DAT600 - Assignment 4

University of Stavanger, Norway

1 TASK1 - FROM PREVIOUS EXAM (DECEMBER 2012)

The constraints that are given in hours:

$$\frac{x}{4} + \frac{y}{3} \leq 40$$
$$\frac{x}{3} + \frac{y}{2} \leq 35$$
$$x \geq 10$$
$$x, y \geq 0$$

Profit per unit produced:

$$p_x = 200 - \frac{100}{4} - \frac{20}{3}$$
$$p_x = 168.33$$
$$p_y = 300 - \frac{100}{3} - \frac{20}{2}$$
$$p_y = 256.67$$

We want to maximize the profit, z, where x is number of produced units of x and y is number of produced units of y

$$z = p_x \cdot x + p_y \cdot y$$
$$z = 168.33x + 256.67y$$

2 DETAILED EXPLANATION OF THE MINIMUM CUT AND MAXIMUM FLOW PROBLEMS

In this assignment, we analyze two key network flow problems using the PuLP library in Python: the Minimum Cut (Bottleneck) Problem and the Maximum Flow Problem. The full source code is available at https://github.com/Tinnitussen/dat600-assignments/tree/main/assignment-4. Below is a detailed explanation with selected code fragments to illustrate our approach.

a Minimum Cut (Bottleneck) Problem

Concept Overview: The minimum cut problem seeks to partition the nodes of a directed network into two groups such that the source (**S**) is in one group and the sink (**t**) is in the other. An edge is considered to "cross the cut" if it goes from a node in the source side (assigned the value 0) to a node in the sink side (assigned the value 1). The objective is to minimize the sum of the capacities of these crossing edges, which represents the weakest link or bottleneck in the network.

Model Formulation: In our model:

- We assign each node a binary variable using:

```
group = {node: pulp.LpVariable(f"group_{node}", cat="
    ↪ Binary") for node in nodes}
```

  A value of 0 indicates a node is in the source group, while 1 indicates it is in the sink group.

- For each directed edge (i, j) with a given capacity, we create a binary variable that indicates whether the edge crosses the cut:

```
cut_edge[(i, j)] = pulp.LpVariable(f"cut_{i}_{j}", cat
    ↪ ="Binary")
problem += cut_edge[(i, j)] >= group[j] - group[i]
```

  This constraint ensures that if node i is in group 0 and node j is in group 1, then the variable `cut_edge[(i, j)]` must be 1.

- We force the source and sink to be in their respective groups:

```
problem += group['S'] == 0
problem += group['t'] == 1
```

- Finally, the objective is set to minimize the sum of the capacities of the crossing edges:

```
problem += pulp.lpSum([capacity * cut_edge[(i, j)] for
    ↪  (i, j, capacity) in arcs])
```

Output Summary: The solver produced a minimum cut capacity of **30**. In the solution, the node assignments were:

- All nodes except the sink (**t**) were assigned to group 0.
- The sink (**t**) was assigned to group 1.

This indicates that the only edges crossing from group 0 to group 1 are those entering **t** (specifically, from nodes **V4** and **V5**) with capacities that sum to **30**.

b Maximum Flow Problem

Concept Overview: The maximum flow problem determines the greatest amount of flow that can be sent from the source (**S**) to the sink (**t**) through the network without exceeding the capacities on the edges. Each edge is assigned a flow variable, and the model maximizes the total flow leaving the source while ensuring that, for every intermediate node, the flow into the node equals the flow out of the node (flow conservation).

Model Formulation: Key steps in our maximum flow model include:

- Creating a flow variable for each edge that is bounded between 0 and the capacity of the edge:

```
flow = {(i, j): pulp.LpVariable(f"flow_{i}_{j}",
    ↪ lowBound=0, upBound=cap)
      for (i, j, cap) in arcs}
```

- Defining the objective function to maximize the total flow leaving the source (**S**):

```
max_flow += pulp.lpSum([flow[(i, j)] for (i, j, cap)
    ↪ in arcs if i == 'S'])
```

- Adding flow conservation constraints for each node (except **S** and **t**):

Assignment 2 in Computer Science (DAT600), IDE, UiS 2025.

```
for node in nodes:
    if node not in ['S', 't']:
        inflow = pulp.lpSum([flow[(i, j)] for (i, j,
            ↪ cap) in arcs if j == node])
        outflow = pulp.lpSum([flow[(i, j)] for (i, j,
            ↪ cap) in arcs if i == node])
        max_flow += (inflow == outflow)
```

Output Summary: The maximum flow model confirms the result predicted by the max-flow min-cut theorem: the maximum flow from **S** to **t** is also **30**. This means that the largest possible flow through the network, given the edge capacity limits is **30**.

Conclusion: The detailed modeling of both the minimum cut and maximum flow problems demonstrates that our network's bottleneck has a capacity of 30. The agreement between the minimum cut result and the maximum flow (as predicted by the max-flow min-cut theorem) validates our approach. For further details and complete code, refer to the repository at https://github.com/Tinnitussen/dat600-assignments/tree/main/assignment-4.

REFERENCES