Segment Based TSP Solver

20180063 Dohyun Kim

November 12, 2021

1 Introduction

There are lots of ways to solve TSP, such as nearest neighbor algorithm or genetic algorithm. In this paper, we proposed "Segment Based TSP Solver", which is the combination of nearest neighbor algorithm and genetic algorithm. Our algorithm is faster to find the optimal point then general GA and find better optimal point than NN.

2 Nearest Neighbor Algorithm

Nearest neighbor algorithm is fast and can find a reasonable optimal point. The key idea of this algorithm is that for a given node, traverse to the nearest node. The pseudo code is as Figure 1. Even though we got a reasonable optimal point by NN, we might want to find better result. However, NN easily stuck in local minima and it is hard to escape if we use only NN.

3 Genetic Algorithm

Genetic algorithm is also popular way to solve TSP. The general GA for TSP usually treats a node as a gene. However, we thought that treating a node as a gene is too complex and has high randomness. So, we proposed Segment Based GA (SEG-GA) to make the problem simpler.

3.1 Terminologies

There are two terms for SEG-GA. **Segment** is a segment of edges of the graph. **Pivot** is the start node of the segment. **Pivot list** is the list of pivots, which is a subset of vertices of the graph.

As you can see in the Figure 2, there are three segments (red, blue, and green). Suppose pivot list is [1, 0, 3] (note that, the order is important), which means 1, 0, and 3 are the pivots.

3.2 Gene and Individual

SEG-GA treats a pivot as a gene and a pivot list as an individual. This is because if the pivot list is fixed, then there can be only one corresponding route. To explain in more details, let us first describe how the segment is generated.

Procedure Nearest Neighbor Heuristic

10 Select an arbitrary node j
20 Set $l = j$ and $W = \{1, 2, ..., n\} \setminus \{j\}$
30 While $W \neq \emptyset$ do
40 Let $j \in W$ such that $c_{lj} = \min\{c_{li} \mid j \in W\}$
50 Connect l to j and set $W = W \setminus \{j\}$ and $l = j$.
60 Connect l to the node selected in 10 to form a Hamiltonian cycle.

Figure 1: Pseudo code for Nearest Neighbor algorithm

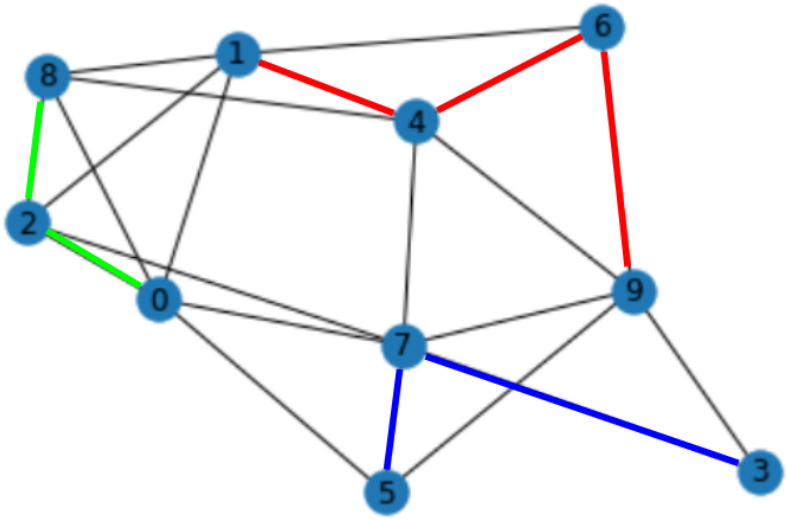

Figure 2: Example graph for TSP. Pivot list is [1, 0, 3]

If a pivot is given, we can generate a segment from that pivot by nearest neighbor algorithm. We traverse using NN until we reach to the another pivot. So, that will be the segment. For example, as Figure 2, suppose we generate a segment starting from 1. Since the nearest node from the 1 is 4, we traverse to 4. In the same manner, we will traverse to 6 and 9. When we reach to the 9, the next nearest node is 3. However, 3 is the pivot. So, the traveling will be over, and the segment, from 1 to 9, is generated. (It is the red segment) After generating the segment, the next node to travel is the next pivot, i.e., 0 in this example. Similarly, the green segment will be created. Since the last pivot is 3, we will travel to 3 and the last blue segment will be generated. So, the route will be 1-4-6-9-0-2-6-3-7-5-1. Note that, once the pivot list is fixed, the corresponding route is deterministic.

To summarize, the pivot list is the individual. It is much simpler compared to general GA. Also, by not fixing the length of the pivot list, we can guarantee the diversity of individuals. Note that, if the length of the pivot list is less than 3, the algorithm will be same as general NN. So, we should set the length to be larger than 2. In our experiment, the length ranges from 3 to the number of vertices.

3.3 Fitness Function

We did not use fitness function but distance of the route for evaluation. Note that, smaller distance means better.

3.4 Initialization

A pivot list, i.e., an individual, is created by sampling nodes from the nodes of the graph with random size. We repeat this process to make an initial population.

3.5 Selection

Selection is based on the distance value. We pick from the population in order of distance value (note, smaller means better). For example, if we decided to select 10 individuals, top 10 individuals will be selected.

3.6 Crossover

First, we randomly choose two individuals from the population and they will be parents. Then, we choose a random cutting point and perform single point crossover. Since a pivot list cannot has duplicate pivots, we should discard duplicated pivot. The details are in Figure 3.

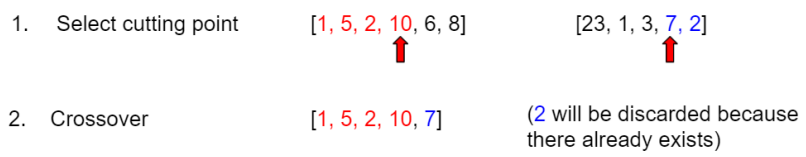

Figure 3: Example of crossover

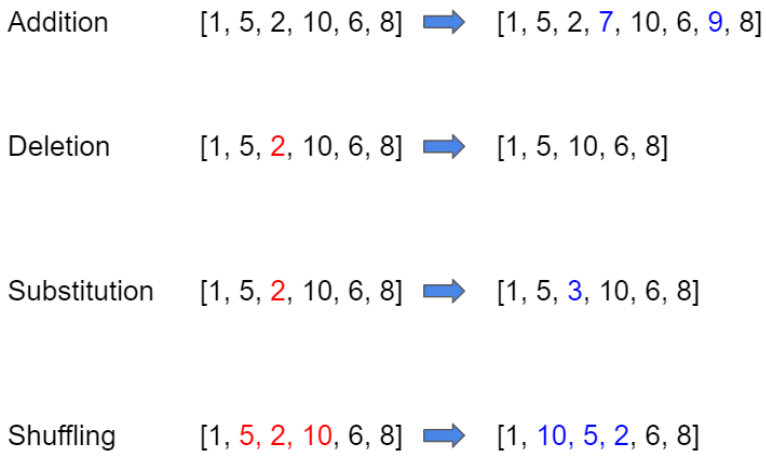

Figure 4: Example of four kinds of mutations

3.7 Mutation

There are four kinds of mutations: addition, deletion, substitution, and shuffling. An individual can be mutated only once. So it should choose only one mutation among four kinds of mutations. It will be chosen based on the certain probabilities.

Figure 4 describes four kinds of mutations. To easy to describe, let us define **remain** as a set of nodes which are in the graph but not in the pivot list.

3.7.1 Addition

For each node in remain, with the chance of certain probability, it will be inserted at the random position of the pivot list. Note that, inserting multiple nodes are allowed.

3.7.2 Deletion

For each node in pivot list, with the chance of certain probability, it will be deleted from the pivot list. Note that, deleting multiple nodes are allowed.

3.7.3 Substitution

For each node in pivot list, with the chance of certain probability, it will be deleted from the pivot list and new node from the remain will be inserted at that position. Note that, substituting multiple nodes are allowed.

3.7.4 Shuffling

Choose a sub list from the pivot list. Then, it will be shuffled.

Hyperparameter	Value
NPOP	50
NBEST	15
NCHILD	10
NMUT	25
PMUT1	0.1
PMUT2	0.1
PMUT3	0.1
PCMUT1	0.25
PCMUT2	0.25
PCMUT3	0.25
PCMUT4	0.25
MAXFIT	1000

Table 1: Hyperparameters

TSP file	berlin52.tsp	eil76.tsp	lin105.tsp	ch150.tsp	d198.tsp
SEG-GA	7985	611	16049	7092	17630
NN	9140	661	19110	7749	18522
Best	7542	538	14379	6528	15780
SEG-GA / Best	1.05	1.13	1.12	1.08	1.12

Figure 5: Result table

4 Experimetns

4.1 Hyperparameters

The hyperparameters for our experiments are in Table 1.

NPOP means the population size. **NBEST** means the selection size, **NCHILD** indicates the number of children(offspring) generated by crossover, and **NMUT** is the number of individuals that will be mutated. However, we did not hard code **NBEST**, **NCHILD**, and **NMUT** but is calculated by **NPOP** * 0.3, **NPOP** * 0.2, and **NPOP** * 0.5, respectively.

PMUT1 is the chance of probability for the addition mutation. **PMUT2**, **PMUT3**, and **PMUT4** are for deletion, substitution, and shuffling, respectively. **PCMUT1** indicates the probability of choosing addition mutation. **PCMUT2**, **PCMUT3**, and **PCMUT4** are for deletion subtitution, and shuffling, respectively.

Lastly, **MAXFIT** is the total number of fitness evaluation.

4.2 Results

We tested several tsp file, and selected results are in the Figure 5. In the Figure 5, SEG-GA is distance by SEG-GA, NN is distance by NN, and Best is the best known solution. And a graph for berlin52.tsp is given as Figure 6.

As you can see in the table, SEG-GA distance is about 1.1 times larger than the best known solution. So, we can say that our algorithm works well in general.

5 Conclusion

We proposed SEG-GA to solve TSP problem. It shows better result than NN and fast convergence than GA. However, it is hard to get the best know solution by this algorithm. So, another attempts,

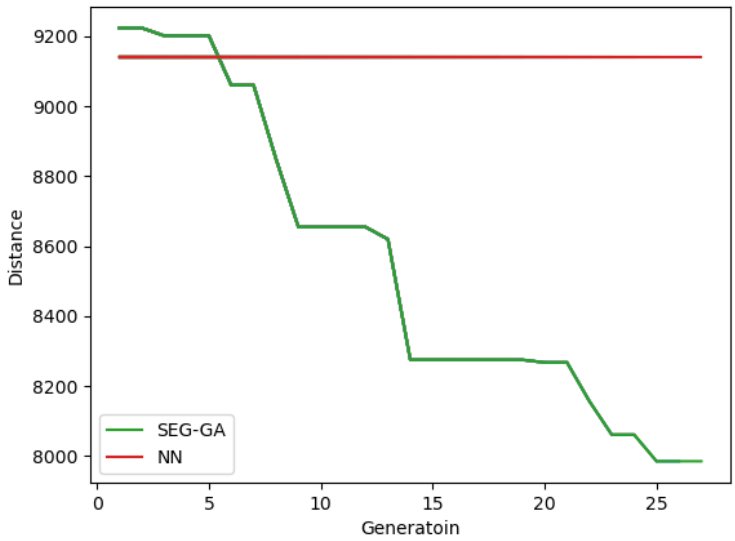

Figure 6: berlin52.tsp

such as Greedy algorithm, is needed to find the better solution. Also, it is still too slow so that it took huge amount of time when running large tsp file, such as rl11849.tsp. The bottleneck of the code is calculating distance of the route, so the optimization will be needed.