Combinatorial Optimization

Matching

Dias, Schiewe, Pattanaki *Instructor: Pattanaki*

§ Week IV §

Problem 1: Maximal Matching

Find a maximum match in the following graph:

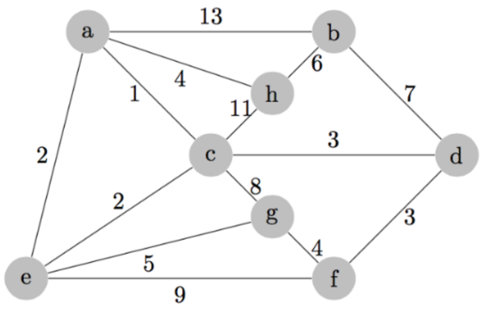

Figure 1: Undirected weighted graph

Solution:

There are two approaches:

- **Without** weights:

  The resulting graph would look like the following:

  Now, we proceed with the algorithm:

  1. Choosing (a, b) as the first edge in our matching: $M = \{(a, b)\}$;
  2. Following the algorithm, we need to find an augmenting path. For example, $c - a - b - d$;
  3. The next step in the algorithm, the goals is: the new matching M is the combined edges that are in the previous matching but not in the path and edges that are in the path but no in the previous path. The following equation resumes that:

  $$M = \{(M \setminus E(P)) \cup (E(P) \setminus M)\} \tag{1.1}$$

  4. With the given path and the current M:

  $$M = \{(\emptyset) \cup ((c, a), (b, d))\} \tag{1.2}$$

  5. Resulting $M = \{(c, a), (b, d)\}$.
  6. The next path we can pick is, for instance, $g - c - a - b - d - f$;

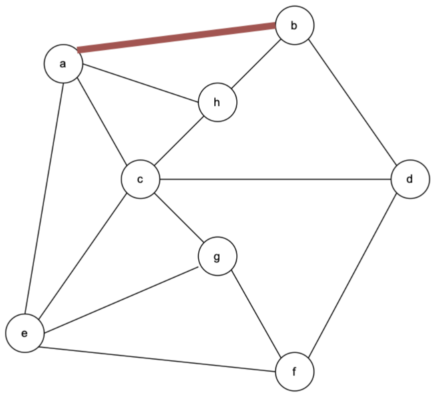

Figure 2: 1^{st} iteration

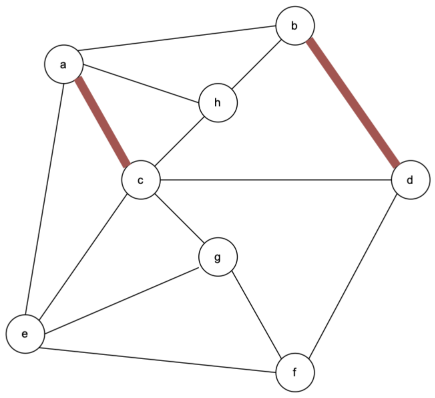

Figure 3: 2^{nd} iteration

7. Applying the same procedure:

$$M = \{() \cup ((g, c), (a, b), (d, f))\} \tag{1.3}$$

8. Resulting matching is $M = \{(g, c), (a, b), (d, f)\}$.

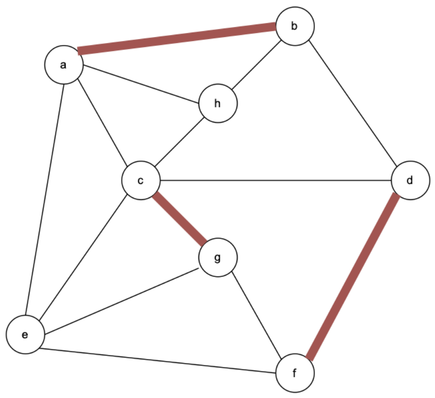

Figure 4: 3^{rd} iteration

9. There is no remaining augmenting path left.

- **With** weights:

With this approach, we can follow the same strategy as in Kruskal's algorithm for minimum spanning tree.

1. Sort all edges by decreasing weight value;
2. For each edge added, check if the current matching is not violated. Otherwise, do no add such an edge;
3. Finish it when all edges have been checked.

Using the graph provided as input:

1. First edge added is (a, b), followed by edge (c, h) and (e, f), resulting in the match $M = \{(a, b), (c, h), (e, f)\}$;
2. Edges $(c, g), (b, d), (b, h), (g, e), (g, f), (d, f), (c, d), (a, e), (c, e), (a, c)$ cannot be added due to violating the matching M.

The resulting output is:

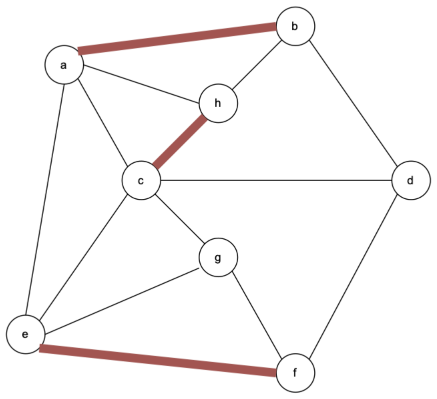

Figure 5: 4^{st} iteration

Problem 2: Ford vs Edmond

Find the maximum flow in the following graph using Ford-Furkelson;

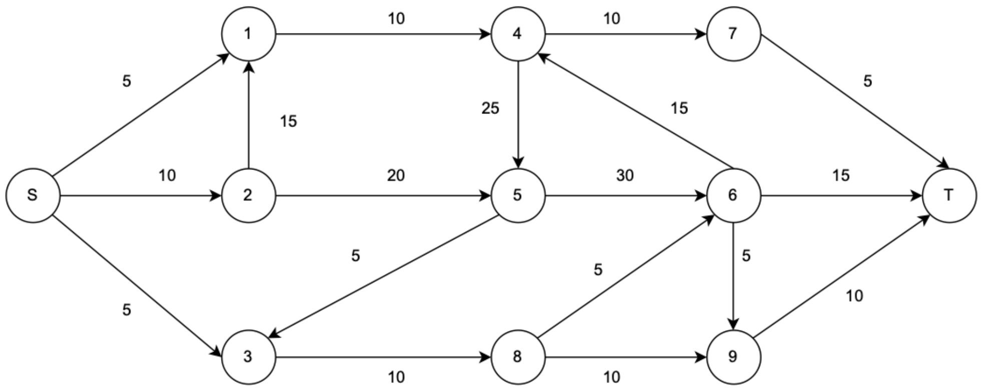

Figure 6: Directed weighted graph

Solution:

Applying Ford-Furkelson, we can highlight a few augmenting paths between source and sink:

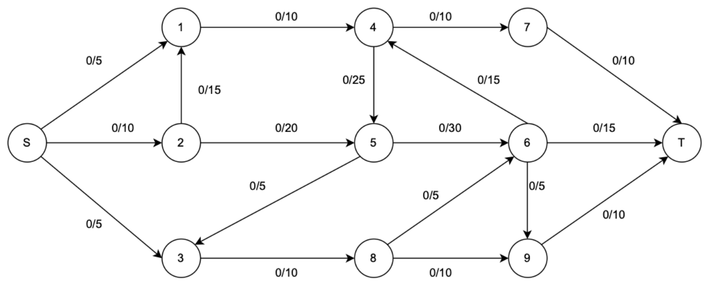

Figure 7: Set all flows to zero.

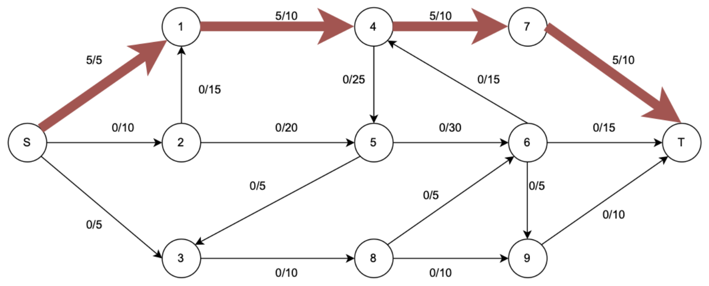

Figure 8: First path

1. Path $S - 1 - 4 - 7 - T$ with bottleneck capacity equal to 5;

2. Path $S - 2 - 5 - 6 - T$ with bottleneck capacity equal to 10;

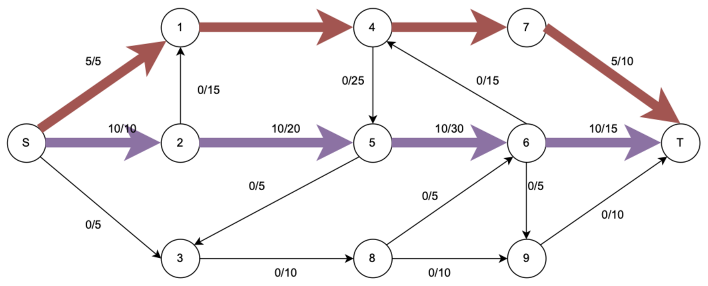

Figure 9: Second path

3. Path $S - 3 - 8 - 9 - T$ with bottleneck capacity equal to 5.

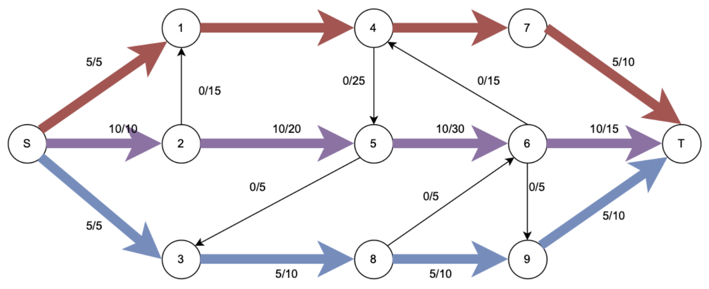

Figure 10: Third path

4. No other path can leave the source.

With that the maximum flow is 20.

Problem 3: Flows and matchings

How can Ford-Fulkerson be used for maximum matchings?
Solution:

The problem of finding the maximum matching can be reduced to maximum flow in the following manner. Let $G(V, E)$ be the bipartite graph where V is divided into X and Y. We will construct a directed graph $G'(V', E')$ in which V' which contains all the nodes of V along with a source node s and a sink node t. For every edge in E, we add a directed edge in E' from X to Y . Finally we add a directed edge from s to all nodes in X and from all nodes of Y to t. Each edge is given unit capacity.

Let f be an integral flow of G_0 of value k. Then we can make the following observations:

1. There is no node in X which has more than one outgoing edge where there is a flow.
2. There is no node in Y which has more than one incoming edge where there is a flow.
3. The number of edges between X and Y which carry flow is k.

By these observations, it is straightforward to conclude that the set of edges carrying flow in f forms a matching of size k for the graph G. Likewise, given a matching of size k in G, we can construct a flow of size k in G_0. Therefore, solving for maximum flow in G0 gives us a maximum matching in G. Note that we used the fact that when edge capacities are integral, Ford-Fulkerson produces an integral flow.

Problem 4: Gale-Shapley

Gale-Shapley is an algorithm developed by two American scientists: David Gale and Lloyd Shapley, to find solutions to stable matching problems. In their formulation, the polynomial algorithm (linear in the input size) works as a truthful mechanism from the point of view of the proposing participants, for whom the solution will always be optimal. In addition, the Gale-Shapley algorithm produces stable matchings on complete bipartite graphs. Consider now complete graphs. Are all matchings stable?

Solution:

No. Consider K4 with vertices $\{1, 2, 3, 4\}$ and preferences, for example, vertex 4 will have to be paired with someone, let's say i, leaving only one other pair, let's say j, k who are paired together. i prefers j or k over 4; also either j or k prefers i over its actual partner. Hence no matching is stable.

Problem 5: Tic-Tac-Toe

A positional game consists of a set X of positions and a family $W_1, W_2, \cdots, W_m \subset X$ of winning sets (Tic-Tac-Toe has 9 positions corresponding to the 9 boxes, and 8 winning sets corresponding to the three rows, three columns, and two diagonals). Two players alternately choose positions; a player wins when they collect a winning set.

Suppose that each winning set has size at least a and each position appears in at most b winning sets (in Tic-Tac Toe $a = 3$ and $b = 4$). Prove that Player 2 can force a draw if $a \geq 2b$.

Hint: Form a bipartite graph G with bipartition (X, Y) where $Y = \{W_1, W_2, \cdots, W_m\} \cup \{W_1', W_2', \cdots, W_m'\}$ with edges xW_j and xW_0 whenever $x \in W_j$. How can Player 2 use a matching in G?

Solution:

Let $Y' \subseteq Y$ and define S to be the set of all edges incident with a vertex in Y'. Since every vertex in Y has degree at least a we must have $|S| \geq a|Y'|$ .On the other hand, every vertex in X has degree at most 2b, so we must have $2b|N(Y')| \leq |S|$. Combining these gives us $|Y'| \leq 2b|N(Y')|$ and together with the assumption $a \geq 2b$ we find $N(Y')| \geq |Y'|$.

For every $1 \leq i \leq m$ let Q_i be the set consisting of the two vertices in X which are matched to W_i and $W_i^{'}$. Now the sets $Q_1, \cdots, Q_m$ are disjoint two element subsets of X and every W_i contains Q_i. Here is a strategy which will guarantee the second player a draw (or better). For each move made by the first player, if the first player chooses a position $xæQ_i$ for some $1 \leq i \leq m$ then the second player responds by choosing the other position in Q_i if it is available. Otherwise, the second player just plays arbitrarily. It follows from a straightforward induction that after every turn of the second player, there is no set Q_i for which player 1 has chosen one element, and player 2 none. It follows from this that player 1 can never choose both members of a set Q_i, and from this that player 1 cannot choose all members of any W_i. Thus, player 1 cannot win when player 2 adopts this strategy.