Student Name: Matthew Battagel
Student ID: C1946077

Combinatorial Optimisation
Coursework

Task 1

Write down a definition of a neighbourhood N that is specific to this problem, i.e., specify the conditions for when any given ranking R2 is a neighbour of another R1, and illustrate your definition with an example. Justify your choice of definition by showing how the cost of a neighbouring solution R2 can easily be computed from the cost of current solution R1.

In this instance, a ranking is defined as a list of integers with possible values from 1 through to 46. These numbers represent the ID of each driver and the index within the list their ranking. A neighbour is made when swapping two random non adjacent edges with each-other from an existing neighbour. The method for this includes a new list where indexes 0 -¿ random1, random1 -¿ random2, and random2 -¿ 46 are all separate arrays given two random indexes random1 and random2. These random indexes have two conditions; they cannot be the same nor can they be next to each-other. The array from indexes random1 -¿ random2 is then reversed so all values are relocated to their inverse indexes. These three arrays are then concatenated back into a ranking in the order that they were taken apart. The resulting ranking will be a 2-change neighbour of the original ranking and therefore the neighbourhood is all of the possible rankings made by this method.

The following is an example of a ranking and a possible neighbour. Say we have 8 unique drivers with IDs from 1 through 8. The current ranking is an ordered list of these IDs, [1, 2, 3, 4, 5, 6, 7, 8]. To find a neighbour we will implement the method above. Random1 will be 3 and random2 will be 6. Split the array with on these edges into three arrays, [1, 2], [3, 4, 5], and [6, 7, 8]. Note that it is possible to choose the edge to the left of ID 1 and to the right of ID 8. In this scenario the respective left or right array would be empty. The next step is to reverse the second array to [5, 4, 3] and then to combine all 3 arrays together again. The resulting ranking is [1, 2, 3, 5, 4, 3, 6, 7, 8] and is a neighbour apart of the neighbourhood of the original ranking. The original ranking is also apart of the new rankings neighbourhood as with the same random indexes chosen the ranking will go back to the original order.

Using a 2-change neighbourhood is advantageous when performing optimisation problems as the cost function can utilise the previously calculated cost, in this case the Kemeny score, to calculate the new cost. This saves valuable operations per cycle of the algorithm which overall will reduce the amount of time it takes to run. Compare this to randomly arranging the list of integers every cycle. Firstly, the cost of one ranking is in no way related to the new ranking, and secondly, there is no relation between the two rankings so achieving a low cost for ranking R1 is not inherited to its neighbour in any form. 2-change neighbours give the advantage of providing inheritance of a good score so that the simulated annealing algorithm can gradually improve this score over time.

The scores between neighbours are related as the first and last arrays generated when creating a neighbour will keep the same Kemeny score between their edges as in the current ranking. This means the only intensive calculation needed is for the second, reversed, array. This saves computation as calculating a Kemeny score from scratch has the time complexity of $O(n^2)$. Instead the Kemeny score for the neighbour can be calculated as $C(x') = C(x_0)$ - old_middle + new_middle.

Task 2

All code is contained within one python file named c1946077_code.py. The code should be executed by doing "python3 c1946077_code.py 1994_Formula_One.wmg.

Task 3

Give the values of the parameters TL, TI and a (in f(T) = a×T), as well as num non improve, which seem to give the best solutions for your program. For this "best" choice of parameters provide screenshots of the output of your program. Write a short summary (max 300 words) of your results, indicating the range of different values you tried for the parameters, which parameters' variation had the biggest effect on the output solution, etc. Extra marks available for deeper analysis and presentation/visualisation of results, e.g., via use of graphs showing results of varying the different SA parameters Also offer some speculation on the presence of local optima in this problem. Are there many?

To find the optimal parameters, I will first find the "best" parameters which are guaranteed to find the global optima and then judge their time complexity to determine if they are suitable. If not then these values will have to be reduced to find the optimal parameters for our scenario where we want the results instantaneously.

To achieve this wrote a helper function to iterate over various changing parameters. I would then plot a graph of these results to determine which parameters had the largest effect on the resulting Kemeny score. Note that each instance of a chosen value for a parameter is retried 10 times to reduce anomalous results due to the random indexing. On each of the graphs the blue dots represent an individual result and the line the average of these points.

Firstly, I iterated over values for the initial temperature (TI). The helper function I had written allowed me to select an initial value for TI then slowly increase it overtime exponentially. This would try a large range of values from 10 to 10^{15} to hopefully be able to view the whole trend. The graph produced has a log scale to better represent the wide range of numbers.

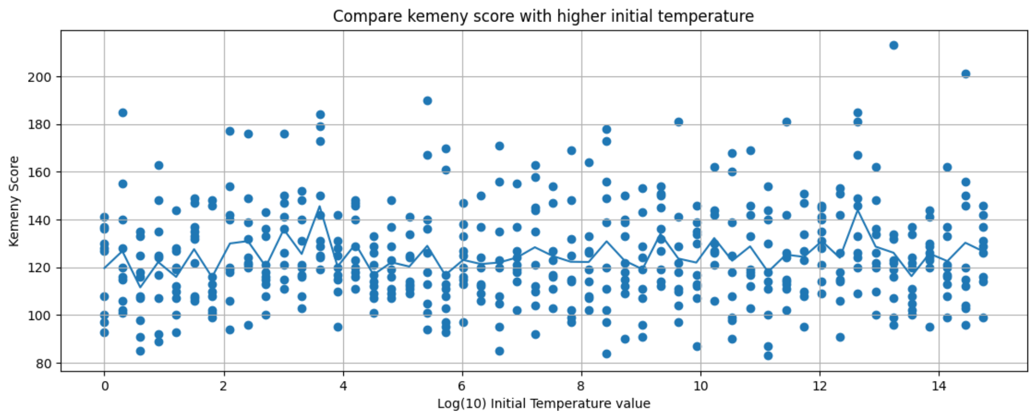

Figure 1: Initial Temperature vs Kemeny Score Analysis

From this analysis I have determined that the size of the initial temperature does not have a large effect on the accuracy of the Kemeny score. The average value for the Kemeny score does not change even though the amount of computation needed is greater with larger initial temperatures. I will settle on a small initial temperature number of 100 to allow the algorithm to run as fast as possible.

Secondly, I changed the temperature length (TL). For this I iterated 17 times equally spaced across the range 10 to 100,000 or 10^5. This graph is also logged.

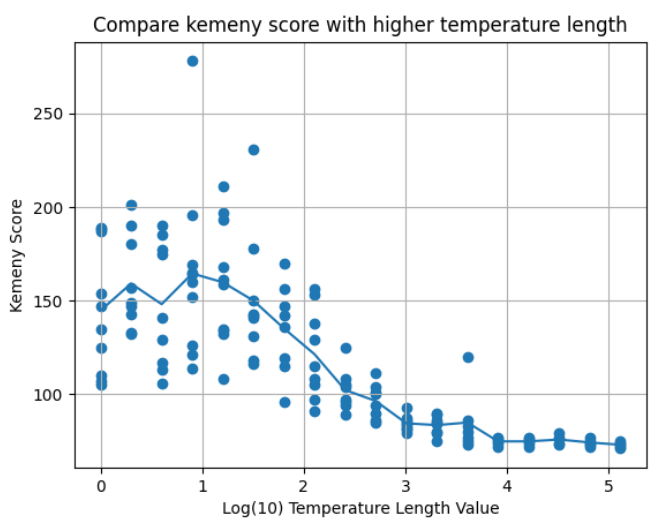

Figure 2: Temperature Length vs Kemeny Score Analysis

The graph shows that the temperature length does have an effect on the Kemeny score. As the temperature length increases, the Kemeny score gets lower and lower. This is true until about 10^4 when it starts to level off. This value of 10000 will be our best temperature length value.

Thirdly, was the temperature multiplier or a in f(T) = a * T. This value was harder to iterate over as it is a decimal between 0 and 1. For this I decided the best method was to use Euler's number to increment the value while never reaching 1. Reaching 1 would break the algorithm as the temperature would never decrease after an iteration. The equation of $1 - e^{-x}$ will gradually tend towards 1 when incrementing x therefore, we can use this to find the value for the temperature multiplier.

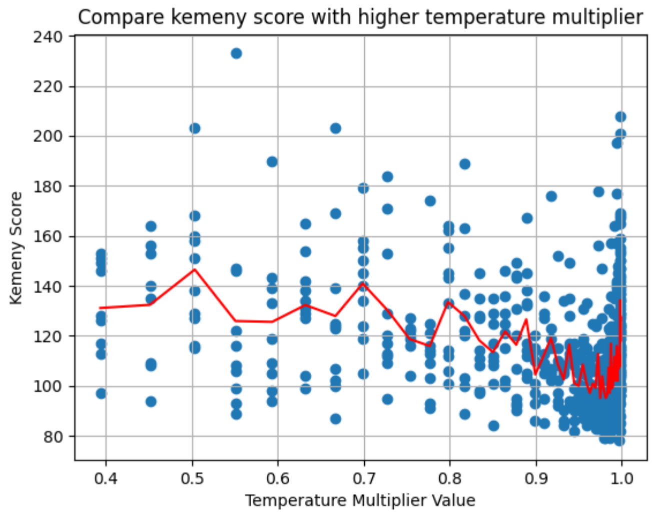

Figure 3: Temperature Multiplier (a) vs Kemeny Score Analysis

The results from the graph show that typically the closer to the value to 1 the lower the Kemeny score however once the value passes 0.97 the Kemeny score increases again. We will take the best value for a as 0.97. Note the average line for this graph is red due to the high number of blue data points.

Lastly, is the stopping condition parameter of num not improved. I increase the value exponentially like the previous initial temperature and temperature length parameters and record the values into the following graph.

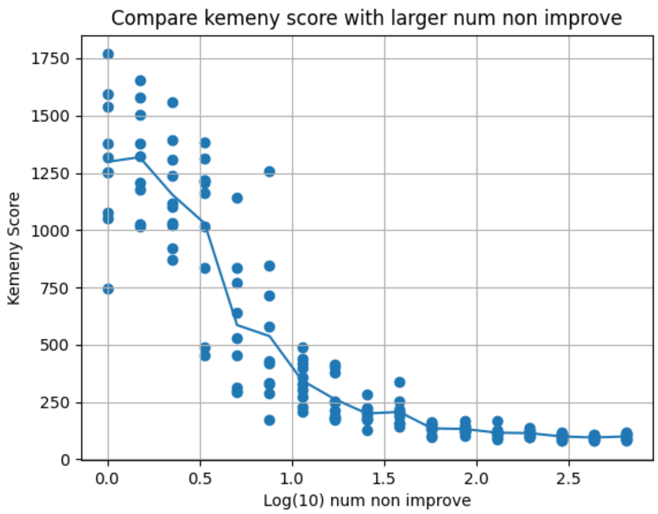

Figure 4: num_not_improved value vs Kemeny Score Analysis

The graph shows a clear correlation between a larger num not improved and a lower Kemeny score. The change in the gradient is greater on this curve than on the previous ones showing that this parameter will have a higher impact on the Kemeny score than increasing other ones. The curve flattens off at around 2.5 giving us the best value for num not improved as $10^{2.5}$ or roughly 300.

This leaves us with the optimal parameters of:

- Initial Temperature: 100
- Temperature Length: 10000
- Temperature Multiplier: 0.97
- Num not improved: 300

With these optimal parameters applied I ran the algorithm to try and determine the global optimal solution (lowest Kemeny score) for this optimisation problem. The ranking is as follows:

Listing 1: Global Optimal Kemeny Score with Best Parameters

```
        20 | Michael Schumacher
        19 | Damon Hill
         1 | Nigel Mansell
         2 | Gerhard Berger
        34 | David Coulthard
         6 | Jean Alesi
        39 | Pedro Lamy
        12 | Mika Hakkinen
         5 | Olivier Panis
         4 | Rubens Barrichello
        29 | Eric Bernard
        41 | Karl Wendlinger
```

```
        32 | Jos Verstappen
         8 | Christian Fittipaldi
         3 | Martin Brundle
        44 | Nicola Larini
        14 | Mark Blundell
         9 | Pierluigi Martini
        10 | J J Lehto
         7 | Heinz-Harald Frentzen
        11 | Franck Lagorce
        15 | Jean-Denis Deletraz
        17 | Mika Salo
        26 | Johnny Herbert
        13 | Michele Alboreto
        30 | Erik Comas
        42 | Aguri Suzuki
        33 | Olivier Beretta
        36 | Jean-Marc Gounon
        16 | David Brabham
        22 | Ukyo Katayama
        25 | Eddie Irvine
        45 | Yannick Dalmas
        18 | Alex Zanardi
        21 | Domenico Schiattarella
        24 | Gianni Morbidelli
        31 | Andrea de Cesaris
        43 | Ayrton Senna
        46 | Philippe Adams
        35 | Philippe Alliot
        37 | Taki Inoue
        23 | Hideki Noda
        27 | Bertrand Gachot
        40 | Roland Ratzenberger
        28 | Paul Belmondo
        38 | Andrea Montermini
        Best kemeny score:
        71
        Completed time in milliseconds:
        260003.9666670491
```

The results look promising as it returned the lowest value that I had every achieved with other parameters. However, the run-time in milliseconds is 260,004 or 4.33 minutes. In certain circumstances this could be an acceptable amount of time but in this scenario an instantaneous value is required. To achieve this I will have to reduce the values of some parameters that are increasing the run-time.

The beat values to reduce can be found in two ways. Firstly, to reduce the parameters which have less of an impact on the overall Kemeny score, namely the temperature length and temperature multiplier. Secondly, to find the values which disproportionately increase in time complexity over their decrease in Kemeny score.

As the initial temperature does not have a large effect on the Kemeny score I will reduce this number to 10. The temperature length does have a large effect on the computation as it directly increases the number of loops the algorithm will go through each time generating a new neighbour which is expensive. Reducing this to 100 will drastically improve the performance of the program. Temperature multiplier can be slightly reduced to improve time complexity as a small change from 0.97 to 0.9 will remove a larger amount of cycles because the temperature will decrease exponentially faster. Lastly, the num not improved score must be reduced but this should be left as in tact as possible because it has the greatest impact on the Kemeny score. Reducing the num not improved value from 300 to 250 brings the algorithm under the average of 2 seconds per run. I determined that 2 seconds would be an acceptable margin of processing time as any longer and I would find my-self getting bored waiting. This margin can be changed and the corresponding parameters adjusted to suit.

Here is the result of the algorithm running with the new optimal parameters.

Listing 2: Final Result from Optimal Parameters

```
        20 | Michael Schumacher
        19 | Damon Hill
         1 | Nigel Mansell
         2 | Gerhard Berger
        34 | David Coulthard
         6 | Jean Alesi
        39 | Pedro Lamy
        12 | Mika Hakkinen
         5 | Olivier Panis
         4 | Rubens Barrichello
        41 | Karl Wendlinger
        29 | Eric Bernard
         8 | Christian Fittipaldi
        32 | Jos Verstappen
         3 | Martin Brundle
        14 | Mark Blundell
         9 | Pierluigi Martini
        10 | J J Lehto
         7 | Heinz-Harald Frentzen
        11 | Franck Lagorce
        15 | Jean-Denis Deletraz
        17 | Mika Salo
        26 | Johnny Herbert
        13 | Michele Alboreto
        30 | Erik Comas
        42 | Aguri Suzuki
        33 | Olivier Beretta
        44 | Nicola Larini
        36 | Jean-Marc Gounon
        16 | David Brabham
        24 | Gianni Morbidelli
        22 | Ukyo Katayama
        25 | Eddie Irvine
        45 | Yannick Dalmas
        18 | Alex Zanardi
        21 | Domenico Schiattarella
```

```
        31 | Andrea de Cesaris
        37 | Taki Inoue
        46 | Philippe Adams
        43 | Ayrton Senna
        35 | Philippe Alliot
        23 | Hideki Noda
        27 | Bertrand Gachot
        40 | Roland Ratzenberger
        28 | Paul Belmondo
        38 | Andrea Montermini
        Best kemeny score:
        74
        Completed time in milliseconds:
        1915.8857079804875
```