Optimization and Decision

Asymmetric traveling salesman problem

Project report

Bologna Master Degree in Mechanical Engineering
Academic Year 2021 – 2022

2nd Semester

Students

Afonso Brandão Oliveira Fernandes, n.º 93204
Duarte Sequeira Correia, n.º 93242
Miguel André Milhazes, n.º 93305

Professors

Susana M. Vieira
Bernardo Marreiros Firme
Miguel de Sousa Esteves Martins

Lisbon, 25th of April 2022

Index

1 Introduction

The asymmetric travelling salesman problem (ATSP) is a special case of the travelling salesman problem (TSP). It is a NP-hard combinatorial problem and is described as, given a set of n nodes and distances for each pair of nodes, find a roundtrip of minimal total length, visiting each node exactly once. In this case, the distance from node i to node j and the distance from node j to node i may be different. Concluding, the goal is to find a Hamiltonian tour of minimal length on a fully connected graph

In order to solve it, three different methods are implemented and tested against two benchmark results obtained in the TSPList [1]. The methods used are the nearest neighbour, tabu search and ant colony optimization. The benchmarks used are *ft53*, with 53 nodes and *kro124p*, with 100 nodes. The best known solution for each problem is 6905 and 36230, respectively.

2 Formulation of the problem

The ATSP has for long been researched and has several possible formulations. In this report, the problem will be formulated as a binary integer linear programming problem. Firstly, the nodes have to be enumerated from 1 to n, n being the number of nodes in the case study, followed by the cost for travelling from one node (i) to the other (j), c_{ij}. The cost function in this problem represents the distance total distance travelled and consists in the sum of costs from travelling from one node to another.

The asymmetric nature of the problem makes it possible for the cost of travelling the arc (i, j) to not be the same as the other way around, (j, i), i.e., $c_{ij} \neq c_{ji}$. These values are retrieved from a text document in matrix form, where c_{ij} is the value in row i and column j. As travelling from one node to the same node doesn't translate to anything in the problem, c_{ij} has a value far superior to the others, in order not to compromise the optimal solution, for every $i = 1, ..., n$.

A binary integer linear programming problem uses binary variables x_{ij}, defined by:

$$\mathbf{x_{ij}} = \begin{cases} 1 & \text{, if arc } (i,\, j) \text{ is part of the optimal solution} \\ 0 & \text{, if not} \end{cases} \tag{2.1}$$

And presented as the following model:

$$Min : \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.2}$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad i = 1, ..., n \tag{2.3}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, ..., n \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, ..., n \tag{2.5}$$

This model does not regard the possibility of sub-tours to appear in the solution. The solution must be a single tour passing through all nodes, not a union of smaller tours.

2.1 Sub-tour Elimination Constraints

To suppress the possibility of having sub-tours, a new constraint has to be added. Using the Dantzig–Fulkerson–Johnson formulation, the last constraint becomes:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset \{1, ..., n\} : S \neq \varnothing \tag{2.6}$$

This guarantees that no subset S can form a sub-tour.

3 Methods and Algorithms

In order to solve the problem, three different methods are implemented: nearest neighbour (NN), tabu search (TS) and ant colony Optimization (ACO).

The choice of algorithms was simple. To begin with, a simple algorithm is tested, the NN algorithm, since it was one of the first algorithms used to solve the TSP and produces a fast solution, although sometimes not the optimal, besides being simple to compute. Then, the TS algorithm is used, with the initial condition guessed with the NN, explained in section 3.2. Through this, it is possible to directly compare the effectiveness of the meta-heuristic algorithm. In the end, a different meta-heuristic algorithm is used, ACO, in order to have some diversification in the methods implemented.

3.1 Nearest neighbour

The nearest neighbour method is a simple algorithm that works with the following steps:

1. Start in a random node;
2. Determine the node with the smallest distance to the starting point and connect to the starting point;
3. The last node is now the new starting point;
4. Repeat the procedure, excluding the nodes already visited
5. If all nodes are visited once, the algorithm is done. If not, go back to step 3.

This type of algorithm has a tendency to become stuck in suboptimal regions, also known as local minimum.

In order to have a better comparison between results and since the problem being studied is relatively small (50 to 100 nodes), instead of starting in a random node, the algorithm was run ($n_{nodes} - 1$) times, so that all the nodes can be the starting point. The results were stored and will be presented in section 4.1.

3.2 Tabu search

Tabu search is a meta-heuristic algorithm based on a local search method. It explores the solution space by replacing recent solutions with the best non visited neighbouring solution. In order to avoid the local minimum problem, TS allows moves that give a worse solution if no improving move is available. Also, it maintains a list of neighbour generation moves that are considered forbidden and solutions that can be obtained through those moves are ignored. The stopping criteria can either be a maximum number of iterations/time or if there is no better solution in the previous iterations. The steps for this algorithm are:

1. Start with an initial solution, in this implementation it is obtained through the NN algorithm;

2. Generate alternative neighbouring solutions to the current solution, changing one node with the other (pairing new nodes), if these moves are in the Tabu list they are discarded;
3. Choose the best solutions out of the alternative ones found previously;
4. Update the Tabu list by removing the moves that expired the Tabu tenure and add the new moves obtained in step 3;
5. If the stopping criteria is reached, the algorithm is done. If not, go back to step 2.

While implementing TS algorithm, two parameters have to be taken into account:

- Tabu tenure (τ):

  The tabu tenure is a pre-specified number that regulates for how many iterations a move remains on the tabu list. Through this, the tabu list is always changing, making the algorithm less prone to get stuck in a local minimum. If $\tau = 0$ the algorithm doesn't store values in the list, making it similar to a NN lookalike algorithm. Tabu tenure can be static or dynamic. In this project, a static τ is used to simplify the algorithm.

- Neighbourhood size:

  The neighbourhood size reflects how big the searching area is when the algorithm is looking for alternative solutions. A bigger neighbourhood size is associated with a more accurate result, but a heavier computational burden.

3.3 Ant colony optimization

Ant colony optimization is a population-based meta-heuristic algorithm. A set of software agents called artificial ants incrementally build solutions by moving through a parameter space representing all possible solutions, in this case, on a weighted graph. The construction of the solution is a stochastic process biased by a pheromone model.

The steps to this iterative algorithm are:

1. An ant completes a tour (a solution), choosing nodes by a stochastic mechanism;
2. Pheromones are laid out according to the quality of the solution, in this case according to how small the distance was;
3. Another ant repeats the procedure;
4. The next ant selects the path to go, again via a stochastic mechanism, but this time biased by the pheromone quantity left in each node;
5. Update the pheromone quantity in each node as every ant tries a path;
6. If the majority of the ants is following the same path, the algorithm can stop and that path is the optimal one.

While implementing this algorithm, a few parameters have to be taken into the account, in this section is a brief explanation of them and in section 4.3 a comparison between them, in order to find optimal values.

- Number of ants (n_{ants}):

  The number of ants affects how many ants are trying new paths. Fewer ants may lead to insufficient exploration, and too many ants can cause an inefficient algorithm.

- Pheromone (τ):

  Pheromones can be considered the base element for the ant colony optimization, since the path taken by an ant is influenced by them. The number of τ increase in the arcs with higher affluence.

- Pheromone evaporation (ρ):

  Pheromone evaporation avoids unlimited accumulation of the pheromone trails leading to quick convergence to suboptimal path, reinforces good solutions and enables the algorithm to forget bad decisions previously taken. It is extremely important in early stages when ants provide various worse solutions.

- Pheromone constant (Q):

  The amount of pheromones laid. If this parameter is too high, it may lead to a quick convergence in a suboptimal path, if it is too low, it can take a long time to find an optimal path.

- Pheromone exponential weight (α):

  Controls the importance given to the pheromone trail. This means that a higher value of α gives more importance to the paths that have been already used, and they might get stuck to non-optimal solution, since the explorations is lower with higher α. However, with low values of α, the ants are continuously searching for new a new path even if they find a good one. This shows that a balance of α is important.

- Heuristic exponential weight (β):

  Controls the importance given to the heuristic information. Higher values of β leads to more exploration, and lower values of β may result in the algorithm finding suboptimal solutions. Its value has to be balanced with the value of α.

4 Results obtained

In this section, the results obtained for the three different algorithms implemented are shown. Note that for the NN and the TS algorithm, the best path has the index 0 as the first point, and $n_{nodes} - 1$ as the final point of the path.

4.1 Nearest neighbour results

For this algorithm, the only possible change is the starting point. So, as stated before, the program was run ($n_{nodes} - 1$) times so that all the nodes can be the starting point. In order to analyse the results, an average path is computed, consisting in the sum of all the paths (the costs in each iteration) divided by the number of times the program was run. Then, the best, the worst and the average results are compared against the benchmark result.

In the following table, the results for the first problem, with 53 nodes, are presented:

Table 4.1: Cost function results for 53 nodes with NN

Solution	Cost	Error
Best	8584	24.32%
Worst	10275	48.81%
Average	9319.87	34.97%
Benchmark	6905	–

The optimal path for this problem obtained with this algorithm is:

[19 – 18 – 17 – 16 – 15 – 52 – 48 – 49 – 50 – 51 – 29 – 28 – 25 – 26 – 27 – 7 – 5 – 8 – 6 – 9 – 33 – 31 – 30 – 0 – 3 – 2 – 1 – 41 – 43 – 42 – 46 – 45 – 44 – 34 – 32 – 21 – 20 – 39 – 35 – 40 – 37 – 36 – 10 – 12 – 14 – 13 – 11 – 38 – 4 – 22 – 47 – 23 – 24 – 19].

For the second problem, with 100 nodes, the results are the following:

Table 4.2: Cost function results for 100 nodes with NN

Solution	Cost	Error
Best	43316	19.56%
Worst	50587	39.63%
Average	46230.61	27.60%
Benchmark	36230	–

The optimal path for this problem obtained with the NN is:

[29 – 95 – 77 – 51 – 4 – 36 – 32 – 75 – 12 – 81 – 94 – 49 – 72 – 43 – 1 – 63 – 39 – 53 – 68 – 67 – 84 – 80 – 24 – 60 – 86 – 50 – 6 – 56 – 11 – 54 – 82 – 8 – 19 – 85 – 26 – 34 – 61 – 59 – 22 – 44 – 31 – 14 – 16 – 10 – 58 – 73 – 71 – 9 – 83 – 37 – 23 – 35 – 98 – 20 – 46 – 90 – 97 – 76 – 57 – 27 – 92 – 66 – 7 – 91 – 62 – 5 – 48 – 89 – 52 – 15 – 21 – 93 – 87 – 78 – 17 – 69 – 64 – 65 – 3 – 96 – 74 – 18 – 55 – 41 – 30 – 88 – 79 – 25 – 0 – 33 – 45 – 13 – 70 – 99 – 47 – 40 – 2 – 42 – 28 – 38 – 29].

Analysing both tables 4.1 and 4.2, it is possible to conclude that this algorithm doesn't have a good accuracy, since the average error to the benchmark result is around 30%. Beside of being a fast algorithm, the NN doesn't provide a solution close to the expected. This might be due to the algorithm getting stuck on local minimums.

4.2 Tabu search results

In order to have reliable results when analysing the performance of this algorithm, 10 trial runs are realized.

Different values of each parameter were tested and the best, worst and average costs of all the trial runs, as well as the error versus the benchmark result, were analysed.

For the problem with 53 nodes, the first parameters being tested are the n_{ite} and τ. The results are presented in the following table:

Table 4.3: Cost function results for 53 nodes with TS for different parameters of n_{ite} and τ

Solution	$n_{ite} = 50$, $\tau = 50$		$n_{ite} = 100$, $\tau = 50$		$n_{ite} = 200$, $\tau = 50$		$n_{ite} = 200$, $\tau = 10$	
	Cost	Error	Cost	Error	Cost	Error	Cost	Error
Best	8035	16.36%	7607	10.17%	7505	8.69%	7530	9.05%
Worst	8233	19.23%	8331	20.65%	8116	17.54%	8116	17.54%
Average	8121.8	17.62%	7947.3	15.09%	7796.9	12.92%	7799.4	12.95%
Benchmark	6905	–	6905	–	6905	–	6905	–

The criteria used to choose the best solution is to evaluate which set of parameters has the best average for the 10 runs and choose the best set found, which is the overall best for this simulation.

The optimal path for this problem obtained with this algorithm is:

[4 – 34 – 52 – 50 – 49 – 51 – 48 – 29 – 28 – 25 – 27 – 26 – 7 – 6 – 5 – 8 – 9 – 33 – 31 – 30 – 0 – 36 – 35 – 40 – 37 – 38 – 2 – 3 – 1 – 43 – 41 – 47 – 42 – 46 – 45 – 44 – 23 – 21 – 20 – 39 – 10 – 11 – 13 – 12 – 14 – 24 – 22 – 19 – 18 – 17 – 16 – 15 – 32 – 4].

The following figure presents the evaluation of the optimal cost and the average cost of the objective function as the number of iterations increase. It is possible to observe that, after 170 iterations, the solution converges.

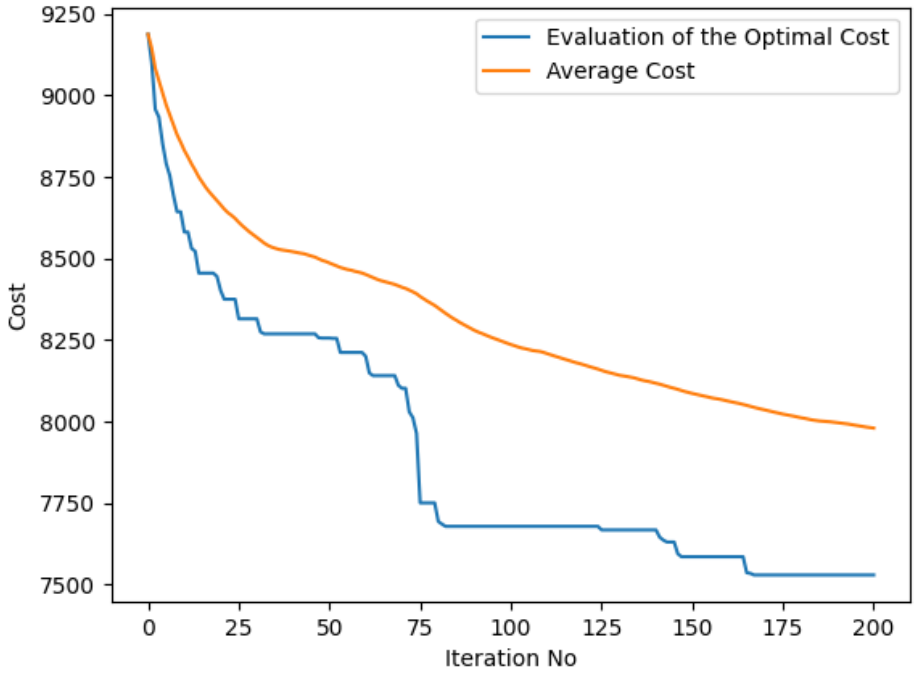

Figure 4.1: Variation of cost through iterations for 53 nodes with TS

For the second problem, with 100 nodes, the results are the following:

Table 4.4: Cost function results for 100 nodes with TS for different parameters of n_{ite} and τ

Solution	$n_{ite} = 50$, $\tau = 50$		$n_{ite} = 100$, $\tau = 50$		$n_{ite} = 200$, $\tau = 50$		$n_{ite} = 200$, $\tau = 10$	
	Cost	Error	Cost	Error	Cost	Error	Cost	Error
Best	41876	15.58%	41941	15.76%	40949	13.03%	40989	13.14%
Worst	43663	20.52%	42675	17.79%	42049	16.06%	42112	16.23%
Average	42697.6	17.85%	42422.8	17.09%	41636.4	14.92%	41749.4	15.23%
Benchmark	36230	–	36230	–	36230	–	36230	–

The criteria used in this simulation is identical to the one used on the previous problem. In this simulation, the best result is included on the parameters that provided the best average solution.

The optimal path for this problem obtained with the TS is:

[18 – 91 – 7 – 30 – 88 – 41 – 79 – 55 – 3 – 65 – 15 – 21 – 69 – 87 – 93 – 17 – 37 – 23 – 98 – 35 – 83 – 9 – 71 – 20 – 58 – 73 – 31 – 16 – 14 – 10 – 97 – 90 – 22 – 44 – 76 – 59 – 61 – 34 – 19 – 85 – 26 – 11 – 56 – 82 – 54 – 42 – 45 – 13 – 70 – 40 – 99 – 47 – 77 – 95 – 4 – 51 – 29 – 38 – 84 – 67 – 43 – 1 – 63 – 53 – 39 – 68 – 72 – 49 – 80 – 24 – 6 – 86 – 50 – 60 – 57 – 66 – 27 – 92 – 0 – 5 – 48 – 89 – 78 – 52 – 25 – 64 – 96 – 74 – 62 – 46 – 8 – 33 – 2 – 28 – 36 – 32 – 75 – 12 – 81 – 94].

The figure 4.2 presents the evaluation of the optimal cost and the average cost of the objective function as the number of iterations increase. It is possible to observe that, after 160 iterations, the solution converges.

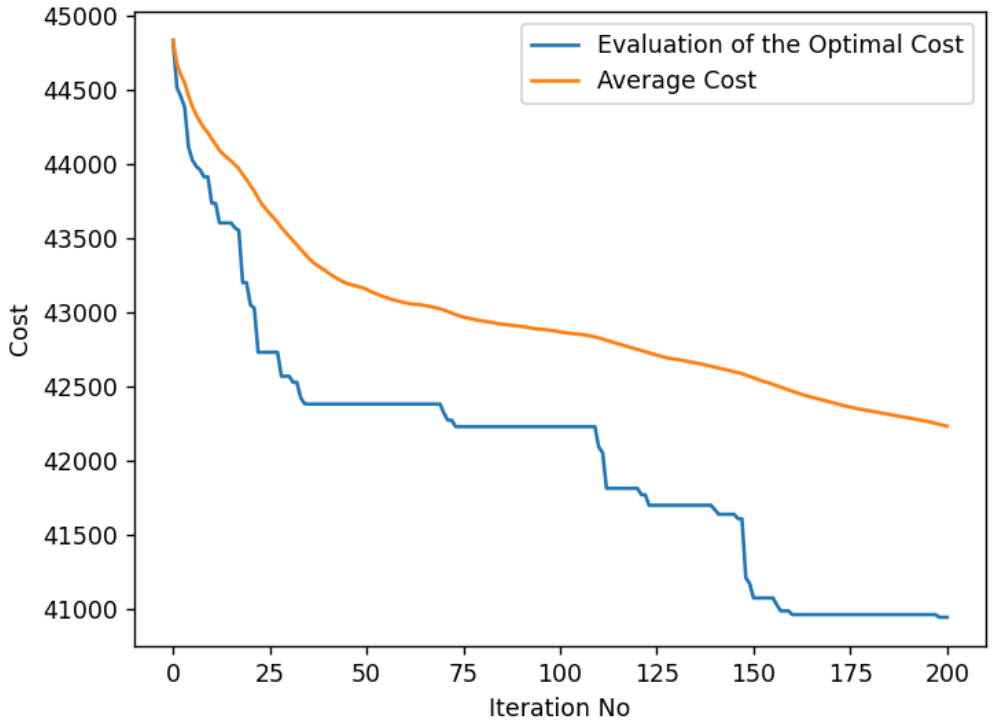

Figure 4.2: Variation of cost through iterations for 100 nodes with TS

Analysing both tables 4.3 and 4.4, it is possible to verify that, as the number of iterations increase, the error decreases. This is expected, since an increase of iterations gives the algorithm more tries to reach an optimal value. The only downside is the bigger computational effort needed. Changing the tabu tenure leads to almost no improvement in both problems. Since there is a big enough n_{ite}, changing the time that a move stays in the tabu list won't affect the final solution due to the fact that the algorithm has enough iterations to explore all the space. A greater value of τ makes the algorithm explore more possible routes, and a lesser value makes it explore less, that translates into a faster converging algorithm with the cost of the possibility of getting stuck in a local minimum, if it is too small.

4.3 Ant colony optimization results

Within this algorithm, a variety of parameters can be tuned in, with the objective of obtaining an optimal solution. All the following values are an average of the results of 10 trial runs.

To begin with, the first problem with 53 nodes (nodes), the number of iterations, n_{ite}, and the number of ants, n_{ant}, are tested with fixed values of $\alpha = 2$, $\beta = 5$, $\rho = 0.3$ and Q=1. The results are shown in table 4.5.

Table 4.5: Cost function results for 53 nodes with ACO for different n_{ite} and n_{ant}

Solution	$n_{ite} = 100$, $n_{ant} = 50$		$n_{ite} = 100$, $n_{ant} = 100$		$n_{ite} = 150$, $n_{ant} = 150$	
	Cost	Error	Cost	Error	Cost	Error
Best	8447	22.32%	8306	20.29%	8223	19.09%
Worst	10256	48.53%	10149	46.98%	10226	48.10%
Average	9098	31.76%	8873.7	28.51%	8907.8	29.01%
Benchmark	6905	–	6905	–	6905	–

As shown in table 4.5, as the number of ants and iterations increases, the associated costs and errors for each test go down, as expected.

The simulation with the largest set of ants ($n_{ant} = 150$) and iterations ($n_{ite} = 150$), produces the best overall result.

Then, the evaporation rate, ρ, is tested. The following parameters are fixed: $n_{ite} = 50$, $n_{ant} = 100$, $\alpha = 2$ and $\beta = 5$. The results are shown in table 4.6.

Table 4.6: Cost function results for 53 nodes with ACO for different values of ρ

Solution	$\rho = 0.1$		$\rho = 0.3$		$\rho = 0.7$	
	Cost	Error	Cost	Error	Cost	Error
Best	8345	20.85%	8447	22.32%	8533	23.58%
Worst	10184	47.49%	10256	48.53%	10525	52.43%
Average	9074.1	31.41%	9098	31.76%	9127.9	32.19%
Benchmark	6905	–	6905	–	6905	–

The evaporation rate, ρ, simulates how fast the pheromones in an arc will disappear after an iteration. The lower this rate, the higher will be the remaining pheromones in the arc, hence, more ants will have the tendency to go through that arc. In table 4.6, the previous statement is confirmed, as the lowest ρ produces the best solution from three tests.

In the end, the α and β parameters are tested, with fixed values of $n_{ite} = 50$, $n_{ant} = 100$ and $\rho = 0.3$. The results obtained are presented in table 4.7:

Table 4.7: Cost function results for 53 nodes with ACO for different values of α and β

Solution	$\alpha = 0$, $\beta = 5$		$\alpha = 2$, $\beta = 0$	
	Cost	Error	Cost	Error
Best	9523	37.91%	24930	261.04%
Worst	12460	80.45%	26602	285.26%
Average	10906	57.94%	25864	274.57%
Benchmark	6905	–	6905	–

Through the two tests, presented in table 4.7, it is possible to confirm the importance of tuning α and β, since the results obtained are not good.

Firstly, for $\alpha = 0$, the ants only search following the heuristic procedure, this means that no pheromones are stored. The major problem with this is the convergence time, since the best paths don't get memorized, there is no reinforcement.

Then, for $\beta = 0$ the results are even worse. This is due to the fact that the ants don't explore different paths, and they get stuck in local minimums.

For all the simulations, the best single solution has a cost of 7855, for $n_{ite} = 150$, $n_{ant} = 150$, $\rho = 0.3$, $\alpha = 2$, $\beta = 5$ and Q=1.

The tour that provided this result is:

[6 – 52 – 49 – 50 – 53 – 51 – 30 – 29 – 26 – 28 – 27 – 4 – 3 – 18 – 17 – 16 – 40 – 39 – 37 – 36 – 41 – 38 – 19 – 20 – 12 – 11 – 13 – 15 – 14 – 35 – 33 – 32 – 34 – 31 – 5 – 2 – 9 – 7 – 10 – 8 – 22 – 21 – 25 – 23 – 48 – 43 – 47 – 44 – 42 – 46 – 45 – 24]

The following figure plots the best and average solution of that simulation.

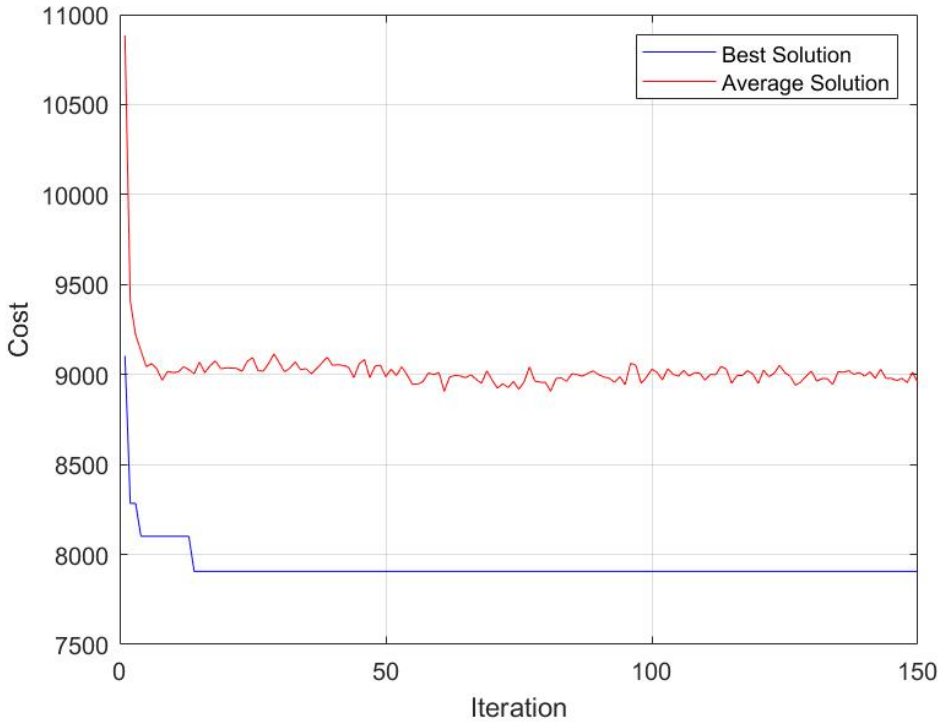

Figure 4.3: Best and Average costs for best solution for 53 nodes with ACO

Analysing figure 4.3 it is possible to observe that, after 20 iterations, the results converge.

For the second problem, with 100 nodes, the same procedure is practised, with fixed values of $\alpha = 2$, $\beta = 5$, $\rho = 0.3$ and Q=1, changing n_{ite} and n_{ant}. The results are presented in table 4.8.

Table 4.8: Cost function results for 100 nodes with ACO for different n_{ite} and n_{ant}

Solution	$n_{ite} = 50$, $n_{ant} = 100$		$n_{ite} = 100$, $n_{ant} = 100$		$n_{ite} = 150$, $n_{ant} = 150$	
	Cost	Error	Cost	Error	Cost	Error
Best	43496	20.06%	42394	17.01%	41536	14.65%
Worst	49478	36.57%	47965	32.39%	47951	32.35%
Average	45564	25.76%	44461	22.72%	43980	21.39%
Benchmark	36230	–	36230	–	36230	–

Similarly to the problem with 53 nodes problem, table 4.8 demonstrates the lowering of the cost and error as n_{ite} and n_{ant} are increased. The simulation with the largest set of ants ($n_{ant} = 150$) and iterations ($n_{ant} = 150$), produces the best overall result, again.

Then, the evaporation rate, ρ is tested. For this, the following parameters are fixed $n_{ite} = 50$, $n_{ant} = 100$, $\alpha = 2$ and $\beta = 5$. The results are shown in table 4.9.

Table 4.9: Cost function results for 100 nodes with ACO for different values of ρ

Solution	$\rho = 0.1$		$\rho = 0.3$		$\rho = 0.7$	
	Cost	Error	Cost	Error	Cost	Error
Best	42753	18.00%	43496	20.05%	42936	18.51%
Worst	48465	33.77%	49478	36.57%	49137	35.63%
Average	44815	23.70%	45564	25.76%	45331	25.12%
Benchmark	36230	–	36230	–	36230	–

For this problem, the best result for $\rho = 0.1$ and $\rho = 0.7$ is practically the same, however, the worst and average values start to diverge. This shows the importance of not only focusing on the best solution.

Again, the α and β parameters are tested with fixed values of $n_{ite} = 50$, $n_{ant} = 100$ and $\rho = 0.3$. The results obtained are presented in table 4.10:

Table 4.10: Cost function results for 100 nodes with ACO for different values of α and β

Solution	$\alpha = 0$, $\beta = 5$		$\alpha = 2$, $\beta = 0$	
	Cost	Error	Cost	Error
Best	49021	35.31%	186130	413.75%
Worst	61713	70.33%	192950	432.57%
Average	55259	52.52%	189540	423.16%
Benchmark	36230	–	36230	–

Similarly to the first problem, the solutions found when $\alpha = 0$ and $\beta = 0$ are worse than the rest. In the first test, the biggest concern is the convergence time, since there is no reinforcement on good paths. In the second test, the ants get stuck in local minimums

For all the trial runs, the best single solution has a cost of 39026, for $n_{ite} = 150$, $n_{ant} = 150$, $\rho = 0.3$, $\alpha = 2$, $\beta = 5$ and Q=1.

The tour that provided this result is:

[34 – 83 – 55 – 27 – 20 – 86 – 12 – 7 – 57 – 51 – 87 – 9 – 35 – 62 – 60 – 77 – 98 – 91 – 23 – 45 – 32 – 15 – 11 – 17 – 21 – 59 – 74 – 72 – 10 – 84 – 38 – 24 – 36 – 99 – 18 – 79 – 53 – 16 – 22 – 94 – 88 – 70 – 65 – 66 – 4 – 26 – 56 – 42 – 31 – 89 – 80 – 97 – 75 – 19 – 90 – 49 – 6 – 63 – 47 – 1 – 92 – 8 – 67 – 28 – 93 – 58 – 61 – 81 – 25 – 73 – 50 – 44 – 2 – 64 – 40 – 54 – 69 – 68 – 85 – 30 – 96 – 78 – 52 – 37 – 5 – 13 – 76 – 33 – 95 – 82 – 39 – 48 – 14 – 71 – 100 – 41 – 3 – 43 – 46 – 29]

The following figure plots the best and average solution of that simulation.

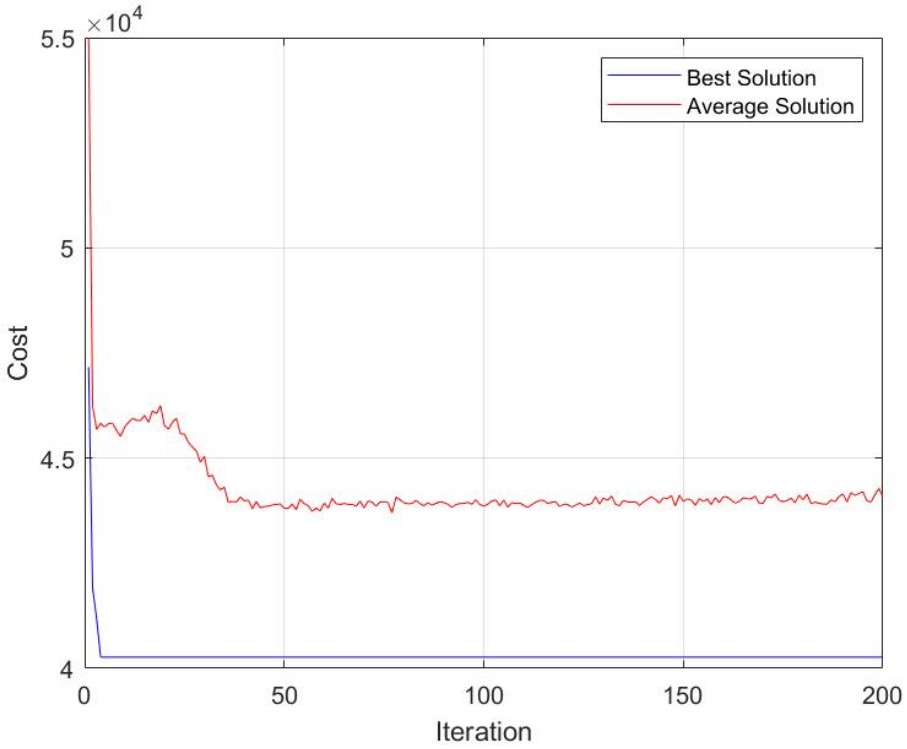

Figure 4.4: Best and Average costs for best solution for 100 nodes with ACO

Analysing figure 4.4 it is possible to observe that, after 10 iterations, the results converge.

Varying all the parameters in ACO causes different results with different meanings. The best solution for each problem is found with the best values for the parameters ρ, α, β and Q, in a number of simulations, iterations, and ants that could be handled in a decent time frame.

5 Comparison of results

In this section, a comparison between the solutions obtained when using the different methods is done in order to discuss which method is the most adequate to solve the asymmetric travelling salesman problem.

Firstly, the nearest neighbour and tabu search are compared. After that, a comparison between the tabu search and ant colony optimization is made.

5.1 Nearest neighbour and Tabu search

A direct comparison between the first two methods implemented is possible, since the TS algorithm uses the NN in order to obtain its first solution.

As shown in figures 5.1 and 5.2, the best average cost for the nearest neighbour algorithm is higher in both cases. This shows that the second algorithm is better for the problem, considering the used sample.

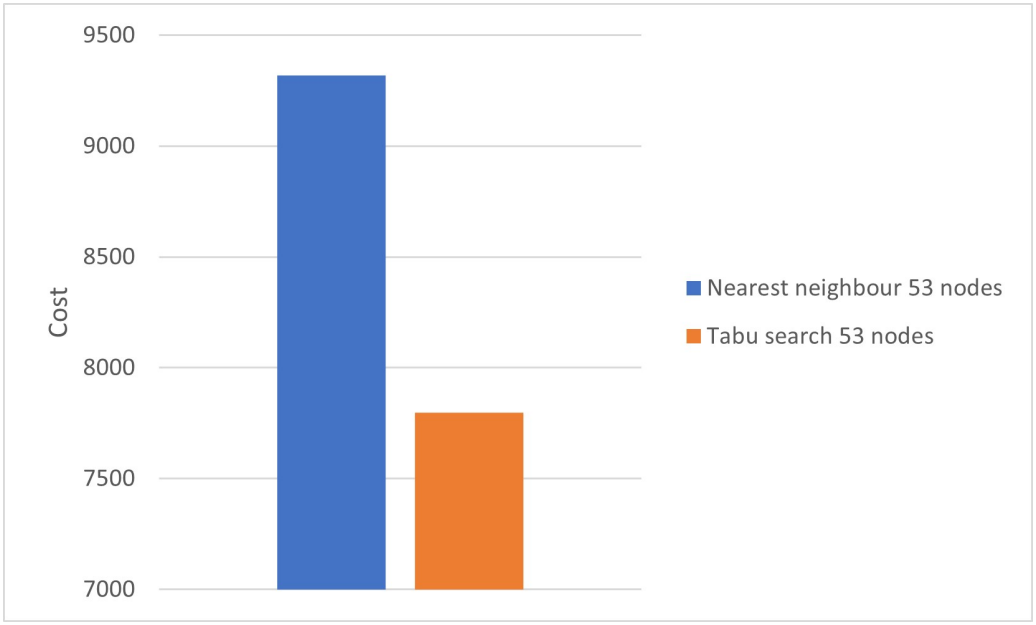

Figure 5.1: Average solution of Nearest Neighbour vs Tabu Search for 53 nodes

For this first problem, the improvement of TS over NN is 16.34%.

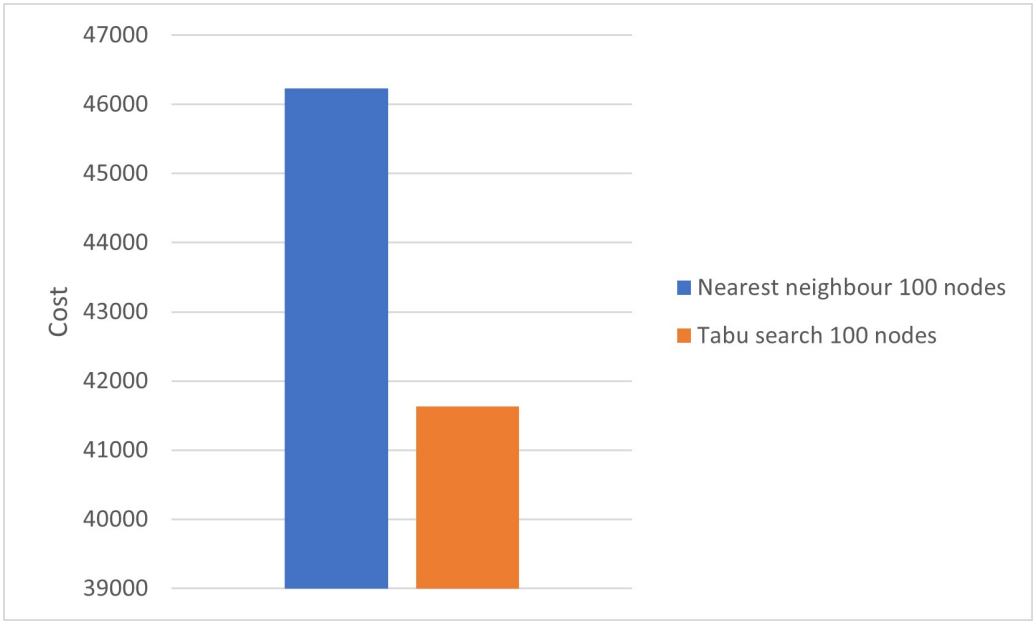

Figure 5.2: Average solution of Nearest Neighbour vs Tabu Search for 100 nodes

For the second problem, the improvement of TS over NN is 9.94%. Both problems have a significant improvement in the average cost when the tabu search algorithm is used.

At a first glance, one might think that the nearest neighbour algorithm uses a close solution for a specific starting position because it is trying to move towards the closest point in every iteration. However, with tabu search, that hypothesis is no longer valid, because moving towards the nearest point could give a longer route in a later move.

On TS, every iteration changes some positions to see if the route is improved and store those positions in the tabu list.

At the end, this shows that, sometimes, choosing a more expensive route on one move, may result in a cheaper total tour.

5.2 Tabu search and Ant colony optimization

First of all, through the results obtained it is possible to conclude that the two meta-heuristics being studied, present better solutions for the problems, in comparison with nearest neighbour algorithm.

The figures 5.3 and 5.4, demonstrate the best solution found in both methods for both problems, where the TS outranks the ACO in the first problem, however, for the second problem, ACO illustrates a better solution.

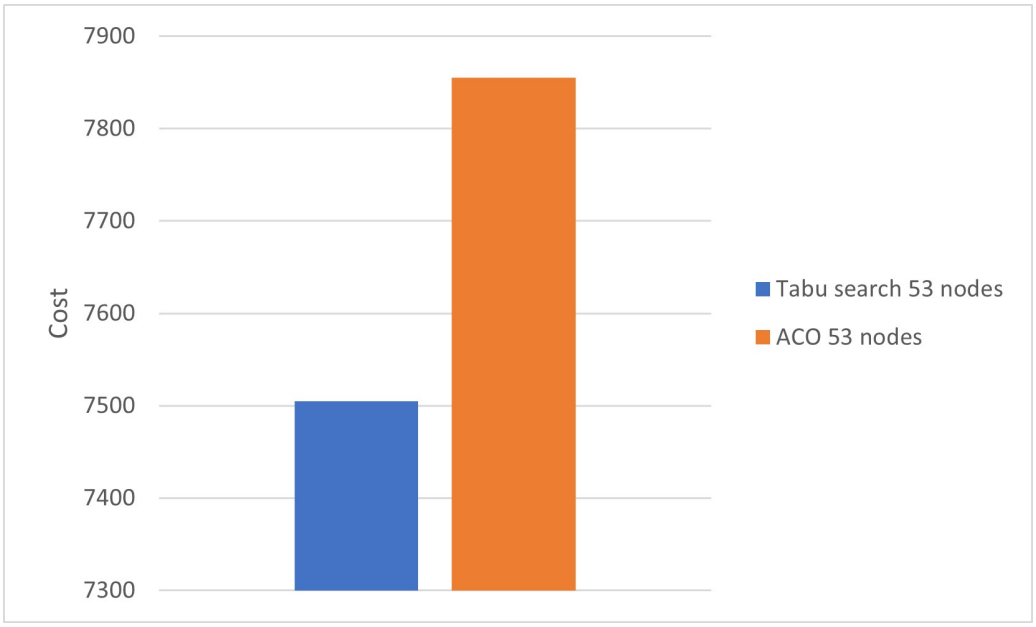

Figure 5.3: Comparison between Tabu search vs Ant Colony Optimization for 53 nodes

For this first problem, the improvement of TS over ACO is 4.66%.

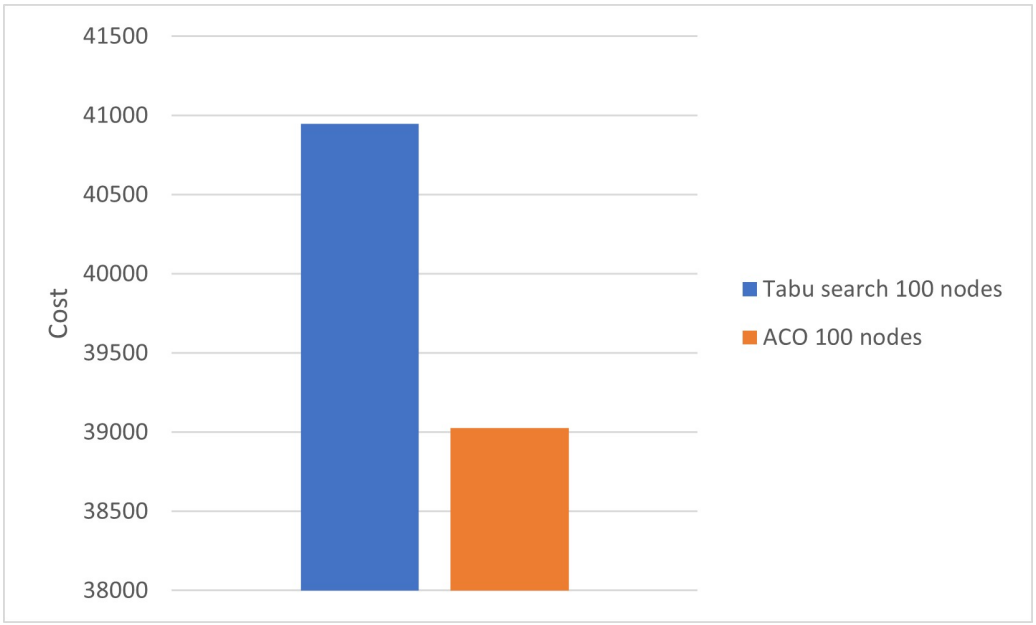

Figure 5.4: Comparison between Tabu search vs Ant Colony Optimization for 100 nodes

For the second problem, the improvement of ACO over TS is 4.93%.

Depending on the quantity of nodes, tabu search or ant colony optimization may present better solutions.

6 Conclusion

Applying the three methods to both problems, the results of each one, in spite of being different, follow the same trend. Changing some parameters within them leads to the similar alterations in the final solution.

Both meta-heuristics algorithms used, tabu search and ant colony optimization, produced the best results in terms of the solution itself.

The nearest neighbour method has the advantage of being the one with the lowest running time. The solution that it gives is far from optimal, but still acceptable. It is possible to conclude that, if a better solution is needed, this algorithm must be paired with others, for instance, the tabu search algorithm.

About the tabu search algorithm, it is clear that a fine-tuning of the parameters is required in order to have good solutions. After this procedure, this algorithm is extremely capable of delivering very good solutions with an acceptable running time. Due to this acknowledgement, it is possible to conclude that this algorithm is useful when dealing with the ATSP problem.

In the other hand, the ant colony optimization method has a significant running time that can be an issue for even bigger problems than the ones dealt in this report. Even tough, it can be argued that, with enough time, it produces the best solution out of the three methods described.

References

[1] Ruprecht-Karls-Universität Heidelberg. *TSPLib*. URL: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`. (accessed: 10.04.2022).

[2] Dorigo Marco and Stützle Thomas. *Ant Colony Optimization*. MIT Press. 2004.

[3] Ilyas Masudin et al. "Capacitated Vehicle Routing Problems: Nearest Neighbour vs. Tabu Search". In: *International Journal of Computer Theory and Engineering* (2019).

[4] M Vieira Susana. *Course notes*. 2022.