TRAVELLING SALESMAN PROBLEM

Solution using Genetic Algorithm

Introduction

Traveling salesman is an all time famous problem which comes from the family of NP-complete problems. A salesman has to complete a round trip covering all the cities. All the cities are connected through roads which have some cost associated with them. The problem is depicted in the form of a weighted undirected graph with cities acting as nodes and roads acting as weighted edges. With this depiction, the problem transforms into search for a hamiltonian cycle with total minimum weight.

Implementation Details

The given problem has 14 cities which are hard-coded in the form of an adjacency matrix. The distance between the cities are given in units of 1000 kms and the cities with no path between them have the value as infinity. The value of infinity is taken as 1001,000 kms for calculation purposes. The state is defined as permutations of letters from A to N which signifies the 14 cities which are required to be visited. The path is calculated as the order of the cities in the permutation with path cost as the sum of the edges between ith and (i+1)th city and edge between last and the first city. The fitness function is defined as 1/path_cost(permutation). The path cost used in the fitness value function does not account for the unit in terms of 1000kms for efficient calculation and better performance.

The algorithm is run for a maximum for 3500 iterations (reduced to 2500 in the final code for saving time of evaluator) without any other stopping criteria. The overall maximum fitness is maintained which

is compared with the best achieved in each generation to give the minimum value of the the path achieved over all the generations.

Improvements

Some important observations of the textbook version to form a base for the improvements:-

1. The best fitness value for subsequent generations vary a lot. This leads to slow convergence and portray inefficient pass on of the information from parent to children as the entire population is swapped with the new population.
2. The adjacency matrix denoting the distance between the cities is sparse i.e. many cities don't have a path between them.
3. The improvement in performance need to incorporate both the minimum value of the path and the number of generations taken to converge to that solution.

- Improved Mutation Function:-

The mutation function is changed in a way to incorporate the steepest hill ascent in order to move to best possible neighbour which could be formed by swapping two cities in the permutation. A nested for loop is used to try all swaps in the given permutation and the best possible variant is returned. The problem of local maximum may arrive with this improvement in the case when the permutation passed for mutation is already the best child. This is prevented by not returning back the child passed for mutation and instead selecting a random other child.

This improvement worked well for this problem as the adjacency matrix is very sparse i.e. many entries in the matrix are infinity which denote no path between the cities. For the textbook version of the genetic algorithm, It becomes difficult to converge on to a

finite possible path. The improved mutation function helps the algorithm to converge quickly to a maximised fitness value.

- Improved Reproduce Function:-

The improved crossover function choses a random length in the range 0 and 13, and explore all the substrings of that length from both the parents to form the best possible child according to fitness value.

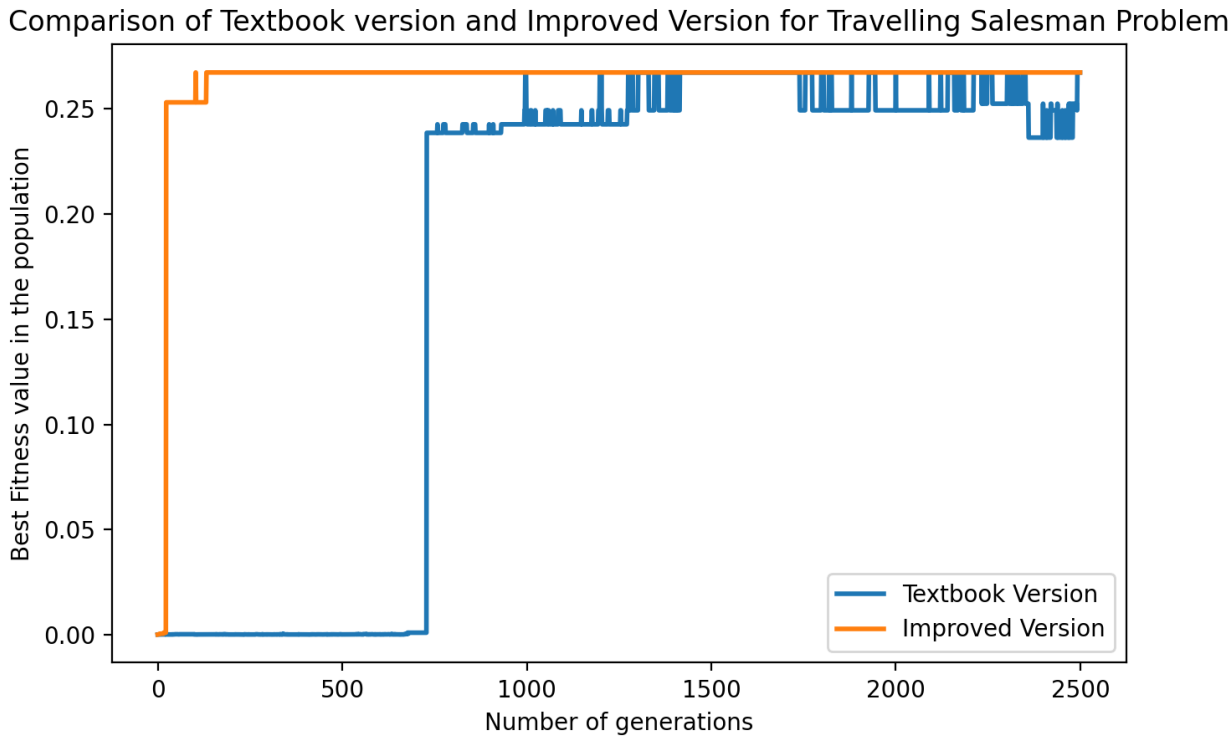
This idea tries to take advantage of a parent solving a partial city tour problem with minimum cost and pass that to the child. The function is not completely converted to the steepest ascent hill climbing as the length of the substring is chosen randomly, thus promoting the randomness and helping with the local maximum problem. Another important advantage of this improvement is to pass on the parent to the next generation if it is better than the children formed. This helps greatly in stabilising the fitness value over the generation which vary greatly in the case of the textbook version. Even if a good fitness value state is formed in a generation it does not pass the information well to the next generations in the textbook version thus converging slowly to the minimum cost.

- Random Selection:-

The probability of selection of the parents for crossover is changed to squared function to favour a parent with more fitness value. The probability of selection was changed to square(fitness value)/ sum of all square(fitness value) for increasing the gap between the probability of choosing a parent with more fitness value than others.

This helps in accelerating the convergence of the algorithm towards a path with minimum cost. Other values of power were also employed for the probability of selection but the best results were achieved with the square function.

Results

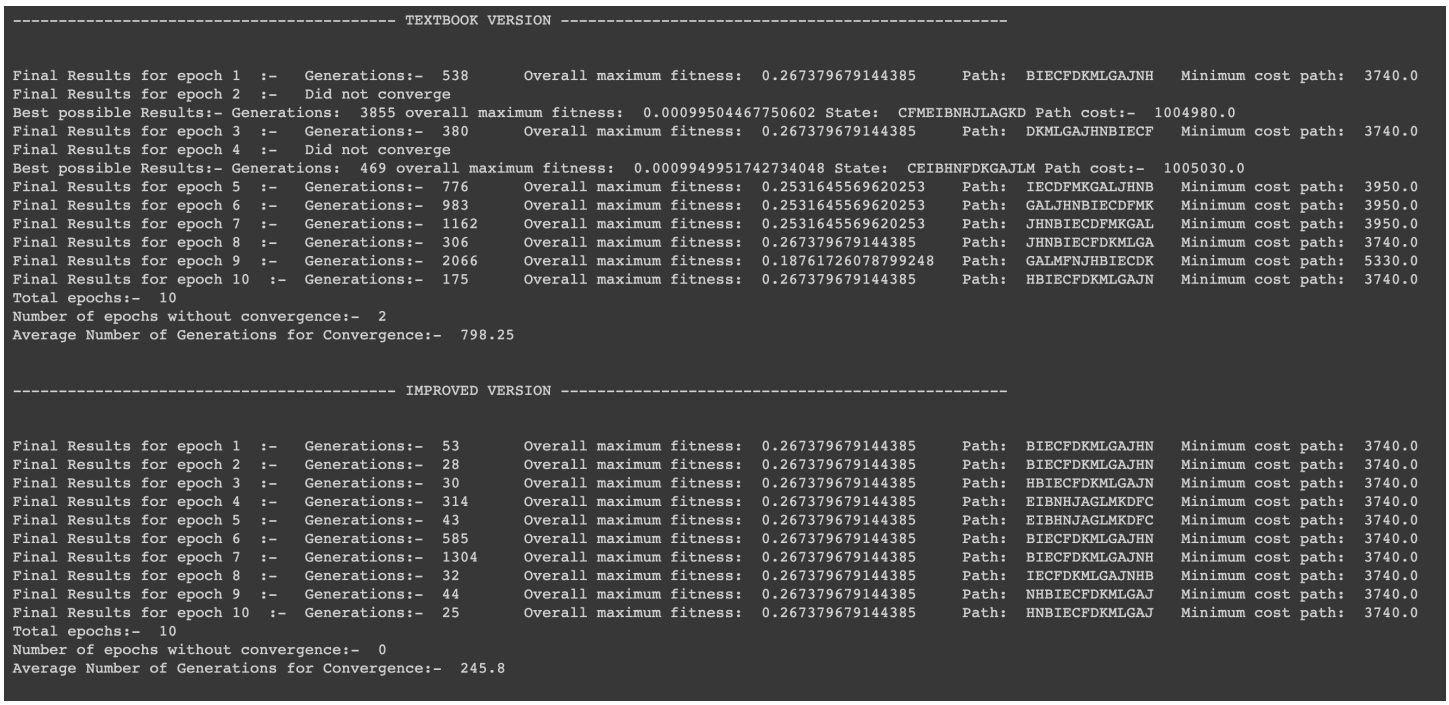



```
Final Results:-
For textbook version, Statistics for best found results:-
Generation:  998        Max Fitness:  0.267379679144385         state with max fitness:  MLGAJNHBIECFDK
Minimum path cost found:  3740.0



For improved version, Statistics for best found results:-
Generation:  105        Max Fitness:  0.267379679144385         state with max fitness:  CEIBNHJAGLMKDF
Minimum path cost found:  3740.0
```

The Improved version reaches the minimum path value of 3740 kms in 105 generations while the textbook version takes 998 generations to converge to the minimum path. As advent from the graph above, the best fitness value for generations seem to be a bit erratic for the textbook version. The improved version however

converges quickly to the minimum path value which is efficiently passed on to the future generations.

The average statistics for 10 runs of both the textbook version and the improved version of the algorithm for the same problem are displayed in the screenshot below.



```
---------------------------------------- TEXTBOOK VERSION ------------------------------------------------


Final Results for epoch 1  :-   Generations:-  538      Overall maximum fitness:  0.267379679144385     Path:  BIECFDKMLGAJNH   Minimum cost path:  3740.0
Final Results for epoch 2  :-   Did not converge
Best possible Results:- Generations:  3855 overall maximum fitness:  0.00099504467750602 State:  CFMEIBNHJLAGKD Path cost:-  1004980.0
Final Results for epoch 3  :-   Generations:-  380      Overall maximum fitness:  0.267379679144385     Path:  DKMLGAJHNBIECF   Minimum cost path:  3740.0
Final Results for epoch 4  :-   Did not converge
Best possible Results:- Generations:  469 overall maximum fitness:  0.0009949951742734048 State:  CEIBHNFDKGAJLM Path cost:-  1005030.0
Final Results for epoch 5  :-   Generations:-  776      Overall maximum fitness:  0.2531645569620253    Path:  IECDFMKGALJHNB   Minimum cost path:  3950.0
Final Results for epoch 6  :-   Generations:-  983      Overall maximum fitness:  0.2531645569620253    Path:  GALJHNBIECDFMK   Minimum cost path:  3950.0
Final Results for epoch 7  :-   Generations:-  1162     Overall maximum fitness:  0.2531645569620253    Path:  JHNBIECDFMKGAL   Minimum cost path:  3950.0
Final Results for epoch 8  :-   Generations:-  306      Overall maximum fitness:  0.267379679144385     Path:  JHNBIECFDKMLGA   Minimum cost path:  3740.0
Final Results for epoch 9  :-   Generations:-  2066     Overall maximum fitness:  0.18761726078799248   Path:  GALMFNJHBIECDK   Minimum cost path:  5330.0
Final Results for epoch 10  :-  Generations:-  175      Overall maximum fitness:  0.267379679144385     Path:  HBIECFDKMLGAJN   Minimum cost path:  3740.0
Total epochs:-  10
Number of epochs without convergence:-  2
Average Number of Generations for Convergence:-  798.25


---------------------------------------- IMPROVED VERSION ------------------------------------------------


Final Results for epoch 1  :-   Generations:-  53       Overall maximum fitness:  0.267379679144385     Path:  BIECFDKMLGAJHN   Minimum cost path:  3740.0
Final Results for epoch 2  :-   Generations:-  28       Overall maximum fitness:  0.267379679144385     Path:  BIECFDKMLGAJHN   Minimum cost path:  3740.0
Final Results for epoch 3  :-   Generations:-  30       Overall maximum fitness:  0.267379679144385     Path:  HBIECFDKMLGAJN   Minimum cost path:  3740.0
Final Results for epoch 4  :-   Generations:-  314      Overall maximum fitness:  0.267379679144385     Path:  EIBNHJAGLMKDFC   Minimum cost path:  3740.0
Final Results for epoch 5  :-   Generations:-  43       Overall maximum fitness:  0.267379679144385     Path:  EIBHNJAGLMKDFC   Minimum cost path:  3740.0
Final Results for epoch 6  :-   Generations:-  585      Overall maximum fitness:  0.267379679144385     Path:  BIECFDKMLGAJHN   Minimum cost path:  3740.0
Final Results for epoch 7  :-   Generations:-  1304     Overall maximum fitness:  0.267379679144385     Path:  BIECFDKMLGAJNH   Minimum cost path:  3740.0
Final Results for epoch 8  :-   Generations:-  32       Overall maximum fitness:  0.267379679144385     Path:  IECFDKMLGAJNHB   Minimum cost path:  3740.0
Final Results for epoch 9  :-   Generations:-  44       Overall maximum fitness:  0.267379679144385     Path:  NHBIECFDKMLGAJ   Minimum cost path:  3740.0
Final Results for epoch 10  :-  Generations:-  25       Overall maximum fitness:  0.267379679144385     Path:  HNBIECFDKMLGAJ   Minimum cost path:  3740.0
Total epochs:-  10
Number of epochs without convergence:-  0
Average Number of Generations for Convergence:-  245.8
```

The textbook version in multiple runs converge to a minimum path cost of 3950 kms, 5330 kms and does not even converge to a possible path for 2 runs. On the other hand, the improved version always converged to the minimum path cost value - 3740 kms. Also, the average generations taken for convergence is 245 for the improved version in comparison to 798 for the textbook version.