French-Azerbaijani University

Network and Algorithms

Project Report

Shortest Paths Problem on Airway Networks

Submitted by:
Parsha Joarder
Naila Ibrahimova
Javid Mammadli

January 3, 2020

Contents

1 Description of the project

The purpose of this project is relation with the airlines and airports that are analyzed in Python using graph theory and involves the identification of the minimum cost of a route between two airports by applying several algorithms that has been learnt during the course.We locate, in this paper, the shortest route using Network Model as the shortest route provides the effective minimum transportation cost,thus Flight Planning Problem is considered, which seeks to compute a cost-minimal flight trajectory on an airway network, given origin and destination airports.

The following are the two concrete tasks required:

- **Codes in Python using NetworkX library**
  Link to Google Colab to view the codes:
  `https://colab.research.google.com/drive/1VnbuxGArXkQirxtRb-M0iMol6Ed_ETW8`
- **Report - clear and vivid description of the project**

The codes has been commented for better comprehensibility.Despite that,there is a clear and vivid description of the followings of why the algorithms has been applied and eventually ends up choosing the optimal solution and present the analysis of the final results.

2 Analysis on the data

Since transportation networks lends itself beautifully to be analysed and modeled as a Graph where vertices naturally correspond to waypoints, and edges to connections between them. One such example is the airline transportation network,therefore the chosen dataset(generic) is related to Airlines Industry with information of airline routes(such as few cities (NODES) connected by airline routes (edges)).It has a source and a destination,also few columns indicating the arrival and departure times for each journey.

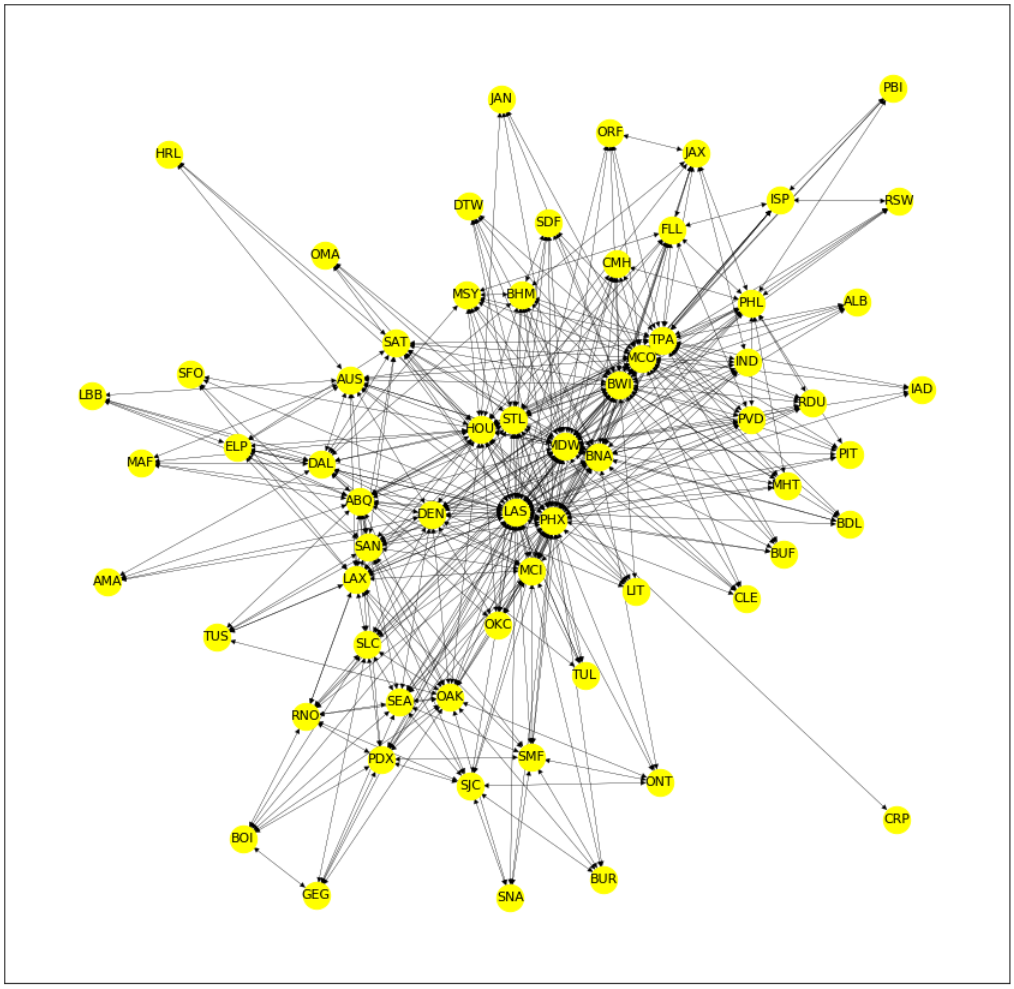
The following are the attributes of the dataset:

Field	**Description**
Year	Flight Date by year
Month	Flight Date by month
DayofMonth	Flight Date by day
DayOfWeek	Flight Date by week
DepTime	Time of Departure
CRSDepTime	Scheduled Departure Time
ArrTime	Time of Arrival
CRSArrTime	Scheduled Arrival Time
AirTime	Duration of flights
Origin	Origin State
Dest	Destination State
Distance	Distance of flights

Database contains 822 routes(EDGES) between 64 airports(NODES).
`Origin` and `Dest` are evidently the nodes,however the edge can be either `Distance` or `Airtime` since the edges represent flight routes, these weights might represent distances or time,moreover it can weigh the paths.

3 Representation of data

Graph type: DiGraph,Weighted
We can model all of these problems as asking questions about graphs, which are simply abstract representations of connections between things.An airline network tracks flights (the edges) between cities (the vertices)
Graphs are directed: in directed graphs the edges points from one node to another, A direct flight from one city to another does not necessarily imply there is also a direct return flight.
Graphs are also weighted, meaning that each edge has a cost or value associated with it. Airline networks are weighted: they might record the time, or the distance of the flight, or some other cost.



4 Algorithms applied to solve problems

4.1 Breadth-first search and Depth-first search on graphs

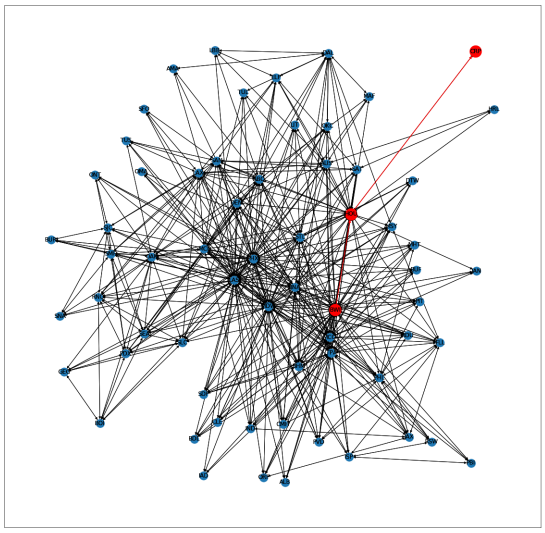
Finding paths from one node to another to explore the graph or the locations reachable
Implementation of finding any path between two vertices has been done with the following algorithms:

- Breadth-first search(BFS) `def bfs(graph, start, goal)`
  - BFS is the algorithm used when we want to find the shortest path,but the claim for BFS is that the first time a node is discovered during the traversal, that distance from the source would give us the shortest path. However,we are dealing with a weighted and directed graph here,but BFS algorithm is suitable to find the shortest path for undirected and unweighted graph. Therefore,this solution is not that feasible for a large network that would have potentially thousands of nodes.
- Depth-first search `def dfs(graph, start, end)`
  - DFS is the algorithm that is optimized not to tell us if a path is the shortest or not,which is not necessary in flights network,however this algorithm tells us if the path even exists!It allows us to determine whether two airports have a path between them. The DFS algorithm does this by looking at all of the adjacent airports of the source node, until it reaches destination. It does this by recursively taking the same steps, again and again, in order to determine if such a path between two airports even exists.

4.2 Dijkstra and Bellman-Ford

Presenting set of algorithms with route planning in flights network. Since determining best connections in transportation networks plays a vital role and the goal is not merely to find a route, but to find the shortest route from one place to another,therefore usage of Routing algorithms has been applied:

- Dijkstra `def dijsktra(G, source, dest, attr)`
  This is the representation of dijkstra function result from 'BWI' to 'CRP' airport.



- Bellman-Ford `def bellman_ford(graph, source, dest)`

The problem can be solved by modeling a transportation network as a graph where edge weights depict travel times/distance on the corresponding connection. In general, Dijkstra's algorithm can solve the problem of finding the quickest path between two nodes,as well as Bellman-Ford algorithm.Both algorithms are applied,however Bellman-Ford just can be eliminated since there is absence of negative edge weights on the graph,thus Dijkstra is an efficient way to plan routes in flight networks for realistic conditions and a flight optimization method based on Dijkstra's algorithm is used to find the optimal minimal time path and to simplify cost calculations.

4.3 Minimum Spanning Tree

Another optimal solution is the minimum spanning tree-Kruskal's algorithm(MST) that constructs a solution step by step, where at every stage it addes increasing cost arcs. Adopting kruskal's algorithm to get a minimum spanning tree is finding a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component). MST cannot do anything about the distance from one connection to another, it can be used to determine the least costly paths with no cycles in this network, thereby connecting all the airports at a minimum cost.The use of a Priority Queue(implementing a heap) will store edges adjacent to vertices you visit. The root of this heap will always have the smallest weighted edge.Main part of kruskal's algorithm where you select the smallest edge every time you visit a vertex.

5 Conclusion

The primary objective of this project is to introduce various graph algorithms including breadth-first graph traversal, Dijkstra's shortest path and Kruskal's minimum spanning tree algorithms,e.t.c. In this work, we modelled flight networks as graphs such that we are able to compute best connections efficiently and after implementing effective graph analysis algorithms the optimal path has been found and the lowest-cost paths are efficiently calculated.
The efficient way of optimization after doing comparisons between the algorithms is the Minimum Spanning Tree,since this algorithm only gives meaningful results when run on a graph, where the relationships have different weights.The Kruskal's algorithm has specifically applies to construct the tree where it greedily grab edges/routes and eventually ends up always grabbing the lowest weight edge, and see if it helps build a spanning tree; if so, we add it, if not we discard it; and keeps doing that until we have the spanning tree.

NOTE* Please refer to the next page for result after applying MST

Link to Google Colab to view the codes:

`https://colab.research.google.com/drive/1VnbuxGArXkQirxtRb-M0iMol6Ed_ETW8`

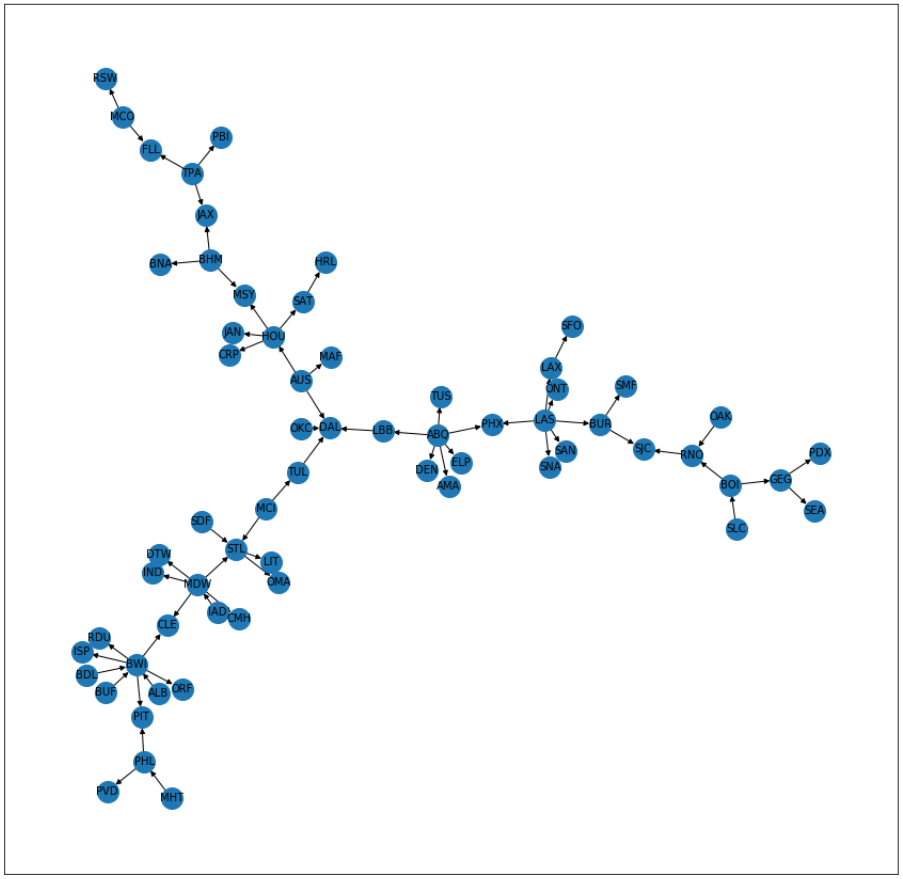


Figure 1: Minimum spanning tree using Kruskal's algorithm