CM3109 – Combinatorial Optimization Coursework

C1722325 – Spyros Lontos

Task 1:

A definition of a neighborhood N specific to our weighted tournament problem is a swap amongst 2 participants in the ranking list. In order to find a neighboring solution, a random choice is picked from the list that contains all of the weights and edges which are stored in the format [[16,9,25], [14,9,27], [14,9,6],…]. Each element [a, b, c] contains the weight amongst the 2 participants (a), the winner amongst the 2 participants (b) and the loser (c). By comparing the positions of (b) and (c) in the ranking we can find out if it agrees with the chosen elements. If Participant C is ranked higher than Participant B, we perform a swap in the ranking amongst those 2. This new ranking sequence after the swap has a new different Kemeny score which can be compared to the previous ranking.

The initial solution for the participants ranking is the same as they were stored in the 'Formula_One_1984.wmg' file and list of weights and edges are found after the 38th line in the same file. As an example of finding a neighbouring solution we first find a randomly chosen element in the list of all the weights and edges. Let's say this random choice gave us [14, 9, 6] which can be seen in line 40 of the given file. We compare the position of driver with number 9 which corresponds to Niki Lauda with the position of driver with number 6 which corresponds to Nigel Mansell. From the element that is randomly chosen we can see that number 9 finished ahead of number 6 with a weight of 14. This means that number 9 should be ranked higher than number 6. Which is not the case with our initial solution. Since the element does not agree with the participants' ranking, we swap their position put Niki Lauda a Rank 6 and Nigel Mansell as Rank 9. Additionally, if the randomly chosen element did agree with the ranking, we would with recursion choose another random choice until we find one that did not agree.

Firstly, we find the initial Kemeny score from the initial ranking solution that is given in the file 'Formula_One_1984.wmg'. Since we had been given the weights between the individual participants we iterate through the list of the weights/edges and see if the ranking solution agrees with each one. When the Participants do not agree with the ranking solution the weight is added to the Kemeny score. When all the weights/edges have been compared the result is the Kemeny score for that specific's Ranking Solution.

A few steps are needed in order to compute the cost of the neighbouring solution R2 from the cost of the current solution R2. Initially, we would find the 2 participants that would be swapped. Then we would find the Participants that are ranked between them. Would then then compare each Participant that was swapped with each of the Participants that were between them and add to the Kemeny score where the ranking/weights/edges do not agree and subtract from the score where it does agree. We would do this with both swapped drivers and we would be returned with the new Kemeny score after the swap has been performed.

An example for this would be:
Let's say we have the initial solution, and the randomly picked element containing the weight/edges has participant 10 and 13. We would need to compare Participant 10 with Participants 11,12. Where the 2 participants e.g.(10, 11) used to agree and their weight wasn't added to the Kemeny score now it will need to be added. And where they used to not agree, and their weight had been added to the Kemeny score now it will be subtracted. After doing it for both chosen participants the new Kemeny score will be created representing the new Ranking solution.

Task 3:

Best Choice of Parameters

The optimal solution for each parameter that I have found are:

- Temperature Length (TL): 7000
- Initial Temperature (T I): 1000
- a: 0.998
- num_non_impove: 1200

```
# Simulated Annealing Parameters
lengthTemp = 7000
initTemp = 1000
a = 0.998
num_non_improve = 1200
```

For these "best" choice of parameters I consistently get very close to the global minimum score in a relatively short amount time. Following is a screenshot of the output of my program while using these chosen parameters. It is split in half and shown side by side since the output is too long vertically.

```
Stopping criterion reached
      -      Final Ranking      -
---------------------------------------
| Position: 1   | Niki Lauda          |
---------------------------------------
| Position: 2   | Alain Prost         |
---------------------------------------
| Position: 3   | Elio de Angelis     |
---------------------------------------
| Position: 4   | Rene Arnoux         |
---------------------------------------
| Position: 5   | Corrado Fabi        |
---------------------------------------
| Position: 6   | Derek Warwick       |
---------------------------------------
| Position: 7   | Michele Alboreto    |
---------------------------------------
| Position: 8   | Nelson Piquet       |
---------------------------------------
| Position: 9   | Patrick Tambay      |
---------------------------------------
| Position: 10  | Andrea de Cesaris   |
---------------------------------------
| Position: 11  | Mauro Baldi         |
---------------------------------------
| Position: 12  | Thierry Boutsen     |
---------------------------------------
| Position: 13  | Teo Fabi            |
---------------------------------------
| Position: 14  | Riccardo Patrese    |
---------------------------------------
| Position: 15  | Nigel Mansell       |
---------------------------------------
| Position: 16  | Stefan Johansson    |
---------------------------------------
| Position: 17  | Jo Gartner          |
---------------------------------------
| Position: 18  | Gerhard Berger      |
---------------------------------------
| Position: 19  | Keke Rosberg        |
---------------------------------------
| Position: 20  | Ayrton Senna        |
---------------------------------------
| Position: 21  | Eddie Cheever       |
---------------------------------------
| Position: 22  | Marc Surer          |
---------------------------------------
| Position: 23  | Jonathan Palmer     |
---------------------------------------
| Position: 24  | Martin Brundle      |
---------------------------------------
| Position: 25  | Huub Rothengatter   |
---------------------------------------
| Position: 26  | Jacques Laffite     |
---------------------------------------
| Position: 27  | Piercarlo Ghinzani  |
---------------------------------------
| Position: 28  | Stefan Bellof       |
---------------------------------------
| Position: 29  | Francois Hesnault   |
---------------------------------------
| Position: 30  | Manfred Winkelhock  |
---------------------------------------
| Position: 31  | Philippe Streiff    |
---------------------------------------
| Position: 32  | Johnny Cecotto      |
---------------------------------------
| Position: 33  | Philippe Alliot     |
---------------------------------------
| Position: 34  | Pierluigi Martini   |
---------------------------------------
| Position: 35  | Mike Thackwell      |
---------------------------------------
Minimum Kemeny Score: 63
Algorithm's Runtime: 3600 ms
Number of Uphill moves: 883

Process finished with exit code 0
```

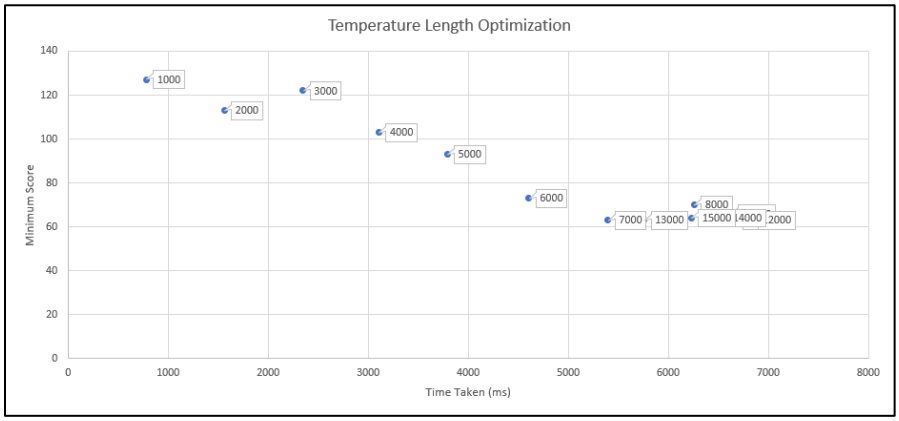
Summary of Results

To find optimal solutions for each Simulated Annealing parameter I needed to test them on different range of values. My testing criteria was finding the minimum cost and the lowest time taken for the algorithm's runtime at those different set of values which would show optimality for each specific parameter. Firstly, for the temperature length I varied the values beginning from 1000 to 15000 with increments of 1000. The results found did show variation, but I needed something more specific. So, I ran the same tests but with temperature length varying from 7000 up to 12000 with increments of 500. Moreover, for Initial Temperature I used values from 100 up to 4900 with increments of 400 but as before I just needed for specific results that could show differences amongst smaller increments, so I opted for a dataset ranging from 1000 to 2000 with increments of 100. Furthermore, for parameter 'a' I used a dataset beginning from 0.980 up to 0.999 with increments of 0.001. Lastly, for the num_non_improve number my chosen range of values was from 400 up to 1200 with increments of 100. By testing each of those ranges for the SA parameters I was able to find enough evidence to identify the optimal solutions for each one. The parameter's variation which had the biggest effect on the output solution was the num_non_improve. This variation resulted in either causing a premature finish of the algorithm before it managed to reach the minimum score or an unreasonable increase in the algorithm's runtime since there where too many iterations where no improvement was made. These variations had a significant effect on the final output so an optimal solution for num_non_improve had to provide an optimal minimum value for the Kemeny score as well as having the shortest amount of runtime.

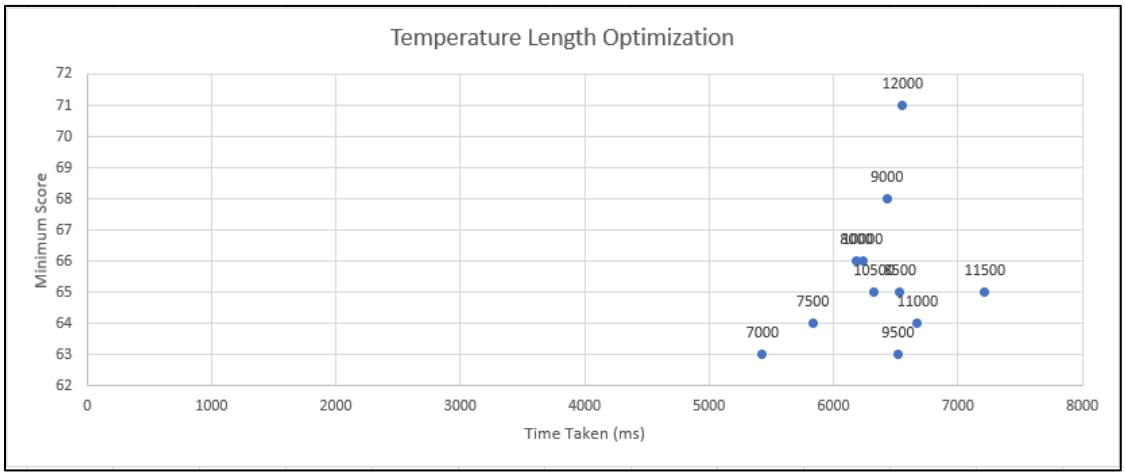
Deeper Analysis for varying SA parameters

Temperature Length

To obtain the more optimal solution for Temperature Length the I ran the Simulated annealing algorithm with different temperature lengths. In order to find this optimal solution, I did not change any other SA parameters. I set "Initial Temperature" as 1000, "a" as 0.999 and the "number_non_improve" at 2000. This gave me results for both Minimum Cost found and Time Taken for the algorithm's runtime at the different temperature lengths which began from 1000 up to 15000 with increments of 1000. I then plotted the results on a Scatter graph. Our optimal solution would be the one with the lowest minimum score and the least Time Taken. From the results taken I can observe that only after the temperature length exceeds 7000, I begin finding the Lowest minimum Score.



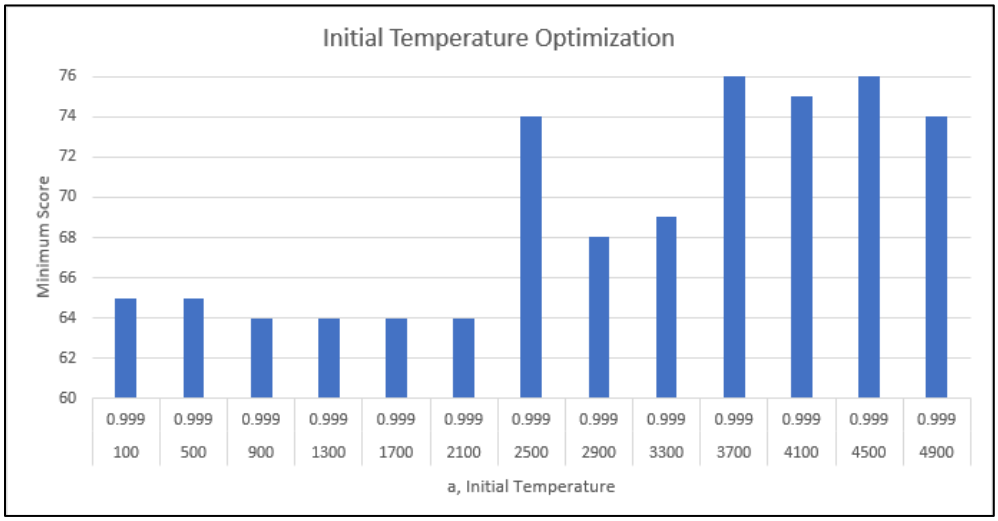
Since the results above 7000(time taken) are very close together it was best to create a new test dataset with different temperatures lengths beginning from 7000 up to 12000 with increments of 500. These will give us results that will clearly show the optimal solution for the Temperature Length.



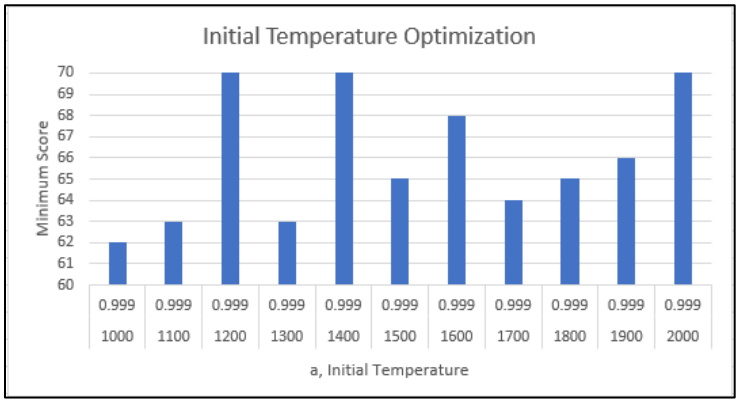
From the results we gathered on the new dataset on varying Temperature Lengths we can see that most of our results lie close to our expected minimum score 63-66. The more efficient ones in relation to the time taken where temperature lengths 7000 and 7500. With these results we can conclude that the optimal solution for the **Temperature Length is 7000**.

Initial Temperature

To discover the more optimal solution for the Initial Temperature I ran the Simulated annealing algorithm with a set of different initial temperatures. The set SA parameters that were not changed during the tests were “Temperature Length” set at 7000, “a” at 0.999 and “num_non_improve” at 2000. For the set of the initial temperatures I began from 100 up to 4900 with increments of 400.



From these results it is obvious that the initial temperature should be below 2100 in order to get an optimal solution for the minimum score. To conclude to a more optimal solution I will run the same tests on an Initial Temperature dataset ranging from 1000 to 2000 with increments of 100.

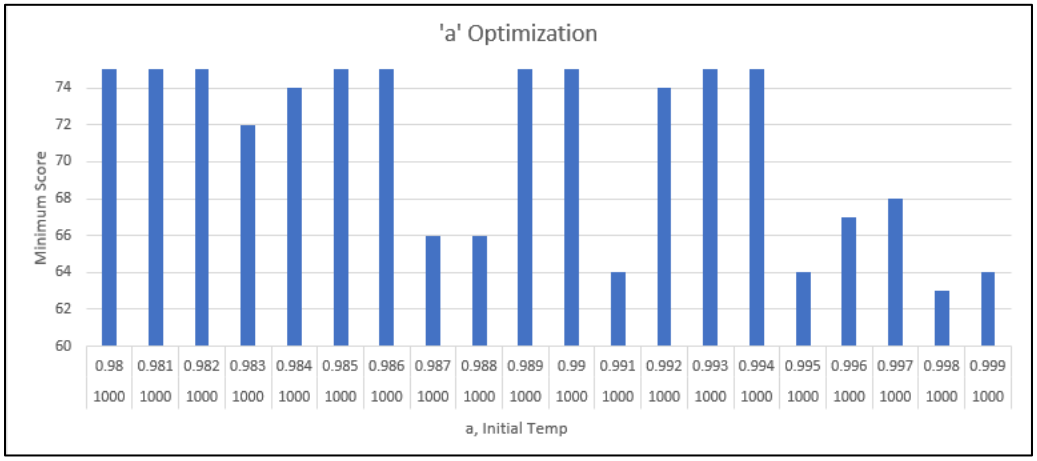


Although there are several Initial Temperature values give a close to optimal result for the minimum score, the best found was **Initial Temperature set at 1000**. It gives us a minimum score result of 62 which is the lowest cost found.

Fixed Parameter 'a'

Cooling schedule consists of 2 parameters, 'a' which is a fixed parameter and the f(t). Since we found the optimal Initial Temperature now, we need to find an optimal solution for 'a' which would give us the best completed cooling schedule.

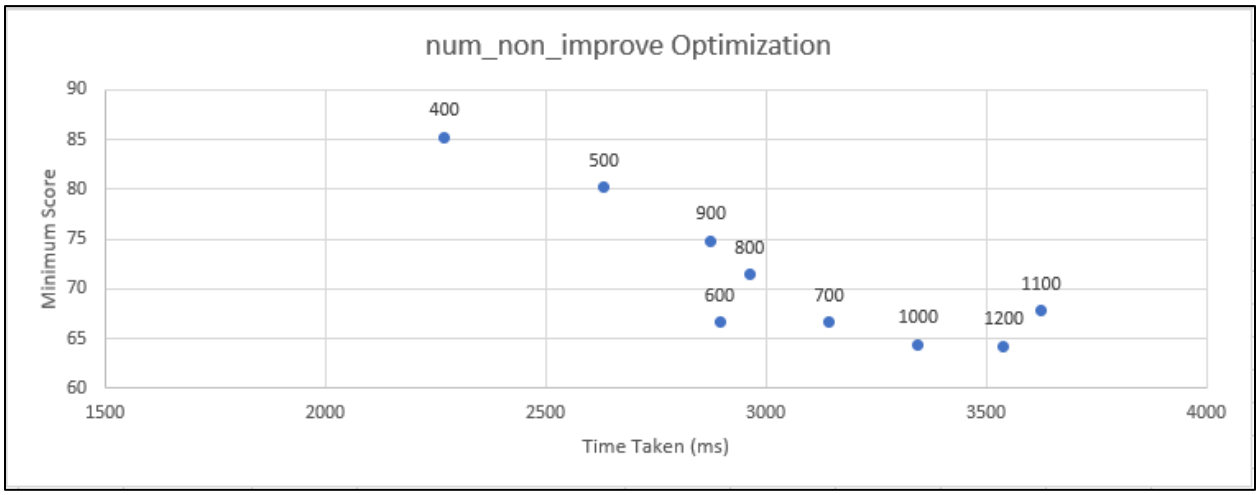
To test this, I used the same values as when finding the Initial Temperature. "Temperature Length" set at 7000, "num_non_improve" at 2000 and instead of setting "a" to a specific value I set the "Initial Temperature" at 1000. To find the optimal solution of 'a' I created a dataset beginning from 0.980 up to 0.999 with increments of 0.001



From these results I can observe that the lowest solutions for the minimum cost where mostly found from 0.995 to 0.999. Since 'a' being equals to 0.998 gives us the lowest score I can conclude that the optimal solution for **fixed parameter 'a' is 0.998**.

Num_non_improve

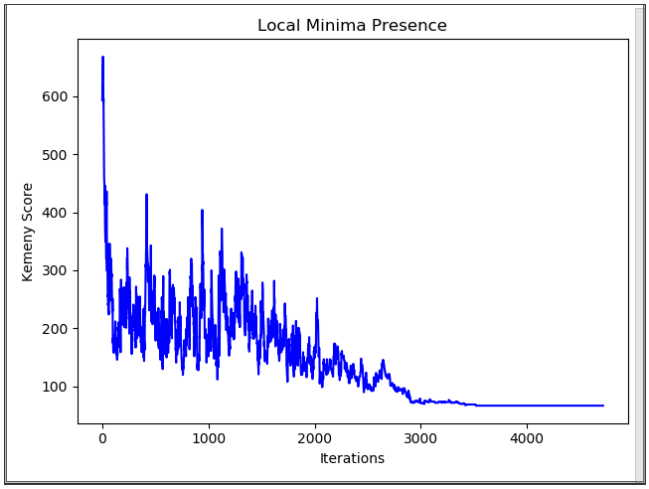
The number of non-improvements should be set optimally so that the algorithm doesn't end prematurely, and it doesn't do too many iterations without any improvements to the minimum cost. From the other tests we found all the other optimal solutions for the Simulated Annealing parameters which were set as 'Temperature Length' at 7000, 'Initial Temperature' at 1000 and 'a' at 0.998. To test the num_non_improve values I created a set beginning from 400 up to 1200 with increments of 100. For each value I ran it 5 times and found an average result which should give more accurate results. We will use the findings to compare amongst minimum cost found with time taken for each minimum cost. This should give us an estimate for the optimal solution where the minimum cost is found for the least amount of time.



From the results gathered we can observe that the we reach very close the minimum score only after num_non_improve 1000, and 1200 finds the lowest minimum cost only by a very close margin. The previous values cause a premature end in the algorithm without let it reach the minimum score. I weigh minimum score over time taken so for me finding the best solution for the minimum cost is more important for some minor difference in the time taken so my optimally chosen solution for **num_non_improve is be 1200**.

Local Optima

By using the best choice of SA parameters, I plotted a graph by taking the Kemeny score at each iteration of the algorithm. This shows all of the uphill moves and downhill moves made and each trough represents a local minimum. For this specific problem a local minimum is a local optimum. By observation of the graph it's clear that there are many local optima in our Weighted tournament problem.



Task 4:

Decision Variables

For a general tournament T with n participants and with tournament matrix $[a_{ij}]$ the decisions variable for each pair of participants would be 1 if there is weight between them and 0 otherwise.

$$X_{ij} = \begin{cases} 1 & \text{if rank of i is below rank of j} \\ 0 & \text{otherwise} \end{cases}$$

Objective Function

Our objective function for a general tournament T would be to minimize the Kemeny score. Each edge $i \rightarrow j$ potentially contributes the Kenemy score between i and j to this Kemeny score total. So, if we let C_{ij} denote the weight, we arrive at the following objective function (to be minimized):

$$\begin{aligned} & C_{11}X_{11} + C_{12}X_{12} + \ldots + C_{1n}X_{1n} + \\ + \; & C_{21}X_{21} + C_{22}X_{22} + \ldots + C_{2n}X_{2n} + \\ & \vdots \qquad \vdots \qquad\qquad \vdots \\ + \; & C_{n1}X_{n1} + C_{n2}X_{n2} + \ldots + C_{nn}X_{nn} \end{aligned}$$

Constraints

- Each weight should only by accounted for once and only once.
- For a given tournament T with n participants Let participants be i, j, k if i is ranked above j and j is ranked above k then i is ranked above k.
- For all i, j such that $i \neq j$ exactly one of "i is ranked above j" or "j is ranked above i" must hold.
- For all I,j such that $X_{ij} = 1$ their weight C_{ij} is bigger than 0.
- For all I,j such that $X_{ij} = 0$ their weight C_{ij} is also 0.
- X_{ij} is binary for all i, j.