Task 1

In the context of this task, I believe a suitable definition of the neighbourhood R2 given R1 is to swap two random rankings of the drivers around with a restriction on the maximum distance between the values being swapped,

Eg.

Given:

R_1 = [A, B, C, D, E, F]

R2 could be:

R_2 = [A, E, C, D, B, F] or,

R_2 = [C, B, A, D, E, F] or,

R_2 = [A, F, C, D, E, B]

Initially I thought of implementing a full random solution however this could result in the calculation of all Kemeny scores if the random ranking to change was the initial and final value. This controlled random solution allows for there to be a greater variance of rankings faster than a simple adjacent swap whilst not impacting the computation time too much by limiting the Kemeny ranks calculated as just the values between the start and end. Through testing a value of around 4 seems to be strong, and adjacency swap is possible with this configuration by setting the maximum distance to 1.

Task 3

In this fig 1 visualisation we can see the correlation matrix between different input and output variables. It is clearly shown the 2 main affecting variables are the num_non_improve and to a lesser extent initial temp. We see that whilst increasing the num_non_improve input variable leads to an improvement in the optimal solution, it comes at great cost to the overall runtime.

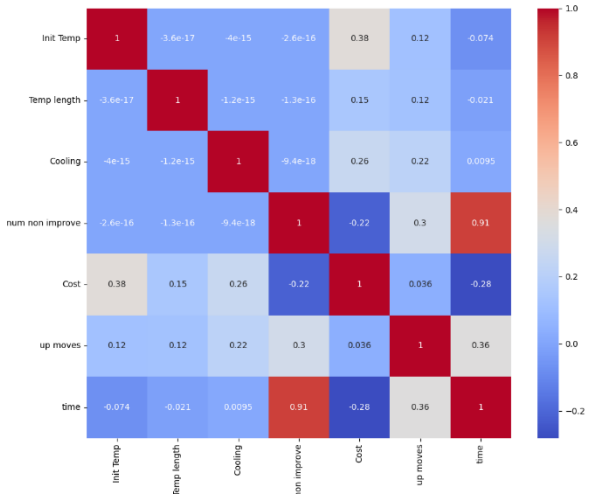
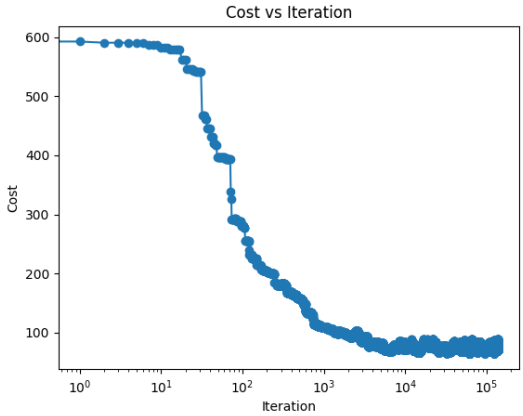


Figure 1 - Correlation Matrix

Temp length does have an impact but as the improvement of accuracy when increasing the TL value has a 6:-1 cost improvement to time ratio, maintaining a high value for this variable makes sense.

We can also see that the cooling period increasing has a seasonable effect on the number of up moves, whilst not too greatly impacting time. This is beneficial as an increase in up moves can be indicative of better exploring local optima.



Showing local optima can give us a greater insight into how the overall number of iterations impacts the cost at the end of the solution. From fig 2 we can see that the major improvement in cost is present from 10^1 to 10^4 and not much improvement is gathered after that point. Finding values of num_non_improve and temperature length that result in a total iteration around 10^4 would be an optimal balance between time and cost.

Figure 2 - Total iterations to cost

	Init Temp	Temp length	Cooling	num non improve	Cost	up moves	time
count	2688.000000	2688.000000	2688.00000	2688.000000	2688.000000	2688.000000	2688.000000
mean	28.500000	43.500000	0.93990	12585.714286	107.987723	2355.512277	4297.528865
std	31.888529	51.112346	0.07181	16569.867543	88.385916	8548.262907	6127.264621
min	1.000000	1.000000	0.80000	100.000000	63.000000	0.000000	3.000000
25%	4.250000	4.250000	0.90000	1000.000000	68.000000	17.000000	407.285000
50%	15.000000	20.000000	0.97000	5000.000000	74.000000	119.000000	1618.305000
75%	42.500000	62.500000	0.99950	20000.000000	81.000000	985.500000	5870.460000
max	100.000000	150.000000	0.99990	50000.000000	593.000000	162512.000000	51482.640000

Figure 3 - Overall basic statistics

From figure 3 we can gather that the best cost outcomes (bottom quartile) hover around the high 60s, therefore finding an input variable set that minimizes time whilst maintaining a cost value in line with this will be the ideal solution. I have found that total iterations of 30,000 - 100,000 can manage this low cost whilst allowing for the solution to be generating in around a second as can be seen in figure 4. This also fits with my conclusions from figure 3.

```
Final data line found
New Rank        Original Rank   Driver Name
1               9               Niki Lauda
2               10              Alain Prost
3               2               Rene Arnoux
4               3               Elio de Angelis
5               7               Corrado Fabi
6               22              Derek Warwick
7               13              Michele Alboreto
8               16              Nelson Piquet
9               18              Patrick Tambay
10              20              Andrea de Cesaris
11              33              Mauro Baldi
12              11              Thierry Boutsen
13              31              Teo Fabi
14              21              Riccardo Patrese
15              30              Gerhard Berger
16              6               Nigel Mansell
17              29              Jo Gartner
18              1               Keke Rosberg
19              14              Ayrton Senna
20              24              Eddie Cheever
21              12              Marc Surer
22              15              Jonathan Palmer
23              26              Martin Brundle
24              19              Huub Rothengatter
25              4               Jacques Laffite
26              23              Stefan Bellof
27              25              Francois Hesnault
28              28              Stefan Johansson
29              5               Piercarlo Ghinzani
30              8               Manfred Winkelhock
31              17              Johnny Cecotto
32              32              Pierluigi Martini
33              27              Philippe Alliot
34              34              Mike Thackwell
35              35              Philippe Streiff

Overview

Best Kemeny Score:      64
Up moves:               461
Time taken:             830.0 milliseconds

Detailed time breakdown (average time for each procedure)
Read File       ---     2.0 (ms)
K Score         ---     1.83 (ns)
Update Score    ---     22.65 (ns)
```

Therefore, the finalised optimum values used to obtain solution in figure 4 are

```
TI = 1
TL = 50
cooling = 0.9999
numNonImprove = 2000
```

And from the output we can see the response was near instant being under a second and a Kemeny score of 64 is as close to optimum we can consistently achieve within this time frame. Further breakdown of the timing shows us that file reading and Kemeny score are quite efficient at 2ms and 1.86 nano seconds each. It is the update score's function that has the slowest processing speed. Most likely due to the sudo-random neighbourhood creation, however the benefits of more diverse neighbourhoods as discussed earlier are worth it in my opinion.

Figure 4 - Optimal input variables output