The Traveling Salesman problem

Amanur Rahman Saiyed
Indiana State University
Terre Haute, IN 47809 , USA
asaiyed@sycamores.indstate.edu

April 11, 2012

Abstract

The Traveling Salesman Problem, deals with creating the ideal path that a salesman would take while traveling between cities. The solution to any given TSP would be the Shortest way to visit a finite number of cities, visiting each city only once, and then returning to the starting point. We also must assume that if there are two cities, city A and city B for example, it costs the same amount of money to travel from A to B as it does from B to A. For the most part, the solving of a TSP is no longer executed for the intention its name indicates. Instead, it is a foundation for studying general methods that are applied to a wide range of optimization problems.

Contents

1 Statement Of The Problem

The traveling salesman problem involves a salesman who must make a tour of a number of cities using the shortest path available and visit each city exactly once and only once and return to the original starting point. For each number of cities n ,the number of paths which must be explored is $n!$, causing this problem to grow exponentially rather than as a polynomial. There are bunch of algorithms offering comparably fast running time and still yielding near optimal solutions.

2 History of The TSP

The Traveling Salesman Problem (TSP) is a problem whose solution has eluded many mathematicians for years. Currently there is no solution to the TSP that has satisfied mathematicians. Historically, mathematics related to the TSP was developed in the 1800's by Sir William Rowan Hamilton and Thomas Penyngton Kirkman, Irish and British mathematicians, respectively. Hamilton was the creator of the Icosian Game in 1857. It was a pegboard with twenty holes that required each vertex to be visited only once, no edge to be visited more than once, and the ending point being the same as the starting point.This kind of path was eventually referred to as a Hamiltonian circuit.However, the general form of the TSP was first studied by Karl Menger in Vienna and Harvard in the late 1920's or early 1930's.

TSP?s were first studied in the 1930?s by mathematician and economist Karl Menger in Vienna and Harvard. It was later investigated by Hassler Whitney and Merrill Flood at Princeton.

In 1994, Applegate, Bixby , Chvatal, and Cook solved TSP containing 7,397 cities. Later in 1998, they solved it using 13,509 cities in United States. In 2001, Applegate, Bixby, Chvtal, and Cook found the optimal tour of 15,112 cities in Germany. Later in 2004, TSP of visiting all 24,978 cities in Sweden was solved; a tour of length of approximately 72,500 kilometers was found and it was proven that no shorter tour exists. This is currently the largest solved TSP.

Table 1: Summarizes the Milestones of TSP

Year	Research Team	Size of instance
1954	G. Dantzig, R. Fulkerson, and S. Johnson	49 cities
1971	M. Held and R.M. Karp	64 cities
1975	P.M. Camerini, L. Fratta, and F. Maffioli	67 cities
1977	M. Grotschel	120 cities
1980	H. Crowder and M.W. Padberg	318 cities
1987	M. Padberg and G. Rinaldi	532 cities
1987	M. Padberg and G. Rinaldi	2,392 cities
1994	David L.Applegate, Robert E.Bixby, Vasek Chvatal, and William J. Cook	7,397 cities
1998	David L.Applegate, Robert E.Bixby, Vasek Chvatal, and William J. Cook	13,509 cities
2001	David L.Applegate, Robert E.Bixby, Vasek Chvatal, and William J. Cook	15,112 cities
2004	David L.Applegate, Robert E.Bixby, Vasek Chvatal, and William J. Cook	24,978 cities

3 Solution methods of TSP

Introduction

Suppose a salesperson needs to travel from a city to all the other cities exactly once to sell his products and return back to the city he started from. He wants to do this while covering the minimum total distance. How can he do that? This is where solving the TSP comes in. Some solution methods of TSP include:

3.1 Exact Solutions

- Brute-force method.
- Branch and Bound.

3.1.1 Brute force method

When one thinks of solving TSP, the first method that might come to mind is a brute-force method. The brute-force method is to simply generate all possible tours and compute their distances. The shortest tour is thus the optimal tour. To solve TSP using Brute-force method we can use the following steps:

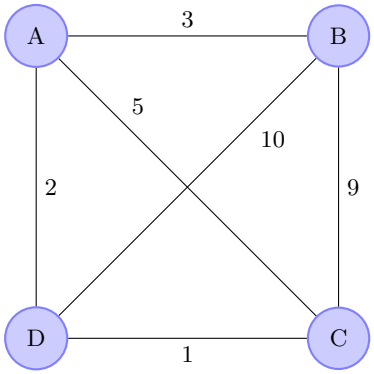
Step 1. calculate the total number of tours.

Step 2. draw and list all the possible tours.

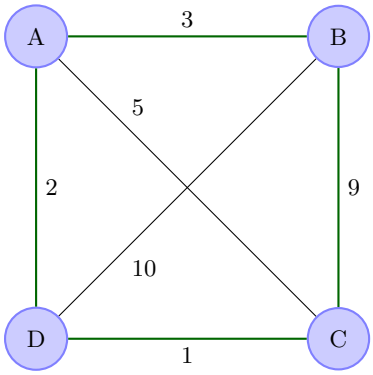
Step 3. calculate the distance of each tour.

Step 4. choose the shortest tour, this is the optimal solution.

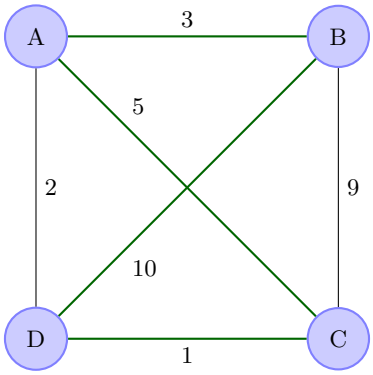
3.1.2 Example for Brute Force Technique



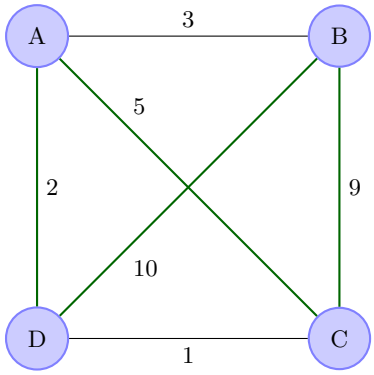
Here, there are 4 nodes. There is a possibility of the following 3 paths



$A \rightarrow B \rightarrow C \rightarrow D \rightarrow A = 15$



$A \rightarrow B \rightarrow D \rightarrow C \rightarrow A = 19$



$A \to C \to B \to D \to A = 26$

The best distance path is $A \to B \to C \to D \to E \to A$, of value 15.

3.1.3 Branch and Bound

The Branch and Bound strategy divides a problem to be solved into a number of sub-problems. It is a system for solving a sequence of sub-problems each of which may have multiple possible solutions and where the solution chosen for one sub-problem may affect the possible solutions of later sub-problems.

Step 1: Choose a start node.

Step 2: Set bound to a very large value, let's say infinity.

Step 3: Choose the cheapest arc between the current and unvisited node and add the distance to the current distance and repeat while the current distance is less than the bound.

Step 4: If current distance is less than bound, then we are done

Step 5: Add up the distance and bound will be equal to the current distance.

Step 6: Repeat step 5 until all the arcs have been covered.

4 Approximate solutions

- Nearest Neighbor.
- Greedy approach.

4.1 Nearest Neighbor

This is perhaps the simplest and most straight forward TSP heuristic. The key to this algorithm is to always visit the nearest city, then return

to the starting city when all the other cities are visited.

Nearest Neighbor, $O(n^2)$

Step 1. Select a random city.

Step 2. Find the nearest unvisited city and go there.

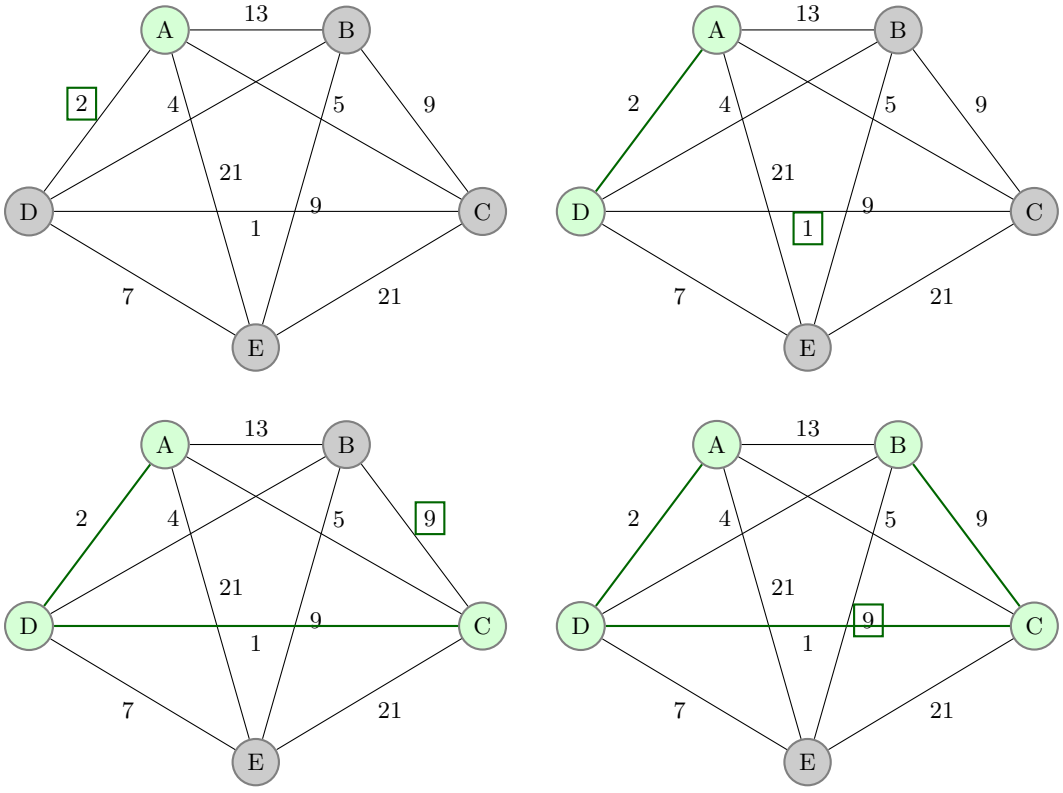
Step 3. Are there any unvisited cities left? If yes, repeat step 2.

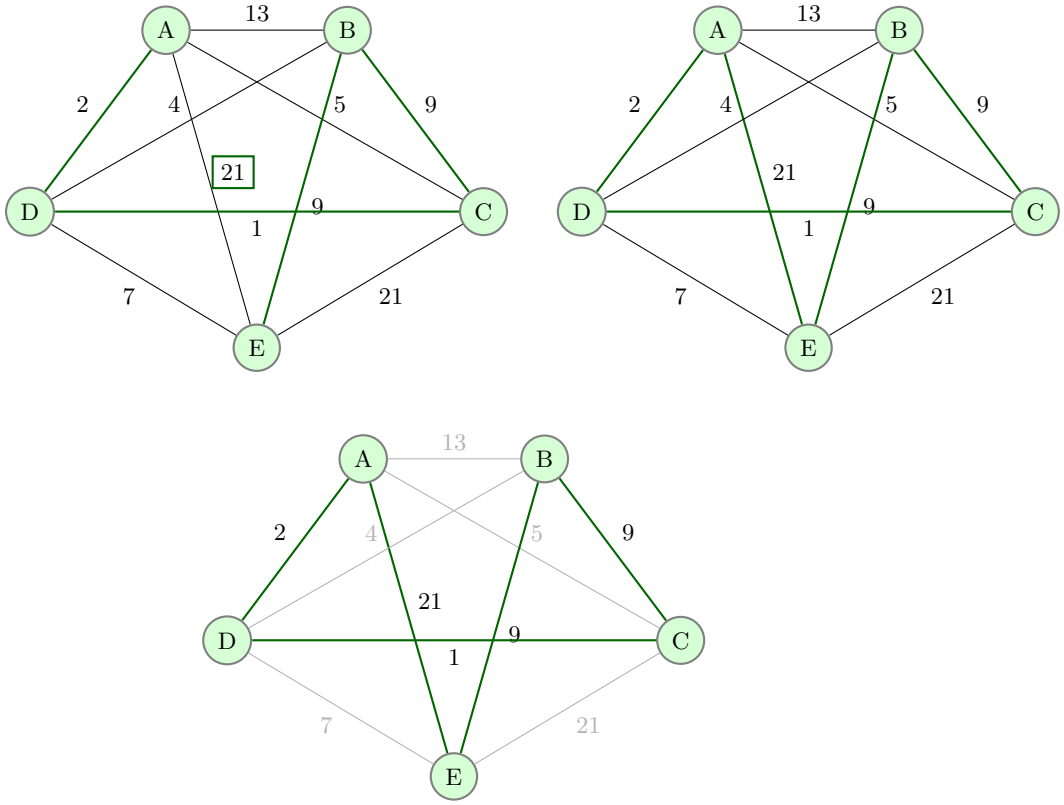
Step 4. Return to the first city.

4.1.1 Example for Nearest Neighbor Method

This is the step-wise approximate solution by nearest neighbor method.

This case has 5 nodes. We start with the node A and perform the nearest neighbor algorithm.





The total distance of the path $A \to D \to C \to B \to E \to A$ obtained using the nearest neighbor method is $2 + 1 + 9 + 9 + 21 = 42$.

4.2 Greedy

Greedy algorithm is the simplest improvement algorithm. It starts with the departure Node 1. Then the algorithm calculates all the distances to other $n - 1$ nodes. Go to the next closest node. Take the current node as the departing node, and select the next nearest node from the remaining $n - 2$ nodes. The process continues until all the nodes are visited once and only once then back to Node 1. When the algorithm is terminated, the sequence is returned as the best tour.

Greedy, $O(n^2 log_2(n))$

5 Optimization Techniques

Once a tour has been generated by some tour construction heuristic, we might wish to improve that solution. There are several ways to do this, but the most common ones are the 2-opt and 3-opt local searches. Their performances are somewhat linked to the construction heuristic used.

5.1 2-opt and 3-opt

The 2-Opt algorithm was first proposed by Croes [1958], although the basic move had already been suggested by Flood [1956]. The 2-opt algorithm basically removes two edges from the tour, and reconnects the two paths created. This is often referred to as a 2-opt move.

The 3-opt algorithm works in a similar fashion, but instead of removing two edges we remove three. A 3-opt move can actually be seen as two or three 2-opt moves.

We finish our search when no more 3-opt moves can improve the tour. If a tour is 3-optimal it is also 2-optimal.

When talking about the complexity of these k-opt algorithms, one tends to omit the fact that a move can take up to $O(n)$ to perform.A naive implementation of 2-opt runs in $O(n^2)$,this involves selecting an edge (c_1, c_2) and searching for another edge (c_3, c_4), completing a move only if $\text{dist}(c_1, c_2) + \text{dist}(c_3, c_4) > \text{dist}(c_2, c_3) + \text{dist}(c_1, c_4)$.

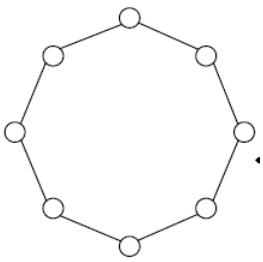


Figure 1: 2-Opt move

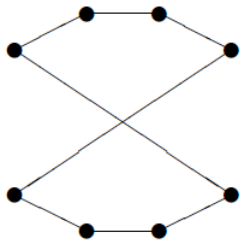


Figure 2: 2-Opt move

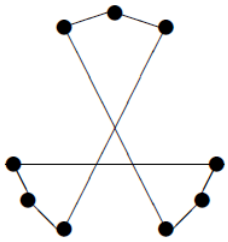


Figure 3: 3-Opt move

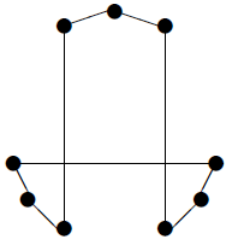
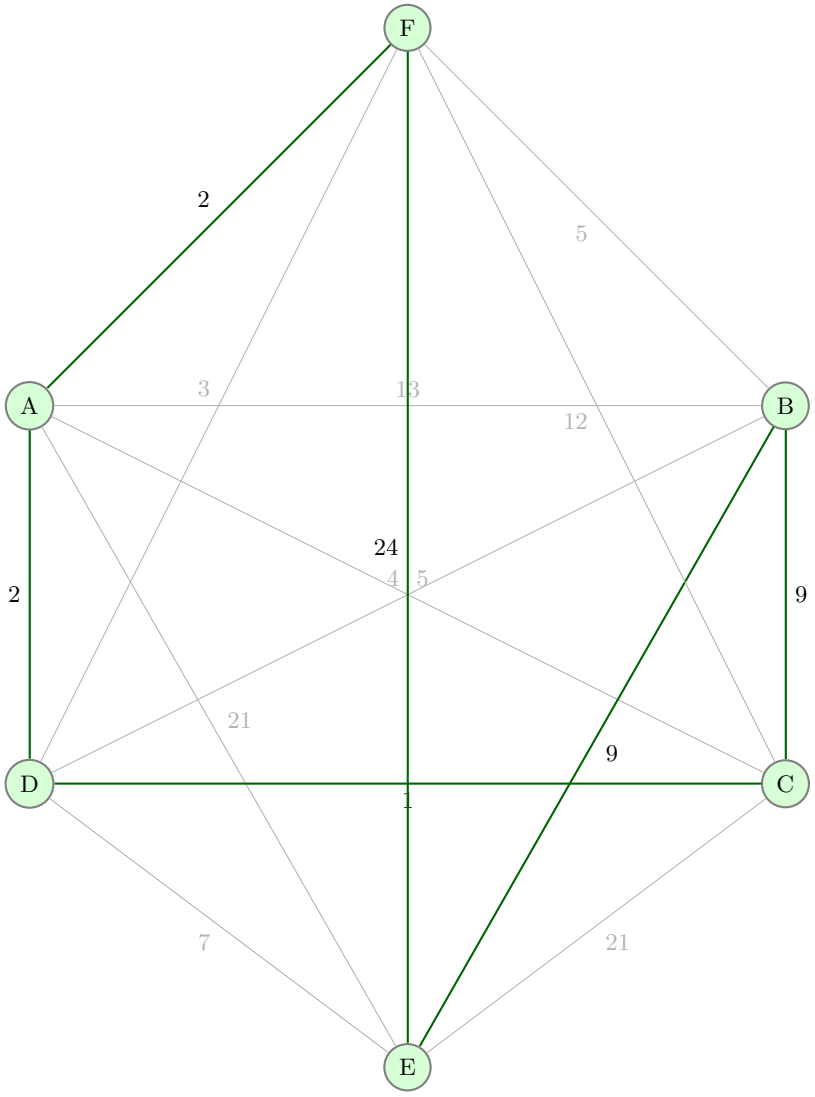


Figure 4: 3-Opt move

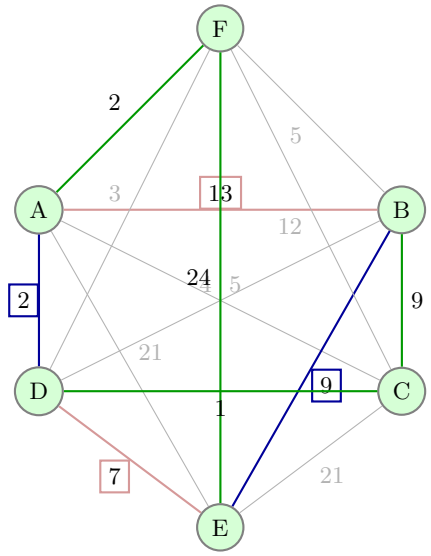
5.1.1 Example for 2-opt Technique

Assume that this is the solution obtained from nearest neighbor method or some kind of approximation method.



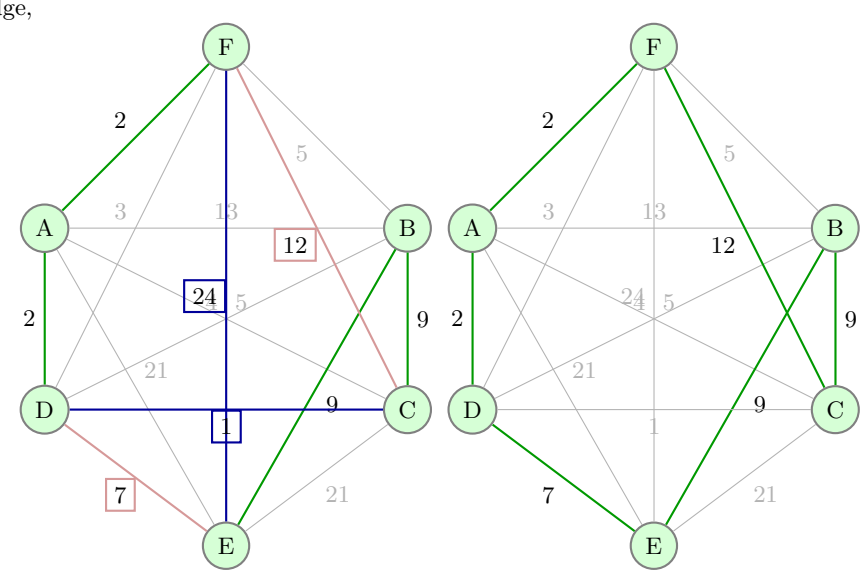
Here, we can see that the total distance of the tour is 47. We have to apply 2-opt algorithm over this approximate solution to improve the solution.

For this, we first select an edge. Let the edge be AD. We should now select an edge such that it is not adjacent to the edge being considered, AD. Here, we only one such edge, BE. We can replace AD and BE with AB and DE.



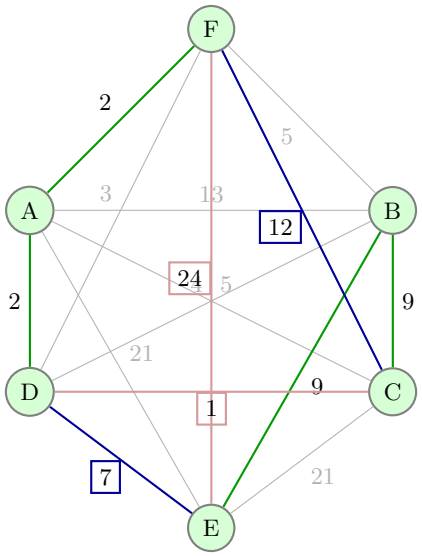
Here we can see that that the sum of replaceable edges is higher than the sum of the original edges. So, we do not replace the edges

We now consider the next edge, DC. Its has only one non adjacent edge,



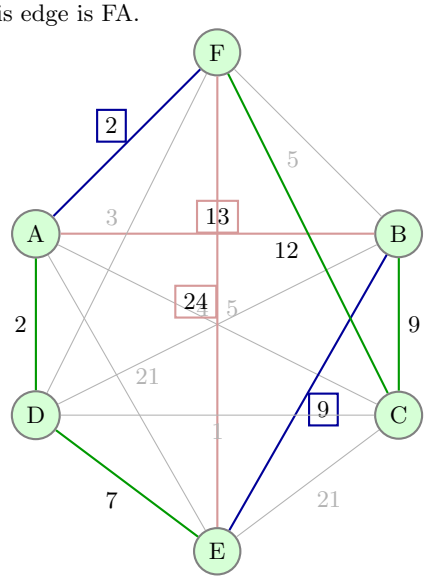
Here, as we have the sum of current edges greater than the sum of replaceable edges, we replace them with the replaceable edges.

We now consider the edge DE. This edge has the non adjacent edge CF.



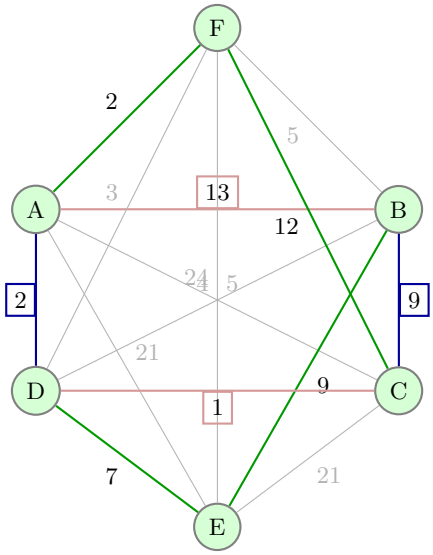
As the replaceable edges are larger than the original edges, we don't replace the original edges.

We go to the next step where the edge EB is selected. The edge that is non adjacent to this edge is FA.



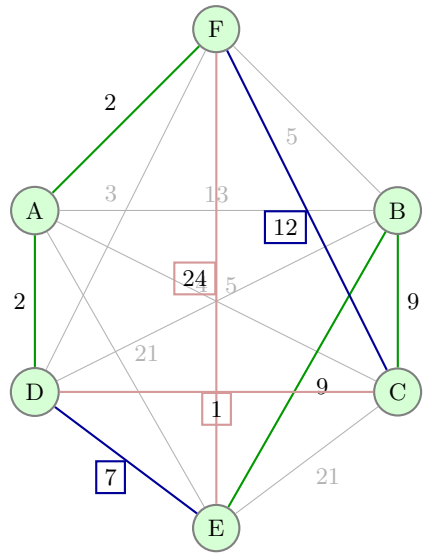
No edge replacement is necessary in this case.

We now move on to the next edge, BC. The non adjacent edge to this edge is AD.



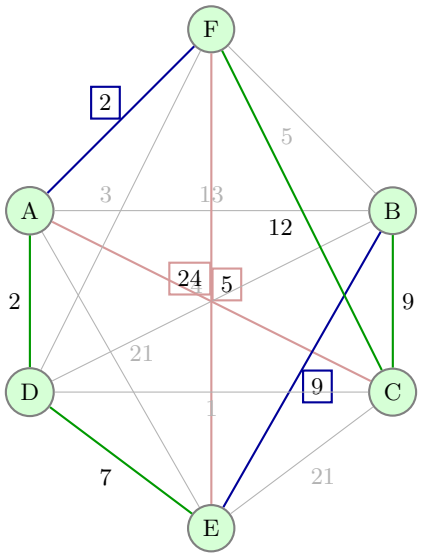
This scenario doesn't warrant any change in the path.

Now we consider the next edge CF. The non adjacent edge of this edge is DE.



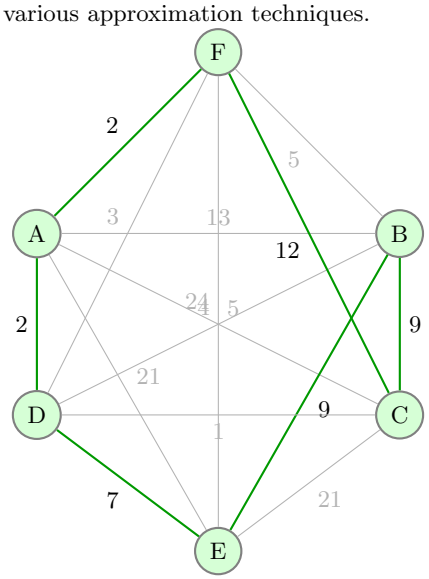
Here, there is no need to change any path as the replaceable edges are of smaller size when compared to the existing edges.

The next edge to be considered is FA. The non adjacent edge for FA is EB.



This is also the case where there is no need for change in the path.

Finally, we arrive at the optimized solution from the approximate solution obtained from various approximation techniques.



We finally have a solution of path size 41. This is an improvement from the earlier path.

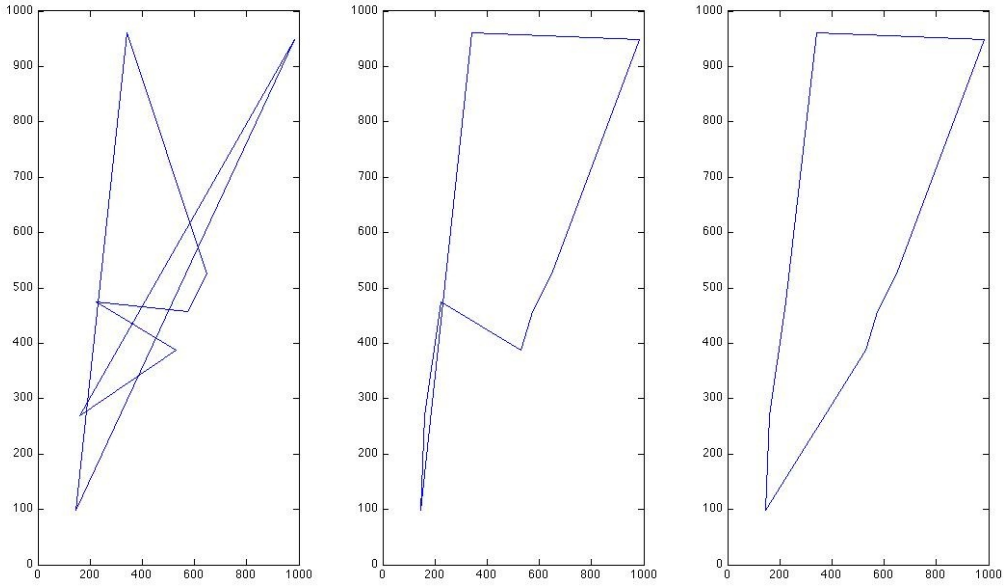


Figure 5: Comparison of algorithms

References

[1] David L.Applegate, Robert E.Bixby, Vasek Chvatal, William J.Cook *"The Traveling Salesman Problem"*, year = 2001.

[2] D.S. Johnson and L.A. McGeoch, *"Experimental Analysis of Heuristics for the STSP "*, The Traveling Salesman Problem and its Variations, year = 2002.

[3] Cook, William., "History of the TSP " *The Traveling Salesman Problem*, year = 2009.

[4] *http://en.wikipedia.org/wiki/Traveling salesman problem*

[5] *http://www.tsp.gatech.edu/*