Operations Research

Minimum cost maximum flow project

Zsombor Malatinszki

2023.06.11.

The program, written in C# (as a windows form app), has the following functionalities:

- Adding a new graph in the specified input format to the program
- Parsing and displaying the input file with DOT language
- Calculating the max flow
- Calculating the min cut
- Calculating the min cost max flow

Instructions to run the program:

1, Install Visual Studio

- During installation, makes sure to add „.NET desktop development” in the „Workloads” tab, under „Desktop & Mobile”

2, Make sure GraphViz is installed on your machine (64-bit, add to system path)

3, Unzip the compressed folder, and double click on „MinCutMaxFlow.sln”

4, Press F5 to start the program

5, To view the code, in the solution explorer, right click on „MinCutMaxFlow.cs”, and then click „View Code”

6, Make sure that the path to which the program is downloaded doesn’t have special characters or spaces

Alternatively, the .exe file can be found in MinCutMaxFlow/bin/Debug/net6.0-windows/MinCutMaxFlow.exe (points 2 and 6 are still necessary).

Regarding the input format:

The program takes the inputs as specified in the pdf, with the following restrictions:

- the nodes must be numbered sequentially, without skips in numbering (eg. nodes 1,2,3,4 are accepted, but 1,3,4,5 are not
- only costs greater than or equal to 1 are accepted
- make sure that the path of the the input file does not contain any special characters

Algorithms used in the program:

In the **max flow** calculation (CalculateMaxFlow function), the Edmonds-Karp algorithm is used. The Edmonds-Kapr algorithm is a specific implementation of the Ford-Fulkerson algorithm, using breadth-first search (BFS).

The algorithm follows these steps:

1. Initialize the flow in all edges to 0.
2. While there exists a path from the source to the sink in the residual graph (using Breadth-First Search):
   - Find the minimum residual capacity along the path.
   - Update the flow along the path by adding the minimum residual capacity.
   - Update the residual capacities of the forward and backward edges.
3. Return the maximum flow.

The maximum flow value can be found in a time complexity of O(V * E^2), where V is the number of vertices and E is the number of edges in the network.

In the **min cut** calculation (CalculateMinCut function), the same algorithm is used as before, with the addition of a depth-first search (DFS) on the residual graph, once the maximum flow has been calculated. With the DFS, all nodes that can be visited from the source are visited. The edges that will be in the min cut are the ones between node(u) that we could visit and node(v) that we could not in the residual graph, but only for those nodes that had an edge in the original graph.

In the **min cost max flow** calculation (CalculateMinCostMaxFlow function), the Bellman-Ford algorithm is used to find the paths with the least cost („least distance"). The algorithm follows these steps:

1. Initialize the distance from the source vertex to all other vertices as positive infinity (int.MaxValue), except the distance to the source vertex itself, which is set to 0.
2. Relax edges repeatedly. In each iteration, consider all edges and update the distance to each destination vertex if a shorter path is found:
   - for each edge (u, v) with cost c, if the distance to u plus w is less than the current distance to v, update the distance to v as the distance to u plus c.
3. After numNodes-1 iterations, all shortest paths from the source vertex have been found unless there is a negative cycle present.
4. To check for the presence of a negative cycle, perform an additional iteration and if any distance is updated, then a negative cycle exists in the graph.
5. Additionally, if the sinkNode is no longer reachable, the algorithm ends