Network Flow

Acknowledgement
This notes is compiled based on the following textbooks and a variety of Internet resources, such as Wikipedia for example.

- Introduction to Algorithms by Cormen, Lerserson, Rivest and Stein.
- Network Flows by Ahuja, Magnanti and Orlin.

1 LP Modeling of Maximum Flow

Given a directed graph $G(V, E)$, where each edge $e \in E$ is associated with a **nonnegative capacity** $u(e)$. A **flow** f from a **source vertex** $s \in V$ to a **sink vertex** $t \in V$ is a function $f : E \to [0, \infty)$, such that

- For any edge $e \in E$, $0 \leq f(e) \leq u(e)$, i.e., the amount of flow on any edge must be non-negative and below the flow capacity of the edge.
- for every vertex v and $v \neq s$ and $v \neq t$:

  $$\sum_{\forall e(w,v) \in E} f(e) = \sum_{\forall e(v,w) \in E} f(e),$$

  i.e., the "incoming" flow and "outgoing" flow for very vertex must be balanced.
- The **value** of the flow f denoted $|f|$ is:

  $$\begin{aligned} |f| &= \sum_{\forall e(s,w)} f(e) - \sum_{\forall e(w,s)} f(e) \\ &= \sum_{\forall e(w,t)} f(e) - \sum_{\forall e(t,w)} f(e), \end{aligned}$$

  i.e., the total amount of flow out of s, or the total amount of flow into t.

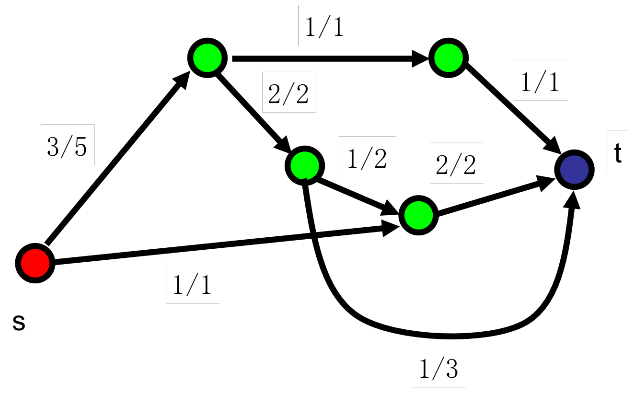

Figure 1: Illustraing a network flow f, where the pair of numbers f_e/u_e denote the flow value $f(e)$ and capacity $u(e)$ of an edge e. The value of the flow is 4.

Figure 1 illustrates a flow on a network.

A flow functiion f is a **maximum flow** if its flow value is maximum. The flow as shown in Figure 1 is also a maximum flow on the network.

Clearly the maximum flow problem is a linear programming problem:

$$\begin{array}{lll} \text{maximize} & \sum_{\forall e(s,w)} f(e) - \sum_{\forall e(w,s)} f(e) & \\ \text{subject to} & 0 \leq f(e) \leq u(e) & \forall e \in E \\ & \sum_{\forall e(w,v) \in E} f(e) = \sum_{\forall e(v,w) \in E} f(e) & \forall v \in V - \{s, t\} \end{array}$$

To see this LP model, let's look at a concrete example. Figure 2 shows the same network as from Figure 1. To find the maximum flow in this network, we will assign a variable x_e to each edge $e \in E$. The LP model of the maximum flow problem is:

$$\begin{array}{lll}
\text{maximize} & x_1 + x_2 & \text{or } x_6 + x_8 \\
\text{subject to} & 0 \le x_1 \le 5 & \\
 & 0 \le x_2 \le 1 & \\
 & 0 \le x_3 \le 1 & \\
 & 0 \le x_4 \le 2 & \\
 & 0 \le x_5 \le 2 & \\
 & 0 \le x_6 \le 1 & \\
 & 0 \le x_7 \le 2 & \\
 & 0 \le x_8 \le 3 & \\
 & x_1 = x_3 + x_4 & \text{for vertex 1} \\
 & x_4 = x_5 + x_8 & \text{for vertex 2} \\
 & x_3 = x_6 & \text{for vertex 3} \\
 & x_2 + x_5 = x_7 & \text{for vertex 4}
\end{array}$$

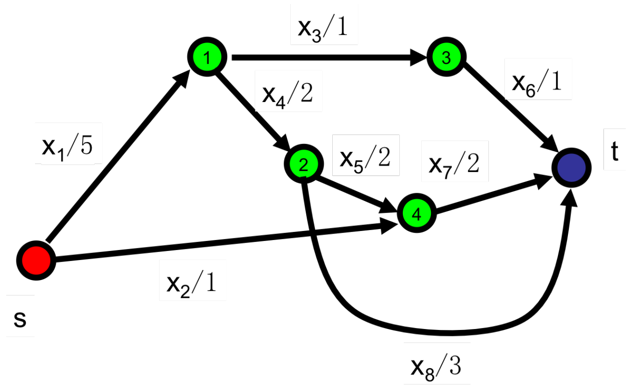

Figure 2: Illustraing the LP modeling of maximum flow.

2 The Maximum Flow Modeling of Maximum Bipartite Matching

Definition 1 *An undirected graph $G(V, E)$ is* **bipartitie** *if the vertex set V can be partitioned into two disjoint subsets U and W, such that all edges in E connects a vertex in U to a vertex in W. (See Figure 4 for illustrations.)*

Definition 2 *A* **matching** *M in a graph $G(V, E)$ is a subset of edges, such that no two edges from M share a common vertex.*

Definition 3 *Given a bipartitie graph, its* **maximum bipartite matching** *is a matching of the graph with the most number of edges. (See Figure 4 for illustrations.)*

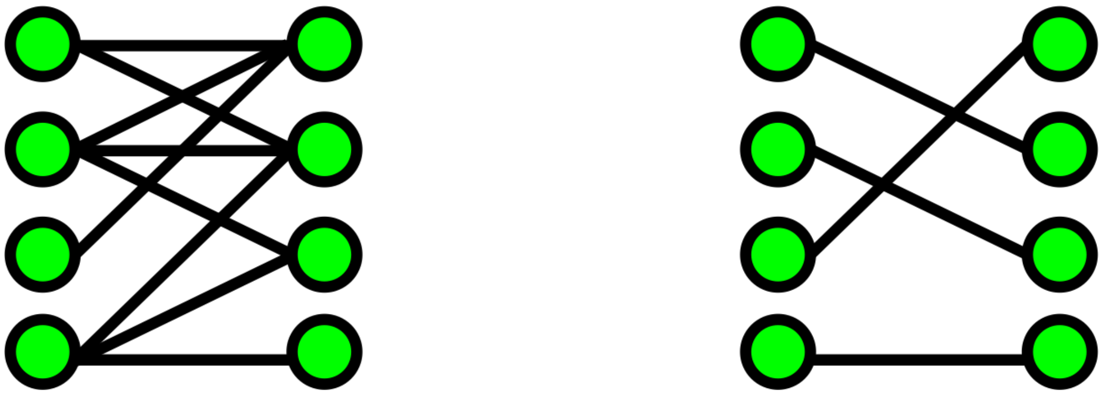

Figure 3: A bipartite graph and its maximum matching.

An interesting observation is that we can reduce the maximum bipartite matching problem to the maximum flow problem. More specifically, let the given bipartite graph be $G(U \cup W, E)$, where U and W are the two disjoint subsets of vertices. We introduce a "dummy source" vertex s and a "dummy sink" vertex t. We introduce a directed arc from s to all vertices of U, and a directed arc from all vertices of W to t. All the undirected edges from E will be converted to directed arcs from U to W. All directed edges will have a unit flow capacity of 1. Clearly, the maximum bipartitie matching correspnds to the maximum flow from s to t.

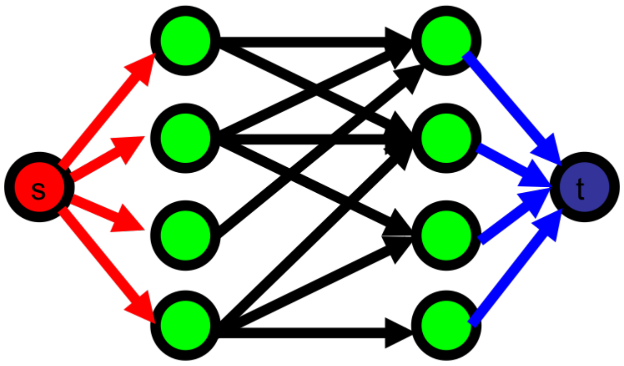

Figure 4: A bipartite graph and its maximum matching.

3 LP Modeling of Minimum Cost Flow

Given a directed graph $G(V, E)$, where each edge $e \in E$ is associated with a **nonnegative capacity** $u(e)$ and a **nonnegative cost per unit flow** $c(e)$.

A **feasible flow** f from a **source vertex** $s \in V$ to a **sink vertex** $t \in V$ with **demand** b is a function $f : E \to [0, \infty)$, such that

- For any edge $e \in E$, $0 \le f(e) \le u(e)$, i.e., the amount of flow on any edge must be non-negative and below the flow capacity of the edge.
- for every vertex v and $v \neq s$ and $v \neq t$:

$$\sum_{\forall e(w,v)\in E} f(e) = \sum_{\forall e(v,w)\in E} f(e),$$

  i.e., the "incoming" flow and "outgoing" flow for very vertex must be balanced.

- The **value** of the flow f is exactly the demand b, i.e.,

$$\begin{aligned} b &= \sum_{\forall e(s,w)} f(e) - \sum_{\forall e(w,s)} f(e) \\ &= \sum_{\forall e(w,t)} f(e) - \sum_{\forall e(t,w)} f(e) \end{aligned}$$

  i.e., the total amount of flow sent from s to t is the demand b.

- The **cost** of the flow f is:

$$\sum_{\forall e \in E} f(e)c(e)$$

A feasible flow functiion f is a **minimum cost flow** if its cost is minimized. Clearly the minimum cost flow problem is also a linear programming problem:

$$\begin{array}{lll} \text{minimize} & \sum_{\forall e(s,u)} f(e)c(e) & \\ \text{subject to} & 0 \le f(e) \le u(e) & \forall e \in E \\ & \sum_{\forall e(s,v)} f(e) - \sum_{\forall e(u,s)} f(e) = b & \\ & \sum_{\forall e(u,t)} f(e) - \sum_{\forall e(t,v)} f(e) = b & \\ & \sum_{\forall e(u,v)\in E} f(e) = \sum_{\forall e(v,u)\in E} f(e) & \forall v \in V - \{s,t\} \end{array}$$

4 Augmenting Path Algorithm

For the moment, let's consider the following heuristic approach for solving the maximum flow problem. We will prove its optimality in the next section. The idea of the heuristic is very simple. We will send flow from the source to the sink as long as possible. In other words, if we can find a path from s to t that's not saturated (i.e., with remaining capacity), we will send as much flow through the path as we can. This path is called an **augmenting path** since it will increase (or augment) the flow value.

In order to implement the above idea, we will need the concept of residue graph, which facilitates in finding augmenting paths.

Definition 4 *Let $G(V, E)$ be a network, where each edge $e \in E$ is associated with a positive flow capacity $u(e)$. Let f be a flow function on G. The residue graph $G'(V', E')$ with capacity u' of G with respect to the flow function f is the graph obtained as follows:*

- *$V' = V$, i.e., the residue graph G' has the same set of vertices as the original graph G.*
- *For every arc $e(v, w)$:*
  - *If $f(e) = 0$, add arc $e'(v, w)$ to G' with a capacity $u(e') = u(e)$.*
  - *If $0 < f(e) < u(e)$, add arc $e'(v, w)$ to G' with a capacity $u(e') = u(e) - f(e)$. We will also introduce the reverse arc $e''(w, v)$ to G' with a capacity $u(e'') = f(e)$.*
  - *If $f(e) = u(e)$, we only introduce the reverse arc $e''(w, v)$ to G' with a capacity $u(e'') = u(e)$.*
- *Merge all multi-arcs (if exist) by summing up their capacities.*

Figure 5 illustrate the idea of the algorithm. There are three iterations, in each iteration, the algorithm first builds the residue graph, and then use depth first search to find a unsaturated path from s to t, and augment the flow function as much as possible.

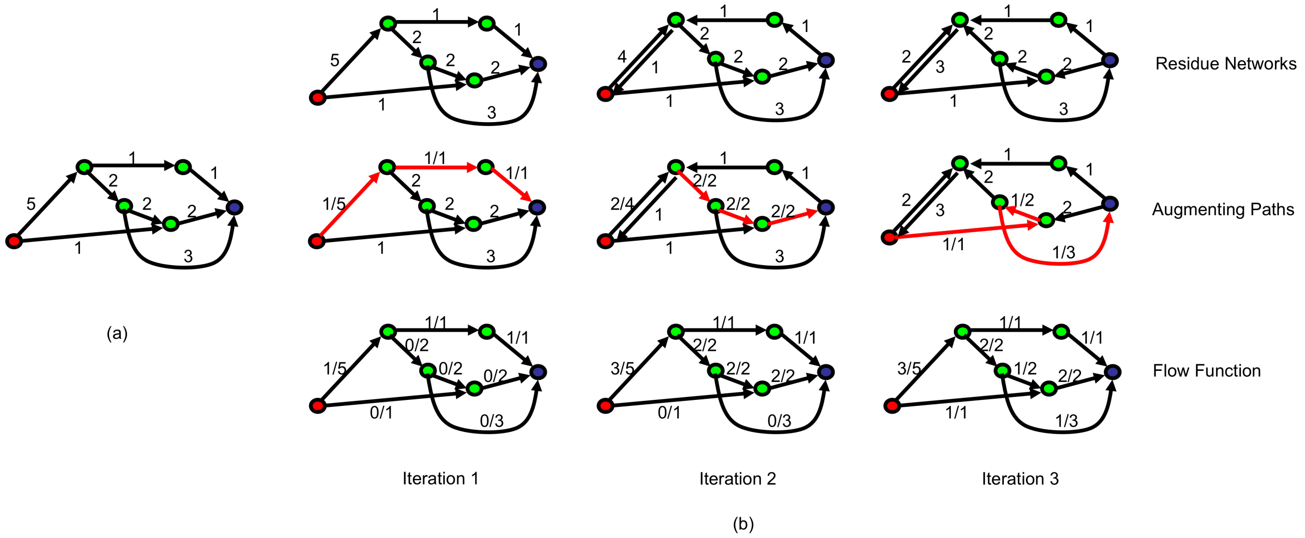

Figure 5: (a) A network. (b) Illustrating the augmenting path algoroithms on the network in (a).

5 Maximum Flow Min-Cut Theorem

An $s-t$ **cut** (or a **source-sink cut**) $C(S,T)$ in a network $G(V,E)$ is a partition of V into dijoint subsets S and T such that $s \in S$ and $t \in T$. The capacity of the $s-t$ cut, denoted by $|C|$ is:

$$|C| = \sum_{\forall e(v,w) \in E, v \in S, w \in T} u(e)$$

Figure 6 shows an $s-t$ cut of a network. The capacity of the cut is 4.

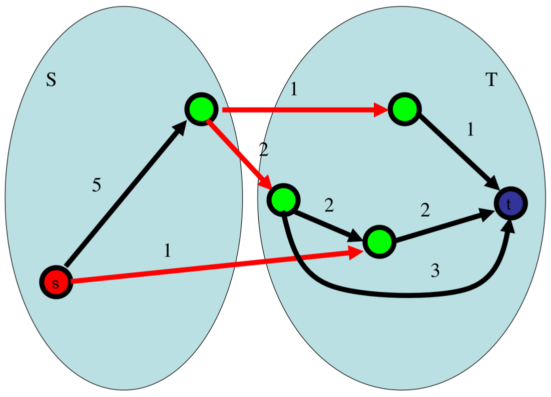

Figure 6: Illustraing the an $s-t$ cut.

A **min-$s-t$ cut** is a an $s-t$ cut with the smallest capacity.

Theorem 1 *The maximum flow is $\leq$ the capacity of an arbitrary $s-t$ cut*

Proof: For ease of explanation, we shall assume that the given network is a **complete digraph**, i.e., every pair of distinct vertices v, w is connected and only connected by a pair of arcs $e(v,w)$ and $e(w,v)$. Note that if the given network is not complete, we can always add a dummy reverse arc with 0 capacity.

Consider an arbitrary $s-t$ cut that partitions V into S and T, with $s \in S$ and $t \in T$.

Thus we have:

$$|f| = \sum_{e(s,w)} f(e) - \sum_{e(w,s)} f(e)$$

Since for any other vertex $v \in S$ and $v \neq s$, the flow through v must be blanaced, i.e., $\sum_{e(v,w)} f(e) - \sum_{e(w,v)} f(e) = 0$, we have:

$$|f| = \sum_{e(s,w)} f(e) - \sum_{e(w,s)} f(e) + \sum_{v \in S - s} \left(\sum_{e(v,w)} f(e) - \sum_{e(w,v)} f(e) \right)$$

$$= \sum_{v \in S} \left(\sum_{e(v,w)} f(e) - \sum_{e(w,v)} f(e) \right)$$

$$= \sum_{v \in S} \left(\left(\sum_{w \in s} f(e(v,w)) + \sum_{w \in T} f(e(v,w)) \right) - \left(\sum_{w \in S} f(e(w,v)) + \sum_{w \in T} f(e(w,v)) \right) \right)$$

$$= \sum_{v\in S}\left(\left(\sum_{w\in S} f(e(v,w)) - \sum_{w\in S} f(e(w,v))\right) + \left(\sum_{w\in T} f(e(v,w)) - \sum_{w\in T} f(e(w,v))\right)\right)$$

$$= \sum_{v\in S}\left(\sum_{w\in S} f(e(v,w)) - \sum_{w\in S} f(e(w,v))\right) + \sum_{v\in S}\left(\sum_{w\in T} f(e(v,w)) - \sum_{w\in T} f(e(w,v))\right)$$

Since every arc has been cited twice, due to symmetry, $\sum_{v\in S}\left(\sum_{w\in S} f(e(v,w)) - \sum_{w\in S} f(e(w,v))\right) = 0$. Thus, we have:

$$\begin{aligned}|f| &= \sum_{v\in S}\left(\sum_{w\in T} f(e(v,w)) - \sum_{w\in T} f(e(w,v))\right)\\ &= \sum_{v\in S}\sum_{w\in T} f(e(v,w)) - \sum_{v\in S}\sum_{w\in T} f(e(w,v))\\ &\le \sum_{v\in S}\sum_{w\in T} f(e(v,w))\\ &\le \sum_{v\in S}\sum_{w\in T} u(e(v,w)) = |C(S,T)|\end{aligned}$$

□

From the above theory, for any flow f and any cut C, $|f| \le |C|$. Thus if one can find a flow function f that saturates a cut C, i.e., $|f| = |C|$, then the flow function f must be the maximum flow.

Theorem 2 *The maximum flow is equal to the minimum $s - t$ cut.*

Proof: We will prove that with the augmenting path algorith, we will eventually saturates the min-cut of the network. For ease of argument, we shall assume that there one and only one arc between any pair of vertices. This assumption can be achieved with the following transformation. Let v, w be two vertices, such that there both the arc $e(v,w)$ and $e(w,v)$ are in the network. We will remove $e(w,v)$ by adding a "dummy vertex" d, and two arcs $e(w,d)$ and $e(d,v)$, each with the same capacity as $e(w,v)$.

Let f be a flow function for a network G. Let G' be the residue graph of G with respect to f. Suppose there is no augmenting path from s to t in the residue graph G'. Since all the edges will have a positive capacity in G', G' must be disconnected. Let S be the subset of vertices that are reachable from s, and $T = V - S$. Obviously, $t \in T$, and (S,T) form an $s - t$ cut in G'.

Observe that for $w \in S$ and $v \in T$:

- If (w,v) is in G, then $0 = u'(w,v) = u(w,v) - f(e(w,v))$, i.e., $u(w,v) = f(e(w,v))$ is saturated.
- If (w,v) is not in G, then its reverse arc (v,w) must be in G, and $0 = u'(w,v) = f(e(v,w))$, i.e., $f(e(v,w)) = 0$.

Thus the flow function f saturates every arc from S to T, and therefore the $s - t$ cut. Therefore f must be maximum flow and the cut $C(S,T)$ must be the min-cut. □