Yuthika Tia Harripersad

u23562732

COS 314 Assignment 1

1.Initial solution generation method

The initial solution for both the Simulated Annealing and Tabu Search methods is generated by randomly ordering the cities given.

2.Perturbation method

The perturbation method used for both searches is swapping two cities. A copy of the current solution is made and is adjusted slightly so that a neighbour can be created. This adjustment involves generating 2 random numbers, which is divided by the number of cities. The remainder of this value is taken for both numbers and used as the first and second cities to swap. If the second index is equivalent to the first index, the second index is re-generated until it is no longer equivalent to the first index. This ensures that the neighbour will always be different from the current solution. Both indexes are re-generated if they are equal to 0, as the start and end of the tour must be the first node.

3.Neighbourhood definition

The neighbourhood is a set of solutions which are similar to the current solution. In the travelling salesman scenario, the neighbourhood of a solution is any solution where there are only two changes when compared to the current solution. This change is due to the swapping of two random cities which are not the same. However, all solutions must also start with node 1. This definition applies to both searches.

4.Acceptance criterion

The Simulated Annealing uses the metropolis criterion and determines if the solution is accepted by calculating if the cost of the neighbour is less than the cost of the current solution. If the cost of the neighbour is less than the cost of the current solution, then it is always accepted. However, if the cost of the neighbour solution is greater or equal to the current solution, then we can accept or reject this solution based on a probability rule. A random value between 0 and 1 is generated. If this random value is less than or equal to the exponential value of the ((-1*difference in cost between current solution and neighbour)/current temperature), then the neighbour is accepted as a solution. If the neighbour is not accepted, it is rejected.

The Tabu Search bases the acceptance criterion of the cost of the solution. If the solution has a cost lower than the cost of the candidate solution, then it is accepted. If otherwise, the solution is rejected.

5.Stopping criteria

For the Simulated Annealing, if no improvements are made for more than the maximum iterations, then the search algorithm must stop and can return the current best solution.

For the Tabu Search, the search algorithm stops when the maximum iterations specified has been met.

6.Experimental setup

The Simulated Annealing requires the initial temperature and maximum iterations to be initialised to appropriate values. For the Travelling salesman problem, the initial temperature was calculated by generating the first 10 neighbours and calculating the difference in cost between each neighbour and the initial solution. All these differences were added together to get a total difference. This total difference in cost was then divided by 10 to get the average cost difference. The average cost difference was multiplied by 1.8, to provide an initial temperature slightly higher than the average cost difference. This value was set as the initial temperature.

The cooling schedule for the temperature continually decreased the temperature by multiplying it 0.95 after each iteration.

The maximum number of iterations for Simulated Annealing was defined as an integer and given the value of 500 multiplied by the number of cities.

The Tabu Search requires a maximum number of iterations and the maximum length of the tabu list to be defined. A user must enter the value of the max iterations as a parameter when the search function is called. For the values generated in the table below, the maximum iterations were equivalent to the number of cities multiplied by 100. The maximum length of the tabu list changes according to the number of cities. The maximum length is equivalent to the number of cities multiplied by 2.

7. Table

Problem instance	Algorithm	Seed value	Cost	Best solution	Runtime
8 cities	Tabu SA	1742666829 1742666829	286.502 286.502	1,4,8,6,2,5,7,3 1,3,7,5,2,6,8,4	7.21836 ms 20.0448 ms
12 cities	Tabu SA	1742668281 1742667964	351.33 378.811	1,12,6,11,8,2,10,4,9,7,3,5 1,5,3,7,9,8,11,2,4,10,6,12	11.8085 ms 46.3202 ms
15 cities	Tabu SA	1742668417 1742669716	386.989 462.613	1,7,3,10,12,5,8,2,6,15,13,4,11,14,9 1,7,3,10,12,5,8,9,11,4,13,6,2,15,14	12.5889 ms 103.112 ms
20 cities	Tabu SA	1742668976 1742669343	376.363 539.671	1,9,16,3,14,8,7,17,2,5,20,13,11,4,18,15,10,12,19,6 1,6,10,19,14,3,12,2,8,17,7,5,13,11,20,4,15,18,16,9	15.4735 ms 85.8555 ms
25 cities	Tabu	1742667127	445.034	1,24,21,16,11,14,4,8,23,18,6,10,25,20,13,15,2,17,22,19,5,9,7,12,3	21.4933 ms

	SA	1742669142	739.08	1,21,16,6,23,10,20,15,2,17,22,4,8,11,14, 19,13,5,25,18,9,7,12,24,3	113.002 ms

8.Analysis

For smaller problem instances like 8 cities, both searches are able to get the same cost and best solution. However, the Simulated Annealing is seen to take almost three times the time that the Tabu Search took.

For 12 ,15 and 20 cities, the Tabu search has a lower cost than the Simulated annealing, resulting in different best solutions. The Tabu Search also takes much less time to get the lower cost for all three instances.

For 25 cities, there is a clear performance difference with the Tabu Search returning a cost that has a large difference from the cost returned by the Simulated Annealing, as well as a much lower runtime.

The Tabu Search's runtime is seen to increase gradually as the problem instance becomes larger, with the largest gap in runtime being between the 20 and 25 cities. The Simulated Annealing's runtime increases at a much larger rate with the runtime of the 15 cities problem instance being more than double the runtime of the 8 cities problem instance.

Even though both searches are able to find the best solution with the same cost for smaller problem instances, the Tabu Search is still faster and more efficient. The Tabu Search is especially seen to perform better in larger problem instances, as Simulated Annealing is more computationally expensive and therefore slower. Even though Simulated Annealing takes more time, it was not able to provide better solutions than the Tabu Search.

The performance of the Tabu Search is a result of using the tabu list, which avoids using solutions which have already been generated previously. This leads to a faster way to find better solutions, however the requirement to determine if the neighbour has been generated increases the runtime by a bit. The performance of the Simulated Annealing is a result of its acceptance criterion, initial temperature and max iterations. Due to the possibility of accepting worse solutions, the search can become slower as it explores more of the search space. This possibility can lead to better solutions being found, however when compared to the Tabu Search, the solutions found for the current larger problem instances do not appear to be as good.

During the implementation of the local searches, both searches were seen to rely on their initial configurations and parameters to improve their performance. The Simulated Annealing required well thought out maximum number of iterations, initial temperature and cooling schedule. If the maximum number of iterations or initial temperature were too low, worse solutions would be returned. The Tabu Search required an efficient stopping criterion, like the maximum number of iterations, to allow the search to return good solutions.

In conclusion, the Tabu Search is seen to perform better in all scenarios than Simulated Annealing, however both searches will not always return the absolute best solution that exists.