# SHORTEST FLIGHT PATH

Chinmayi C.R. - 181IT113

Department of Information Technology

National Institute of Technology Karnataka

Surathkal, India 575025

Email: chinmayicr27@gmail.com

Krishna Poojitha Vantakula - 181IT223

Department of Information Technology

National Institute of Technology Karnataka

Surathkal, India 575025

Email: krishnapoojitha2001@gmail.com

**In mathematics, and more specifically in graph theory, a graph is a structure consisting of a set of nodes in which some pairs of the nodes are related in some manner. A graph is typically depicted in diagrammatic form as a set of dots or circles for the vertices, joined by lines or curves for the edges joining the vertices. These nodes correspond to vertices and each of the related pairs of vertices is called an edge (also called lines).**

**In particular there are two properties of edges that determine the type of a graph. These two properties are edge weight and edge directionality. The edges may be directed or undirected. A directed graph is a graph containing a set of objects (also called as vertices or nodes) that are connected together, where all the edges in the graph are directed from one node to another. A directed graph is also termed as a digraph or a directed network. An undirected graph is a set of objects (also called as vertices or nodes) that are connected together, where all the edges are bidirectional. An undirected graph is also depicted as an undirected network. The graphs can also be classified as weighted or unweighted. A weight is a numerical value assigned to each individual edge. If edges in the graph contain weights, then the graph is said to be weighted. If the edges do not contain weights, the graph is said to be unweighted. In weighted graphs, relationship between nodes exists, containing a value which is basically the magnitude and this magnitude is important to the relationship we are considering. In an unweighted graph the existence of a relationship between the nodes depends on interest.**

**Graphs can be used to represent various types of connections, relations in all the basic systems present around us. Many practical problems can be represented by graphs and thereafter can be solved efficiently.**

**Shortest path algorithms are a family of algorithms which are designed to solve the shortest path problem between required objects very efficiently. In general, the shortest path algorithms typically operate on some input graph provided. Sometimes even cycles are present in the graph being provided. Each of these subtle differences are what makes one of the algorithms perform better than others for a certain type of graph. There are also different types of shortest path algorithms. Shortest path algorithms have many applications in our daily routine. The mapping of software like Google or Apple maps makes use of shortest path algorithms**

**to provide the users with the best service in locating landmarks. They are also important for a variety of other purposes like the road network, railways and logistics research. Shortest path algorithms are also very important for computer networks, like the Internet which is considered to be a vital part of our living.**

**There are two main types of shortest path algorithms, single-source and all-pairs. Both types have algorithms that perform best in their own way. All-pairs algorithms take longer time to run because of the added complexity of finding paths from all vertices to all other vertices. All pair shortest path algorithms return values that can be used to find the shortest path, even if those return values vary in type or from algorithm to algorithm.**

**Dijkstra's original algorithm finds the shortest path between two given nodes in a graph, but a more common variant fixes a single node as the source node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree between the nodes. It can also be used for finding the shortest path from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. This is our basic concern at present to determine the shortest path between a source and a destination. The all pair shortest path algorithm is used to find all pair shortest path problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph. Another all pair shortest path algorithm is matrix multiplication algorithm.**

## I. INTRODUCTION

The air transportation industry has evolved rapidly over the past few years. The growth of air passenger demands has triggered airlines to enhance their quality of service. The airlines should mainly focus on the route network development which is considered as the initial problem addressed by the airlines. It aims to determine a set of routes to be operated in an airline's network. It takes passengers demand, airport interior facilities, aircraft characteristics, and then generate a set of origin-destination pairs to serve. The schedule design problem aims to define the departure time and arrival time of each flight.

This problem consists in showing only interesting airports with respect to a specific flight. Cost and time are the major criteria of finding shortest path of this model.

This problem consists of locating in the network what routes represent business opportunities that are attractive regarding time or cost criteria, and passing through a specific flight. We model our ideas with the help of Dijkstra's algorithm. Dijkstra's algorithm is a type of single source shortest path algorithm but we have modified the algorithm to match our objective of finding cheaper and faster routes.

There are hundreds and thousands of flights all around the globe, and are bound to have different routes with multiple stops between places. In such a scenario, it is too confusing to choose which flight would be the best one to travel through. Graphs are often used to model such networks in which one travels from one point to another. A graph refers to a collection of vertices (or nodes) and a collection of edges that connect pairs of vertices. Here, we use graph theory to find the best route for the flight journey.

## II. LITERATURE SURVEY

### A. *Public transport route planning: Modified Dijkstra's algorithm*

*By Alican Bozyigit, G. Alankus, Efendi Nasiboglu*

This paper uses modified Dijkstra's algorithm to find and implement an efficient public transport route planning. Users have a variety of preferences when it comes to the definition of an ideal route, which lead to various criteria for planning and evaluation of routes. There are many algorithms to find shortest path in the literature. Dijkstra's Algorithm is the most commonly used one in order to find the shortest path between two vertices. The Dijkstra's algorithm is modified by implementing penalty study in their model. It is observed that this modified algorithm is quite efficient for route planning in the public transport network in terms of the number of transfers, distance of proposed route and walking distance. The alteration is done by using a penalty function for adding a penalty to alternative cost if it is needed.

Thus with the growth of airlines facilities,passengers try to choose their travel based on individual requirements. If the passenger needs to reach destination in less time, then he/she needs to be provided with all possible routes providing lesser time. Similarly economy should be taken into consideration. Routes with cheaper fares are to be provided to passengers.

All of these are very essential in terms increase in travel through flights.

### B. *Shortest path for aerial vehicles in heterogeneous environment using RRT**

*By P. Parpatara, B. Hérissé, R.Pepy, Y. Bestaoui*

Recently, trajectory planning is a high-demand method used for the aerial vehicles such as missiles, drones, and future Unmanned Combat Aerial Vehicles (UCAV). An efficient/optimal trajectory known *a priori* before the mission can increase the probability to complete the mission gradually. The purpose of this paper is to develop an efficient trajectory planning algorithm for an aerial vehicle traveling in 2-dimensional vertical plane while avoiding obstacles.

Trajectory planning in the vertical plane is a challenging problem with many constraints since the vehicle travels in a heterogeneous environment where the air density decreases with altitude. Most aerial vehicles depend on the aerodynamic forces to control their course in the air. As a consequence, the maneuverability of the aerial vehicles decreases with altitude.

The RRT$^*$ algorithm together with the Dubins' paths in heterogeneous environment is capable of finding a solution for the trajectory planning of the aerial vehicle in vertical plane while avoiding obstacles. The solutions keeps on improving during the remaining time that results in finding a near-optimal solution.

*C. The Approximate Shortest Distance Route Intelligent System for Traveling in Taiwan*
*By Chin-Jung Huang, Ying-Hong Lin*

In a well-known problem, there are N! possible routes for tourists to visit N cities, with each city passed through once before the return to the departure city. It is difficult to find the shortest route among N! possible routes quickly and effectively. This research proposes a method that integrates the Hungarian method and the branch-and-bound method in operation research, nearest neighbor in data mining, and rule based inference in artificial intelligence to find the approximate shortest distance route and the distance. It also uses object-oriented programming to construct the approximate shortest distance route intelligent system for traveling (ASDRST).

A traveling route passes through n cities. Dij represents the distance from city i to city j. If a tourist sets out from city 1, passes through cities 2,3,…,n in order, and then returns to city 1 with each city in the route passed through exactly one time.

By using the computing approach for finding the approximate shortest distance above, we employed object-oriented programming, constructing a highly practical and easy to use Approximate Shortest Distance Route Intelligent System for Traveling (ASDRST). The ASDRST only requires a personal computer and can find the approximate shortest distance route and its corresponding distance quickly and effectively.

It takes about 22 minutes of computing time passing through 60 cities in Taiwan to find the approximate shortest distance route by train. The method is acceptable for practical use. The maximum city number in a route is 255, meeting the actual needs of general tourists.

The method can also solve the route distance for the user-assigned route in user-defined visiting order. This provides the user with the opportunity to compare the route with the approximate shortest distance route and the distance to decide what kind of journey to take.

*D. An application of Dijkstra's Algorithm to shortest route problem.*
*By Ojekudo, Nathaniel Akpofure*

The model takes in all aspects of the business, helping management to plan and decide different levels at the various stages in the industry, e.g. knowing what to pay and what to charge. A network representation is essential in an industry because it helps to determine and monitor the flow of goods from the industry to its final destination. Taking the crude oil as an example, network representation can help determine the various stages, from crude oil purchase, shipping to refineries, refining it, to send it for storage and distribution for sale purposes. Network optimization is a special type of linear programming model. Network models have three main advantages over linear programming.

## III. PROBLEM STATEMENT

The implementation of Dijkstra's algorithm using the graph theory to find the shortest flight duration and cheaper cost of flight route between two given airports. This is the network of airports represented in the graph form. There are 8 airports connected to each other in an undirected graph. Furthermore, we use Dijkstra's algorithm to calculate shortest duration and cheapest flight route between source and destination.

Provided list of airports in our model are:

1. Mangalore [IXE]
2. Bangalore [BLR]
3. Chennai [MAA]
4. Hyderabad [HYD]
5. Visakhapatnam [VTZ]
6. Mumbai [BOM]
7. Delhi [DEL]
8. Kolkata [CCU]

The passenger provided with given list of airports is allowed to select source and destination of travel.

Thereafter he/she is given the opportunity to select the type of journey they prefer,faster or cheaper. On providing their option they get the best route considering their desired option.

This would help passengers to choose their route of journey in a simple manner, which is considered to be economical and time saving. This is an effective way which can be used by passengers, avoiding the confusion of choosing their route from all the available routes.

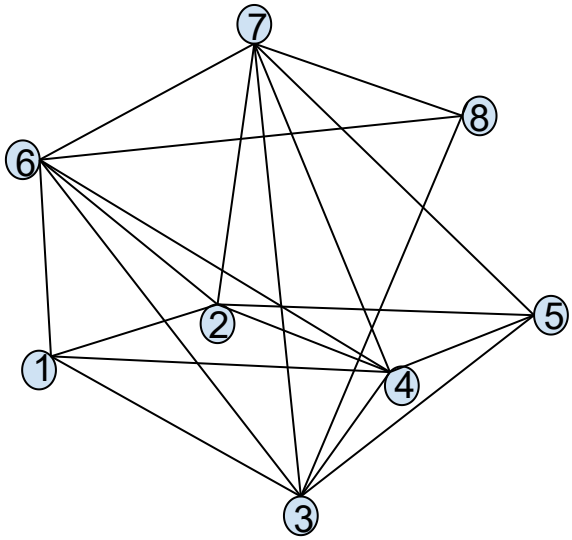


FIG. 1: Network of airports and flight routes

## IV. METHODOLOGY

Source and destination are taken as inputs from the user. The possible routes are obtained and the route with shortest duration and least cost is found using Dijkstra's algorithm.

- Each node in the above directed, weighted graph represents an airport and the edge represents the route.
- Each edge is given weights depending on user's choice. User is given a choice of finding flights based on shortest duration and cheapest flights.
- Time is given as weight when user chooses for shortest duration flights and cost is given as weight in the latter.
- Dijkstra's algorithm finds a shortest path tree from a single source node, by building a set of nodes that have minimum distance from the source. The algorithm is modified version of Dijkstra's algorithm in the sense that the algorithm is stopped as soon as the destination node is found. Our model

doesn't require calculation of all the minimum distances from a given node.

**Algorithm:**

1. Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
2. Push the source vertex in a min-priority queue in the form (distance, vertex), as the comparison in the min-priority queue will be according to vertices distances. This is done till the destination is reached.
3. Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
4. Update the distances of the vertices that are connected to the popped vertex in case of current vertex distance + edge weight < next vertex distance, then push the vertex (insert in the queue)
   with the new distance to the priority queue.
5. If the popped vertex is visited before, just proceed without using it.
6. Apply the same algorithm again until the priority queue is empty.

**Representation of weights of edges**

Considering the nodes as airports and flight route as the edge the graph can be represented as shown below:

1. Mangalore [IXE]
2. Bangalore [BLR]
3. Chennai [MAA]
4. Hyderabad [HYD]
5. Visakhapatnam [VTZ]
6. Mumbai [BOM]
7. Delhi [DEL]
8. Kolkata [CCU]

**Based on duration**

| | IXE | BLR | MAA | HYD | VTZ | BOM | DEL | CCU |
|---|---|---|---|---|---|---|---|---|
| IXE | 0 | 60 | 105 | 115 | INF | 100 | INF | INF |
| BLR | 60 | 0 | INF | 70 | 95 | 105 | 170 | INF |
| MAA | 110 | INF | 0 | 80 | 80 | 120 | 170 | 140 |
| HYD | 105 | 75 | 75 | 0 | 70 | 90 | 140 | INF |
| VTZ | INF | 100 | 90 | 75 | 0 | INF | 140 | INF |
| BOM | 90 | 105 | 110 | 85 | INF | 0 | 125 | 160 |
| DEL | INF | 165 | 170 | 130 | 135 | 130 | 0 | 130 |
| CCU | INF | INF | 150 | INF | INF | 175 | 150 | 0 |

Table 1: Representing time (in minutes) as weights

The duration is zero when the source and destination are the same airports.

The duration is taken as infinite (INF) in case of no available direct route from source to destination.

**Based on cost**

| | IXE | BLR | MAA | HYD | VTZ | BOM | DEL | CCU |
|---|---|---|---|---|---|---|---|---|
| IXE | 0 | 4541 | 7051 | 3766 | INF | 3706 | INF | INF |
| BLR | 2695 | 0 | INF | 2131 | 3554 | 1889 | 3640 | INF |
| MAA | 2748 | INF | 0 | 1622 | 2649 | 2791 | 411 | 4037 |
| HYD | 3848 | 1810 | 1648 | 0 | 2747 | 1896 | 2686 | INF |
| VTZ | INF | 3598 | 2450 | 2901 | 0 | INF | 4270 | INF |
| BOM | 3696 | 4035 | 3031 | 1596 | INF | 0 | 2438 | 4295 |
| DEL | INF | 4035 | 4106 | 2967 | 4058 | 2320 | 0 | 2944 |
| CCU | INF | INF | 4515 | INF | INF | 4448 | 3292 | 0 |

Table 2: Representing cost (in rupees) as weights

The cost is zero when the source and destination are the same airports.

The cost is taken as infinite (INF) in case of no available direct route from source to destination.

This output shows minimum cost for the direct route from Bangalore to Delhi.

**Sample output 1:**

**----------Welcome----------**

**List of airport:**

1. Mangalore
2. Bangalore
3. Chennai
4. Hyderabad
5. Visakhapatnam
6. Mumbai
7. Delhi
8. Kolkata

Enter the source:
2
Enter the destination:
7

Enter your choice for search of flights:
10-Based on time constraint
20-Based on cost constraint
0-exit

20

Travel through required source (2) to destination (7) has minimum cost of 3640 rupees and the route is [ (2) Bangalore (7) Delhi ]

This output shows minimum cost for the indirect route from Mangalore to Visakhapatnam.

**Sample output 2:**

**----------Welcome----------**

**List of airport:**

1. Mangalore
2. Bangalore
3. Chennai
4. Hyderabad
5. Visakhapatnam
6. Mumbai
7. Delhi
8. Kolkata

Enter the source:
1
Enter the destination:
5

Enter your choice for search of flights:
10-Based on time constraint
20-Based on cost constraint
0-exit

20

Travel through required source (1) to destination (5) has minimum cost of 6513 rupees and the route is [ (1) Mangalore (4)Hyderabad (5) Visakhapatnam ]

This output shows the shortest duration to travel the direct path from Chennai to Hyderabad.

**Sample output 3:**

**----------Welcome----------**

**List of airport:**

1. Mangalore
2. Bangalore
3. Chennai
4. Hyderabad
5. Visakhapatnam
6. Mumbai
7. Delhi
8. Kolkata

Enter the source:
3
Enter the destination:
4

Enter your choice for search of flights:
10-Based on time constraint
20-Based on cost constraint
0-exit

10

Travel through required source (3) to destination (4) has minimum time of 80 minutes and the route is [ (3) Chennai (4) Hyderabad ]

This output shows the shortest duration to travel the indirect path from Kolkata to Mangalore.

**Sample output 4:**

**----------Welcome----------**

**List of airport:**

1. Mangalore
2. Bangalore
3. Chennai
4. Hyderabad
5. Visakhapatnam
6. Mumbai
7. Delhi
8. Kolkata

Enter the source:
8
Enter the destination:
1

Enter your choice for search of flights:
10-Based on time constraint
20-Based on cost constraint
0-exit

10

Travel through required source (8) to destination (1) has minimum time of 260 minutes and the route is [ (8) Kolkata (1) Mangalore ]

ıner.

## V. CONCLUSION

The shortest path problem is about finding a path between two vertices in a graph such that the total sum of the edge weights is the minimum. This method provides an easier, efficient and a logical approach toward solving the problem intelligently.

This project gives a fast method and an efficient algorithm for approximation and reduction of a large network and a fast estimate calculation for the shortest and cheaper paths. It shows how graph theory can solve complicated problem consisting networks in an efficient

The user can get the shortest path (less time taken) to reach the desired destination just by entering required source and destination. Furthermore, the cheapest flight can be calculated. Among the various flights, finding the shortest duration path could be cumbersome without the knowledge of graph theory.
This model can be applied on a larger scale of networks where computing details becomes very difficult due to complexity.

## REFERENCES


[1] Alican Bozyigit, G. Alankus and Efendi Nasiboglu, Public transport route planning: Modified Dijkstra's algorithm, 2017

[2] *P. Parpatara, B. Hérissé, R.Pepy, Y. Bestaoui , Shortest path for aerial vehicles in heterogeneous environment using RRT*, 2015*

[3] *Chin-Jung Huang and Ying-Hong Lin , The Approximate Shortest Distance Route Intelligent System for Traveling in Taiwan*

[5] *Jinhau Lu and Chi Dong*, *Research of shortest path algorithm based on the data structure, 2012*

[6] *Q. Sun, J. H. Shen, J. Z. Gu, An improved Dijkstra algorithm [J]. Computer Engineering and Applications, vol. 3 ,2002*

[7] *L. B. Chen, R. T. Liu, A Dijkstra's shortest path algorithm [J]. Harbin University of Technology, vol.3,2008*