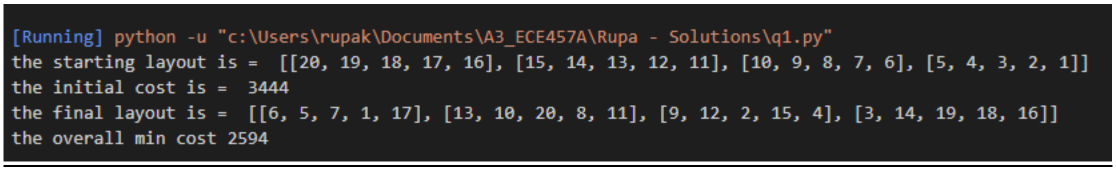
Rupakeerthana Vemulapalli
20778222

Assignment 3

Question 1:

Base Case:

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	None

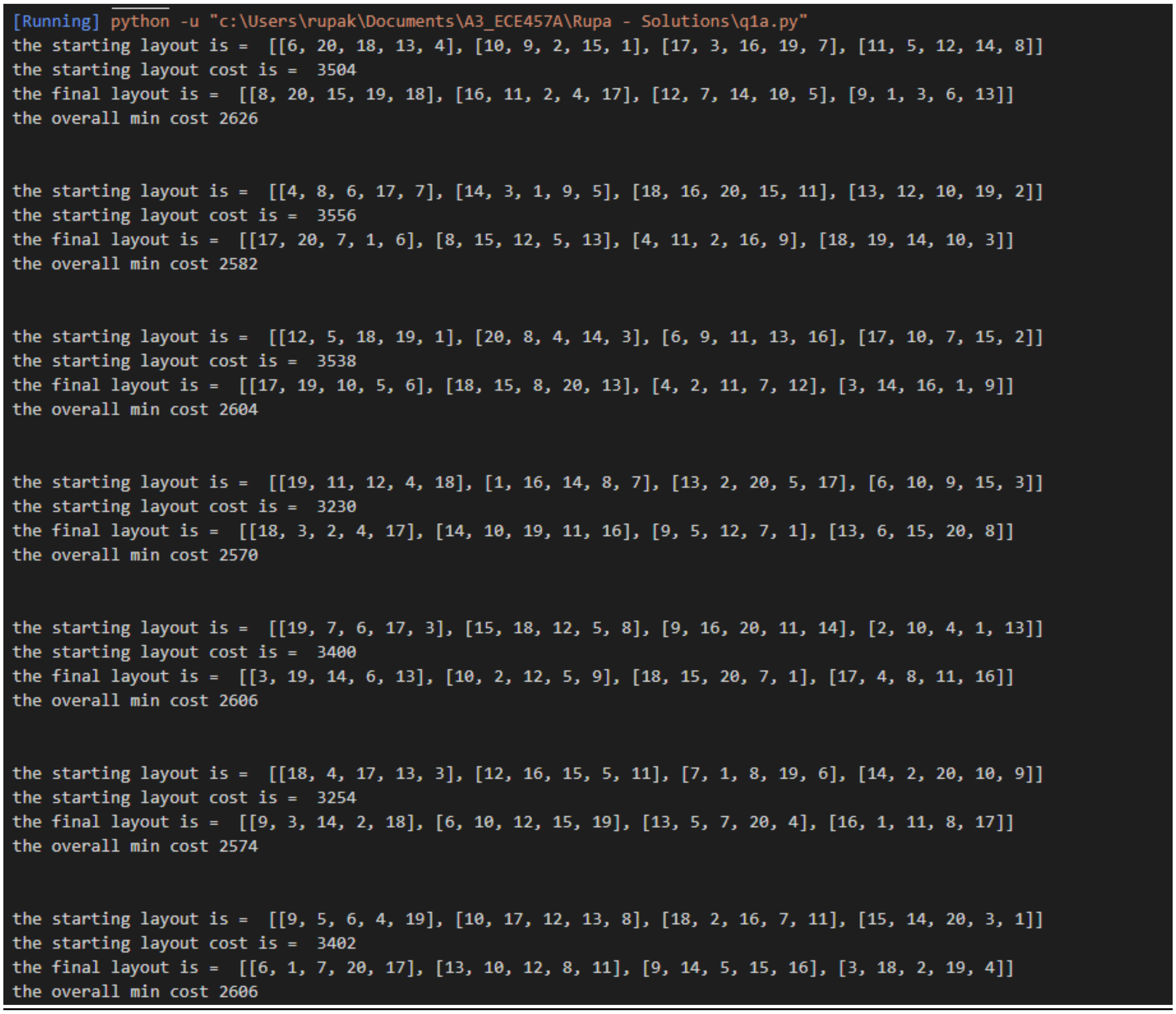


```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[6, 5, 7, 1, 17], [13, 10, 20, 8, 11], [9, 12, 2, 15, 4], [3, 14, 19, 18, 16]]
the overall min cost 2594
```

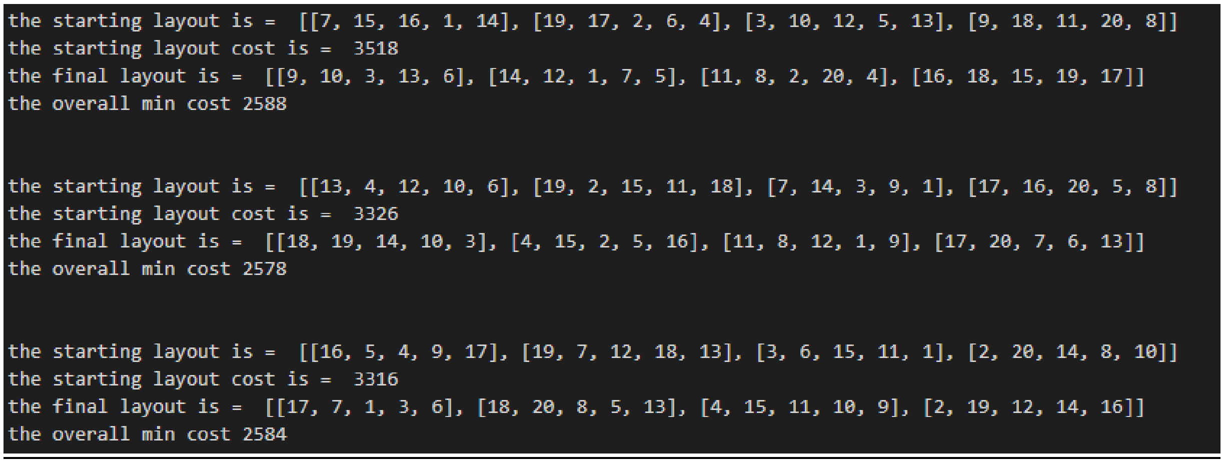
Min cost = 2594

Case a: Changing the initial starting point (initial solution) 10 times

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	None
**Changing the initial starting point (initial solution) 10 times	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1a.py"
the starting layout is =  [[6, 20, 18, 13, 4], [10, 9, 2, 15, 1], [17, 3, 16, 19, 7], [11, 5, 12, 14, 8]]
the starting layout cost is =  3504
the final layout is =  [[8, 20, 15, 19, 18], [16, 11, 2, 4, 17], [12, 7, 14, 10, 5], [9, 1, 3, 6, 13]]
the overall min cost 2626


the starting layout is =  [[4, 8, 6, 17, 7], [14, 3, 1, 9, 5], [18, 16, 20, 15, 11], [13, 12, 10, 19, 2]]
the starting layout cost is =  3556
the final layout is =  [[17, 20, 7, 1, 6], [8, 15, 12, 5, 13], [4, 11, 2, 16, 9], [18, 19, 14, 10, 3]]
the overall min cost 2582


the starting layout is =  [[12, 5, 18, 19, 1], [20, 8, 4, 14, 3], [6, 9, 11, 13, 16], [17, 10, 7, 15, 2]]
the starting layout cost is =  3538
the final layout is =  [[17, 19, 10, 5, 6], [18, 15, 8, 20, 13], [4, 2, 11, 7, 12], [3, 14, 16, 1, 9]]
the overall min cost 2604


the starting layout is =  [[19, 11, 12, 4, 18], [1, 16, 14, 8, 7], [13, 2, 20, 5, 17], [6, 10, 9, 15, 3]]
the starting layout cost is =  3230
the final layout is =  [[18, 3, 2, 4, 17], [14, 10, 19, 11, 16], [9, 5, 12, 7, 1], [13, 6, 15, 20, 8]]
the overall min cost 2570


the starting layout is =  [[19, 7, 6, 17, 3], [15, 18, 12, 5, 8], [9, 16, 20, 11, 14], [2, 10, 4, 1, 13]]
the starting layout cost is =  3400
the final layout is =  [[3, 19, 14, 6, 13], [10, 2, 12, 5, 9], [18, 15, 20, 7, 1], [17, 4, 8, 11, 16]]
the overall min cost 2606


the starting layout is =  [[18, 4, 17, 13, 3], [12, 16, 15, 5, 11], [7, 1, 8, 19, 6], [14, 2, 20, 10, 9]]
the starting layout cost is =  3254
the final layout is =  [[9, 3, 14, 2, 18], [6, 10, 12, 15, 19], [13, 5, 7, 20, 4], [16, 1, 11, 8, 17]]
the overall min cost 2574


the starting layout is =  [[9, 5, 6, 4, 19], [10, 17, 12, 13, 8], [18, 2, 16, 7, 11], [15, 14, 20, 3, 1]]
the starting layout cost is =  3402
the final layout is =  [[6, 1, 7, 20, 17], [13, 10, 12, 8, 11], [9, 14, 5, 15, 16], [3, 18, 2, 19, 4]]
the overall min cost 2606
```



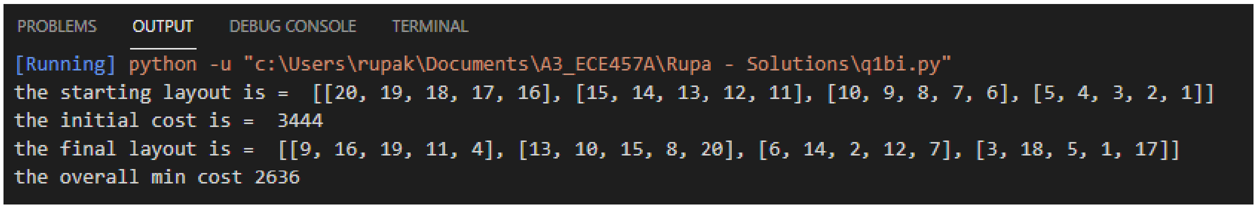
```
the starting layout is =  [[7, 15, 16, 1, 14], [19, 17, 2, 6, 4], [3, 10, 12, 5, 13], [9, 18, 11, 20, 8]]
the starting layout cost is =  3518
the final layout is =  [[9, 10, 3, 13, 6], [14, 12, 1, 7, 5], [11, 8, 2, 20, 4], [16, 18, 15, 19, 17]]
the overall min cost 2588


the starting layout is =  [[13, 4, 12, 10, 6], [19, 2, 15, 11, 18], [7, 14, 3, 9, 1], [17, 16, 20, 5, 8]]
the starting layout cost is =  3326
the final layout is =  [[18, 19, 14, 10, 3], [4, 15, 2, 5, 16], [11, 8, 12, 1, 9], [17, 20, 7, 6, 13]]
the overall min cost 2578


the starting layout is =  [[16, 5, 4, 9, 17], [19, 7, 12, 18, 13], [3, 6, 15, 11, 1], [2, 20, 14, 8, 10]]
the starting layout cost is =  3316
the final layout is =  [[17, 7, 1, 3, 6], [18, 20, 8, 5, 13], [4, 15, 11, 10, 9], [2, 19, 12, 14, 16]]
the overall min cost 2584
```

Analysis:

By running multiple solutions, we have achieved the most optimal solution at *starting point 4 where cost is 2570.*

<u>Case bi: Changing the tabu list size smaller than the original (original length = 20, new length = 5)</u>

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	5
Stopping Criterion	250 iterations
Aspiration Criteria	None
**Changing the tabu list size smaller than the original (original length = 20, new length = 5)	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1bi.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[9, 16, 19, 11, 4], [13, 10, 15, 8, 20], [6, 14, 2, 12, 7], [3, 18, 5, 1, 17]]
the overall min cost 2636
```

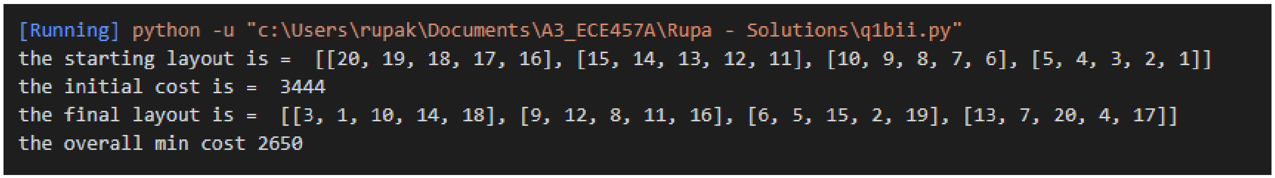
Min cost = 2636

Analysis:

By making the tabu list size smaller, we have a more sub-optimal solution compared to the base case.

<u>Case bii: Changing the tabu list size larger than the original (original length: 20, new length = 50)</u>

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	50
Stopping Criterion	250 iterations
Aspiration Criteria	None
** Changing the tabu list size smaller than the original (original length = 20, new length = 50)	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1bii.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[3, 1, 10, 14, 18], [9, 12, 8, 11, 16], [6, 5, 15, 2, 19], [13, 7, 20, 4, 17]]
the overall min cost 2650
```

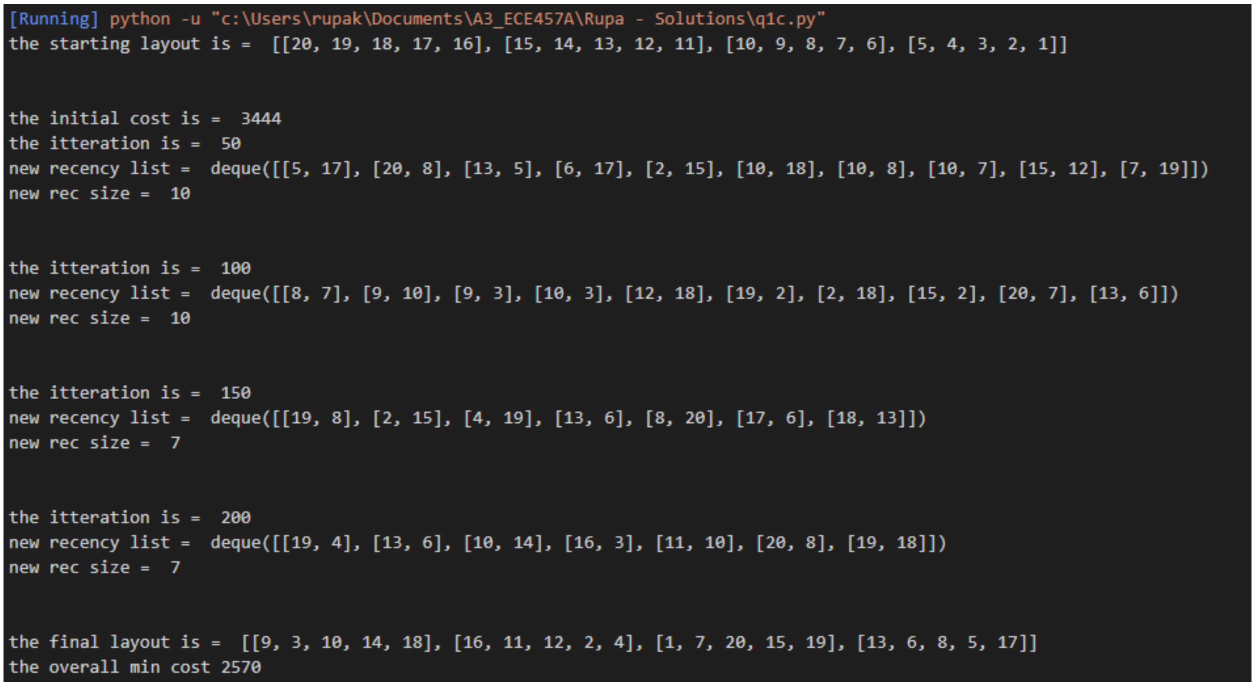
Min cost = 2650

Analysis:

By running with a larger tabu list size, we have achieved a less optimal solution than having a smaller tabu list size compared to the previous. This means that maybe we get stuck in a sub-optimal solution (local minima) when we try to find the optimal cost with a larger tabu list size.

<u>Case c: Changing the tabu list size dynamically</u>

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	Dynamic (between 1-20)
Stopping Criterion	250 iterations
Aspiration Criteria	None
** Changing the tabu list size by producing a random number between 1 and 20 every 50 iterations	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1c.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]


the initial cost is =  3444
the itteration is =  50
new recency list =  deque([[5, 17], [20, 8], [13, 5], [6, 17], [2, 15], [10, 18], [10, 8], [10, 7], [15, 12], [7, 19]])
new rec size =  10


the itteration is =  100
new recency list =  deque([[8, 7], [9, 10], [9, 3], [10, 3], [12, 18], [19, 2], [2, 18], [15, 2], [20, 7], [13, 6]])
new rec size =  10


the itteration is =  150
new recency list =  deque([[19, 8], [2, 15], [4, 19], [13, 6], [8, 20], [17, 6], [18, 13]])
new rec size =  7


the itteration is =  200
new recency list =  deque([[19, 4], [13, 6], [10, 14], [16, 3], [11, 10], [20, 8], [19, 18]])
new rec size =  7


the final layout is =  [[9, 3, 10, 14, 18], [16, 11, 12, 2, 4], [1, 7, 20, 15, 19], [13, 6, 8, 5, 17]]
the overall min cost 2570
```

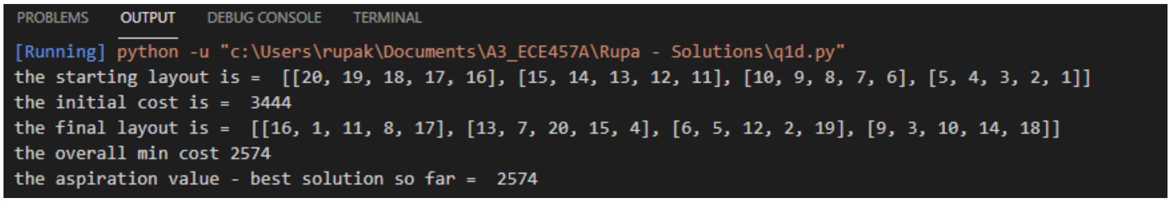
Min cost = 2570

Analysis:

Changing the tabu list size diversifies the balance between exploration and exploitation. This actually resulted in us reaching the optimal solution.

<u>Case d: Aspiration Criteria – *best solution so far*</u>

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	Best solution so far
** Aspiration Criteria – *best solution so far*	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1d.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[16, 1, 11, 8, 17], [13, 7, 20, 15, 4], [6, 5, 12, 2, 19], [9, 3, 10, 14, 18]]
the overall min cost 2574
the aspiration value - best solution so far =  2574
```

Best solution so far = 2574

Analysis:

The aspiration criteria was equal to the minimum cost found overall, because the same aspiration criteria was used.

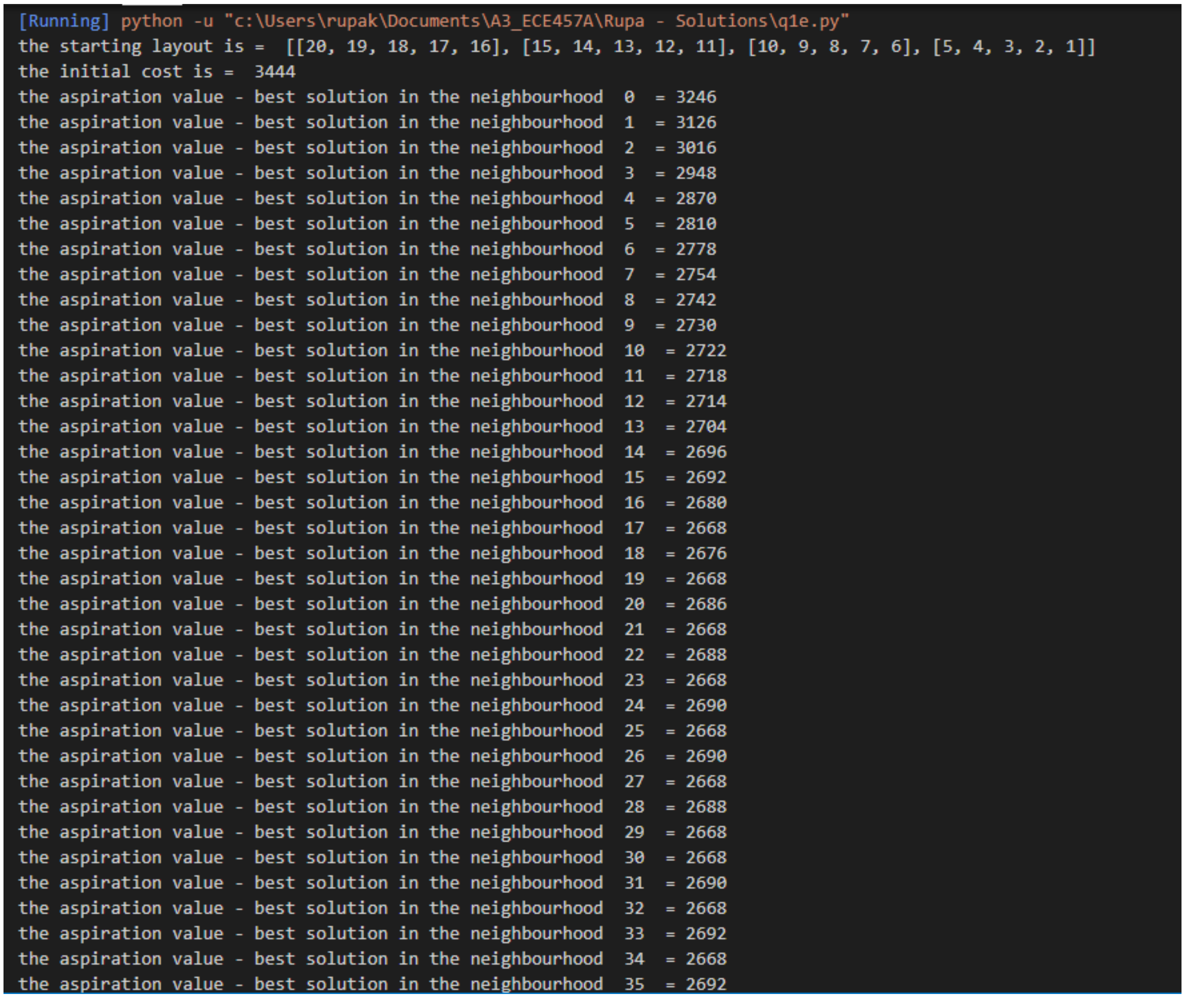
Case e: Aspiration Criteria – *best solution in the neighbourhood*

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking the whole neighbourhood
Tabu list type	Recency-based
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	Best solution in the neighbourhood
** Aspiration Criteria – *best solution in the neighbourhood*	

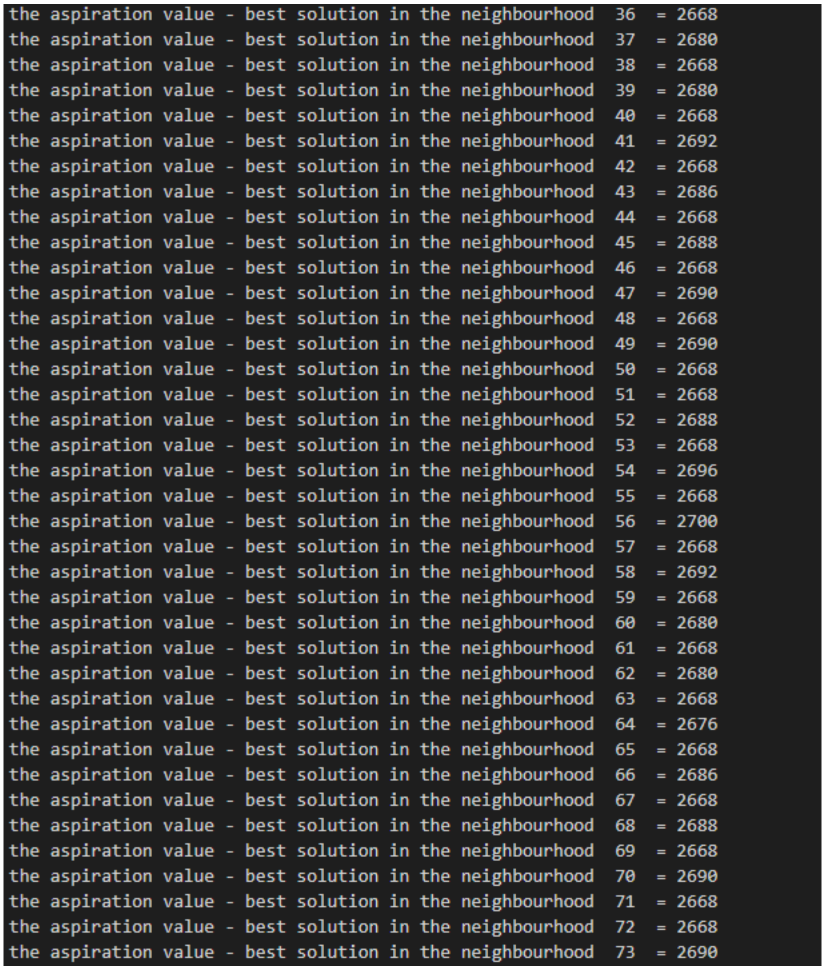
Min cost = 2698

Analysis:

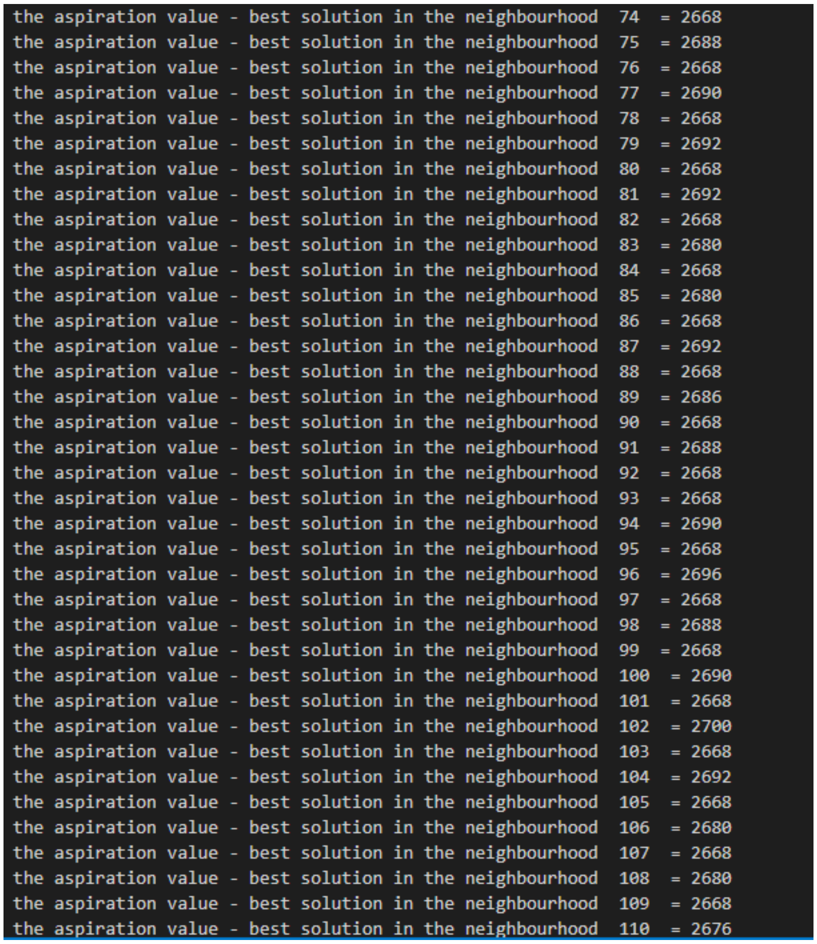
The aspiration criteria didn’t really help me to achieve an optimal solution.



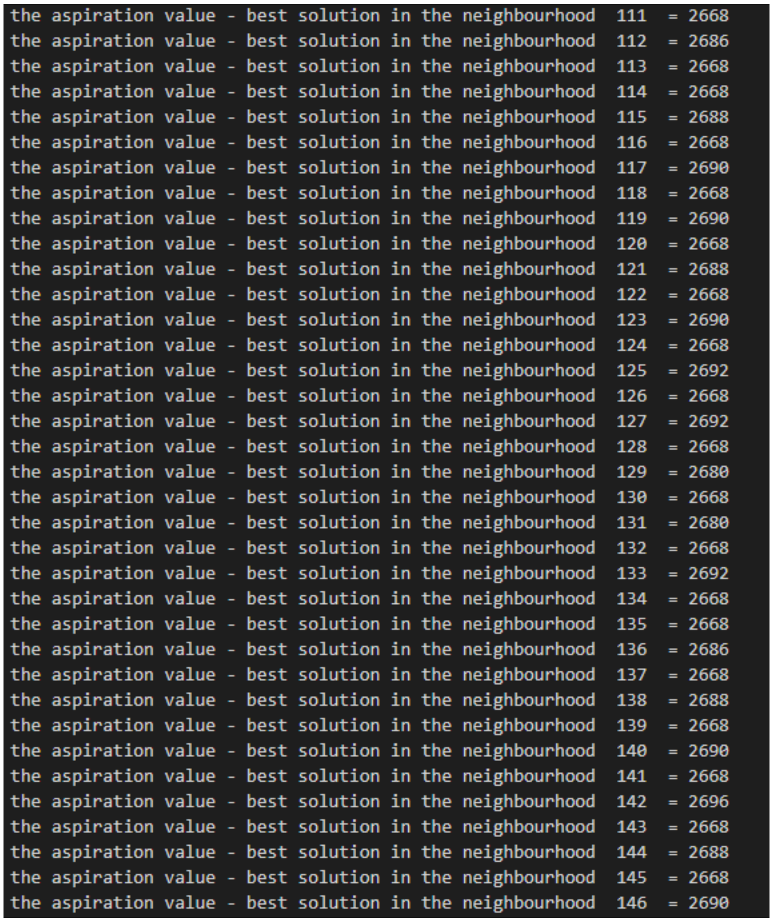
```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1e.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the aspiration value - best solution in the neighbourhood  0  = 3246
the aspiration value - best solution in the neighbourhood  1  = 3126
the aspiration value - best solution in the neighbourhood  2  = 3016
the aspiration value - best solution in the neighbourhood  3  = 2948
the aspiration value - best solution in the neighbourhood  4  = 2870
the aspiration value - best solution in the neighbourhood  5  = 2810
the aspiration value - best solution in the neighbourhood  6  = 2778
the aspiration value - best solution in the neighbourhood  7  = 2754
the aspiration value - best solution in the neighbourhood  8  = 2742
the aspiration value - best solution in the neighbourhood  9  = 2730
the aspiration value - best solution in the neighbourhood  10  = 2722
the aspiration value - best solution in the neighbourhood  11  = 2718
the aspiration value - best solution in the neighbourhood  12  = 2714
the aspiration value - best solution in the neighbourhood  13  = 2704
the aspiration value - best solution in the neighbourhood  14  = 2696
the aspiration value - best solution in the neighbourhood  15  = 2692
the aspiration value - best solution in the neighbourhood  16  = 2680
the aspiration value - best solution in the neighbourhood  17  = 2668
the aspiration value - best solution in the neighbourhood  18  = 2676
the aspiration value - best solution in the neighbourhood  19  = 2668
the aspiration value - best solution in the neighbourhood  20  = 2686
the aspiration value - best solution in the neighbourhood  21  = 2668
the aspiration value - best solution in the neighbourhood  22  = 2688
the aspiration value - best solution in the neighbourhood  23  = 2668
the aspiration value - best solution in the neighbourhood  24  = 2690
the aspiration value - best solution in the neighbourhood  25  = 2668
the aspiration value - best solution in the neighbourhood  26  = 2690
the aspiration value - best solution in the neighbourhood  27  = 2668
the aspiration value - best solution in the neighbourhood  28  = 2688
the aspiration value - best solution in the neighbourhood  29  = 2668
the aspiration value - best solution in the neighbourhood  30  = 2668
the aspiration value - best solution in the neighbourhood  31  = 2690
the aspiration value - best solution in the neighbourhood  32  = 2668
the aspiration value - best solution in the neighbourhood  33  = 2692
the aspiration value - best solution in the neighbourhood  34  = 2668
the aspiration value - best solution in the neighbourhood  35  = 2692
```



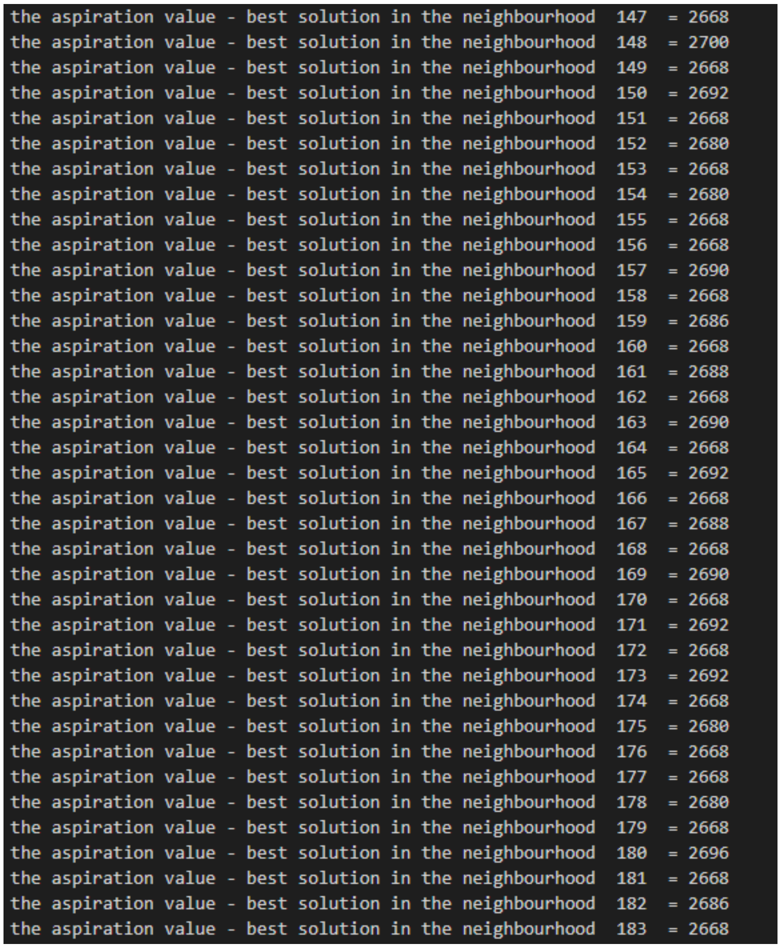
```
the aspiration value - best solution in the neighbourhood  36  = 2668
the aspiration value - best solution in the neighbourhood  37  = 2680
the aspiration value - best solution in the neighbourhood  38  = 2668
the aspiration value - best solution in the neighbourhood  39  = 2680
the aspiration value - best solution in the neighbourhood  40  = 2668
the aspiration value - best solution in the neighbourhood  41  = 2692
the aspiration value - best solution in the neighbourhood  42  = 2668
the aspiration value - best solution in the neighbourhood  43  = 2686
the aspiration value - best solution in the neighbourhood  44  = 2668
the aspiration value - best solution in the neighbourhood  45  = 2688
the aspiration value - best solution in the neighbourhood  46  = 2668
the aspiration value - best solution in the neighbourhood  47  = 2690
the aspiration value - best solution in the neighbourhood  48  = 2668
the aspiration value - best solution in the neighbourhood  49  = 2690
the aspiration value - best solution in the neighbourhood  50  = 2668
the aspiration value - best solution in the neighbourhood  51  = 2668
the aspiration value - best solution in the neighbourhood  52  = 2688
the aspiration value - best solution in the neighbourhood  53  = 2668
the aspiration value - best solution in the neighbourhood  54  = 2696
the aspiration value - best solution in the neighbourhood  55  = 2668
the aspiration value - best solution in the neighbourhood  56  = 2700
the aspiration value - best solution in the neighbourhood  57  = 2668
the aspiration value - best solution in the neighbourhood  58  = 2692
the aspiration value - best solution in the neighbourhood  59  = 2668
the aspiration value - best solution in the neighbourhood  60  = 2680
the aspiration value - best solution in the neighbourhood  61  = 2668
the aspiration value - best solution in the neighbourhood  62  = 2680
the aspiration value - best solution in the neighbourhood  63  = 2668
the aspiration value - best solution in the neighbourhood  64  = 2676
the aspiration value - best solution in the neighbourhood  65  = 2668
the aspiration value - best solution in the neighbourhood  66  = 2686
the aspiration value - best solution in the neighbourhood  67  = 2668
the aspiration value - best solution in the neighbourhood  68  = 2688
the aspiration value - best solution in the neighbourhood  69  = 2668
the aspiration value - best solution in the neighbourhood  70  = 2690
the aspiration value - best solution in the neighbourhood  71  = 2668
the aspiration value - best solution in the neighbourhood  72  = 2668
the aspiration value - best solution in the neighbourhood  73  = 2690
```



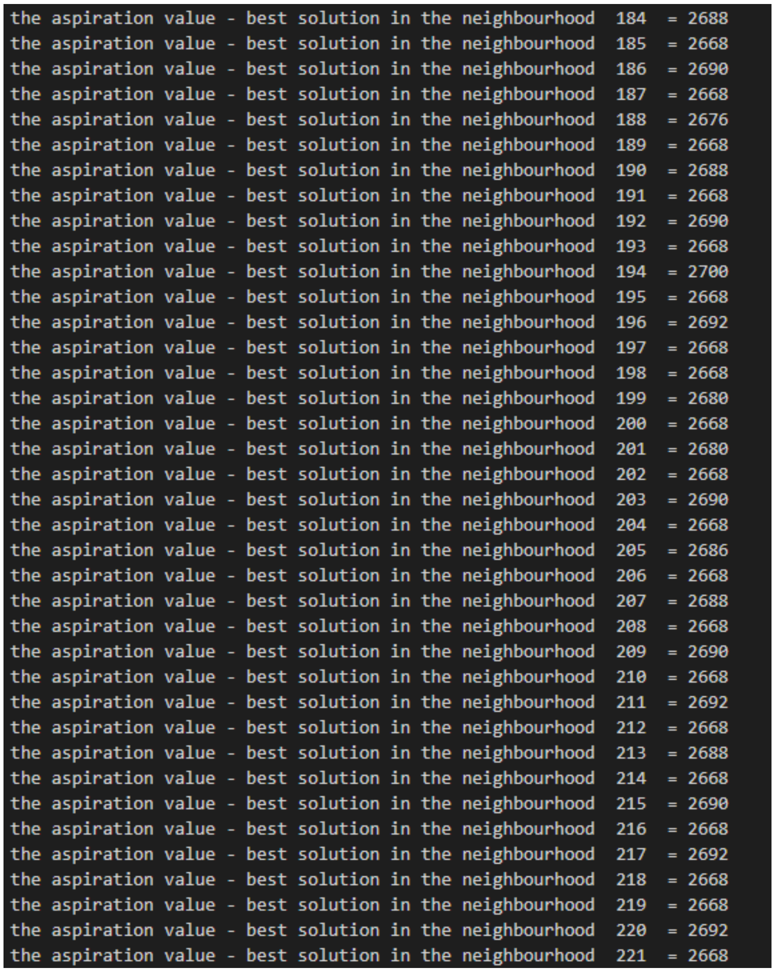
```
the aspiration value - best solution in the neighbourhood  74  = 2668
the aspiration value - best solution in the neighbourhood  75  = 2688
the aspiration value - best solution in the neighbourhood  76  = 2668
the aspiration value - best solution in the neighbourhood  77  = 2690
the aspiration value - best solution in the neighbourhood  78  = 2668
the aspiration value - best solution in the neighbourhood  79  = 2692
the aspiration value - best solution in the neighbourhood  80  = 2668
the aspiration value - best solution in the neighbourhood  81  = 2692
the aspiration value - best solution in the neighbourhood  82  = 2668
the aspiration value - best solution in the neighbourhood  83  = 2680
the aspiration value - best solution in the neighbourhood  84  = 2668
the aspiration value - best solution in the neighbourhood  85  = 2680
the aspiration value - best solution in the neighbourhood  86  = 2668
the aspiration value - best solution in the neighbourhood  87  = 2692
the aspiration value - best solution in the neighbourhood  88  = 2668
the aspiration value - best solution in the neighbourhood  89  = 2686
the aspiration value - best solution in the neighbourhood  90  = 2668
the aspiration value - best solution in the neighbourhood  91  = 2688
the aspiration value - best solution in the neighbourhood  92  = 2668
the aspiration value - best solution in the neighbourhood  93  = 2668
the aspiration value - best solution in the neighbourhood  94  = 2690
the aspiration value - best solution in the neighbourhood  95  = 2668
the aspiration value - best solution in the neighbourhood  96  = 2696
the aspiration value - best solution in the neighbourhood  97  = 2668
the aspiration value - best solution in the neighbourhood  98  = 2688
the aspiration value - best solution in the neighbourhood  99  = 2668
the aspiration value - best solution in the neighbourhood  100  = 2690
the aspiration value - best solution in the neighbourhood  101  = 2668
the aspiration value - best solution in the neighbourhood  102  = 2700
the aspiration value - best solution in the neighbourhood  103  = 2668
the aspiration value - best solution in the neighbourhood  104  = 2692
the aspiration value - best solution in the neighbourhood  105  = 2668
the aspiration value - best solution in the neighbourhood  106  = 2680
the aspiration value - best solution in the neighbourhood  107  = 2668
the aspiration value - best solution in the neighbourhood  108  = 2680
the aspiration value - best solution in the neighbourhood  109  = 2668
the aspiration value - best solution in the neighbourhood  110  = 2676
```



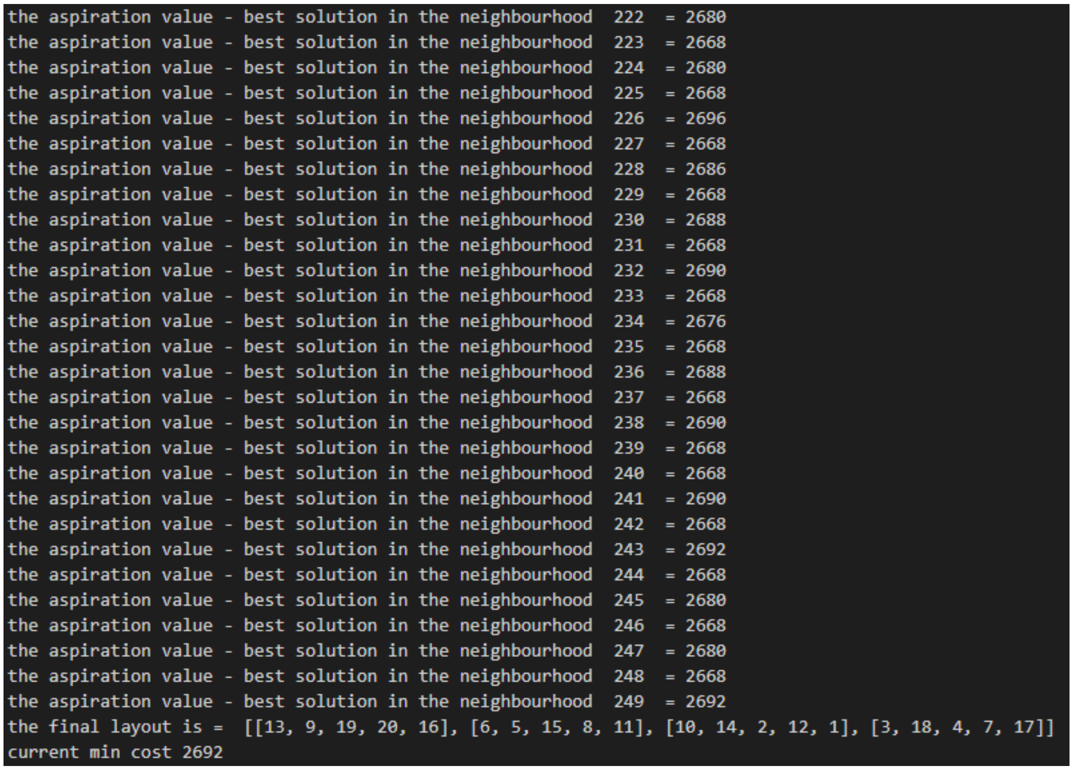
```
the aspiration value - best solution in the neighbourhood  111  = 2668
the aspiration value - best solution in the neighbourhood  112  = 2686
the aspiration value - best solution in the neighbourhood  113  = 2668
the aspiration value - best solution in the neighbourhood  114  = 2668
the aspiration value - best solution in the neighbourhood  115  = 2688
the aspiration value - best solution in the neighbourhood  116  = 2668
the aspiration value - best solution in the neighbourhood  117  = 2690
the aspiration value - best solution in the neighbourhood  118  = 2668
the aspiration value - best solution in the neighbourhood  119  = 2690
the aspiration value - best solution in the neighbourhood  120  = 2668
the aspiration value - best solution in the neighbourhood  121  = 2688
the aspiration value - best solution in the neighbourhood  122  = 2668
the aspiration value - best solution in the neighbourhood  123  = 2690
the aspiration value - best solution in the neighbourhood  124  = 2668
the aspiration value - best solution in the neighbourhood  125  = 2692
the aspiration value - best solution in the neighbourhood  126  = 2668
the aspiration value - best solution in the neighbourhood  127  = 2692
the aspiration value - best solution in the neighbourhood  128  = 2668
the aspiration value - best solution in the neighbourhood  129  = 2680
the aspiration value - best solution in the neighbourhood  130  = 2668
the aspiration value - best solution in the neighbourhood  131  = 2680
the aspiration value - best solution in the neighbourhood  132  = 2668
the aspiration value - best solution in the neighbourhood  133  = 2692
the aspiration value - best solution in the neighbourhood  134  = 2668
the aspiration value - best solution in the neighbourhood  135  = 2668
the aspiration value - best solution in the neighbourhood  136  = 2686
the aspiration value - best solution in the neighbourhood  137  = 2668
the aspiration value - best solution in the neighbourhood  138  = 2688
the aspiration value - best solution in the neighbourhood  139  = 2668
the aspiration value - best solution in the neighbourhood  140  = 2690
the aspiration value - best solution in the neighbourhood  141  = 2668
the aspiration value - best solution in the neighbourhood  142  = 2696
the aspiration value - best solution in the neighbourhood  143  = 2668
the aspiration value - best solution in the neighbourhood  144  = 2688
the aspiration value - best solution in the neighbourhood  145  = 2668
the aspiration value - best solution in the neighbourhood  146  = 2690
```



```
the aspiration value - best solution in the neighbourhood  147  = 2668
the aspiration value - best solution in the neighbourhood  148  = 2700
the aspiration value - best solution in the neighbourhood  149  = 2668
the aspiration value - best solution in the neighbourhood  150  = 2692
the aspiration value - best solution in the neighbourhood  151  = 2668
the aspiration value - best solution in the neighbourhood  152  = 2680
the aspiration value - best solution in the neighbourhood  153  = 2668
the aspiration value - best solution in the neighbourhood  154  = 2680
the aspiration value - best solution in the neighbourhood  155  = 2668
the aspiration value - best solution in the neighbourhood  156  = 2668
the aspiration value - best solution in the neighbourhood  157  = 2690
the aspiration value - best solution in the neighbourhood  158  = 2668
the aspiration value - best solution in the neighbourhood  159  = 2686
the aspiration value - best solution in the neighbourhood  160  = 2668
the aspiration value - best solution in the neighbourhood  161  = 2688
the aspiration value - best solution in the neighbourhood  162  = 2668
the aspiration value - best solution in the neighbourhood  163  = 2690
the aspiration value - best solution in the neighbourhood  164  = 2668
the aspiration value - best solution in the neighbourhood  165  = 2692
the aspiration value - best solution in the neighbourhood  166  = 2668
the aspiration value - best solution in the neighbourhood  167  = 2688
the aspiration value - best solution in the neighbourhood  168  = 2668
the aspiration value - best solution in the neighbourhood  169  = 2690
the aspiration value - best solution in the neighbourhood  170  = 2668
the aspiration value - best solution in the neighbourhood  171  = 2692
the aspiration value - best solution in the neighbourhood  172  = 2668
the aspiration value - best solution in the neighbourhood  173  = 2692
the aspiration value - best solution in the neighbourhood  174  = 2668
the aspiration value - best solution in the neighbourhood  175  = 2680
the aspiration value - best solution in the neighbourhood  176  = 2668
the aspiration value - best solution in the neighbourhood  177  = 2668
the aspiration value - best solution in the neighbourhood  178  = 2680
the aspiration value - best solution in the neighbourhood  179  = 2668
the aspiration value - best solution in the neighbourhood  180  = 2696
the aspiration value - best solution in the neighbourhood  181  = 2668
the aspiration value - best solution in the neighbourhood  182  = 2686
the aspiration value - best solution in the neighbourhood  183  = 2668
```



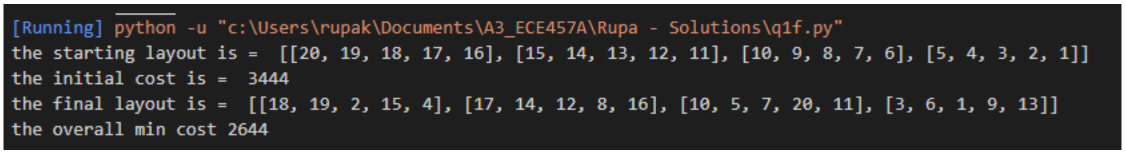
```
the aspiration value - best solution in the neighbourhood  184  = 2688
the aspiration value - best solution in the neighbourhood  185  = 2668
the aspiration value - best solution in the neighbourhood  186  = 2690
the aspiration value - best solution in the neighbourhood  187  = 2668
the aspiration value - best solution in the neighbourhood  188  = 2676
the aspiration value - best solution in the neighbourhood  189  = 2668
the aspiration value - best solution in the neighbourhood  190  = 2688
the aspiration value - best solution in the neighbourhood  191  = 2668
the aspiration value - best solution in the neighbourhood  192  = 2690
the aspiration value - best solution in the neighbourhood  193  = 2668
the aspiration value - best solution in the neighbourhood  194  = 2700
the aspiration value - best solution in the neighbourhood  195  = 2668
the aspiration value - best solution in the neighbourhood  196  = 2692
the aspiration value - best solution in the neighbourhood  197  = 2668
the aspiration value - best solution in the neighbourhood  198  = 2668
the aspiration value - best solution in the neighbourhood  199  = 2680
the aspiration value - best solution in the neighbourhood  200  = 2668
the aspiration value - best solution in the neighbourhood  201  = 2680
the aspiration value - best solution in the neighbourhood  202  = 2668
the aspiration value - best solution in the neighbourhood  203  = 2690
the aspiration value - best solution in the neighbourhood  204  = 2668
the aspiration value - best solution in the neighbourhood  205  = 2686
the aspiration value - best solution in the neighbourhood  206  = 2668
the aspiration value - best solution in the neighbourhood  207  = 2688
the aspiration value - best solution in the neighbourhood  208  = 2668
the aspiration value - best solution in the neighbourhood  209  = 2690
the aspiration value - best solution in the neighbourhood  210  = 2668
the aspiration value - best solution in the neighbourhood  211  = 2692
the aspiration value - best solution in the neighbourhood  212  = 2668
the aspiration value - best solution in the neighbourhood  213  = 2688
the aspiration value - best solution in the neighbourhood  214  = 2668
the aspiration value - best solution in the neighbourhood  215  = 2690
the aspiration value - best solution in the neighbourhood  216  = 2668
the aspiration value - best solution in the neighbourhood  217  = 2692
the aspiration value - best solution in the neighbourhood  218  = 2668
the aspiration value - best solution in the neighbourhood  219  = 2668
the aspiration value - best solution in the neighbourhood  220  = 2692
the aspiration value - best solution in the neighbourhood  221  = 2668
```



```
the aspiration value - best solution in the neighbourhood  222  = 2680
the aspiration value - best solution in the neighbourhood  223  = 2668
the aspiration value - best solution in the neighbourhood  224  = 2680
the aspiration value - best solution in the neighbourhood  225  = 2668
the aspiration value - best solution in the neighbourhood  226  = 2696
the aspiration value - best solution in the neighbourhood  227  = 2668
the aspiration value - best solution in the neighbourhood  228  = 2686
the aspiration value - best solution in the neighbourhood  229  = 2668
the aspiration value - best solution in the neighbourhood  230  = 2688
the aspiration value - best solution in the neighbourhood  231  = 2668
the aspiration value - best solution in the neighbourhood  232  = 2690
the aspiration value - best solution in the neighbourhood  233  = 2668
the aspiration value - best solution in the neighbourhood  234  = 2676
the aspiration value - best solution in the neighbourhood  235  = 2668
the aspiration value - best solution in the neighbourhood  236  = 2688
the aspiration value - best solution in the neighbourhood  237  = 2668
the aspiration value - best solution in the neighbourhood  238  = 2690
the aspiration value - best solution in the neighbourhood  239  = 2668
the aspiration value - best solution in the neighbourhood  240  = 2668
the aspiration value - best solution in the neighbourhood  241  = 2690
the aspiration value - best solution in the neighbourhood  242  = 2668
the aspiration value - best solution in the neighbourhood  243  = 2692
the aspiration value - best solution in the neighbourhood  244  = 2668
the aspiration value - best solution in the neighbourhood  245  = 2680
the aspiration value - best solution in the neighbourhood  246  = 2668
the aspiration value - best solution in the neighbourhood  247  = 2680
the aspiration value - best solution in the neighbourhood  248  = 2668
the aspiration value - best solution in the neighbourhood  249  = 2692
the final layout is =  [[13, 9, 19, 20, 16], [6, 5, 15, 8, 11], [10, 14, 2, 12, 1], [3, 18, 4, 7, 17]]
current min cost 2692
```

<u>Case f: Use less than the whole neighbourhood to select the next solution</u>

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking less than the whole neighbourhood – don't swap 10 neighbouring departments for each department
Tabu list type	Recency-based
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	Use less than the whole neighbourhood to select the next solution
** Aspiration Criteria – *Use less than the whole neighbourhood to select the next solution*	



```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1f.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[18, 19, 2, 15, 4], [17, 14, 12, 8, 16], [10, 5, 7, 20, 11], [3, 6, 1, 9, 13]]
the overall min cost 2644
```

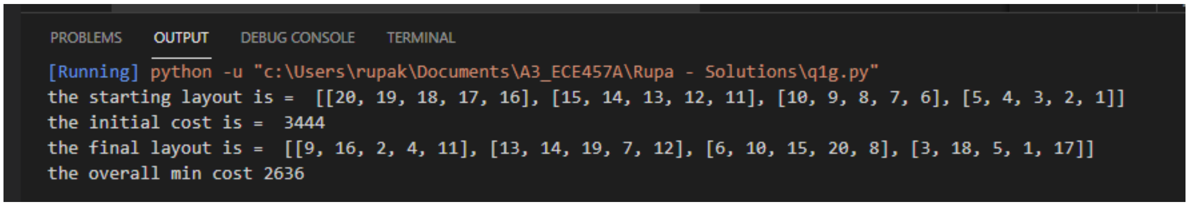
Min cost = 2644

Analysis:

Using less than the neighbourhood function allowed me to get a worse solution because sometimes the solutions that we weren't considering could have been the optimal solution.

Case g: Add a frequency based tabu list to encourage the search to diversify (count = 5)

Feature	Definition
Neighborhood and move operator	Single swaps from current location
Neighbourhood robustness	Checking less than the whole neighbourhood – don’t swap 10 neighbouring departments for each department
Tabu list type	Frequency based tabu list to encourage the search to diversify
Tabu list size	20
Stopping Criterion	250 iterations
Aspiration Criteria	None
** Frequency based tabu list to encourage the search to diversify	



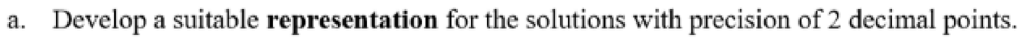
```
[Running] python -u "c:\Users\rupak\Documents\A3_ECE457A\Rupa - Solutions\q1g.py"
the starting layout is =  [[20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1]]
the initial cost is =  3444
the final layout is =  [[9, 16, 2, 4, 11], [13, 14, 19, 7, 12], [6, 10, 15, 20, 8], [3, 18, 5, 1, 17]]
the overall min cost 2636
```

Min cost = 2636

Analysis:

Using a frequency-based list to diversify actually gave me suboptimal solutions compared to the base.

Question 2:



a. Develop a suitable **representation** for the solutions with precision of 2 decimal points.

Kp = (2, 18)

[0, 16] – range

[(16 – 0) * 100] + 1 = 1601

Representation in bits = 11 bits

TI = (1.05, 9.42)

[0, 8.37] – range

[(8.37 – 0) * 100] + 1 = 838

Representation in bits = 10 bits

TD = (0.26, 2.37)

[0, 2.11] – range

[(2.11 – 0) * 100] + 1 = 212

Representation in bits = 8 bits

b. Formulate a **fitness function** that you can use to evaluate a solution.

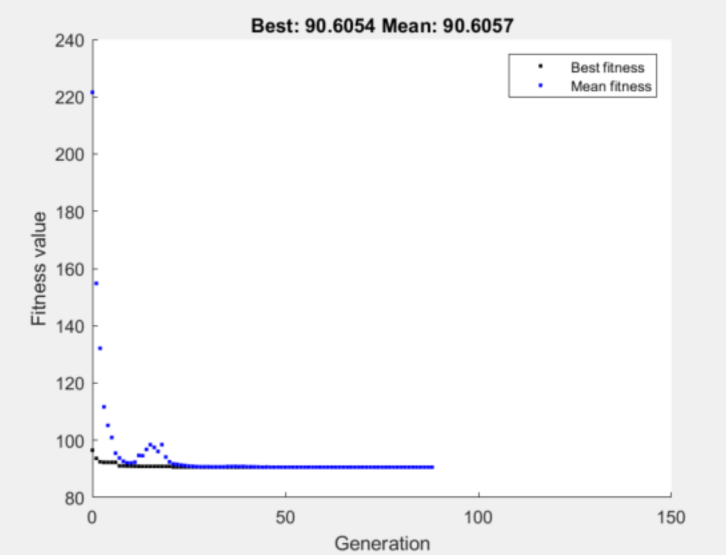
For minimization of all the values, we want a fitness function that maximizes the reward to smaller values. We also want a function that considers all of the parameters.

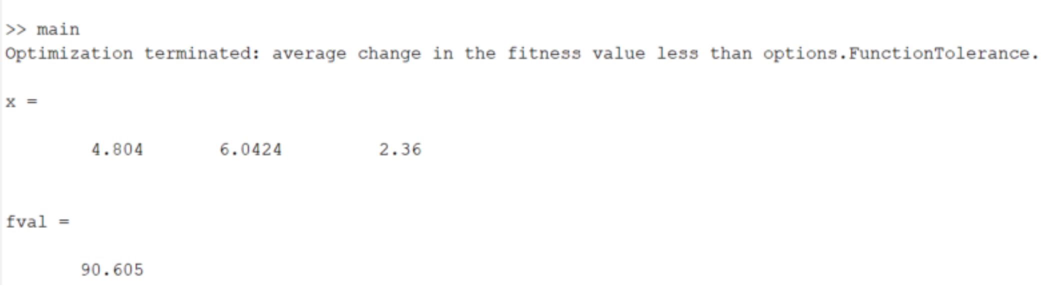
Fitness function for GA that does maximization = 0.7(1/ISE) + 1/t_r + 1/t_s + 1/M_p.

**Key: Placed more weightage to the parameter ISE because it has more weight in this formula

c. **Implement** GA algorithm to solve this problem, use a population of 50 individuals, number of generations of 150, crossover probability of 0.6 and mutation probability of 0.25. Use FPS parent selection strategy and an elitism survival selection strategy keeping the best two individuals across generations. Select proper crossover and mutation operators and solve the problem.

d. **Plot** the fitness of best solution in each generation across the generations.



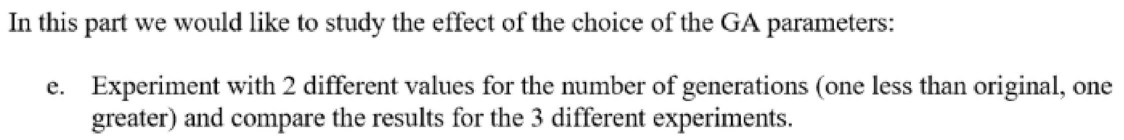

```
>> main
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

        4.804       6.0424         2.36


fval =

       90.605
```

K_P = 4.804

T_i = 6.0424

T_d = 2.36

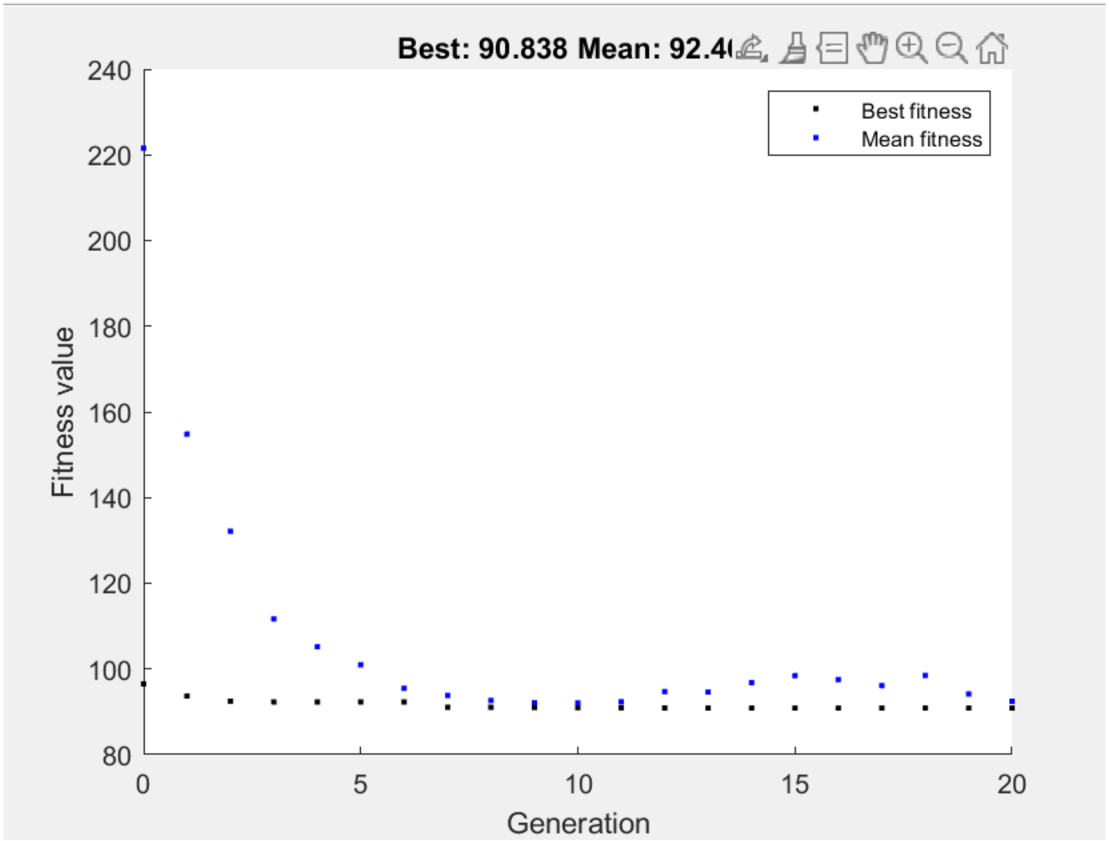
Fval = 90.605

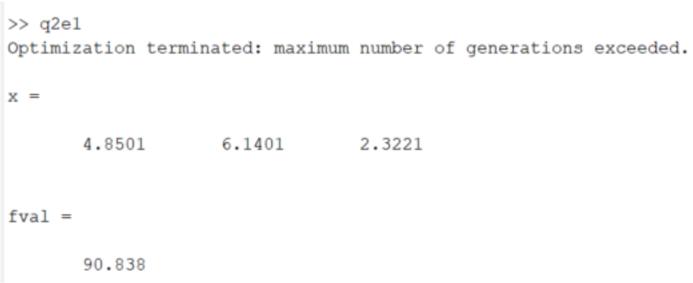


In this part we would like to study the effect of the choice of the GA parameters:

e. Experiment with 2 different values for the number of generations (one less than original, one greater) and compare the results for the 3 different experiments.

Generations = 20





```
>> q2e1
Optimization terminated: maximum number of generations exceeded.

x =

       4.8501       6.1401       2.3221


fval =

       90.838
```

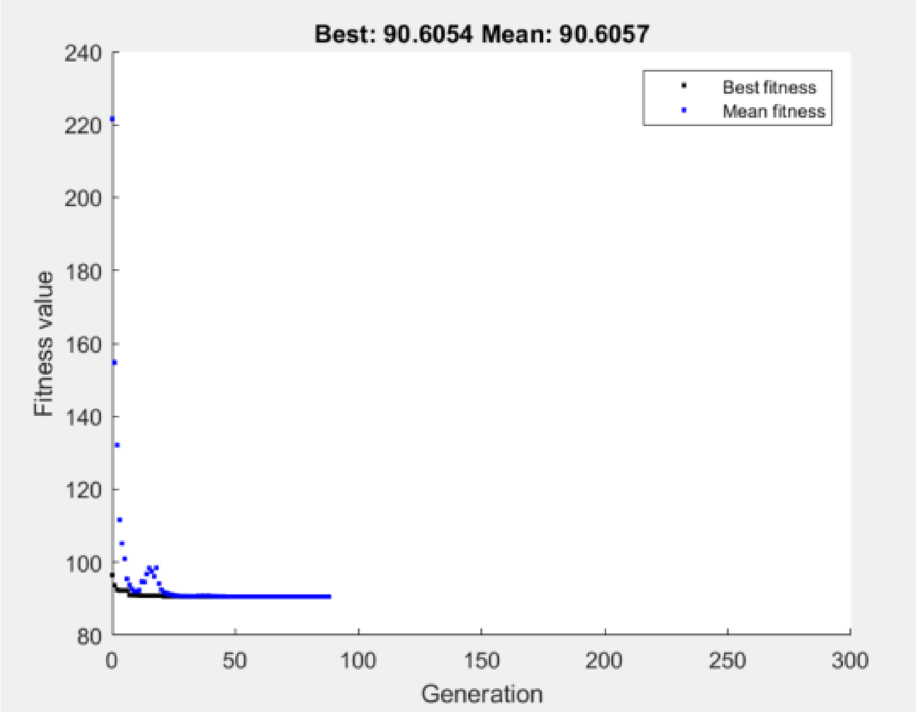
K_P = 4.8501

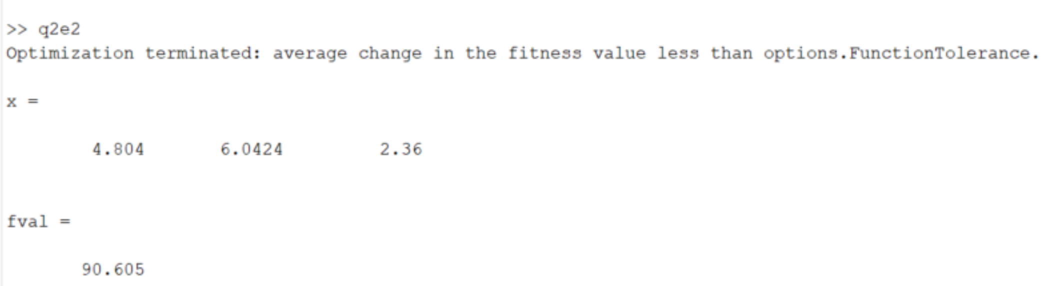
T_i = 6.1401

T_d = 2.3221

Fval = 90.838

Generations = 300





```
>> q2e2
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

        4.804       6.0424        2.36


fval =

       90.605
```

K_P = 4.804

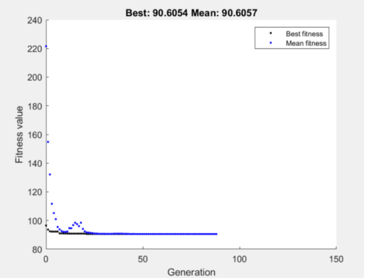
T_i = 6.0424

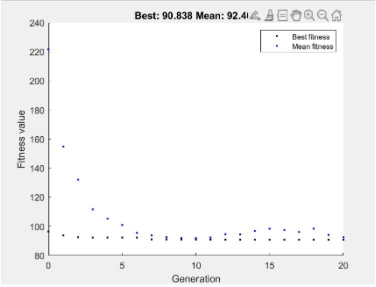
T_d = 2.36

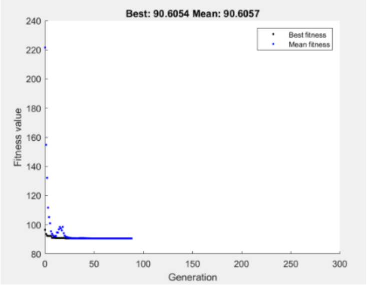
Fval = 90.605

Comparing Results:

	Experiment 1 – Gen 150	Experiment 2	Experiment 3
K_P	4.804	4.8501	4.804
T_i	6.0424	6.1401	6.0424
T_d	2.36	2.3221	2.36
Fval	90.6054	90.838	90.6054



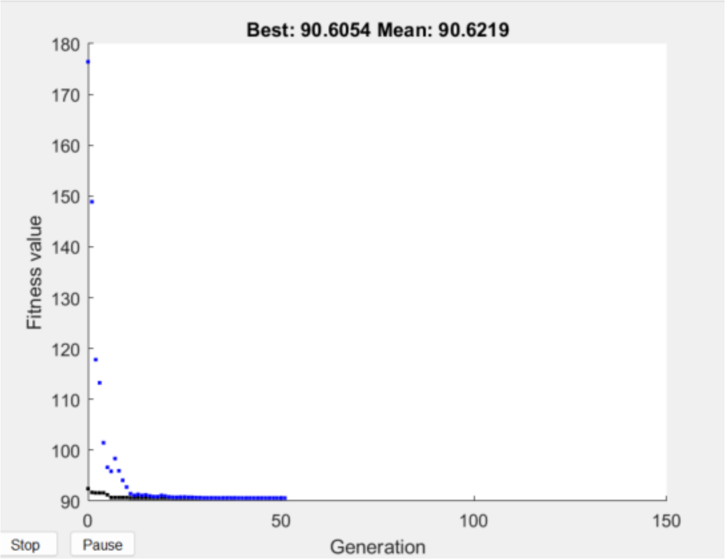


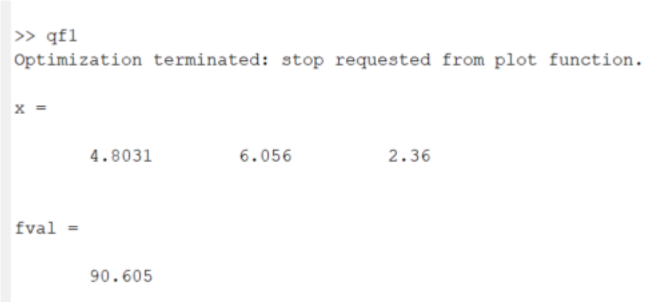


Comparing the values and looking at the behaviour of the graphs , we see that the values have not converged yet at experiment 2. But the values have converged at generation before the max specified in experiment 1 and 3. This is why both the results from the experiments are the same.

f. Experiment with 2 other population sizes (one less than 50, one greater) and compare the results for the 3 different population sizes.

Population size = 200





```
>> qf1
Optimization terminated: stop requested from plot function.

x =

       4.8031        6.056         2.36


fval =

       90.605
```

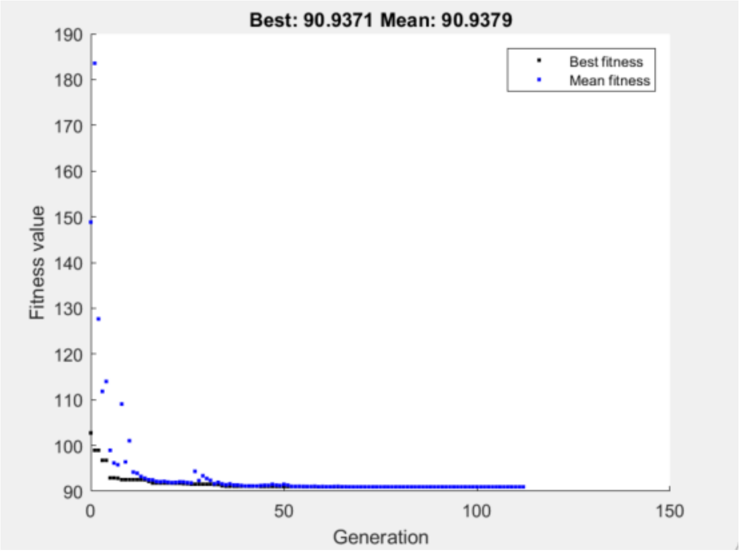
K_P = 4.8031

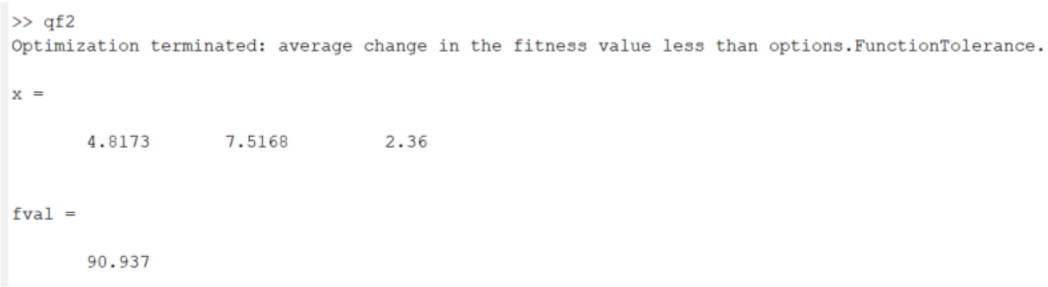
T_i = 6.056

T_d = 2.36

Fval = 90.605

Population size = 10





```
>> qf2
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

       4.8173       7.5168         2.36


fval =

       90.937
```

K_P = 4.8173

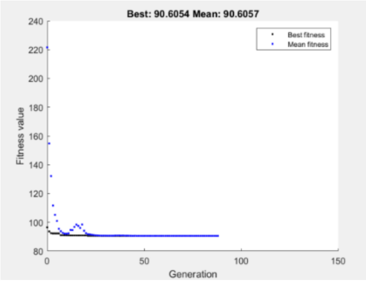
T_i = 7.5168

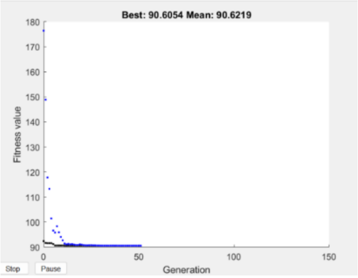
T_d = 2.36

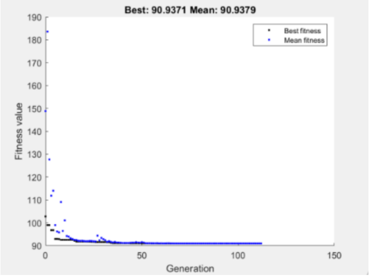
Fval = 90.937

Comparing Results:

	Experiment 1 – Pop = 50	**Experiment 2 – Pop = 200**	**Experiment 3 – Pop = 10**
K_P	4.804	4.8031	4.8173
T_i	6.0424	6.056	7.5168
T_d	2.36	2.36	2.36
Fval	90.654	90.6054	90.9371





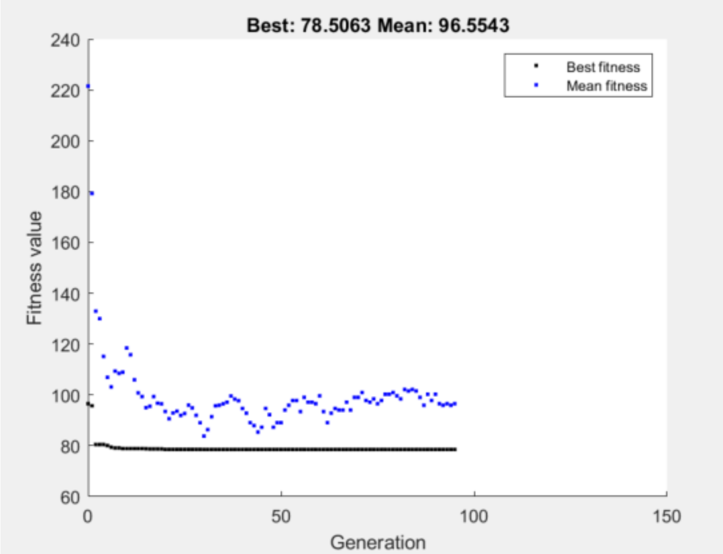


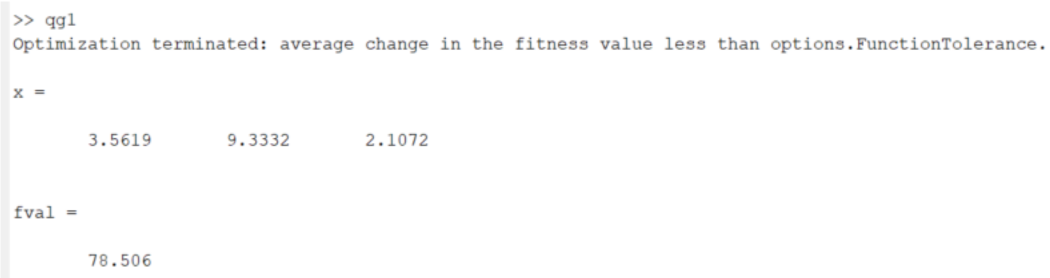
As I was running the simulation, experiment 3 ran the fastest because it had the least population size but experiment 2 took the longest. As well, we can see that we achieved optimal values faster with the greater population size, as we see convergence was reached much more early in experiment 2 than 1. This means that in experiment 1, we were still exploring lots of other population spaces in each generation and sometimes getting sub-optimal solution.

The conclusion is that, if there is time and enough computational capacity, it is better to use a greater population size to achieve convergence faster.

g. Experiment with 2 different crossover probabilities (one less than original, one greater) and compare the results for the 3 different experiments.

Crossover Probability = 0.1





```
>> qg1
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

       3.5619       9.3332       2.1072


fval =

       78.506
```

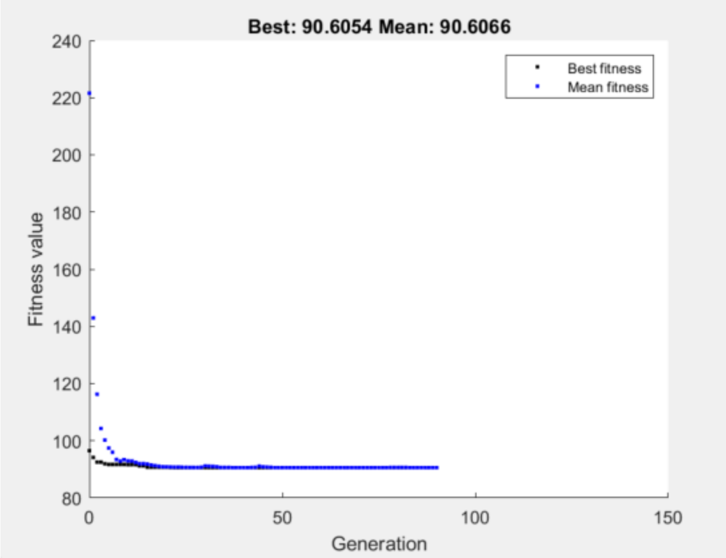
K_P = 3.5619

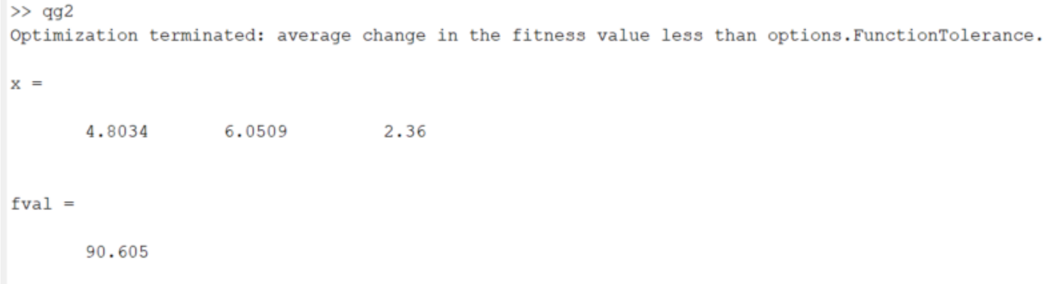
T_i = 9.3332

T_d = 2.1071

Fval = 78.506

Crossover Probability = 0.9




```
>> qg2
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

       4.8034       6.0509         2.36


fval =

       90.605
```

K_P = 4.8034

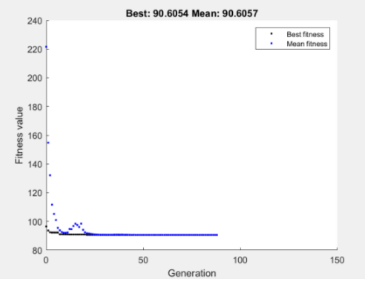
T_i = 6.0509

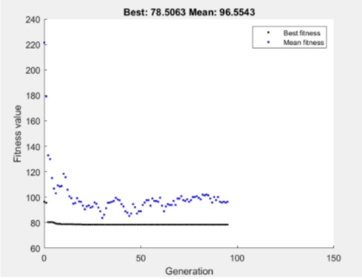
T_d = 2.36

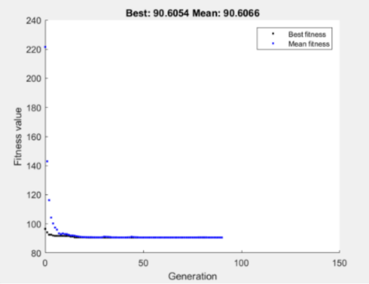
Fval = 90.605

Comparing Results:

	Experiment 1 – crossover P = 0.6	Experiment 2 – crossover P = 0.1	Experiment 3 – crossover P = 0.9
K_P	4.804	3.5619	4.8034
T_i	6.0424	9.3332	6.0509
T_d	2.36	2.1071	2.36
Fval	90.654	78.506	90.605



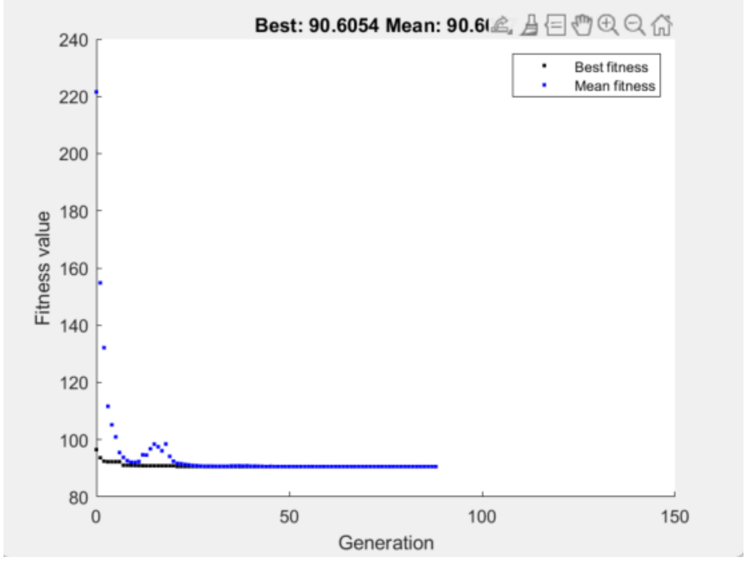


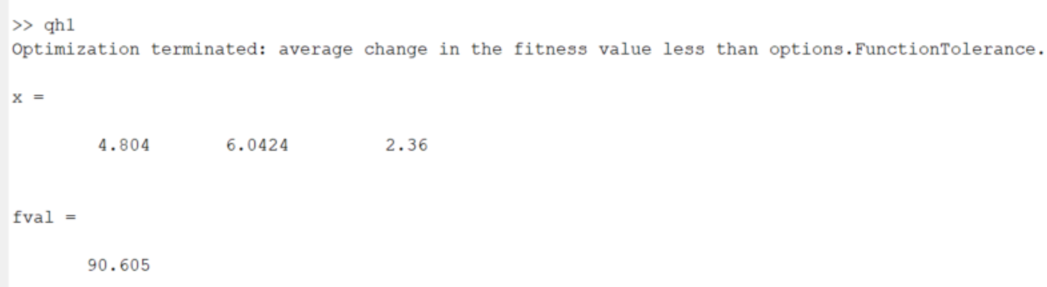


According to this, we can see that decreasing the cross over probability invites more randomness to the f value because we are accepting past solutions, and iteratively not picking the combination of the most optimal solutions. However, as we see in the last graph, for experiment 3, increasing the probability, ensures that we converge quicker because we increase the probability of combining the best solutions.

h. Experiment with 2 different mutation probabilities (one less than original, one greater and compare the results for the 3 different experiments.

Mutation Probability = 0.1





```
>> qh1
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

        4.804       6.0424         2.36


fval =

       90.605
```

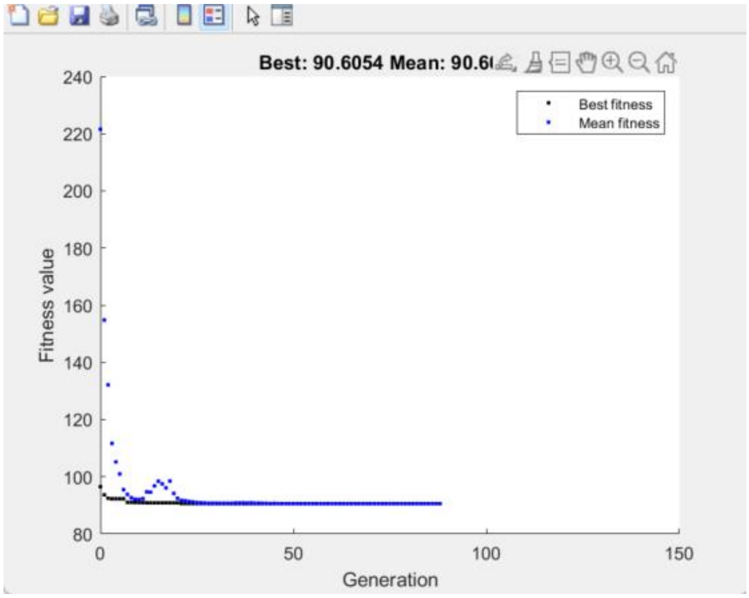
K_P = 4.804

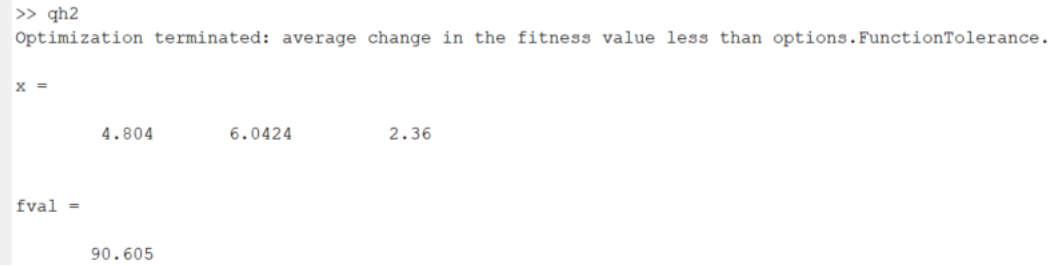
T_i = 6.0424

T_d = 2.36

Fval = 90.605

Mutation Probability = 0.9





```
>> qh2
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

        4.804       6.0424         2.36


fval =

       90.605
```

K_P = 4.804

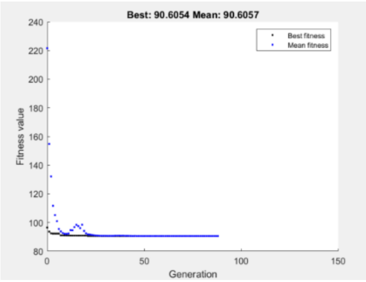
T_i = 6.0424

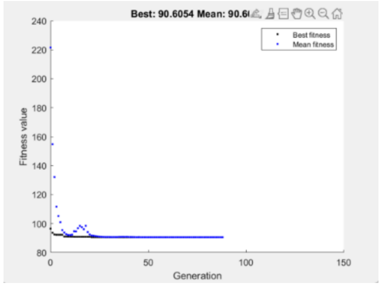
T_d = 2.36

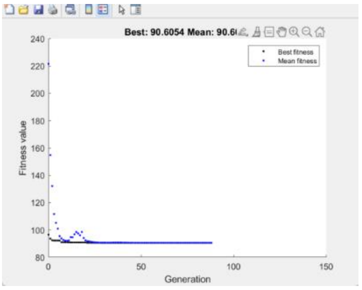
Fval = 90.605

Comparing Results:

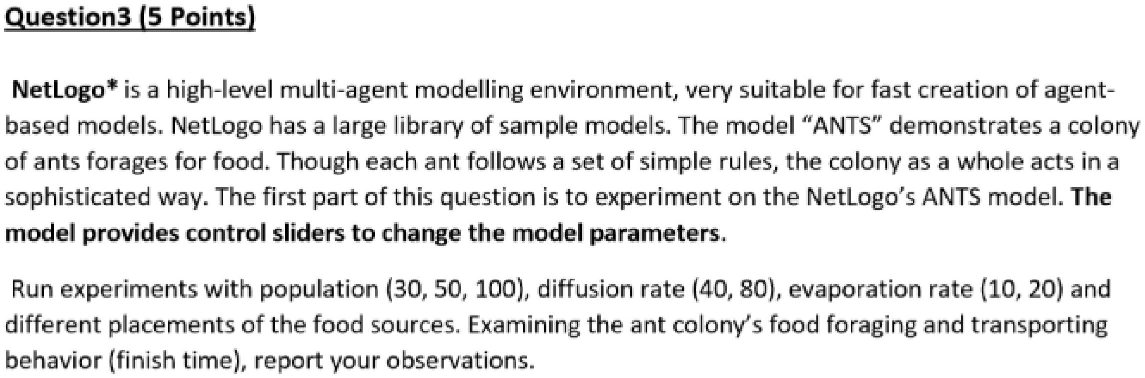
	Experiment 1 – mutation P = 0.25	Experiment 2 – mutation P = 0.1	Experiment 3 – mutation P = 0.9
K_P	4.804	4.804	4.804
T_i	6.0424	6.0424	6.0424
T_d	2.36	2.36	2.36
Fval	90.654	90.605	90.605







Comparing the results of this graph, we see that changing the mutation probability didn't affect the f values or the convergence.



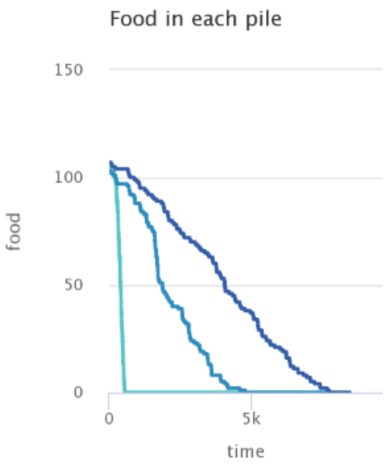
Question3 (5 Points)

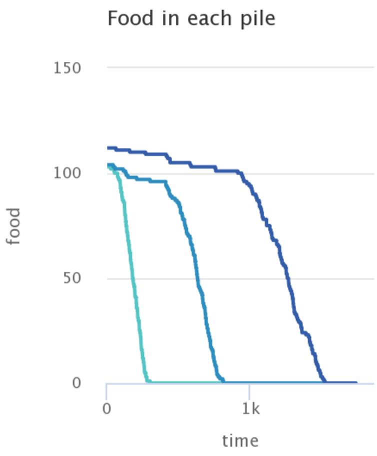
NetLogo* is a high-level multi-agent modelling environment, very suitable for fast creation of agent-based models. NetLogo has a large library of sample models. The model "ANTS" demonstrates a colony of ants forages for food. Though each ant follows a set of simple rules, the colony as a whole acts in a sophisticated way. The first part of this question is to experiment on the NetLogo's ANTS model. **The model provides control sliders to change the model parameters**.

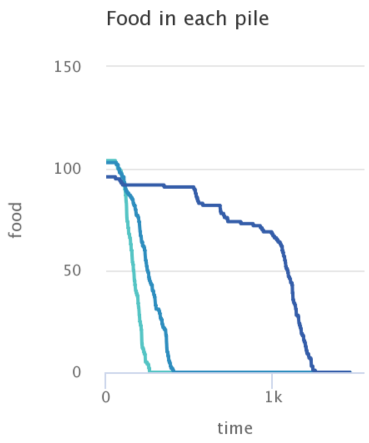
Run experiments with population (30, 50, 100), diffusion rate (40, 80), evaporation rate (10, 20) and different placements of the food sources. Examining the ant colony's food foraging and transporting behavior (finish time), report your observations.

		Impact	
No	**Change**	**Food foraging (exploration of search)**	**Transporting behaviour (finish time – when each of the food piles are complete)**
1	Effect of Increasing Population	As the population increases, the effect that was notice that food foraging also increases	As the population increases, the finish time was much faster for each of the food sources
2	Effect of Increasing Diffusion Rate	Food foraging/exploration was minimal here simply because most of the ants tended to cluster into areas that had more pheromone	As the diffusion increases, the finish time was much faster for each of the food sources (almost by double)
3	Effect of Increasing Evaporation Rate	Lower evaporation also reduces the rate of exploration, so most ants are clustered around one area	Increasing the evaporation reduced the time to consume all of the food sources
4	Effect of changing the location of the food sources	The closer the sources, the fewer the exploration of the ants searching for food	The closer the food sources were to the source of ants, the faster each of the food piles were depleted

No 1 Screenshots:

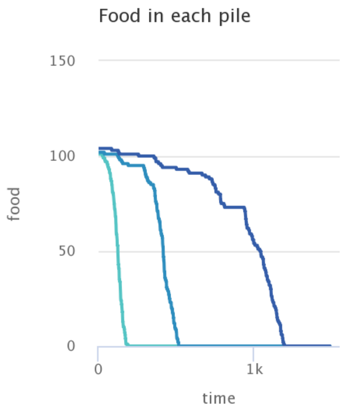


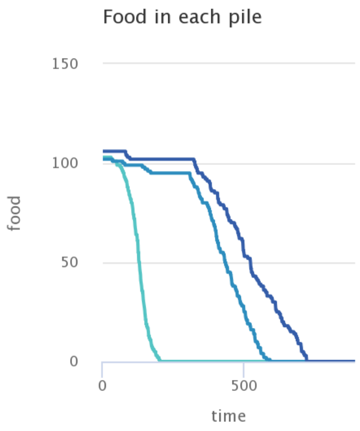




Population = 30, 50, 90

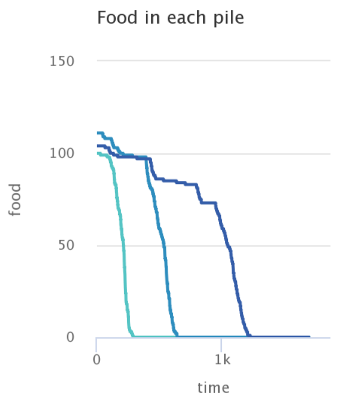
No2 Screenshots:

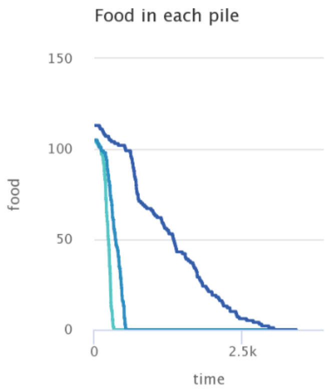




Diffusion Rate: 40, 80

No3 Screenshots:

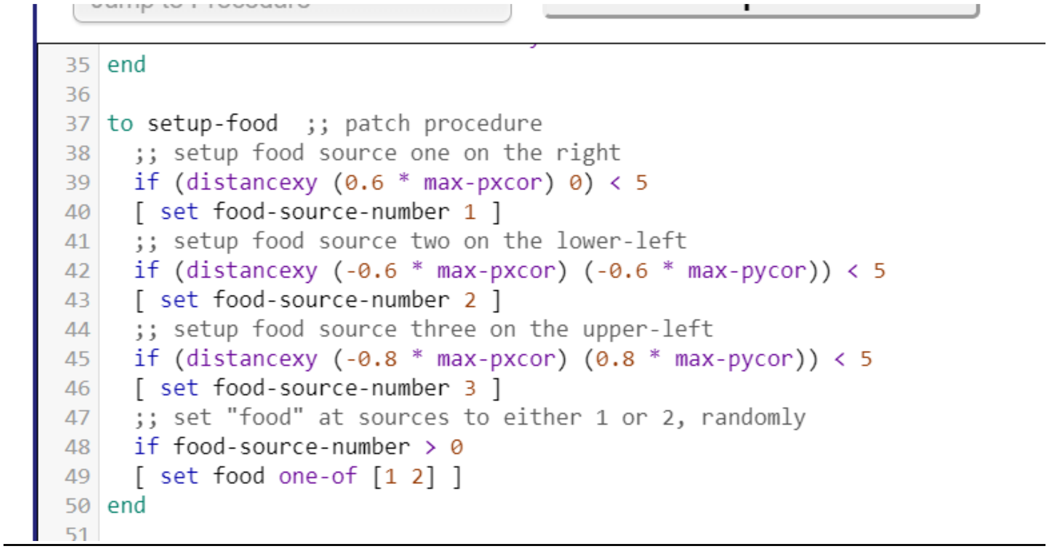




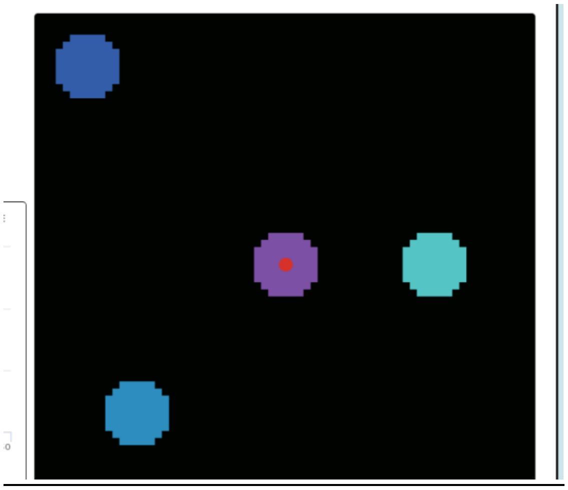
Evaporation Rate: 10, 20

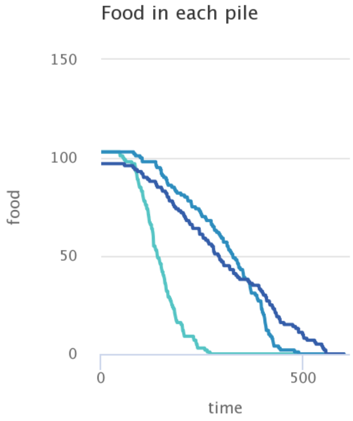
No 4 Screenshots:

Original Food Source Locations:

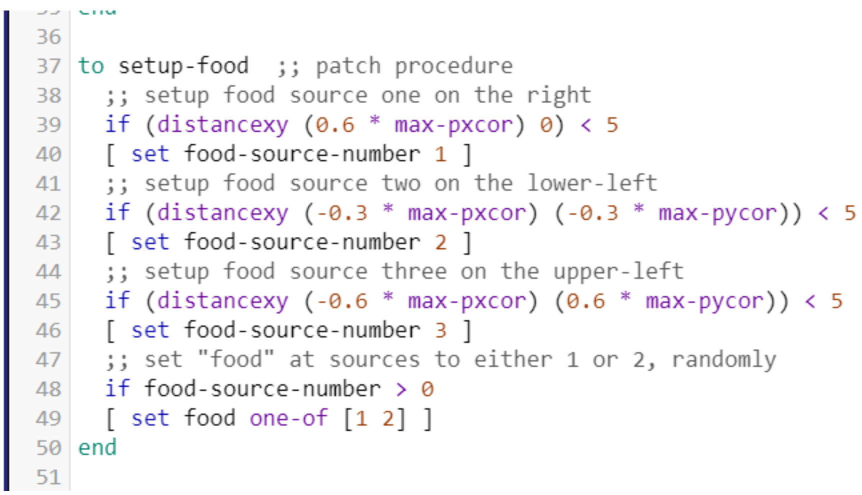


```
end

to setup-food  ;; patch procedure
  ;; setup food source one on the right
  if (distancexy (0.6 * max-pxcor) 0) < 5
  [ set food-source-number 1 ]
  ;; setup food source two on the lower-left
  if (distancexy (-0.6 * max-pxcor) (-0.6 * max-pycor)) < 5
  [ set food-source-number 2 ]
  ;; setup food source three on the upper-left
  if (distancexy (-0.8 * max-pxcor) (0.8 * max-pycor)) < 5
  [ set food-source-number 3 ]
  ;; set "food" at sources to either 1 or 2, randomly
  if food-source-number > 0
  [ set food one-of [1 2] ]
end
```

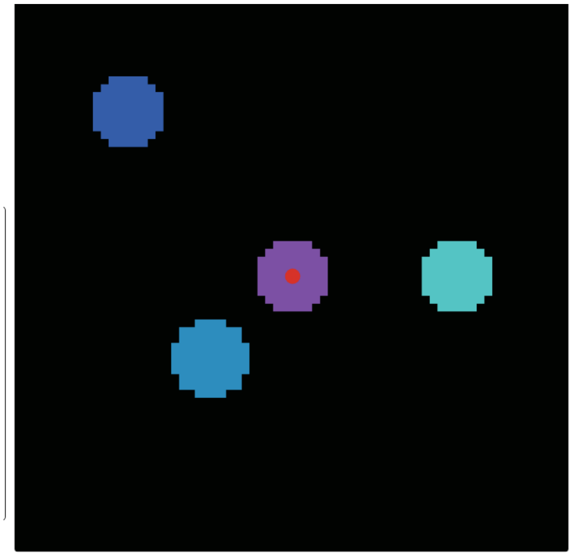


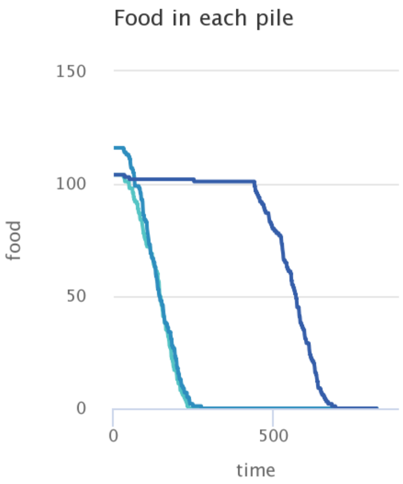


New Food Source Locations:



```
end

to setup-food  ;; patch procedure
  ;; setup food source one on the right
  if (distancexy (0.6 * max-pxcor) 0) < 5
  [ set food-source-number 1 ]
  ;; setup food source two on the lower-left
  if (distancexy (-0.3 * max-pxcor) (-0.3 * max-pycor)) < 5
  [ set food-source-number 2 ]
  ;; setup food source three on the upper-left
  if (distancexy (-0.6 * max-pxcor) (0.6 * max-pycor)) < 5
  [ set food-source-number 3 ]
  ;; set "food" at sources to either 1 or 2, randomly
  if food-source-number > 0
  [ set food one-of [1 2] ]
end

```




Food in each pile
150
100
50
0
food
0
500
time