Final Report: Comparison and Tuning of GA vs ACO on the TSP (TSPLIB) + PLUS Extension: Chu-Beasley Genetic Algorithm (CBGA)

1. Problem Statement and TSPLIB

The Traveling Salesperson Problem (TSP) seeks to find a minimum-cost Hamiltonian cycle that visits a set of nodes exactly once and returns to the origin. For this study, four instances from the TSPLIB library were selected, recording their known optima for the GAP calculation:

- **berlin52:** 52 nodes (Optimum: 7542)
- **eil51:** 51 nodes (Optimum: 426)
- **att48:** 48 nodes (Optimum: 10628)
- **st70:** 70 nodes (Optimum: 675)

2. Experimental Methodology

- **Computational Budget:** A limit based on equivalent iterations was established for each algorithm (500 for GA/CBGA, 100 for ACO with 30 ants).
- **Repetitions (Seeds):** For reasons of computational efficiency and time, **5 seeds** (42 to 46) were executed per algorithm and instance in the base phase, and 3 seeds in the tuning phase.
- **Evaluated Metrics:** Best distance, average distance, standard deviation, average execution time, and percentage GAP with respect to the known optimum.

3. Implementation and Base Parameters

Three metaheuristic algorithms were implemented (in addition to a Random Search or *Dummy* as a baseline). The design decisions and initial parameters were:

- **Classic Genetic Algorithm (GA):**
  - Population size (P): 100
  - Crossover probability (pc): 0.85 (OX Crossover)
  - Mutation probability (pm): 0.15 (Inversion Mutation)
  - Selection: Tournament (k=3)
  - Elitism: 2 individuals
- **Ant Colony Optimization (ACO):**
  - Number of ants (m): 30
  - Pheromone weight (α): 1.0
  - Heuristic weight (β): 2.0
  - Evaporation (ρ): 0.1
  - Deposit intensity (Q): 100.0
- **Chu-Beasley Genetic Algorithm (CBGA):**
  - Population size (P): 50
  - Diversity threshold (Hamming): 10
  - Local search: Fast 2-opt (max. 20 improvements)

- Replacement: The offspring replaces the worst individual if it is better and maintains diversity.

4. Obtained Results (Base Phase)

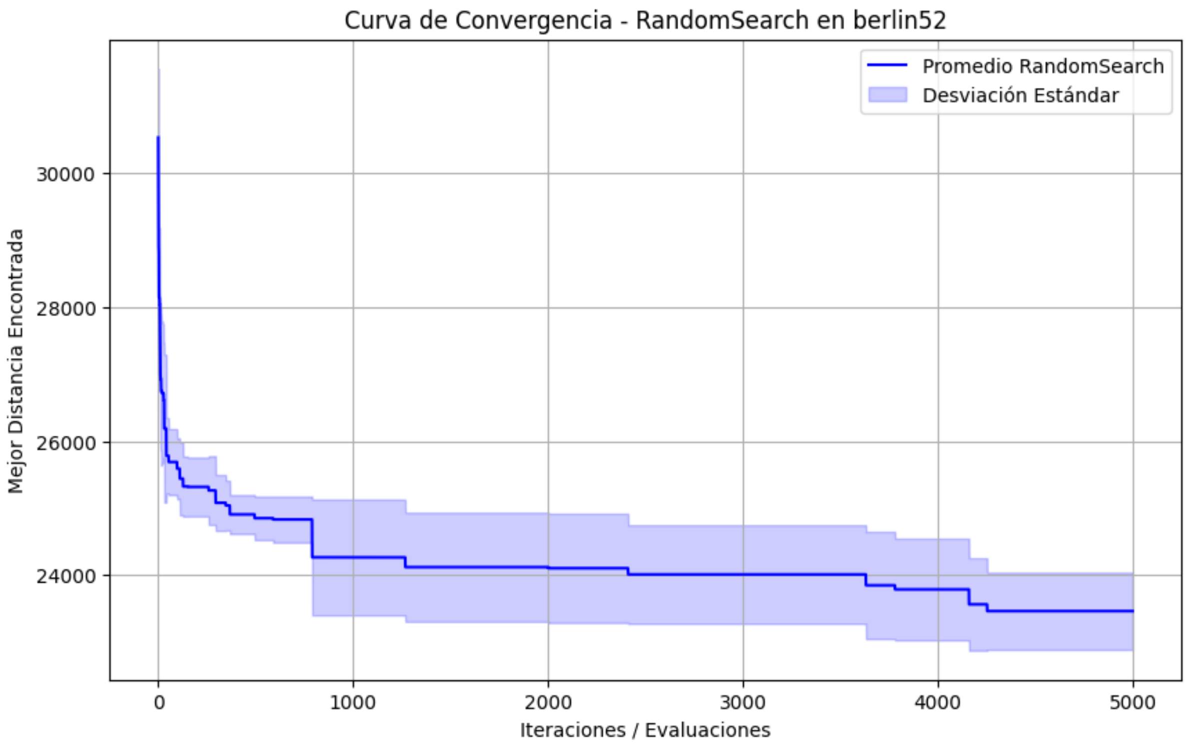
Below are the consolidated results for each algorithm after evaluating the 5 seeds. *(Note: The att48 instance presented anomalous GAPs >200% in all algorithms, suggesting a discrepancy in the pseudo-Euclidean distance function expected by TSPLIB vs the one implemented, but the data is reported exactly as obtained).*

4.0 Random Search (Dummy)

berlin52

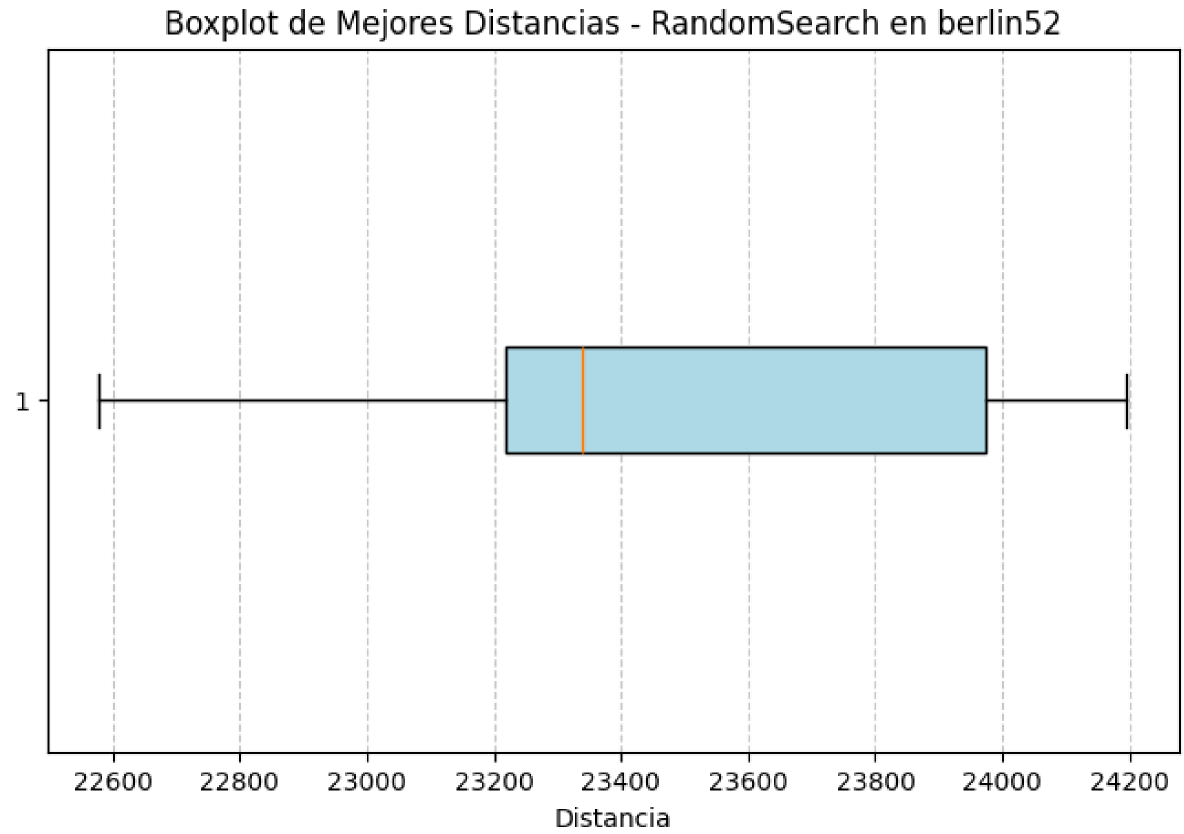
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	22577.15	23460.82 ± 644.17	211.07	0.09

- *[Average convergence curve Dummy - berlin52]*



>

- *[Boxplot of best distances Dummy - berlin52]*

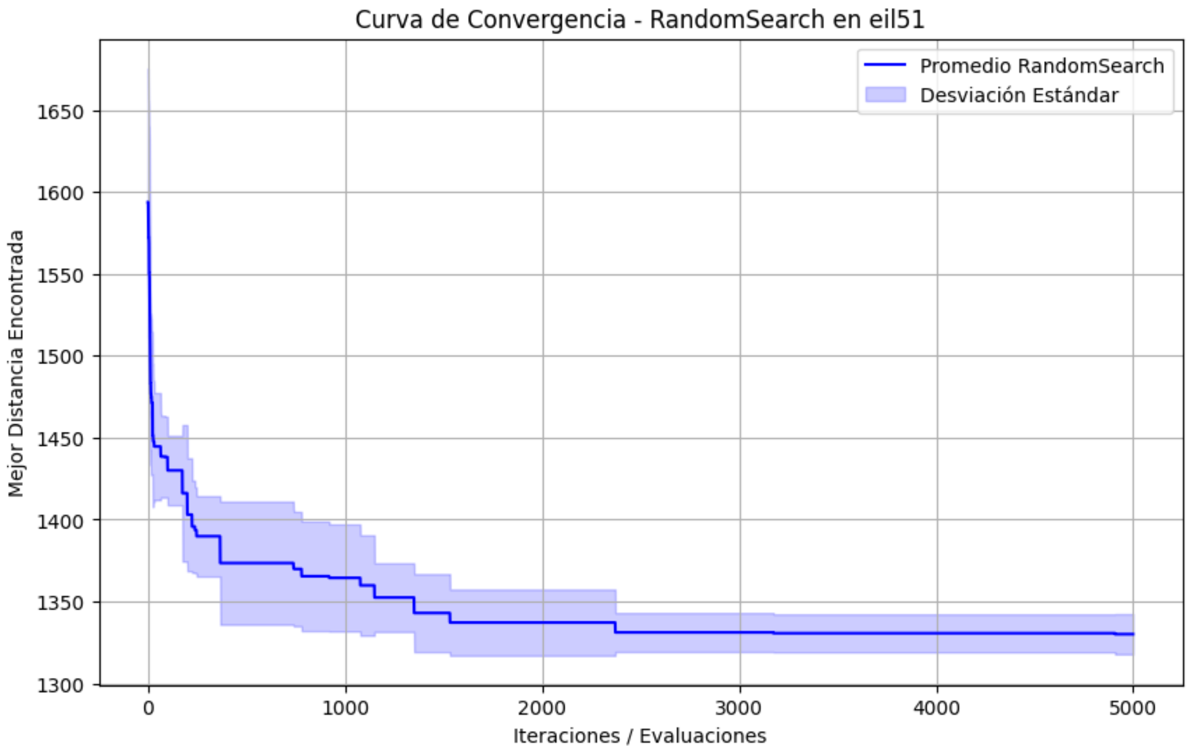


>

eil51

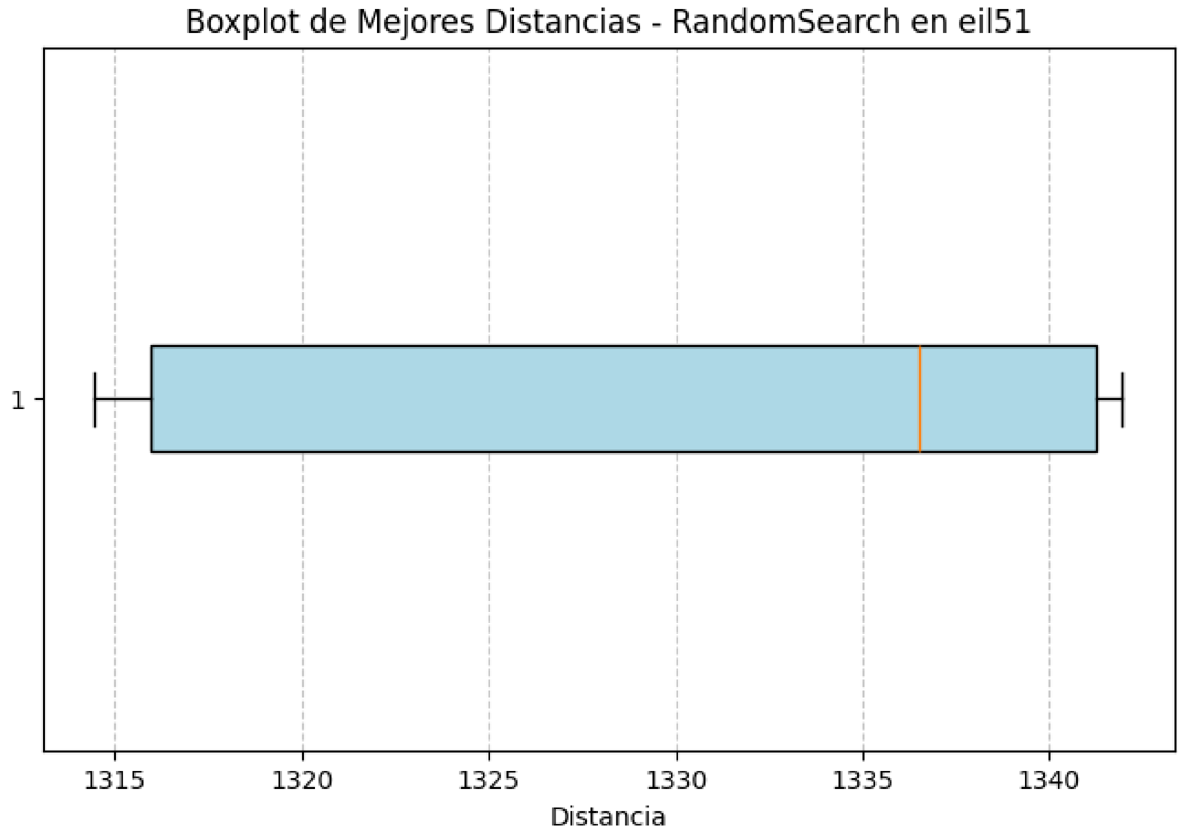
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	1314.43	1330.02 ± 13.71	212.21	0.09

- *[Average convergence curve Dummy - eil51]*



>

- *[Boxplot of best distances Dummy - eil51]*

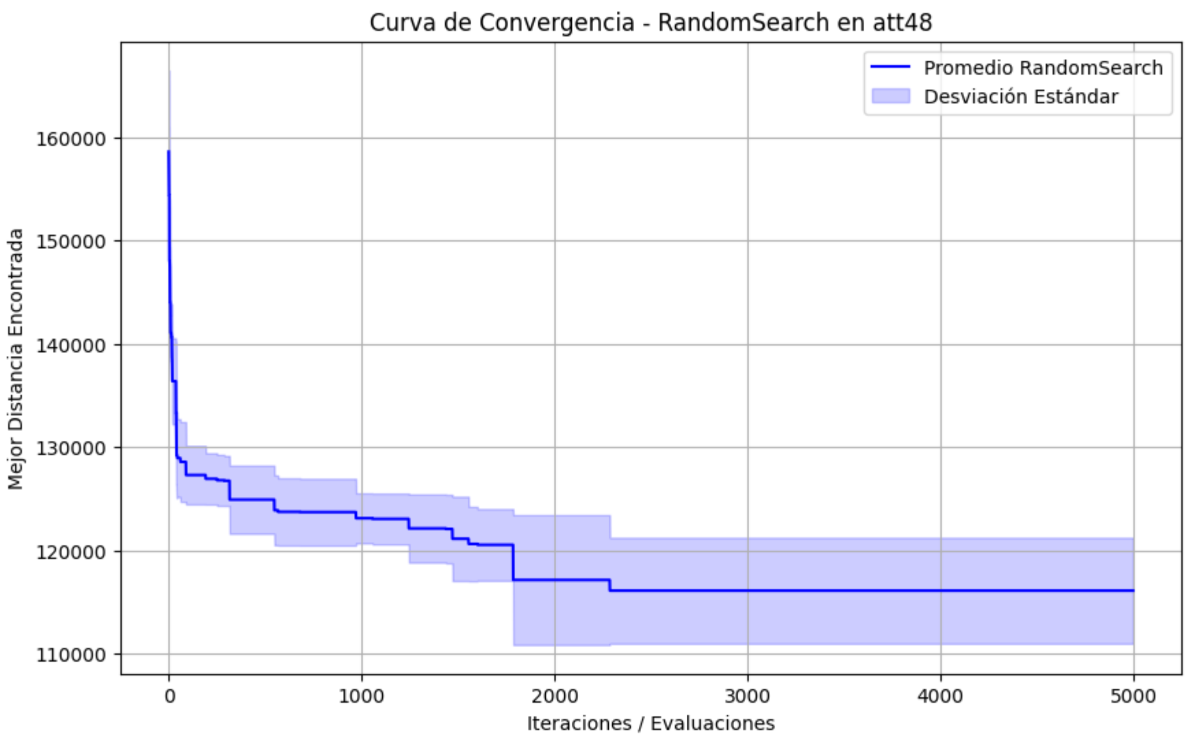


>

att48

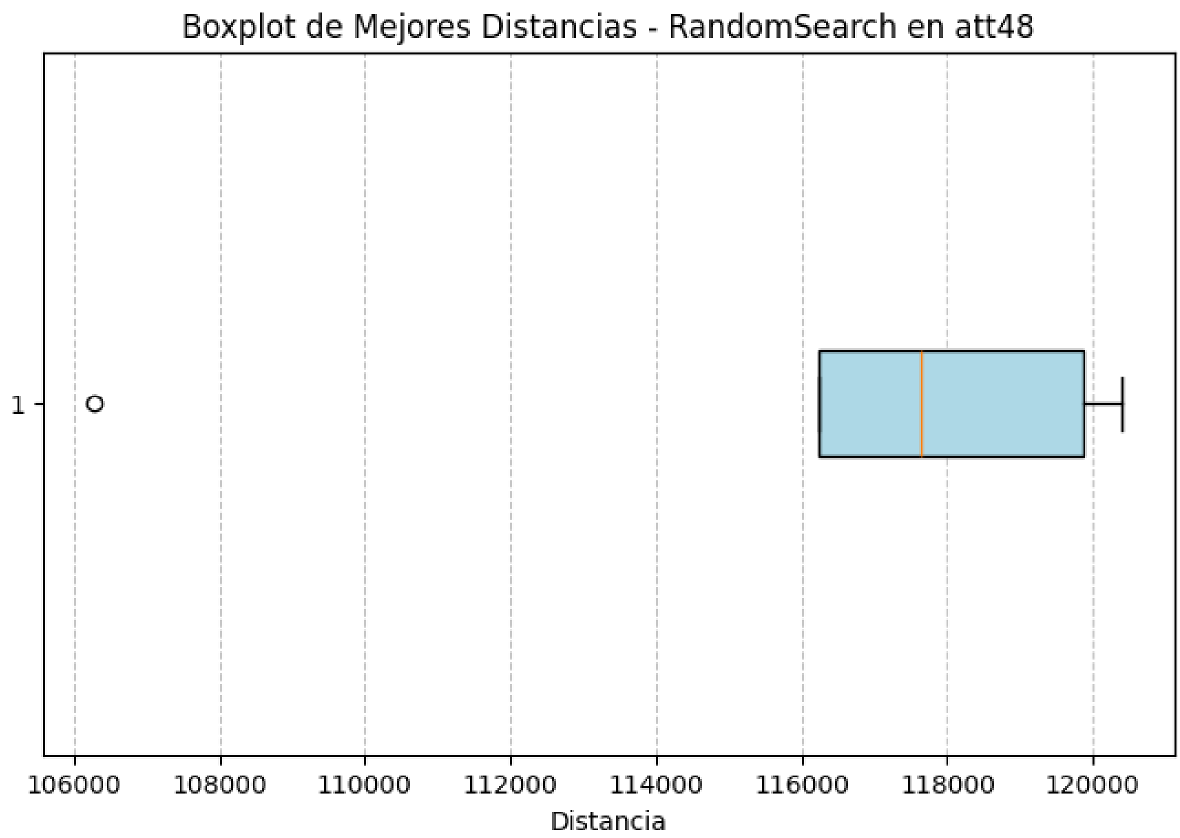
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
att48	10628	106276.78	116083.76 ± 5735.82	992.24	0.11

- *[Average convergence curve Dummy - att48]*



>

- *[Boxplot of best distances Dummy - att48]*

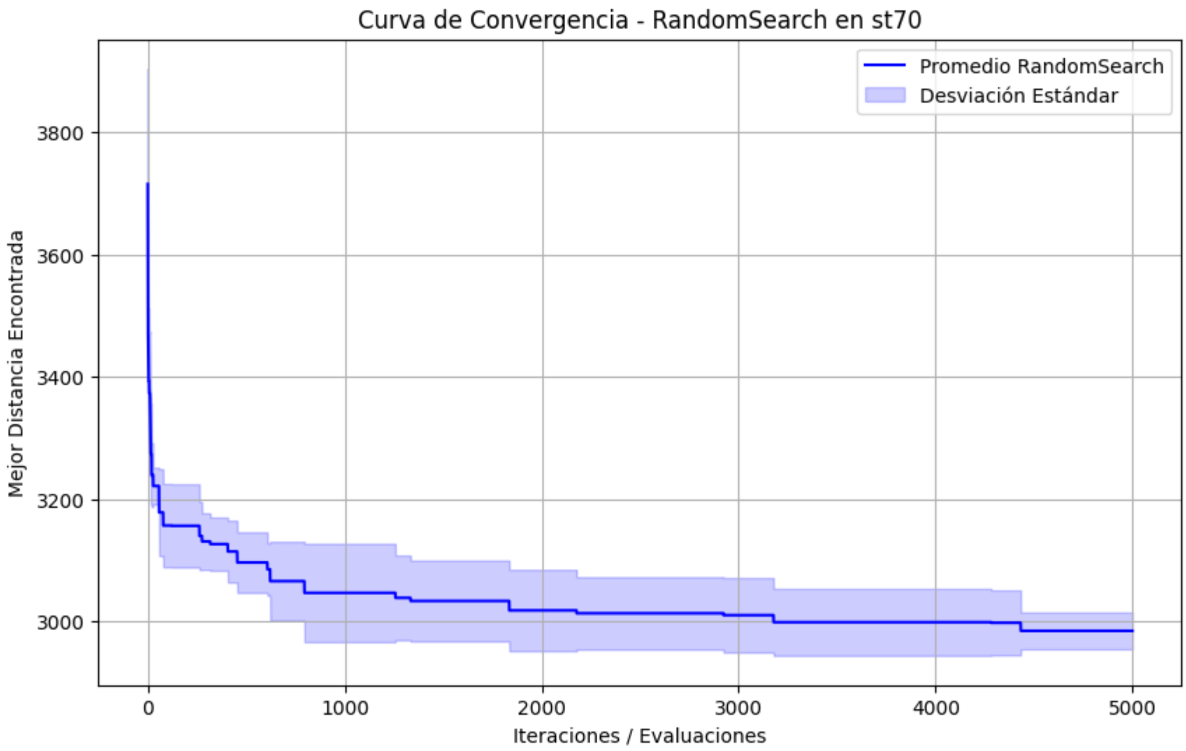


>

st70

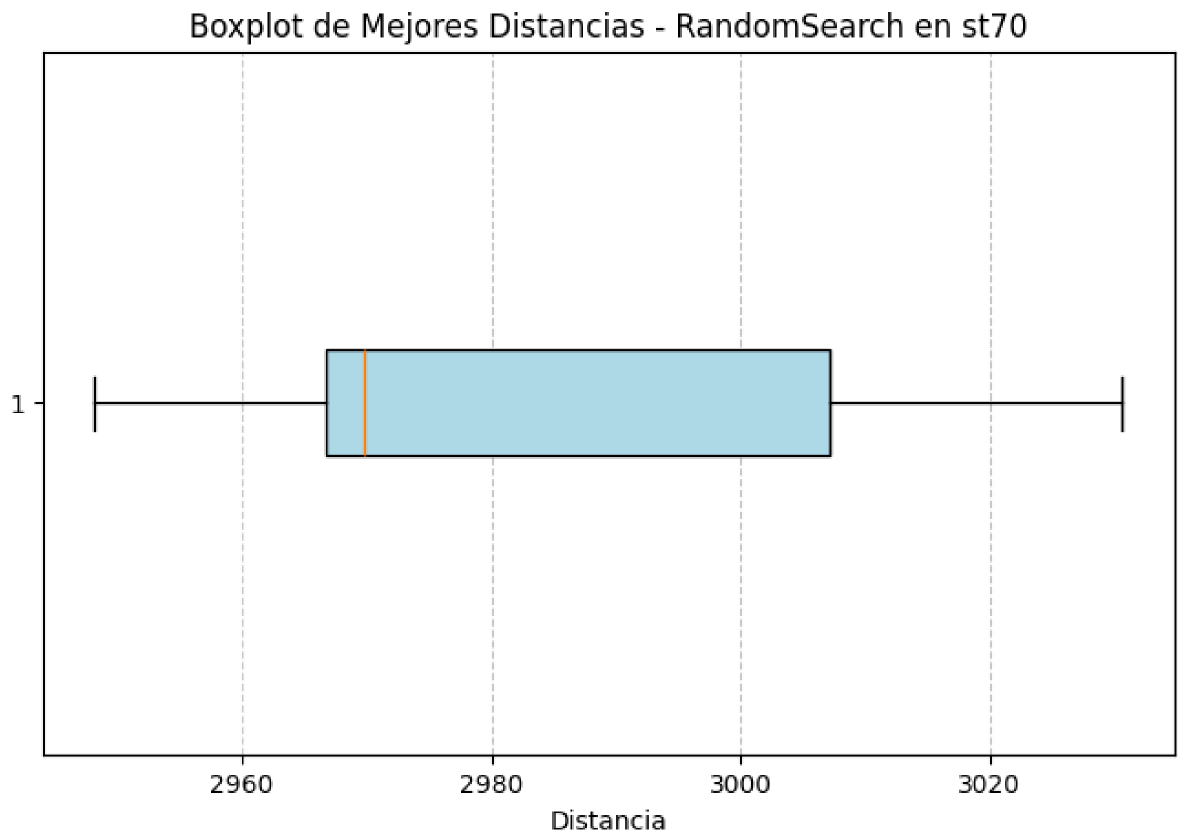
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
st70	675	2948.11	2984.50 ± 33.52	342.15	0.12

- *[Average convergence curve Dummy - st70]*



>

- *[Boxplot of best distances Dummy - st70]*



>

Comparative Table

Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	22577.15	23460.82 ± 644.17	211.07	0.09

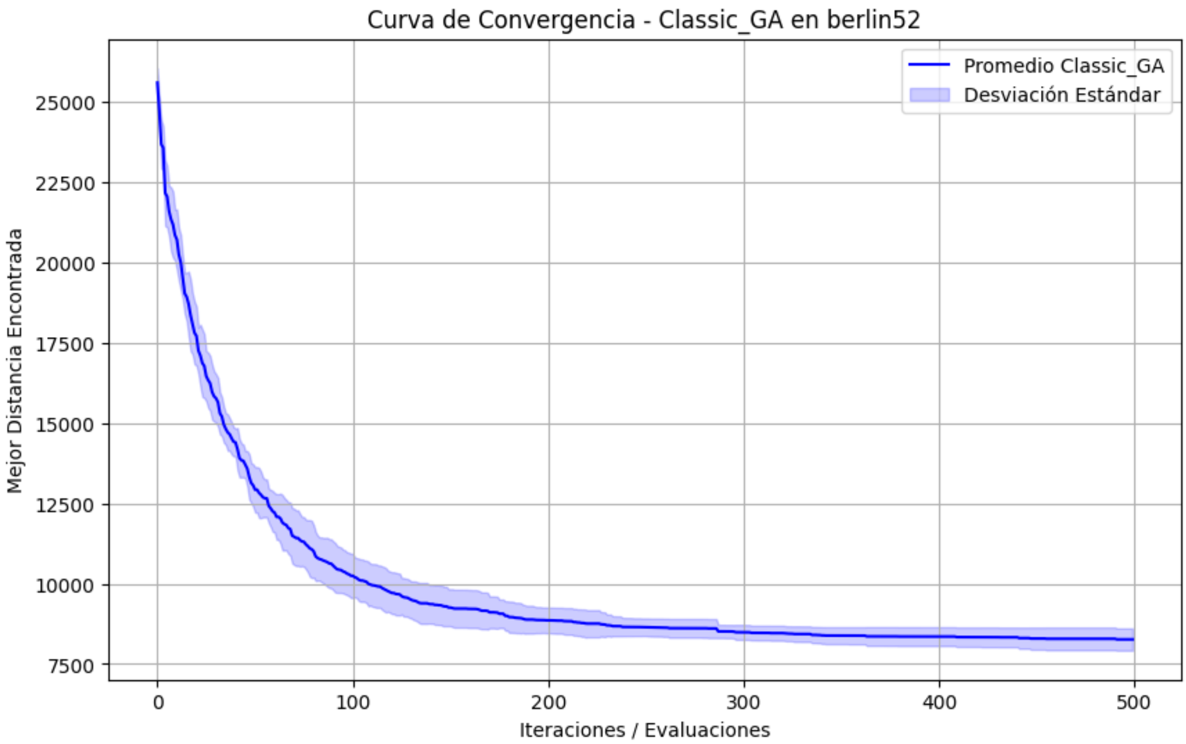
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	1314.43	1330.02 ± 13.71	212.21	0.09
att48	10628	106276.78	116083.76 ± 5735.82	992.24	0.11
st70	675	2948.11	2984.50 ± 33.52	342.15	0.12

4.1. Classic Genetic Algorithm (GA)

berlin52

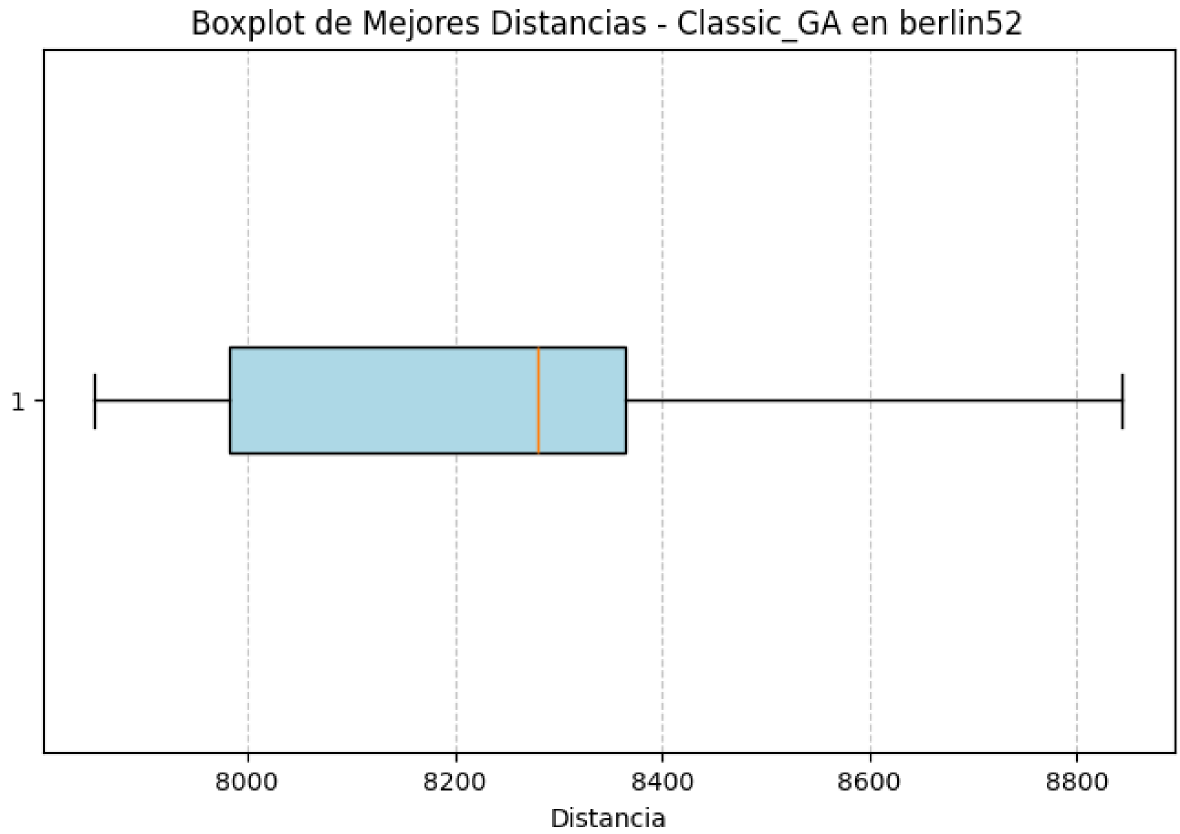
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7851.82	8264.77 ± 386.07	9.58	4.45

- *[Average convergence curve GA - berlin52]*



>

- *[Boxplot of best distances GA - berlin52]*

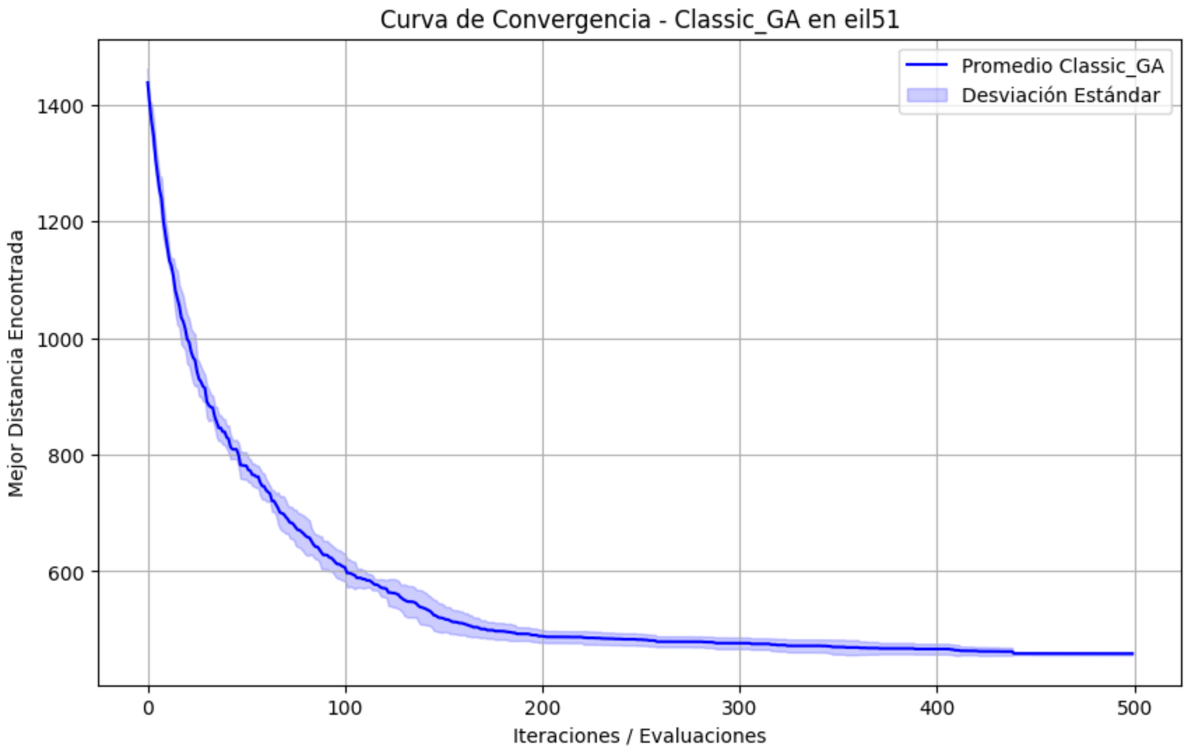


>

eil51

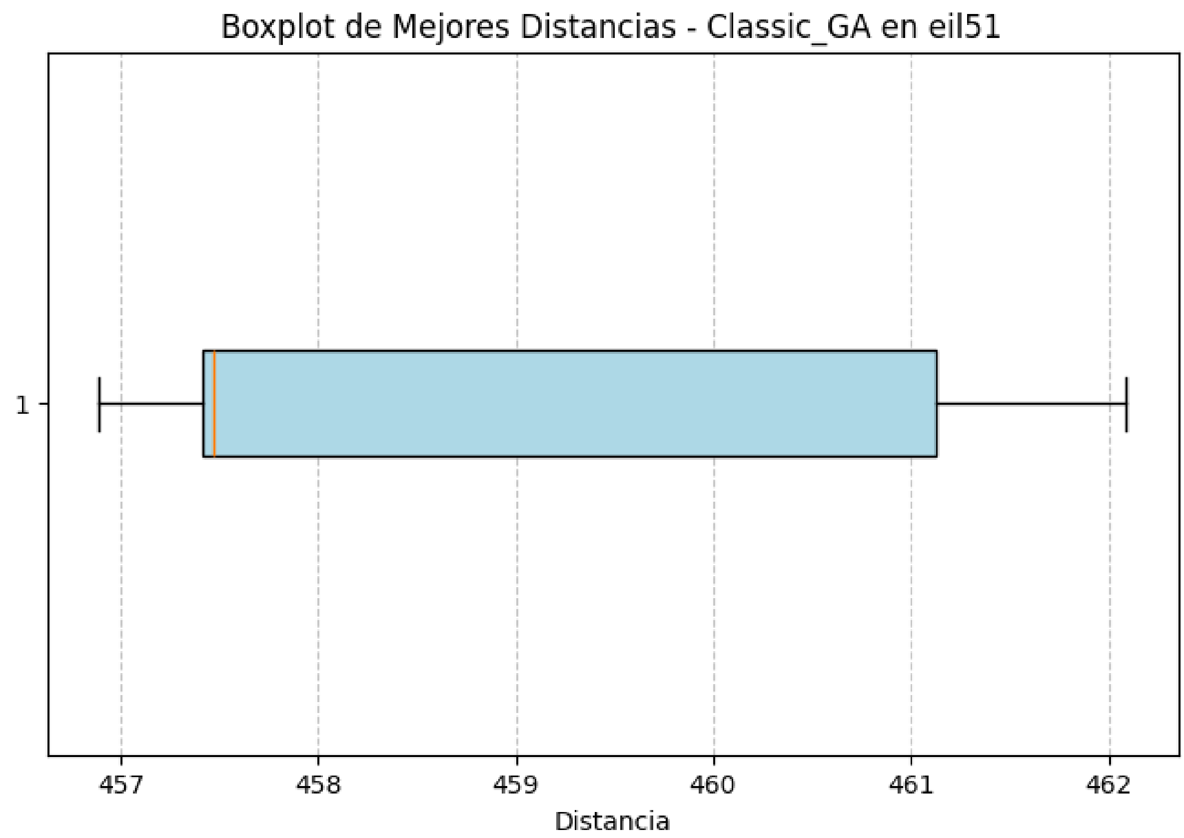
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	456.89	459.00 ± 2.42	7.75	4.42

- *[Average convergence curve GA - eil51]*



>

- *[Boxplot of best distances GA - eil51]*

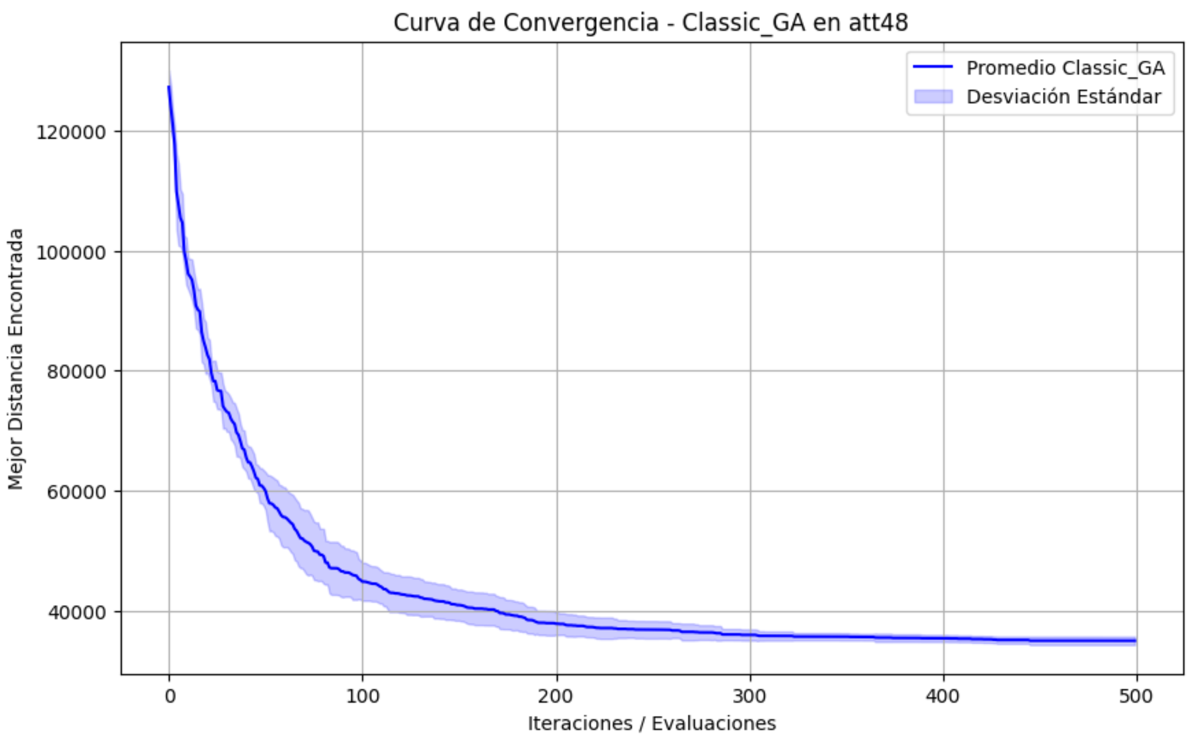


>

att48

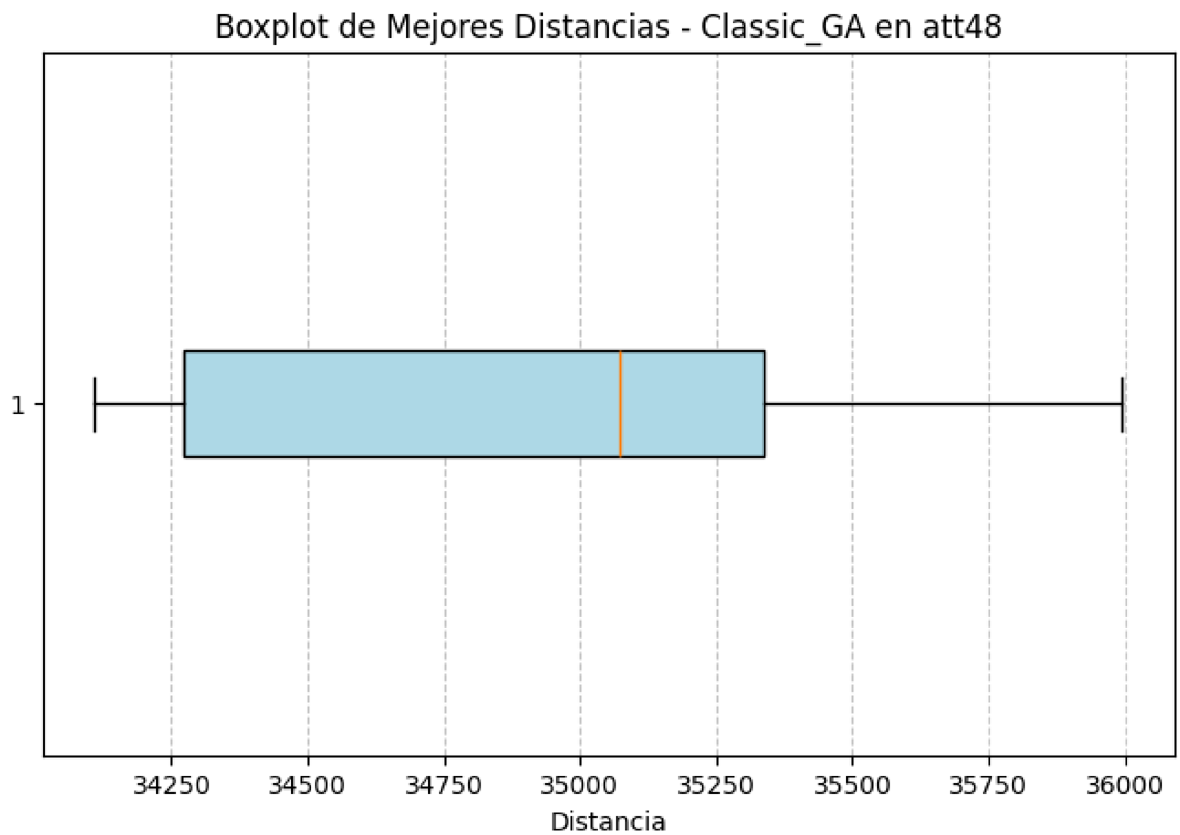
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
att48	10628	34107.07	34957.42 ± 779.50	228.92	4.19

- *[Average convergence curve GA - att48]*



>

- *[Boxplot of best distances GA - att48]*

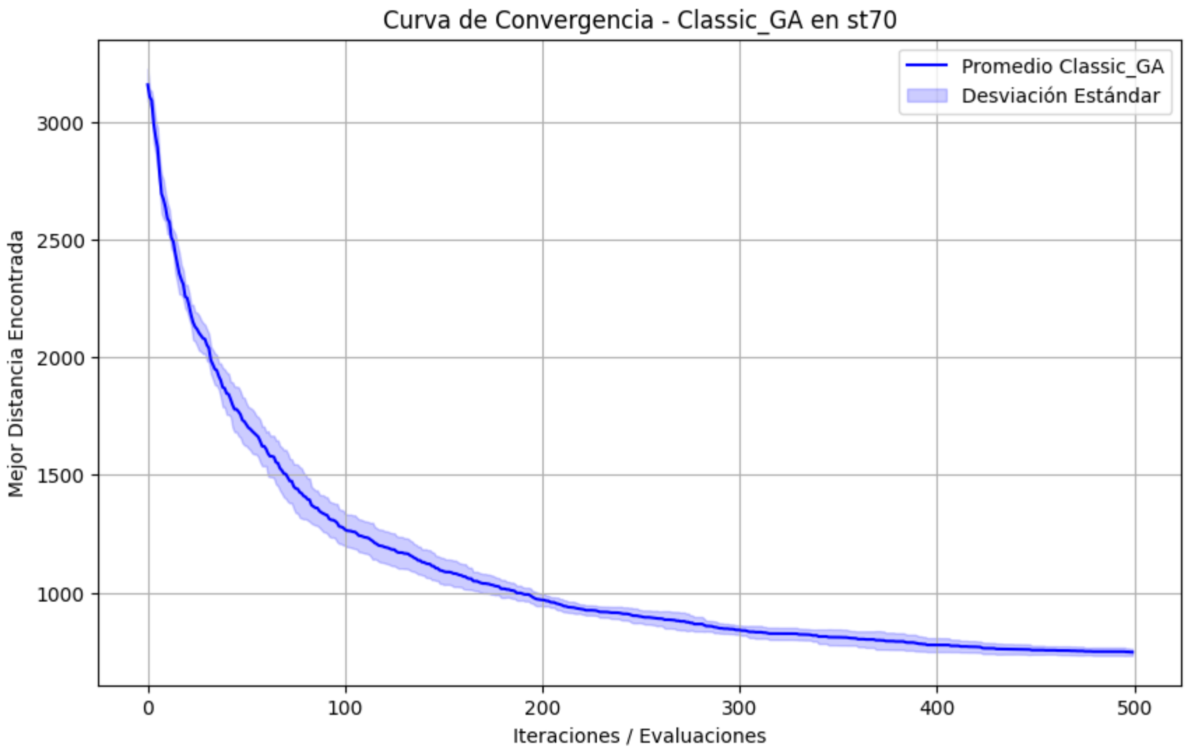


>

st70

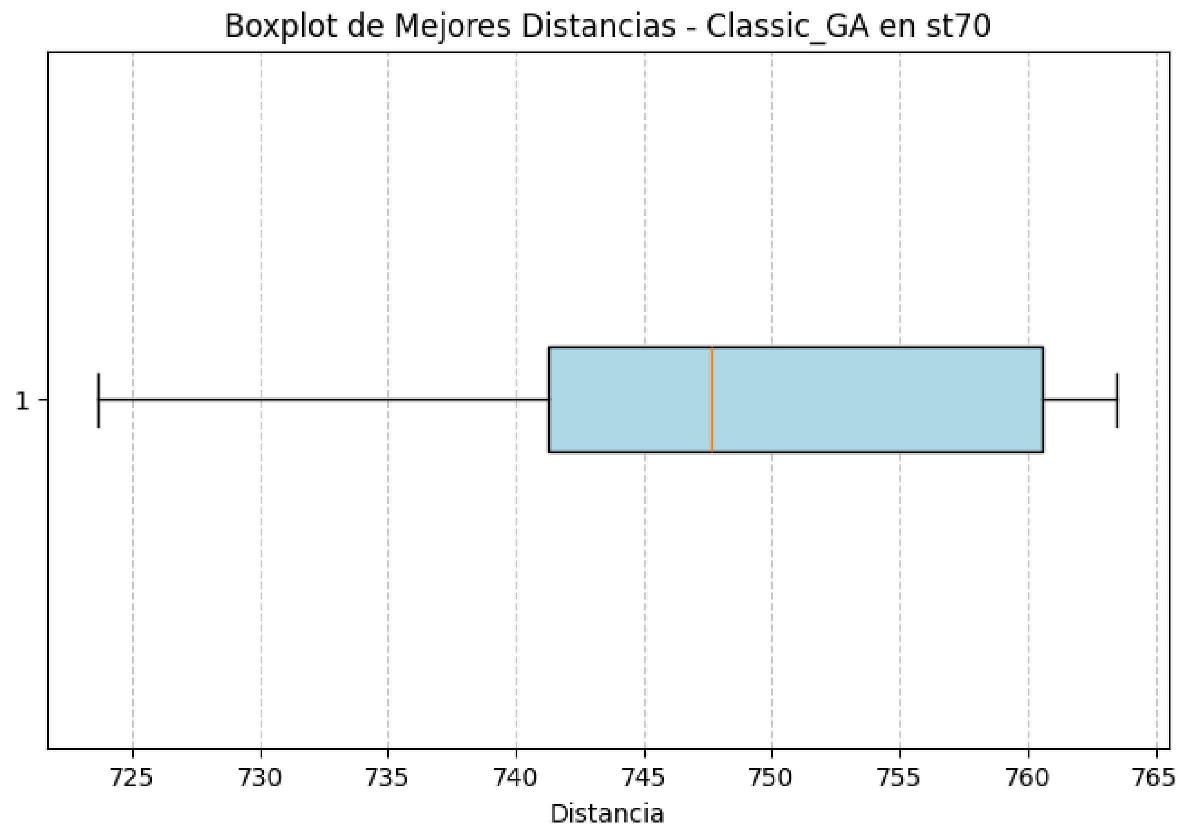
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
st70	675	723.64	747.32 ± 16.08	10.71	5.22

- *[Average convergence curve GA - st70]*



>

- *[Boxplot of best distances GA - st70]*



>

Comparative Table

Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7851.81	8264.77 ± 386.07	9.58	4.45
eil51	426	456.88	459.00 ± 2.42	7.75	4.41

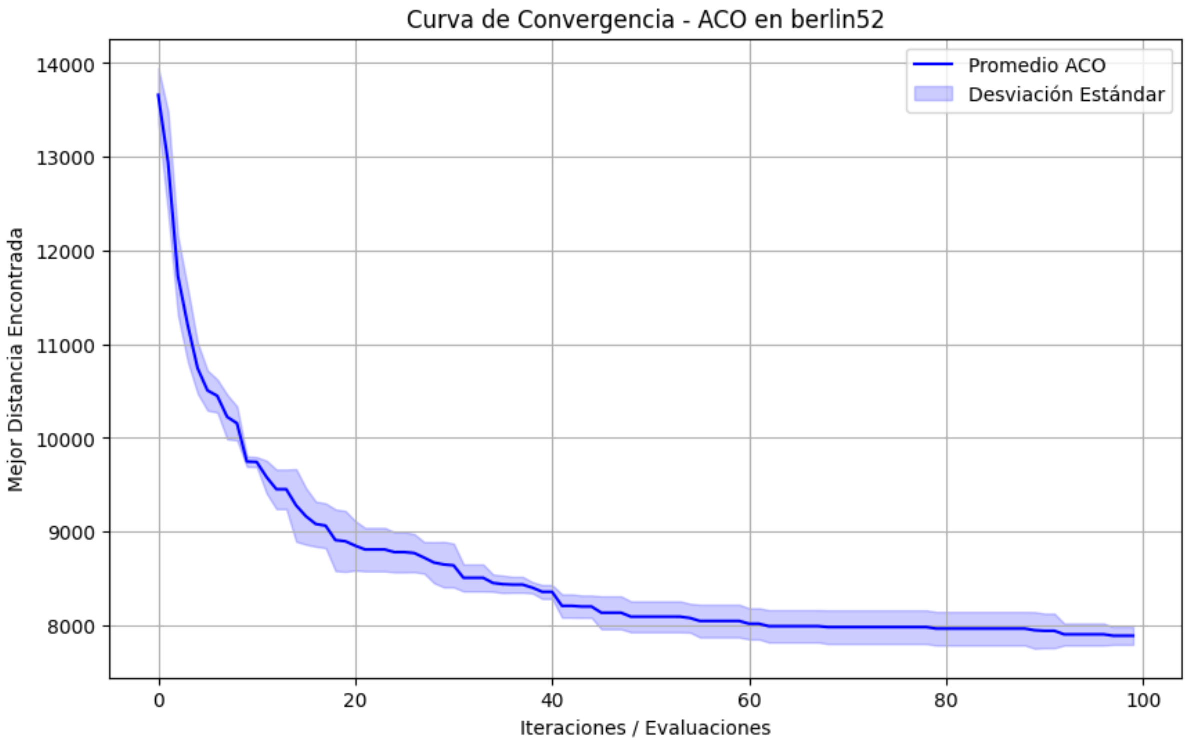
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
att48	10628	34107.06	34957.42 ± 779.50	228.92	4.19
st70	675	723.63	747.32 ± 16.08	10.71	5.21

4.2. Ant Colony Optimization (ACO)

berlin52

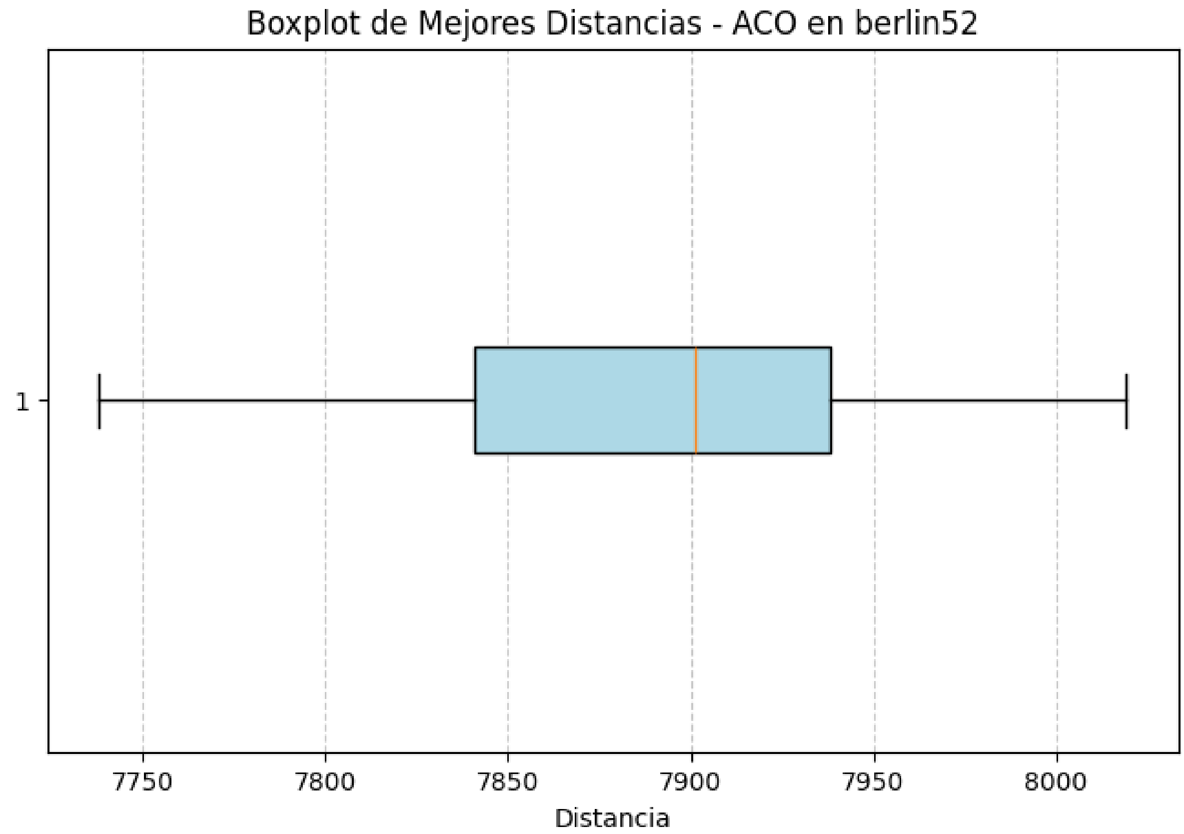
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7737.78	7887.40 ± 105.72	4.58	10.72

- *[Average convergence curve ACO - berlin52]*



>

- *[Boxplot of best distances ACO - berlin52]*

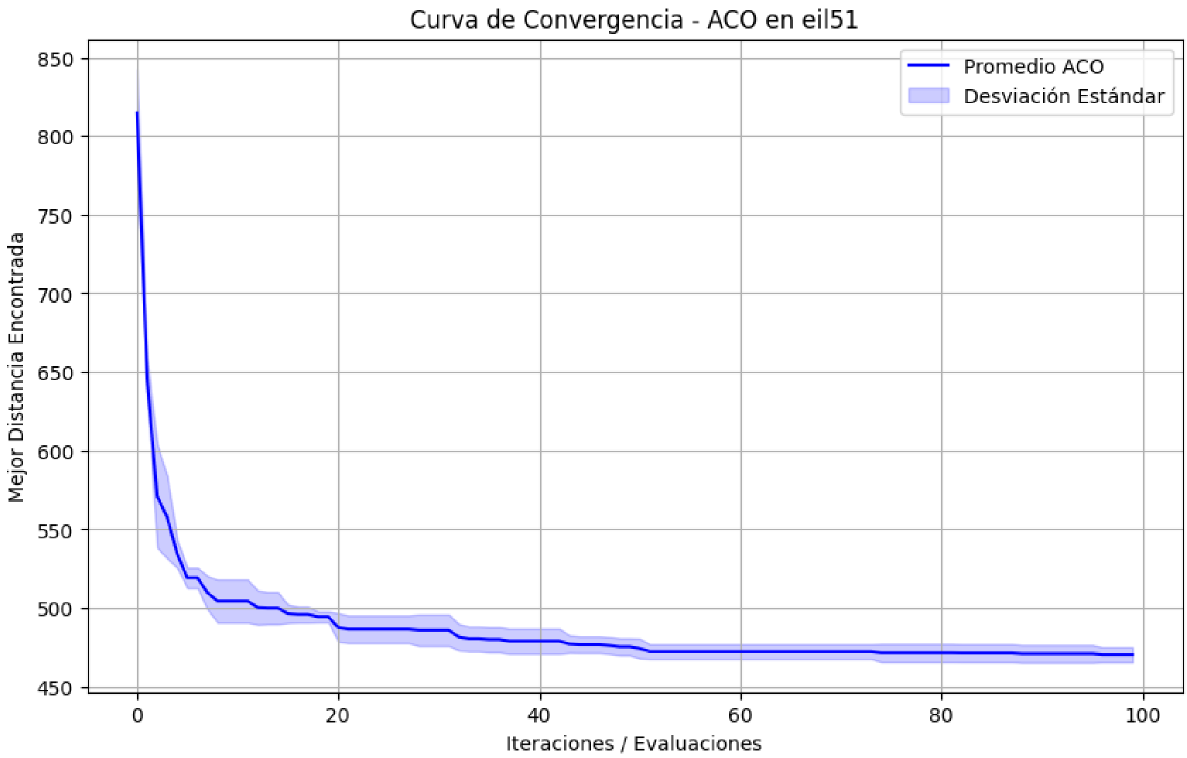


>

eil51

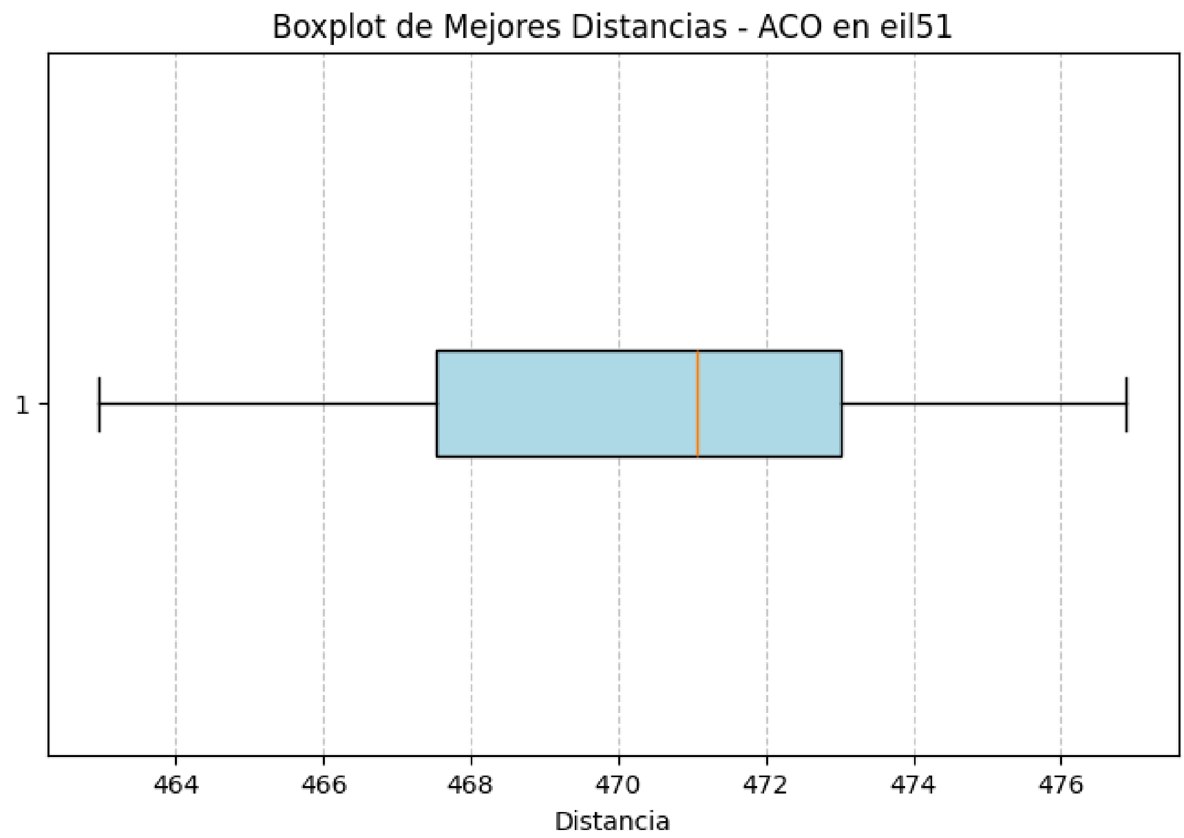
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	462.95	470.29 ± 5.32	10.40	8.88

- *[Average convergence curve ACO - eil51]*



>

- *[Boxplot of best distances ACO - eil51]*

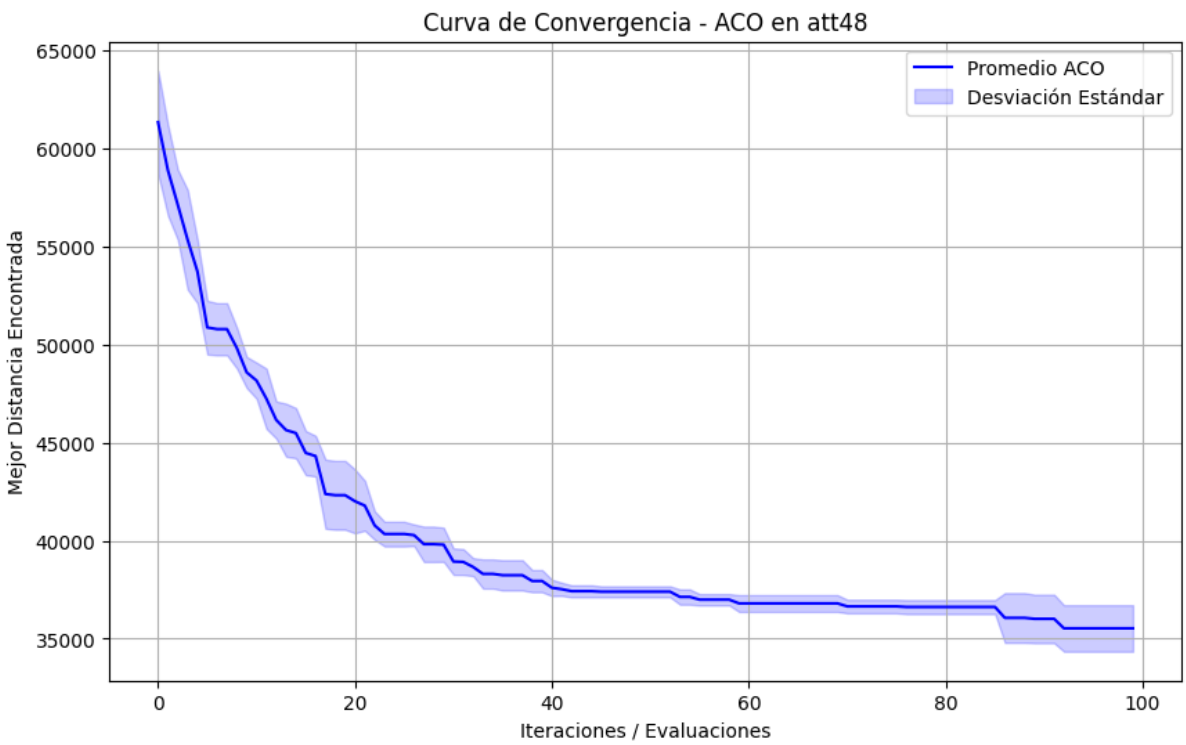


>

att48

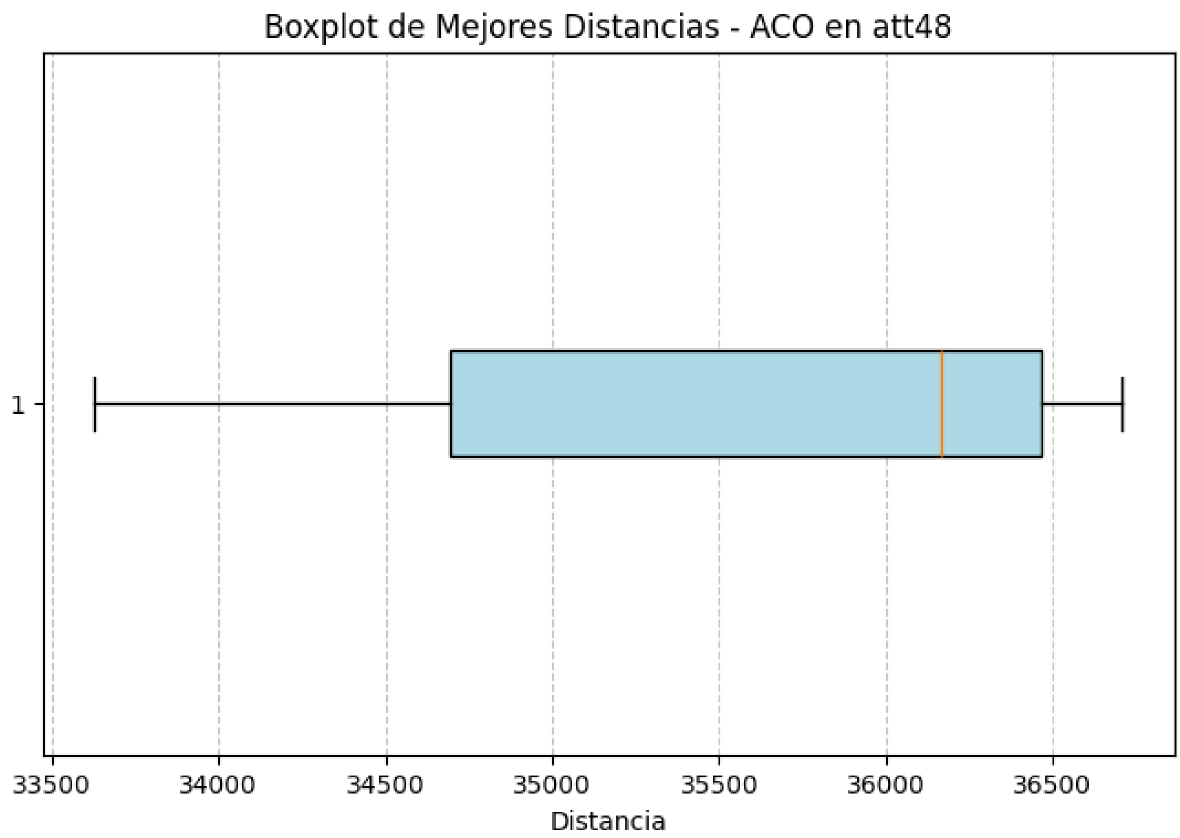
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
att48	10628	33625.62	35531.95 ± 1322.09	234.32	8.52

- *[Average convergence curve ACO - att48]*



>

- *[Boxplot of best distances ACO - att48]*

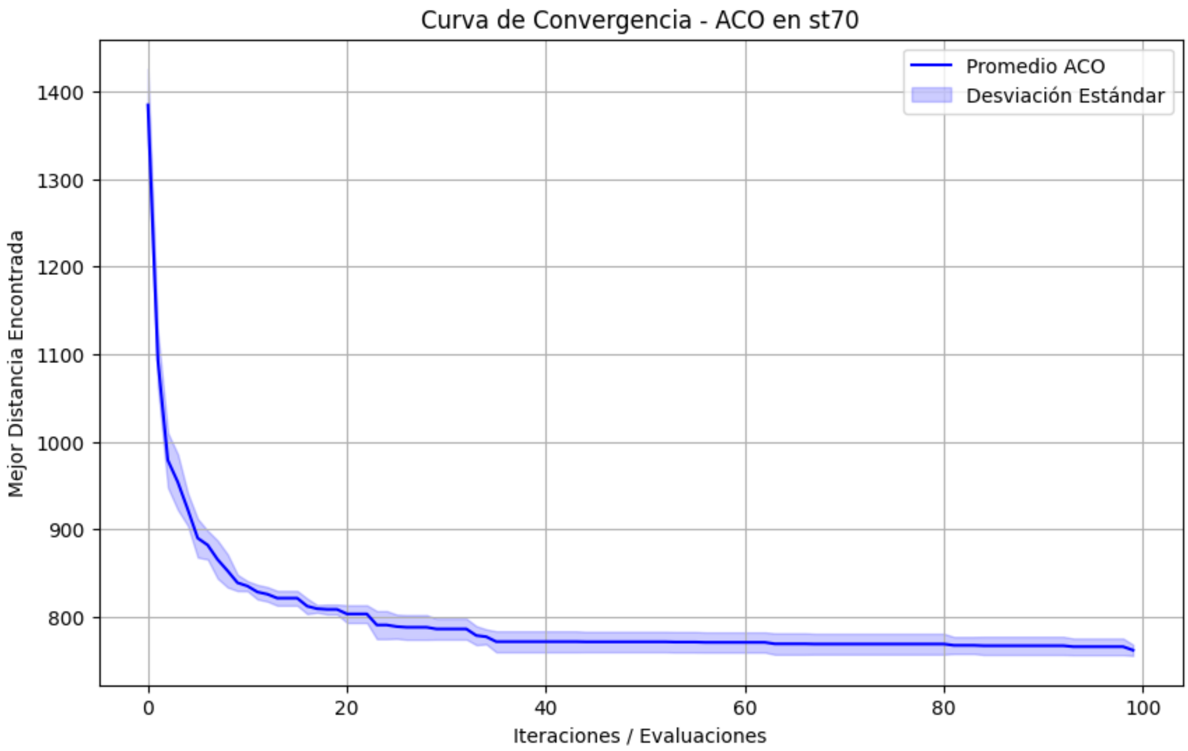


>

st70

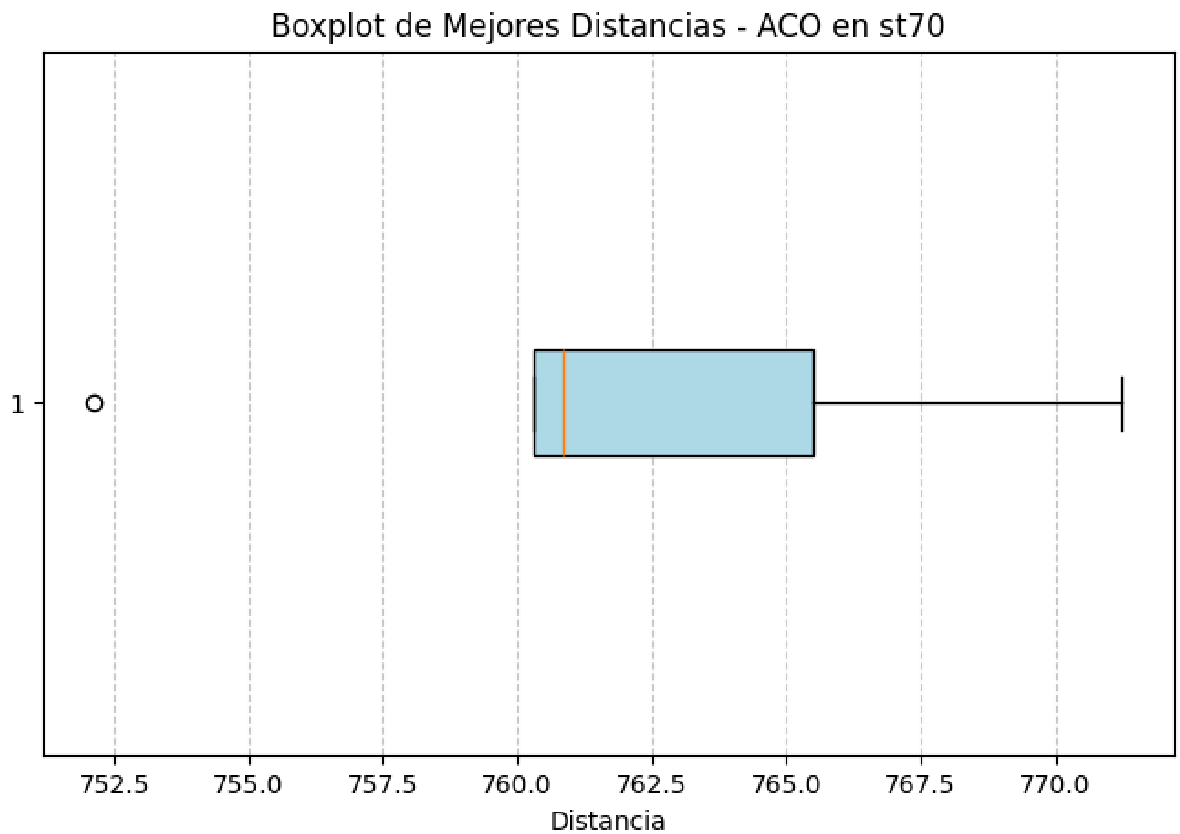
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
st70	675	752.11	762.00 ± 7.07	12.89	14.33

- *[Average convergence curve ACO - st70]*



>

- *[Boxplot of best distances ACO - st70]*



>

Comparative Table

Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7737.78	7887.40 ± 105.72	4.58	10.72

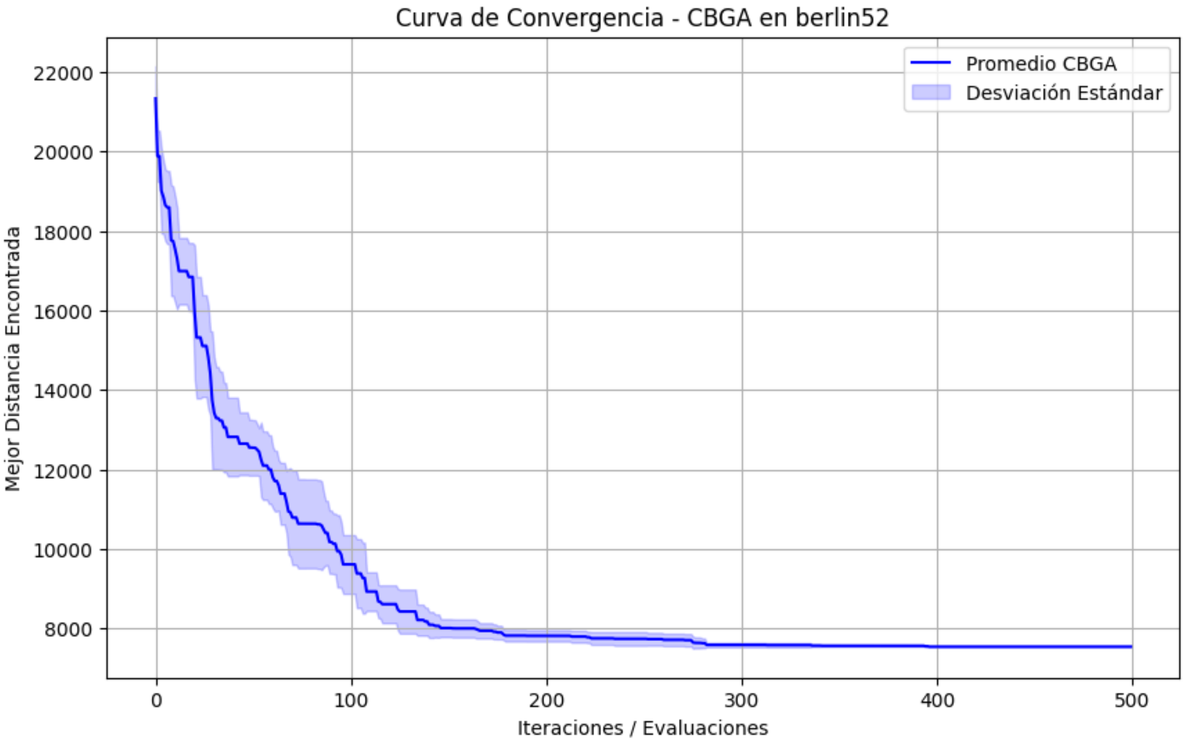
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	462.94	470.29 ± 5.32	10.40	8.87
att48	10628	33625.61	35531.95 ± 1322.09	234.32	8.51
st70	675	752.11	762.00 ± 7.07	12.89	14.32

4.3. Chu-Beasley Genetic Algorithm (CBGA)

berlin52

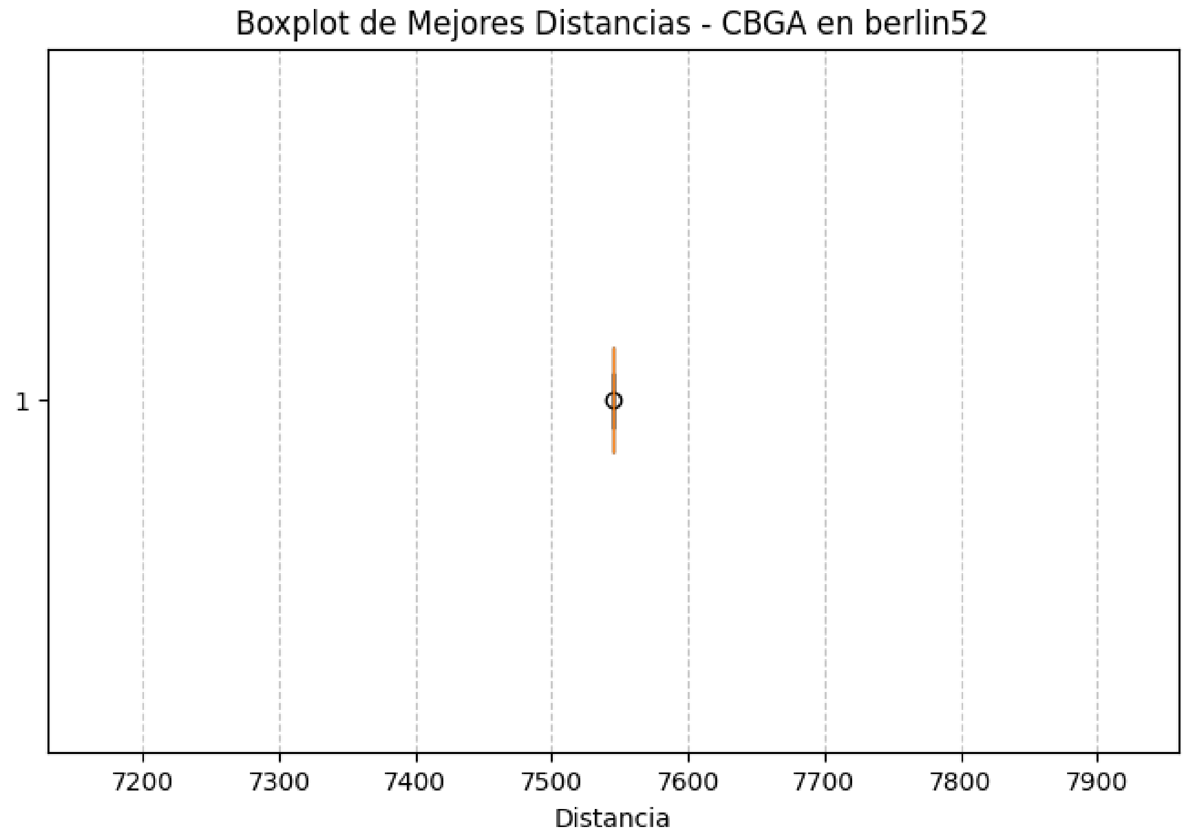
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7544.37	7544.37 ± 0.00	0.03	3.24

- *[Average convergence curve CBGA - berlin52]*



>

- *[Boxplot of best distances CBGA - berlin52]*

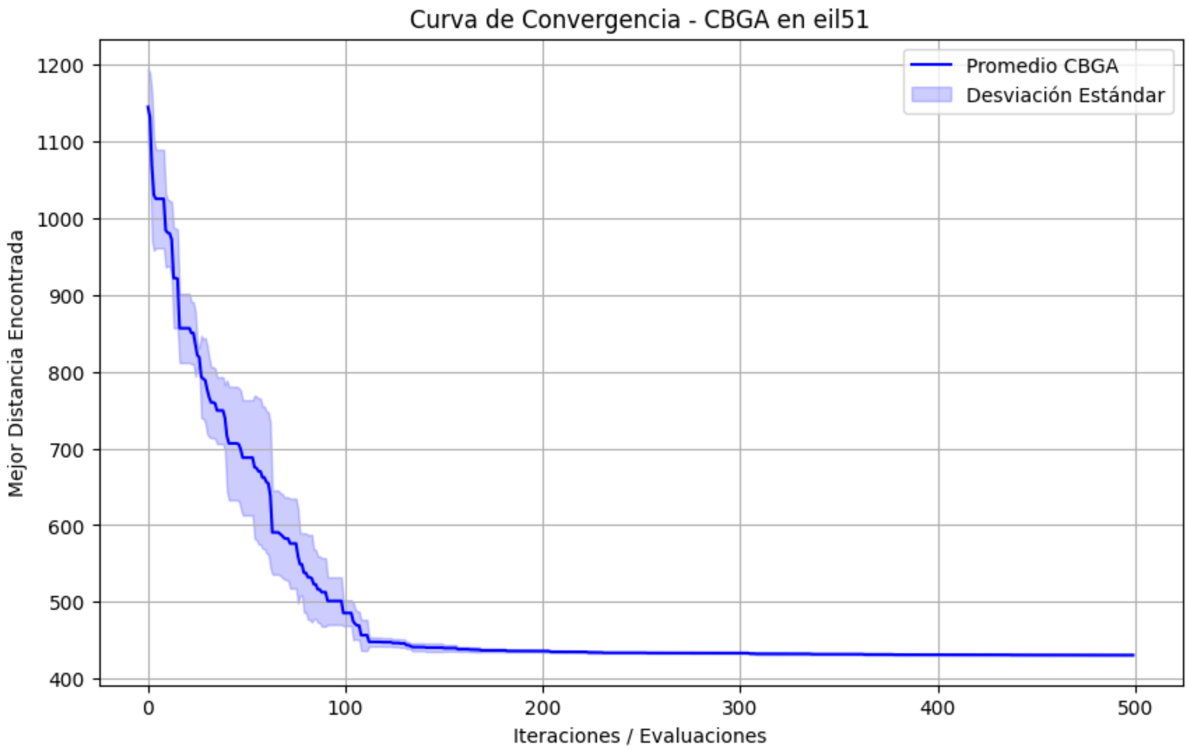


>

eil51

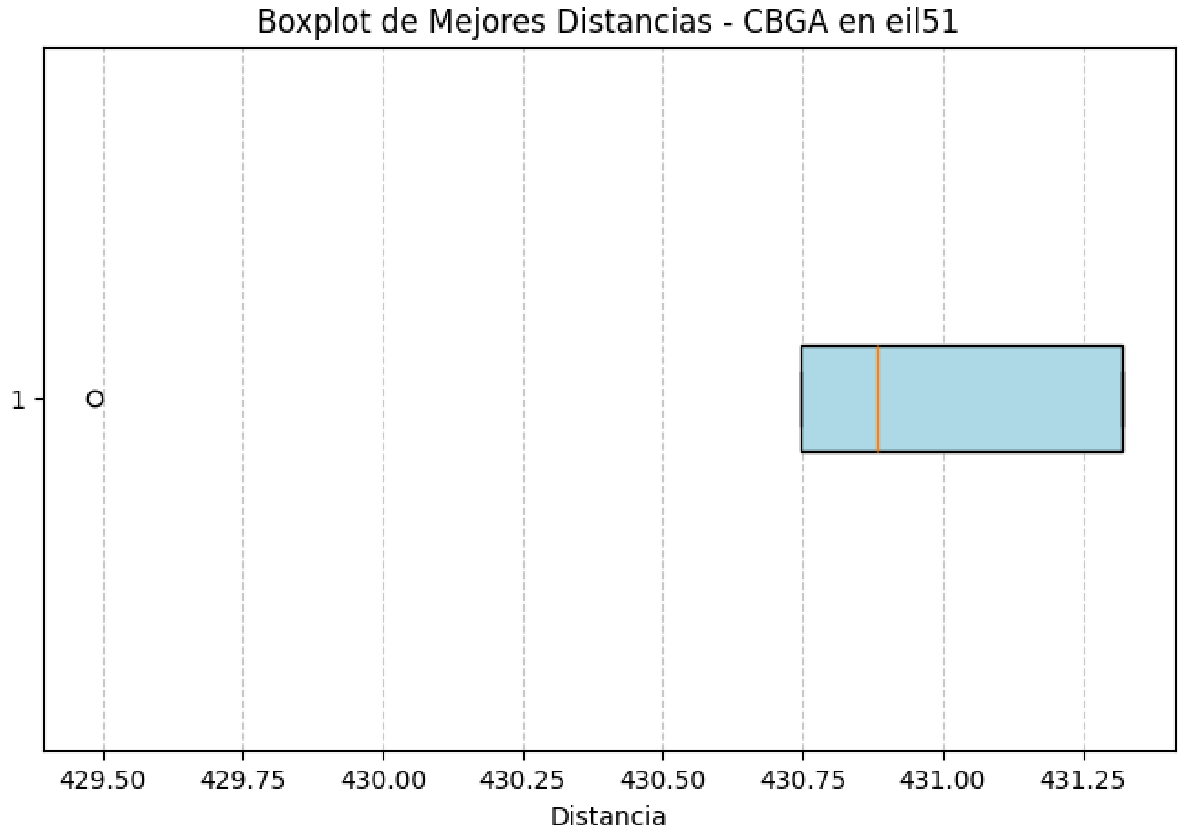
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	429.48	430.75 ± 0.75	1.12	3.01

- *[Average convergence curve CBGA - eil51]*



>

- *[Boxplot of best distances CBGA - eil51]*

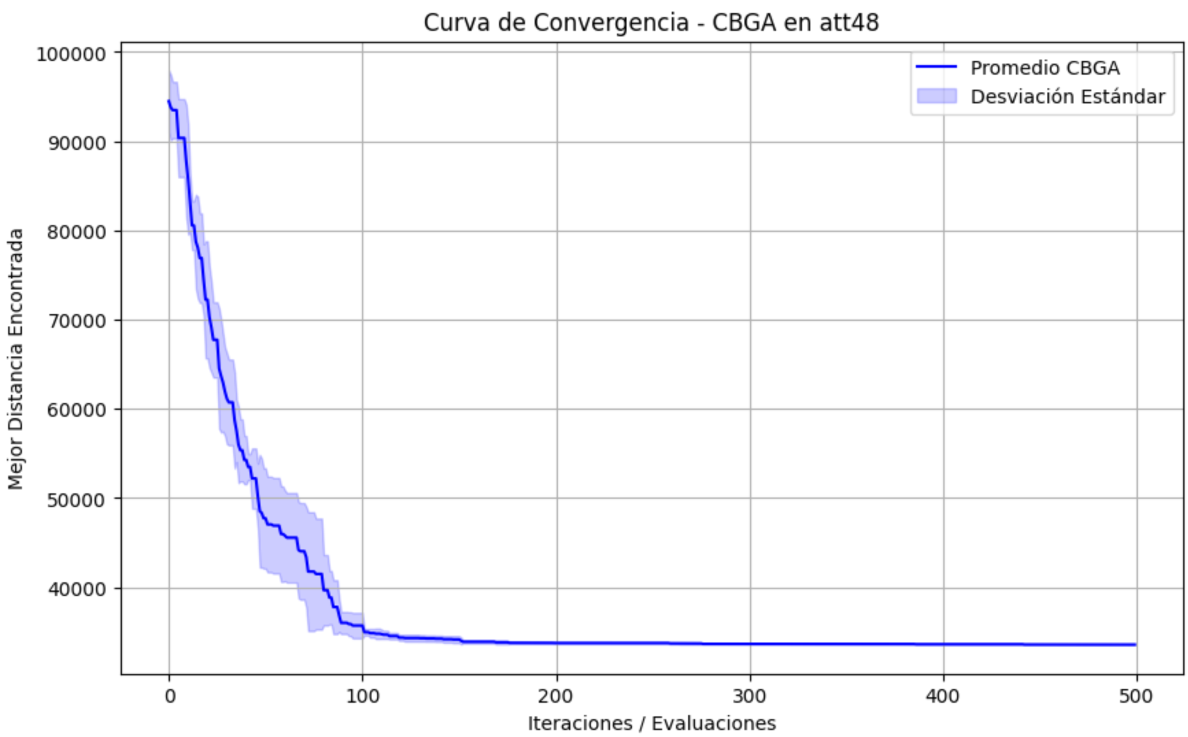


>

att48

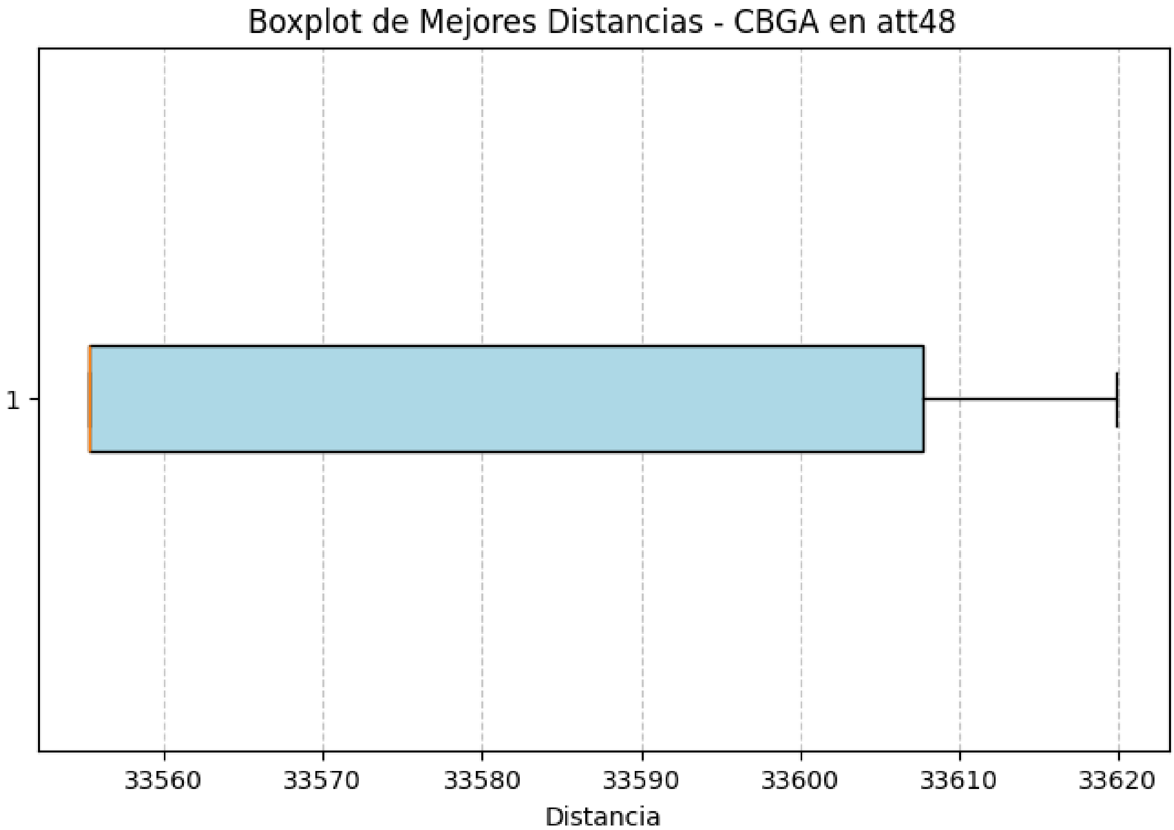
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
att48	10628	33555.28	33578.68 ± 32.34	215.95	2.60

- *[Average convergence curve CBGA - att48]*



>

- *[Boxplot of best distances CBGA - att48]*

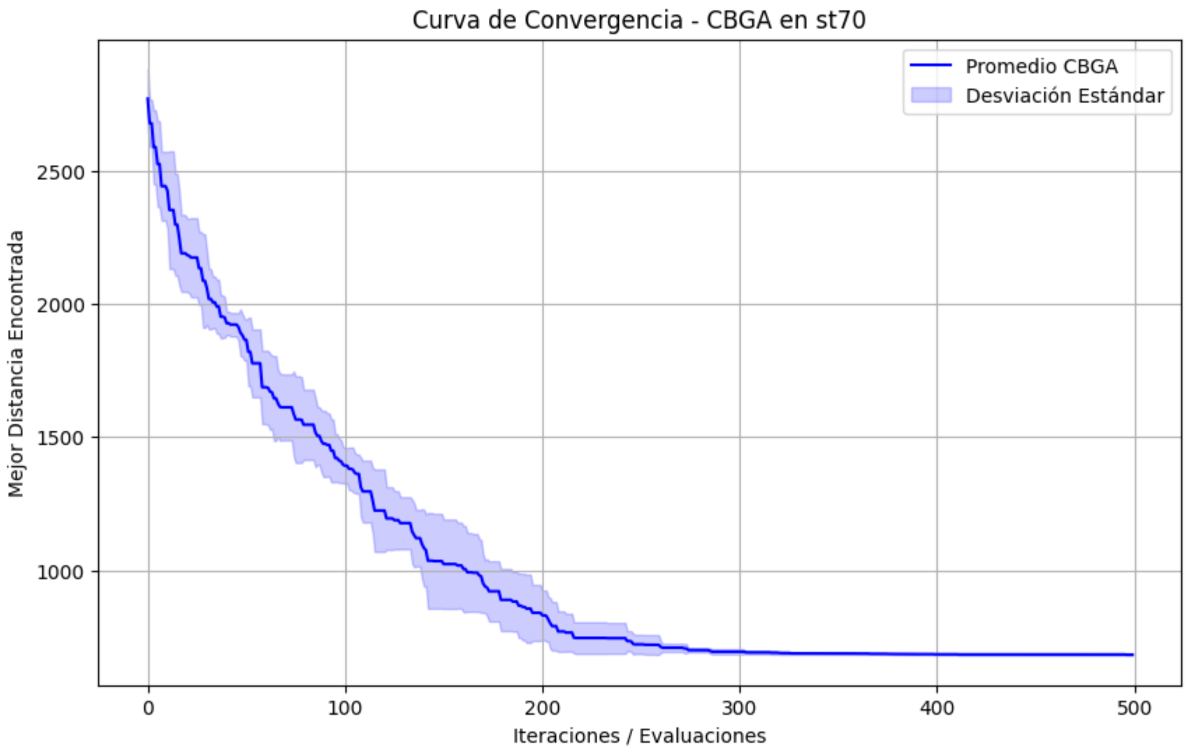


>

st70

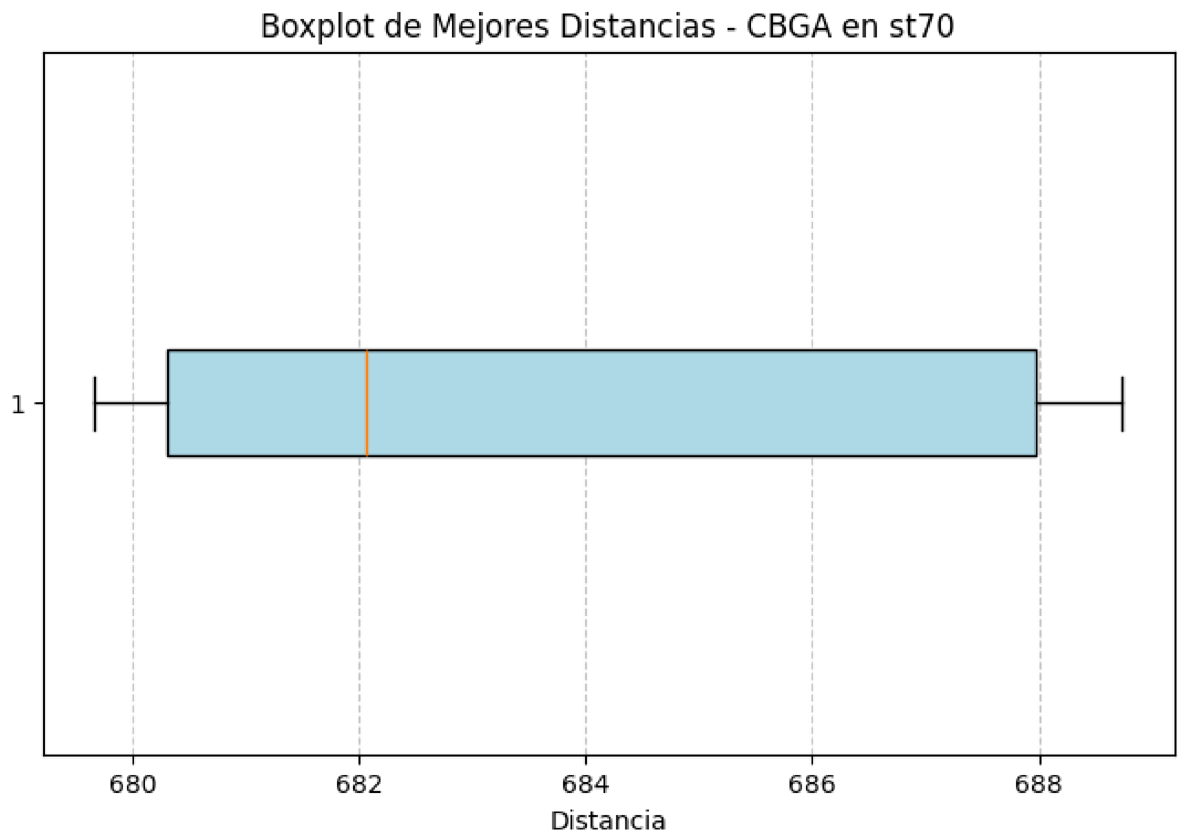
Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
st70	675	679.65	683.75 ± 4.31	1.30	4.90

- *[Average convergence curve CBGA - st70]*



>

- *[Boxplot of best distances CBGA - st70]*



>

Comparative Table

Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
berlin52	7542	7544.36	7544.37 ± 0.00	0.03	3.24

Instance	Optimum	Best Distance	Average Distance	Average GAP (%)	Average Time (s)
eil51	426	429.48	430.75 ± 0.75	1.12	3.00
att48	10628	33555.27	33578.68 ± 32.34	215.95	2.60
st70	675	679.65	683.75 ± 4.31	1.30	4.89

5. Hyperparameter Tuning

A Grid Search was performed on the `berlin52` instance using 3 seeds to find the best configuration for each algorithm.

- **Best GA:** `{'pop_size': 100, 'pm': 0.1, 'pc': 0.85, 'elitism_k': 2, 'iterations': 200}`. Average distance: 8631.69.
- **Best ACO:** `{'num_ants': 40, 'alpha': 2.0, 'beta': 2.0, 'rho': 0.1, 'iterations': 50}`. Average distance: 7623.97.
- **Best CBGA:** `{'pop_size': 50, 'threshold': 20, 'iterations': 200}`. Average distance: 7793.00.

6. Critical Discussion

1. **Which algorithm obtains the lowest average GAP?** The **CBGA** obtains, by a very wide margin, the lowest average GAP. In `berlin52` it achieved an almost perfect GAP of 0.03%, compared to 4.58% for ACO and 9.58% for GA. In `st70`, CBGA achieved 1.30% compared to >10% for the other algorithms.
2. **Which algorithm is more stable (lowest deviation)?** The **CBGA** demonstrated exceptional stability. In `berlin52`, its standard deviation was 0.00 (it found practically the same optimal solution across all seeds). In `eil51`, its deviation was barely 0.75.
3. **Which algorithm reaches good solutions faster?** The **CBGA** is the fastest in execution time (averages of 2.6s to 4.8s), outperforming the classic GA (~4-5s) and being significantly faster than ACO (~8-14s). The inclusion of the 2-opt local search drastically accelerates convergence towards good solutions.
4. **What happens when the dimension increases?** When moving from ~50 nodes to 70 nodes (`st70`), the GA and ACO suffer a degradation in solution quality (GAPs of 10.71% and 12.89% respectively). However, the CBGA scales excellently, maintaining an extremely low GAP of 1.30%.
5. **Does tuning change the winner?** No. Although tuning improved ACO's performance (lowering its average distance to 7623.97 in `berlin52`), the base performance of the CBGA (7544.37) remains superior to the tuned versions of GA and ACO.
6. **Does CBGA improve upon GA? In what sense?** Yes, overwhelmingly. It improves in **quality** (reduces the GAP from 9.58% to 0.03% in berlin52), in **stability** (reduces the standard deviation to almost zero), and in **time** (it is faster by requiring a smaller population, P=50 vs P=100, compensated by the intensification of 2-opt).

7. Conclusions

After evaluating the algorithms under multiple criteria, the following is concluded:

- **Quality and Stability (Winner: CBGA):** The combination of strict diversity control (Hamming distance) and local intensification (2-opt) allows the CBGA to avoid local optima and converge to near-optimal solutions consistently across all executions.

- **Speed (Winner: CBGA / GA):** Population-based algorithms (GA and CBGA) proved to be computationally more efficient than the constructive approach of ACO, which requires a high computational cost for updating and querying the pheromone matrix.
- **Final Verdict:** The **Chu-Beasley Genetic Algorithm (CBGA)** is unquestionably the best algorithm for this problem. Prioritizing the criteria of **solution quality and scalability**, the CBGA demonstrates that a hybrid (memetic) metaheuristic that carefully balances exploration (diversity control) and exploitation (local search) vastly outperforms pure classic approaches.