Report - Project 2
Flow networks and Airlines

Rick Hutchison
Vivienne Prince
Steven Spielman
Miles Tweed

The team was given two data sets regarding airline route and plane information. With this data, the team was tasked with finding the maximum capacity for flights from a source and destination airport with a maximum of one layover. The team also needed to find the maximum capacity flight for the same trip with a single carrier. The team incorporated plane capacity data with their current data to analyze the desired flights and find the maximum capacities for all paths and those with a single carrier.

The team created a data munging script that performs a web scrape to join plane data with their capacities. The team then created a script to take in a source airport, destination airport, and the maximum number of layovers. From here, the user can select to see the maximum flight capacity or the maximum single carrier capacity of the flights from the given source and destination airports. The team decided to look at an example of flights from the source airport of Tampa (TPA) to the destination airport of Chicago O'Hare (ORD).

Data Transformation

The team started with two data files concerning airline routes and plane information. The routes file gave an airline code, airline id, source airport, source airport id, destination airport, destination airport id, codeshare, stops, and equipment. The equipment gives the code of the plane type(s) generally used for that flight. This code is also used in the planes file, which has plane names, the equipment code, and the plane id. With plane information, the team can find capacities for each plane type in the planes file.

The team scraped capacity data from the website:
https://blog.thetravelinsider.info/airplane-types
Now that the capacities have been found, the team joined the planes data with the scrapped capacities to create a new file planes_cap.

Now that the team has airline route information with data for planes joined with their capacities, there is enough available data to assign each flight a route, carrier, and capacity. This will allow the team to find the maximum number of people that can be on each flight as well as the carrier that can transport the maximum number of people between two destinations.

Algorithm Utilization

To find the maximum number of people that can be moved from TPA to ORD, the team found all of the paths in the graph with a source of TPA and the destination of ORD with a maximum of one layover with networkx. To satisfy the condition of having a maximum of one layover, the team specified a depth of 2 in the graph. With all of the paths with a maximum of one layover available, the team used a max flow function (Edmonds-Karp) to find the path with the maximum capacity. The Edmonds-Karp algorithm finds the maximum flow by using the idea of residual networks.

```
max flow = 0
using weights of the edges as flow,

while there exists a path, p, from source to target: (found using BFS - always returns
the shortest path available)
    min_capacity = minimum residual capacity in path
    max_flow = max_flow + min_capacity

    update the residual graph by doing for each edge (u, v) in p:
        (u, v).weight = (u, v).weight - min_capacity
        (v, u).weight = (v, u).weight + min_capacity
```

To find the maximum capacity for a single carrier flight from TPA to ORD, the team used the same approach, just with a couple adjustments. By using a set for the carriers, we isolated the paths available for each carrier with a maximum depth of 2 (1 layover at most), and used Edmonds-Karp again to find the maximum flow for each carrier. The carrier with the biggest max flow is the answer to our problem.

Algorithm Implementation

To create and analyze the graph, the team utilized the python package networkx. The team went with networkx since all members had prior experience using it and it allowed for the utilization of networkx functions to create and work with the graph.

With the graph created, the team implemented their max flow algorithm to find the flight with maximum capacity from TPA to ORD. The algorithm first uses the networkx function nx.all_simple_paths to find all of the paths from TPA to ORD with a cutoff depth of 2 to represent a maximum of one layover. These paths are stored in a list and iterated through to look at the capacities. As mentioned, the maximum capacities for routes with a layover or stop are set to the minimum capacity over the entire path. This is to ensure that it is truly the maximum number of people able to travel the entire flight.

After iterating through the paths and finding the path with the maximum capacity for the flight, the algorithm outputs the flight information as well as the carrier(s) and capacities for each leg in the event of a layover. For our source of TPA and destination of ORD, the maximum capacity is 412 people with 1 layover.

The team then implemented their max carrier algorithm to find the flight with the maximum capacity with a single carrier from TPA to ORD. The algorithm first uses the networkx function nx.all_simple_paths to find all of the paths from TPA to ORD with a cutoff depth of 2 to represent a maximum of one layover, similar to the first algorithm. These paths are stored in a list and iterated through to look at the capacities. In this case, the single carrier has the same maximum capacity along the path in the event of a layover, but we need to distinguish paths with one carrier from those with more than one. To address this problem, the algorithm stores the carriers on the path as a set, that way duplicates are not recorded. If the length of the carriers list for the path is 1, then the flight has the same carrier, and the path is recorded for comparison.

With all of the paths from TPA to ORD with a maximum of one layover and on flights of the same carrier, the capacities can be compared, and the maximum found. The algorithm performs this comparison by iterating through the paths. After iterating through the paths and finding the path with the maximum capacity for the flight, the algorithm outputs the flight information as well as carrier and capacities for each leg in the event of a layover.

In addition to implementing the algorithms to find the maximum flight capacity and maximum single carrier capacity, the team created a user interface to allow the user to select a source airport, destination airport, and the maximum number of layovers to provide wider use and explore other cases. Once the user inputs the needed information, they can select to see the maximum flight capacity, maximum single carrier capacity, or exit the interface. The maximum single carrier capacity is 179 people with carrier NK.

Validation

To validate the output, the team made a few considerations. First, using networkx and its functions, the team wanted to ensure that the graph was being constructed correctly and that the paths identified were truly all of the paths. To validate this, the team analyzed all of the paths in the list to the paths in the data. This confirmed that every path is returned by the networkx functions. For our example, there were not too many flights, making this manual validation possible.

Future Extensions

In the team's initial discussion of future extensions for this project, it had come up to have the algorithms used with source, destination, and even maximum allowed layovers to be variables and used to explore other cases. A change to using a user interface was also mentioned. The team decided to take our solution a step further and implement these ideas so that the program creates a user interface that can take in any source, destination, and maximum layovers to create the graph. Then, the user can even decide which maximum capacity they would like to find.

From where the team will end the project, there are new extensions that can take the project even further. If time data is available for flight and layover length, the minimum layover or overall travel time could be considered as well when identifying the maximum capacity flights. Flight capacity may also be separated by class with ticket price data to look at maximizing revenue.