Proceedings of Science and Mathematics
Faculty of Science, Universiti Teknologi Malaysia
Vol. 3, 2021, page 9-17

The Simulation of Floyd-Warshall Algorithm in Finding the Shortest Path for Travelling Salesman Problem using Microsoft Foundation Classes (MFC)

[1]Noramira Azaly [a], Wan Rohaizad Wan Ibrahim [b]*

[a,b]Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.
*Corresponding author: wrohaizad@utm.my

Abstract The purpose of this project is to investigate the application of the Floyd-Warshall Algorithm (FWA) in modeling the Travelling Salesman Problem and develop a simulation based on FWA. The travelling salesman problem, known as TSP, is a problem in which the salesman must go to the required cities while only taking each path once and returning to the starting point. This project is conducting the salesman from storehouse to 15 different location of customer base in Johor. In this project, the salesman must visit the customer in order to deliver the parcel, but his goal is to take the possible shortest path in order to reduce the cost and time required to travel around the customer's location while using a single path and completing the track at the storehouse. This project makes use of Microsoft Foundation Classes, and the implementation of the Floyd-Warshall algorithm can assist the delivery vehicle in determining the shortest path to deliver the parcel to the specified customer. The simulation results showed that in these scenarios, FWA can generate an optimum solution in less than 10 seconds and with only one execution.
Keywords Floyd-Warshall algorithm; shortest route; Travelling Salesman Problem; heuristic

1 Introduction

The Traveling Salesman Problem (TSP) is an algorithmic problem that seeks the shortest path between a collection of points and places that must be visited. The cities a salesman could visit are represented by the points in the problem statement. The salesman's objective is to minimize both travel costs and distance traveled.

TSP, which focuses on optimization is frequently used in computer science to identify the most efficient path for data to travel between different nodes. Identifying network or hardware optimization strategies is one example of an application. In the 1800s, Irish mathematician W.R. Hamilton and British mathematician Thomas Kirkman created a game that could be solved by discovering a Hamilton cycle [1], which is a non-overlapping path connecting all nodes.

The traveling salesman problem is an optimization problem and has a wide search in the educational world and is said to be NP-hard, which means cannot be clarified by polynomial time [2]. It is one of the most basic knowledge in the field of computer science nowadays. The traveling salesman problem has been used in so many fields today's time. For instance, the

application of manufacturing of microchips, vehicle routing, drilling in printed circuit boards, packet routing, etc.

2 Literature Review

2.1 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is also known as the Floyd algorithm, the Roy-Warshall algorithm, the Roy-Floyd algorithm, or the WFI algorithms. Robert Floyd first published this method in 1962. This approach however is substantially the same as the earlier algorithm reported by Bernard Roy in 1959 and Stephen Warshall in 1962 to determine the transitive closure of the graph. Peter Ingerman, also in 1962, described the modern version of the Warshall algorithm as three nested for-loops [3].

This is an algorithm for dynamic programming. Floyd [4] created it based on a paper presented by Warshall [5]. This method is built around the idea of intermediate vertices. Let ${d_{ij}}^{0}$ represent the weight matrix and ${d_{ij}}^{k}$ represent the shortest path from i to j with its intermediate vertices in the collection $(1,2,\dots,k\}$. Then for ${d_{ij}}^{k} = \min\left({d_{ij}}^{k-1}, {d_{ik}}^{k-1} + {d_{kj}}^{k-1}\right), k > 1$. As a result, ${d_{ij}}^{n}$ will produce the shortest routes matrix for the input graph. The algorithm is described below.

```
for i = 1 to n do
    for j = 1 to n do
        if there exists an edge from i to j then
                d[i,j] = w[i,j]
                r[i,j] = i
            else
                d[i,j] = infinity
                r[i,j] = −1
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if d[i,k] + d[k,j] < d[i,j] then
                d[i,j] = d[i,k] + d[k,j]
                r[i,j] = k
```

Algorithm 1

2.2 MFC Visualization

Microsoft Foundation Classes (MFC) is a collection of simple functions developed by Microsoft Corporation in the late 1980s. The library functions are intended to assist in C++ application development on the Windows platform for purposes such as displaying text, graphics, and images, as well as handling objects such as buttons, edit boxes, static boxes, and menus. MFC includes almost 200 classes for navigating Windows resources [6].

Figure 1 illustrates the overall map and connection between MFC and Windows resources. Programming on Windows involves significant use of the machine's hardware resources. Application Program Interface (API) is a layer above the kernel that allows these resources to be

accessed. Prior to MFC, this process was hard since calls to these resources required several low-level C routines. MFC simplifies this procedure by utilizing a high-level language approach to obtain control over the routines. MFC [6] makes full use of GDI, or Graphic Device Interface, which is a layer in the Windows architecture that protects the program from direct hardware contact. This interface offers a large range of high - level functions that may be accessed through its objects for tasks such as drawing and managing graphics in Windows.

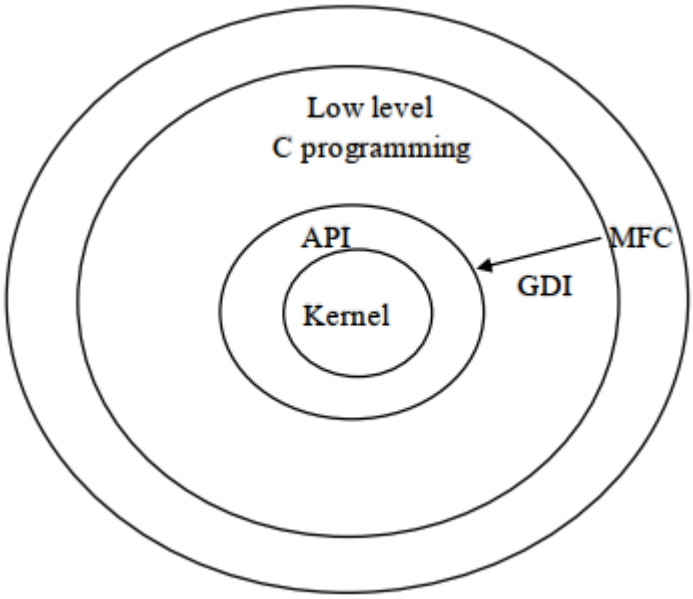

Figure 1: MFC map of access to Windows resources

3 Methodology

The simulation of the Floyd-Warshall algorithm in solving the travelling salesman problem is presented in detail in this chapter. This chapter introduced and utilized knowledge of Microsoft Foundation Classes (MFC) to demonstrate the best solution for the salesman. The goal of this research methodology is to show that combining MFC with the Floyd-Warshall algorithm is the quickest solution to solve the travelling salesman problem.

3.1 Flow Chart for Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is a dynamic algorithm that solves the problem of determining the shortest path [7]. The value of the shortest path is displayed at each side of the path by having graph input that is directed and weighted (V, A) in the form of a collection of location (V) and sides (A). This algorithm computes the lowest value of all routes connecting two locations and executes it all at once [7].

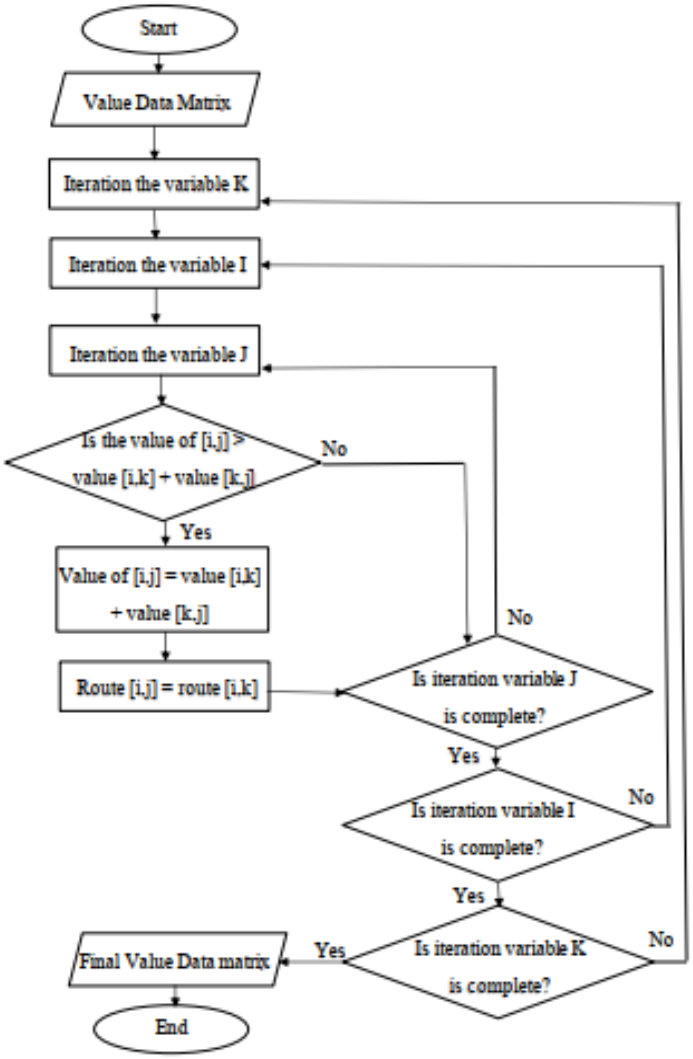

Figure 2: Flow chart of Floyd-Warshall Algorithm

Figure 2 shows how the flow chart operates in this selection path. The Floyd-Warshall flow chart is designed as follows:

1. Processed data is evaluated before being sent into the Floyd-Warshall Algorithm.
2. Assign a zero value to the same location in the matrix until the point is empty.
3. Create a route matrix based on the value of each point.
4. Apply the Floyd-Warshall Algorithm to the matrix in iterations K, I and J.
5. When the iteration is complete, the final result of the route matrix is the best solution for the salesman.

3.2 Computational of Floyd-Warshall Algorithm

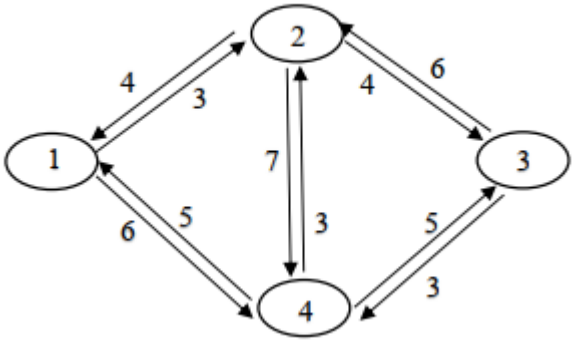

Figure 3: The cycle of 4 customer location

In this part, we elaborate on the Floyd-Warshall algorithm by solving the chosen four customer locations in Johor. Given a Johor map, N in Figure 3.4, we want to find the shortest routes between each pair of locations on this map. This explanation clarifies the Floyd–Warshall algorithm's efficiency.

The matrices for D_4 and R_4 are shown below:

$$D_4 = \begin{bmatrix} 0 & 3 & 5 & 7 \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ 9 & 6 & 3 & 0 \end{bmatrix} \text{ and } R_4 = \begin{bmatrix} - & 2 & 3 & 2 \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ 3 & 2 & 3 & - \end{bmatrix}$$

The following is a complete computation for determining matrices D_1 and R_1. Using step 3 of the Floyd–Warshall algorithm to calculate D_1 yields the following results. It is worth noting that the first row and first column (j = 1) would be the same. Furthermore, the diagonal stays unchanged.

$$d_{23} = \min(d_{23}, d_{21} + d_{13}) = \min(7, 4 + 5) = 7$$

$$d_{24} = \min(d_{24}, d_{21} + d_{14}) = \min(4, 4 + \infty) = 4$$

$$d_{32} = \min(d_{32}, d_{31} + d_{12}) = \min(3, 6 + 3) = 3$$

$$d_{34} = \min(d_{34}, d_{31} + d_{14}) = \min(5, 6 + \infty) = 5$$

$$d_{42} = \min(d_{42}, d_{41} + d_{14}) = \min(6, \infty + \infty) = 6$$

$$d_{43} = \min(d_{43}, d_{41} + d_{13}) = \min(3, \infty + 5) = 3$$

Therefore D_1 is $\begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix}$. R_1 is $\begin{bmatrix} - & 2 & 3 & - \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ - & 2 & 3 & - \end{bmatrix}$ and is calculated by deriving the following values:

$$r_{12} = k = 2$$

$$r_{13} = k = 3$$

$$r_{21} = k = 1$$

$$r_{23} = k = 3$$

$$r_{24} = k = 4$$

$$r_{31} = k = 1$$

$$r_{32} = k = 2$$

$$r_{34} = k = 4$$

$$r_{42} = k = 2$$

$$r_{43} = k = 3$$

This calculation shows how the Floyd-Warshall algorithm works. It is clear that this approach has significantly decreased the amount of calculation required to identify the shortest path and costs between any two random locations $(i, k) \in N$. From this map, which has just four locations, it is apparent that the Floyd-Warshall algorithm is more efficient and evolved in one computation.

4 Result and Analysis

The network of 15 locations scattered on the plane shows in Figure 4 below with marked costs on the edges. On this network, we do a minimum cost and shortest path simulation. In order to satisfy the criteria of a TSP, the salesman begins his work from the warehouse and proceeds to the first customer's location. To solve this problem, the following locations are chosen by Floyd-Warshall algorithm, and an initial solution is created. Without returning to the previously visited location, the salesman continues to travel using the same method, where this algorithm brought the vehicle to the nearest location from the initial move.

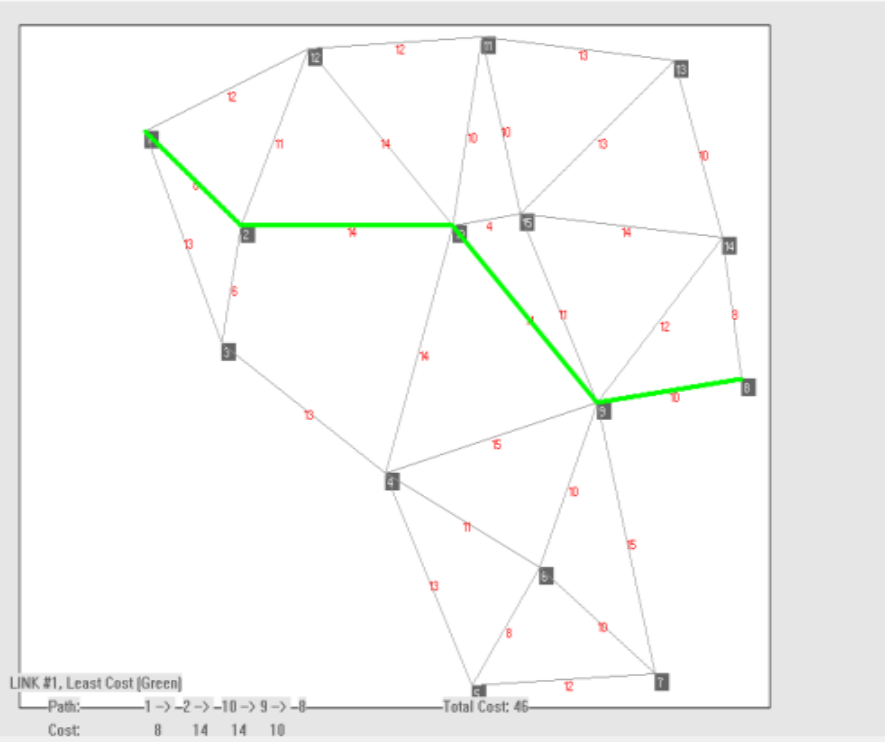

Figure 4: The vehicle traveled to the nearest unvisited location of customer (Green Color)

The diagram above illustrates the first four pathways taken by the salesman as it begins the journey. The vehicle completes the route and provides the following solution:

Path: 1 – 2 – 10 – 9 – 8
Total cost: 46

As seen in the result screen above, the overall minimal cost for the parcel delivery problem utilizing the Floyd-Warshall algorithm is 46. As seen in figure 5 below, the Floyd-Warshall algorithm is implemented in the MFC code to find the total cost as requested from the salesman.

```
//this function compute path using FW algorithm
void CcodeSP::ComputePath()
{
    int i,j,k,m,n,nRel,nSp,nWc;

    //code for shortest, using floyd warshall
    for (i=1;i<=N;i++)
        for (j=1;j<=N;j++)
            for (k=1;k<=N;k++)
                if (eSp[j][i].sd!=99 || eSp[i][k].sd!=99 || eSp[j][k].sd!=99)
                    if (eSp[j][i].sd+eSp[i][k].sd < eSp[j][k].sd)
                    {
                        nSp=1;
                        eSp[j][k].sd=eSp[j][i].sd+eSp[i][k].sd;
                        for (m=1;m<=N;m++)
                            alphaSp[j][k][m]=99;
                        for (m=1; m<=N; m++)
                            if (alphaSp[j][i][m]!=99)
                                alphaSp[j][k][nSp++]=alphaSp[j][i][m];
                        alphaSp[j][k][nSp++]=i;
                        for (m=1; m<=N; m++)
                            if (alphaSp[i][k][m]!=99)
                                alphaSp[j][k][nSp++]=alphaSp[i][k][m];
                    }
}
```

Figure 5: C++ code to calculate the shortest path using Floyd-Warshall algorithm

5 Conclusion and Recommendations

5.1 Conclusion

When utilizing the Floyd-Warshall algorithm to find the shortest path based on the travelling salesman problem, certain essential information is required to conduct movement computation from one location to another location. We can observe in this project that the simulation of finding the shortest path using MFC is improving our understanding of the idea of path planning.

In this project, we present a unique method for computing the shortest path in networks containing cycles. In a number of ways, the proposed approach improves on the Floyd–Warshall algorithm, which is one of the strongest available algorithms for dealing with travelling salesman problems. The Floyd–Warshall algorithm performs identically in calculating the D_0 and R_0 matrices. The Floyd-Warshall algorithm, on the other hand, generates the corresponding matrices significantly faster for stage $j \geq 1$ due to the reduced quantity of work. This has been illustrated graphically with simulation using MFC.

5.2 Recommendations

In this project, it illustrates that parcel delivery problem can be quickly solved using Floyd-Warshall Algorithm. This study indicates that implementing this method helps in the improvement of the parcel delivery problem by using coordinates as the starting point for measuring the problem. Further study on the relocation should be further examined in Floyd-Warshall algorithm. Aspiration analysis may make it possible to make temporary use of the prohibited motion to get the optimal overall.

The goal of this project is to identify the quickest way at the lowest possible cost. Another variable may be introduced into this problem to make it a more challenging and interesting topic to explore. For example, in this project, the selected topic is related to the shortest path and the

lowest cost, but other variables such as path reliability should be considered while selecting the chosen path.

6 References

[1] Barnett, J. H. (2019). Early Writings on Graph Theory: Hamiltonian Circuits and The Icosian Game. Resources for Teaching Discrete Mathematics, 217-224. https://doi.org/10.5948/upo9780883859742.028.

[2] Woeginger, G. J. (2003). Exact algorithms for NP-hard problems: A survey. In Combinatorial optimization-eureka, you shrink! (pp. 185-207). Springer, Berlin, Heidelberg.

[3] Weisstein, Eric, 2009, Floyd-Warshall Algorithm, Wolfram MathWorld.

[4] Robert W. Floyd, Algorithm 97: Shortest path, Communications of the ACM 5 (6) (1962) 345.

[5] S. Warshall.: A theorem on boolean matrices, Journal ofthe ACM 9 (1962) 11-12.

[6] Shepherd, G., & Wingo, S. (1996). MFC Internals: inside the Microsoft Foundation class architecture. Addison Wesley Longman Publishing Co., Inc.

[7] Zhang, L. Y., Jian, M., & Li, K. P. (2010, April). A parallel floyd-warshall algorithm based on tbb. In 2010 2nd IEEE International Conference on Information Management and Engineering (pp. 429-433). IEEE.