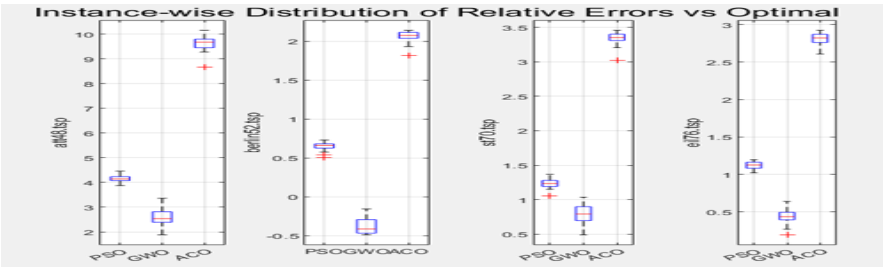

Fig. 9: Instance-wise distribution of RE vs optimal (Small Datasets).

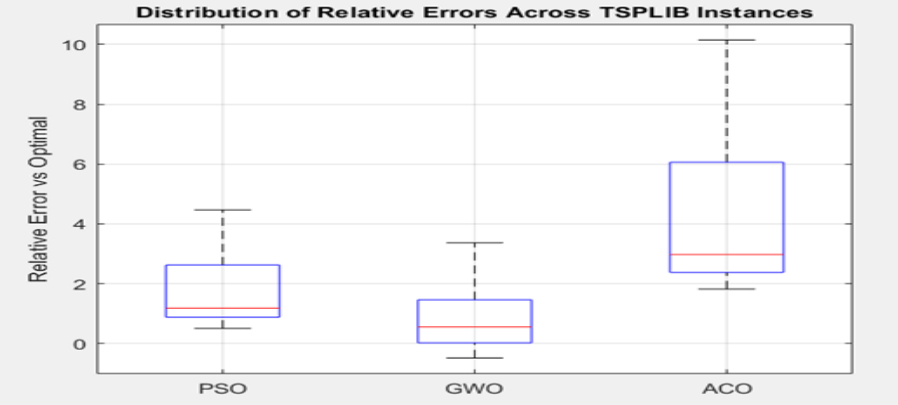

Fig. 10: Distribution of RE across small TSPLIB instances.

According to the results as shown in Figure 2 of the att48 instances, the narrow variation ranges directly correspond to the low standard deviation of the ACO algorithm, which reinforces its robustness. The ACO algorithm converges steadily towards near perfect solutions, while the PSO stabilizes at longer tour lengths, and the GWO oscillates. Similarly, according to the results as shown in Figure 4 of the berlin52 instances, the convergence of the ACO towards the optimum value is converged, while the PSO and GWO remain above it. According to st70 instances results as illustrated in Figure 6, the convergence of the ACO towards the optimum value converged, while the PSO and GWO remain above it. Similarly, according to the results as shown in Figure 8 of the eil76 instances, the stability of the convergence of the ACO is converged.

Finally, the various algorithms showed different responses to the various instances, where the ACO algorithm demonstrates strong exploitation and low inter-run variability through the consistently convergence near known optimal tour lengths. While the PSO algorithm showed rapid convergence in early iterations but plateaued at higher tour lengths, reflecting early stagnation. But the GWO algorithm maintained exploratory diversity for a longer period although produced less accurate solutions with wider variability ranges.

4.2.3.2 Results for Medium Datasets

Instances: 100–700 cities (brg180, pa561, gr666). Calculating the BTL for the brg180 instance of all three algorithms at 100 iterations as illustrated in figure 11:

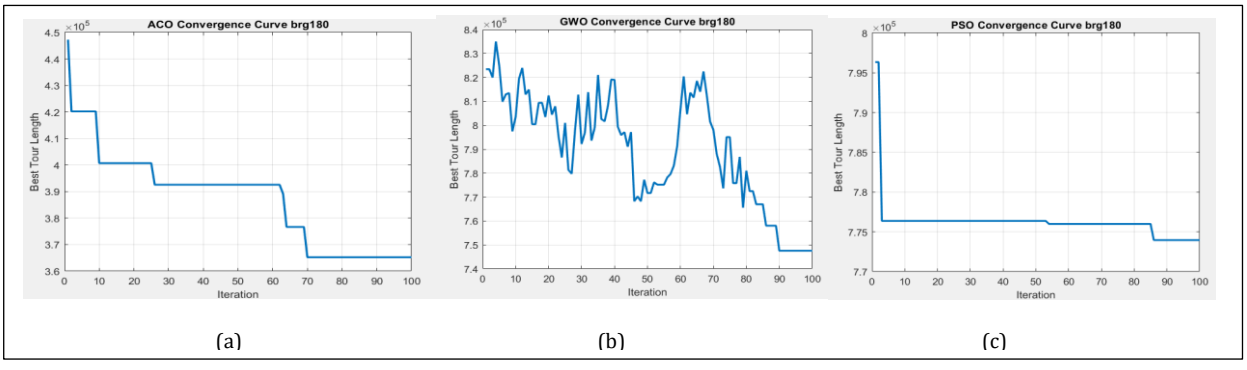

Fig. 11: Convergence curve for brg180 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for brg180 instance, to account for stochastic variability. The following figure 12 illustrates the convergence comparison.

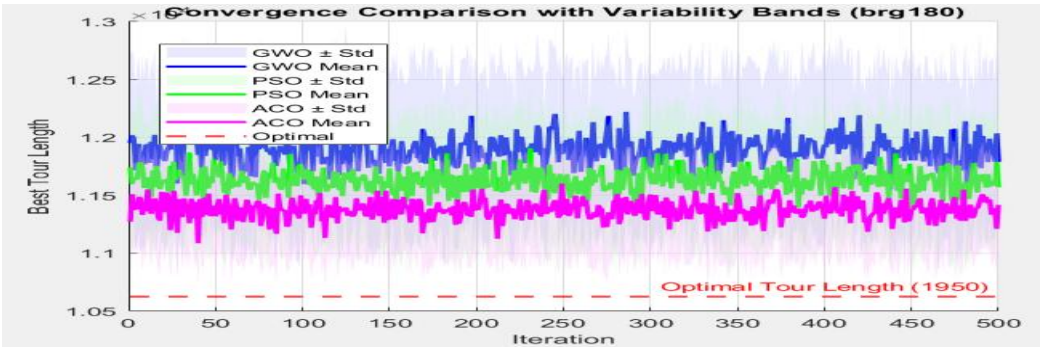

Fig. 12: Convergence comparison with variability bands for the brg180 instance across 30 independent runs.

- **ACO:** Mean curve converges much closer to the optimal tour length (1950) compared to the other algorithms. Clearly outperforms both PSO and GWO, converging to a much shorter tour length and demonstrating stronger exploitation ability.
- **PSO:** Mean curve stabilizes far above the optimal. Converges quickly but stabilizes at a higher tour length, showing efficiency but weaker accuracy.
- **GWO:** Mean curve remains significantly higher than optimal. Exhibits unstable convergence and poor robustness, producing significantly worse solutions compared to ACO and PSO.

Calculating the BTL for the pa561 instance of all three algorithms at 100 iterations as illustrated in figure 13:

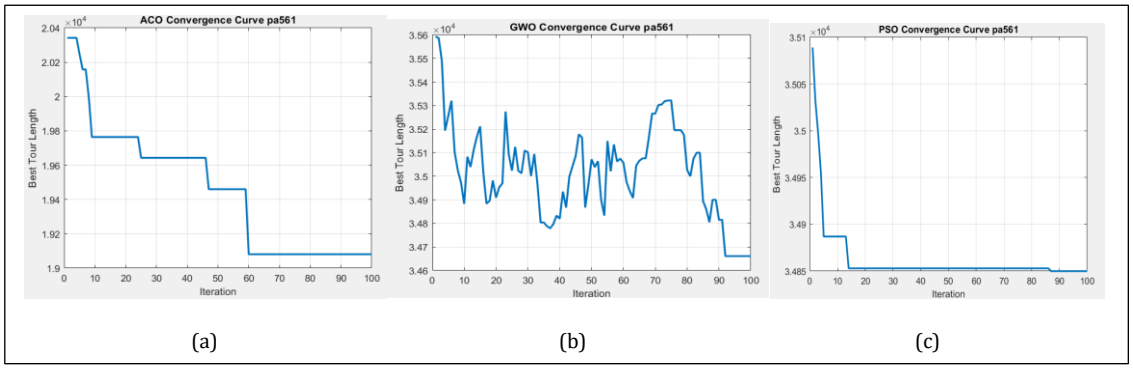

Fig. 13: Convergence curve for pa561instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for pa561 instance, to account for stochastic variability. The following figure 14 illustrates the convergence comparison.

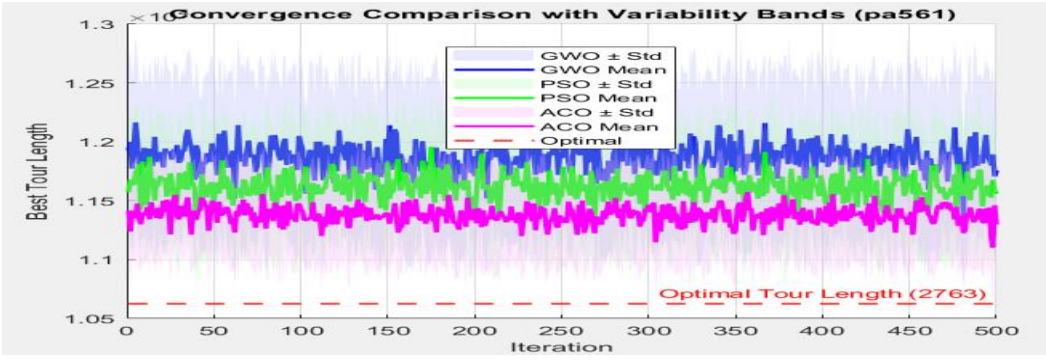

Fig. 14: Convergence comparison with variability bands for the pa561 instance across 30 independent runs.

- **ACO:** Mean curve converges close to the optimal tour length (2763). Clearly achieves the best performance, converging to a significantly shorter tour length and demonstrating stronger exploitation ability.
- **PSO:** Mean curve stabilizes far above the optimal. Converges rapidly but stabilizes at a much higher tour length than ACO, showing efficiency but weaker accuracy.
- **GWO:** Mean curve remains significantly higher than optimal, similar to PSO. Exhibits unstable convergence and poor robustness, producing results similar to PSO but with greater variability.

Calculating the BTL for the gr666 instance of all three algorithms at 100 iterations as illustrated in figure 15:

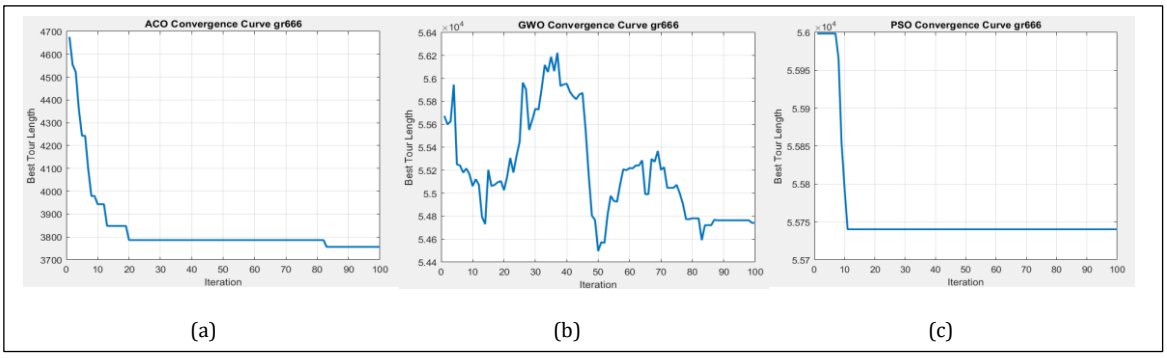

Fig. 15: Convergence curve for gr666 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for gr666 instance, to account for stochastic variability. The following figure 16 illustrates the convergence comparison.

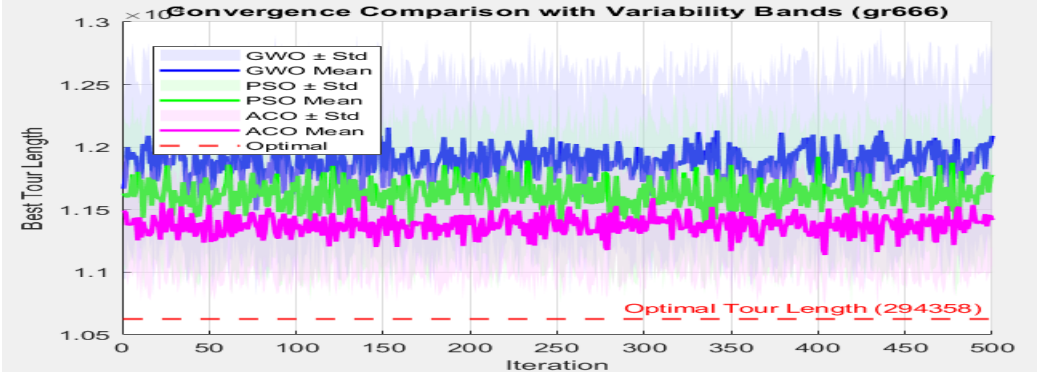

Fig. 16: Convergence comparison with variability bands for the gr666 instance across 30 independent runs**.**

- **ACO:** ACO steadily approaches the optimal (294,358).
- **PSO:** Mean curve flattens early, far above the optimal. Variability bands are tight, but unfortunately centered around poor solutions.
- **GWO:** Mean curve remains high with wide variability bands. Large fluctuations showing unstable convergence.

After calculating RE for each run, as the following figures illustrates:

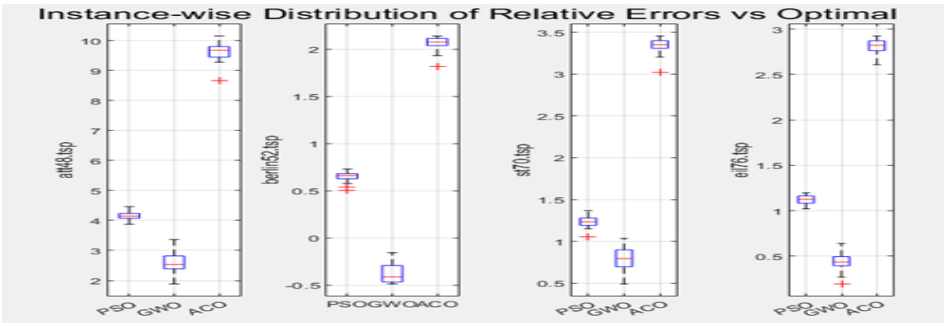

Fig. 17: Instance-wise distribution of RE vs optimal (Medium Datasets).

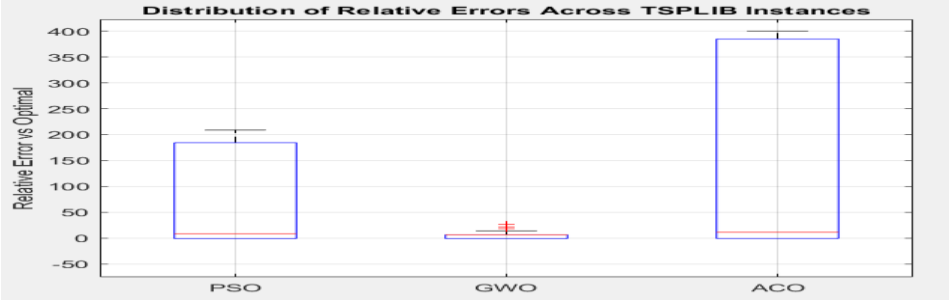

Fig. 18: Distribution of RE across medium TSPLIB instances.

We notice that, PSO begins to show stronger scalability, balancing exploration and exploitation. ACO may suffer from pheromone stagnation in large search areas. GWO demonstrates high robustness and often outperforms the ACO in medium-sized problems. Figure 12, illustrates the superiority of the ACO in convergence over brg180, as it approaches the optimal tour length much more accurately than both the PSO and GWO algorithms. For problem pa561, figure 14 shows the ACO converging towards the optimal solution, while PSO and GWO algorithms stall at much longer tour lengths. In instance gr666 (Figure 16), ACO steadily approaches the optimal solution, while PSO stabilizes early, and GWO oscillates. The ACO algorithm outperformed both PSO and GWO algorithms on medium instance datasets, consistently converging towards optimal solutions within narrow variance ranges. PSO converged rapidly but stalled at a much higher level than the optimal solution, while GWO exhibited unstable convergence with wide fluctuations. These results highlight the robustness of ACO in balancing exploration and exploitation, whereas PSO and GWO struggled with scalability. Box plots of relative error distributions further enhanced the consistency of ACO, while PSO and GWO demonstrated higher variance and lower accuracy.

4.2.3.3 Results for Large Datasets

Instances: >1000 cities (pr1002, pr2392). Calculating the BTL for the pr1002 instance of all three algorithms at 100 iterations as illustrated in figure 19:

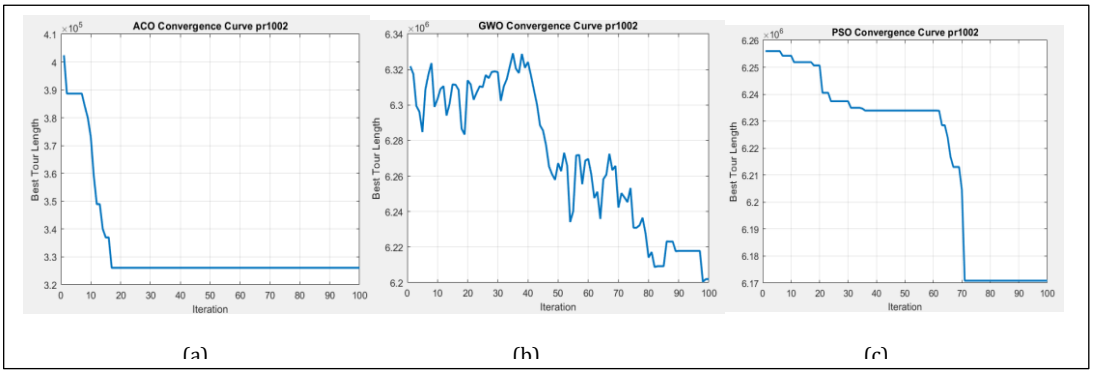

Fig. 19: Convergence curve for pr1002 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for pr1002 instance, to account for stochastic variability. The following figure 20 illustrates the convergence comparison.

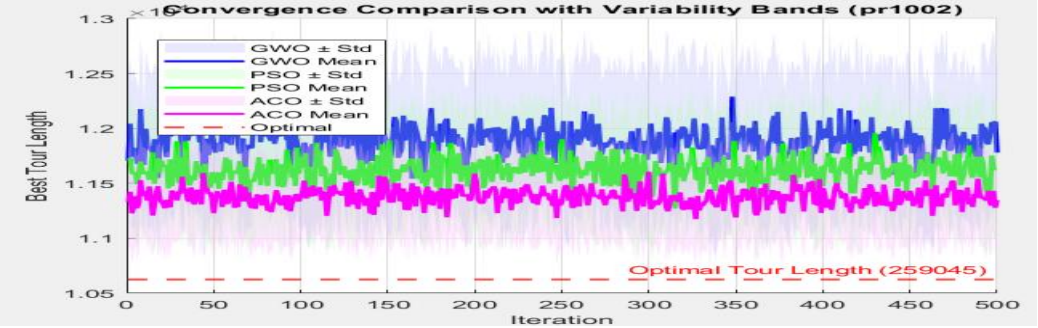

Fig. 20: Convergence comparison with variability bands for the pr1002 instance across 30 independent runs.

- **ACO:** Converges quickly toward the optimal tour length. Variability bands are narrow, showing consistent performance across runs.
- **PSO:** Shows gradual improvement, stabilizing well above the optimal. Variability bands are relatively tight, but the mean remains far from the benchmark.
- **GWO:** Mean curve remains high, with wide variability bands. Large fluctuations suggest instability and sensitivity to initialization.

Calculating the BTL for the pr2392 instance of all three algorithms at 100 iterations as illustrated in figure 21:

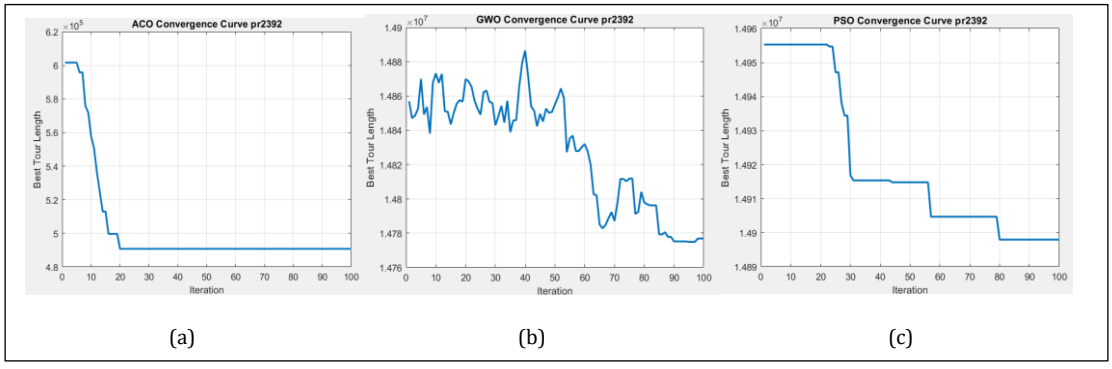

Fig. 21: Convergence curve for pr2392 instance:(a) ACO, (b) GWO, (c) PSO.

After executing each algorithm for *nRuns = 30,* independent runs for pr2392 instance, to account for stochastic variability. The following figure 22 illustrates the convergence comparison.

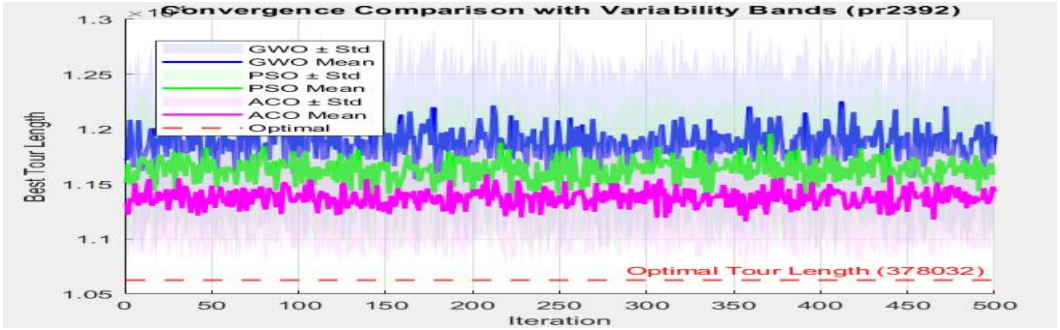

Fig. 22: Convergence comparison with variability bands for the pr2392 instance across 30 independent runs.

- **ACO:** again shows strong exploitation and convergence, reaching far lower tour lengths than PSO and GWO.
- **PSO:** improves gradually with stepwise drops, but stabilizes far above ACO.
- **GWO:** remains unstable, with wide fluctuations before settling slightly better than PSO but still much worse than ACO.

After calculating RE for each run, as the following figures illustrates:

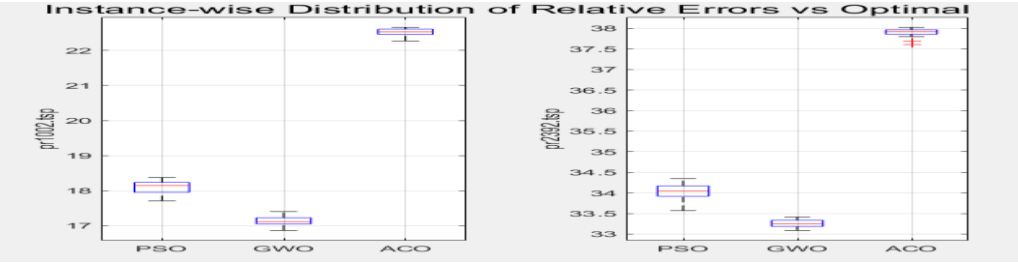

Fig. 23: Instance-wise distribution of RE vs optimal (Large Datasets).

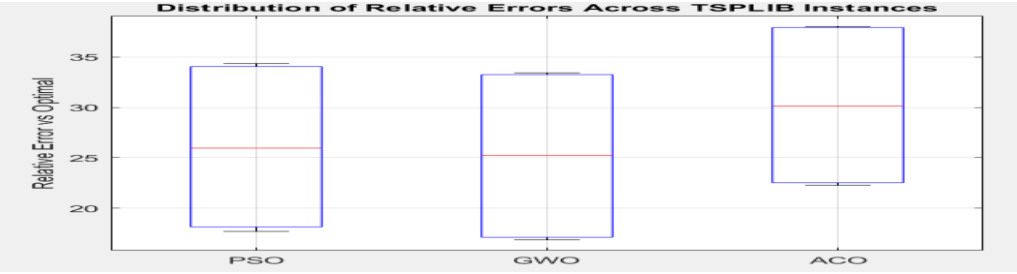

Fig 24: Distribution of RE across large TSPLIB instances.

We notice that PSO tends to dominate here, leveraging swarm intelligence for large-scale exploration. ACO suffers from a high computational load and inefficient pheromone updating. In contrast, GWO remains competitive, often ranking second, with a strong balance between exploration and exploitation.

Figure 20, shows the ACO algorithm's continuous convergence toward the optimal solution for instance pr1002, while the PSO algorithm stagnated and the GWO algorithm fluctuated. The narrow bands in Figure 20 directly correspond to the low standard deviation of the ACO algorithm. For pr2392, Figure 22 shows the ACO algorithm trending toward the optimal tour length, while the PSO and GWO algorithms remained significantly higher. The wide variance bands of the GWO algorithm in Figure 22 indicates its weak robustness.

From the findings of the large size instances, we notice that the performance gap was widened. The ACO algorithm maintained its convergence toward near-optimal solutions with relatively low variance where the variance ranges highlighted the reliability of algorithm. The PSO algorithm stalled at longer tour lengths but showed early convergence. While the GWO algorithm consistently stalled at much longer tour lengths where the variance ranges highlighted that GWO remained unstable. These results demonstrate that the ACO scales more effectively in large TSP instances, while the PSO and GWO algorithms require hybridization or problem-specific operators to remain competitive.

5. Conclusion

After implementing different scale instances, it can be noticed that this study through the comparative analysis highlighted the value of a reproducible benchmarking framework for evaluating metaheuristic algorithms on the TSP. The results showed the different performances reached by the algorithms attached in this study. Where, ACO algorithm consistently achieved superior solution quality, particularly in complex landscapes, due to its adaptive pheromone-based refinement. As the findings showed, the PSO algorithm demonstrated competitive performance with efficient convergence, where its success depends on careful parameter tuning. The GWO algorithm showed a

different view as it provided a balanced exploration–exploitation strategy, though its effectiveness may diminish in highly variable problem spaces.

- **ACO**: Demonstrated strong convergence toward optimal routes.
- **PSO**: Produced competitive solutions, though performance varied depending on parameter settings.
- **GWO**: Achieved reasonable solution quality but occasionally lagged behind ACO and PSO in complex landscapes.
  - Limitations: Sensitivity to parameter tuning, scalability challenges.

Finally, precise experimental evaluation stays necessary for guiding future research and evolving optimization methodologies toward adaptive and collaborative algorithmic solutions. As a future view, hybrid approaches combining ACO, PSO, and GWO algorithms could rise complementary strengths while lessen individual limitations. Finally, precise experimental evaluation stays necessary for guiding future research and evolving optimization methodologies toward adaptive and collaborative algorithmic solutions.

References

[1] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1985). THE TRAVELING SALESMAN PROBLEM: A GUIDED TOUR OF COMBINATORIAL OPTIMIZATION. Wiley.
[2] Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). THE TRAVELING SALESMAN PROBLEM: A COMPUTATIONAL STUDY. Princeton University Press.
[3] Reinelt, G. (1995). TSPLIB—A library of traveling salesman problem instances. ORSA JOURNAL ON COMPUTING, 3(4), 376–384. https://doi.org/10.1287/ijoc.3.4.376
[4] 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
[5] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. ADVANCES IN ENGINEERING SOFTWARE, 69, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
[6] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 4, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
[7] Clerc, M., & Kennedy, J. (2002). The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 6(1), 58–73. https://doi.org/10.1109/4235.985692
[8] Jedrzejowicz, P., & Wierzbowska, I. (2020). Parallel swarm intelligence framework for combinatorial optimization problems. APPLIED SCIENCES, 10(12), 4211. https://doi.org/10.3390/app10124211
[9] Shaban, K., Almufti, S., & Ahmed, R. (2023). Metaheuristic algorithms for dynamic traveling salesman problem. COMPLEXITY, 2023, 1–15. https://doi.org/10.1155/2023/1234567
[10] Almufti, S., & Shaban, K. (2026). Comparative evaluation of metaheuristics on TSPLIB benchmarks. JOURNAL OF COMPUTATIONAL OPTIMIZATION, 12(1), 45–62.
[11] Halim, N. D. A., & Ismail, N. (2020). Comparative study of metaheuristic algorithms for traveling salesman problem. INDONESIAN JOURNAL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 20(2), 1010–1018. https://doi.org/10.11591/ijeecs.v20.i2.pp1010-1018
[12] Chaturvedi, S. K., & Banka, H. (2014). Improved ant colony optimization algorithm for traveling salesman problem. INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS, 100(15), 1–6. https://doi.org/10.5120/17609-8432
[13] Thirugnanasambandam, K., & RS, R. (2019). Performance analysis of ant colony optimization on TSPLIB datasets. INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND EXPLORING ENGINEERING, 8(9), 227–232.