Flow and Matching

CS218, Spring 2018

Outline

• Flow networks

• Max flow (Ford-Fulkerson, Edmonds-Karp)

• Application: maximum bipartite matching
Flow Networks

• A flow network is a digraph $G = (V, E)$ such that each edge $(u, v) \in E$ has a capacity $c(u, v) \geq 0$
• $G$ has two distinguished vertices: a source $s$ and a sink $t$ and self-loops are not allowed
• If $(u, v) \not\in E$ then $c(u, v) = 0$
• If $(u, v) \in E$ then $(v, u) \not\in E$ (restriction can be lifted)
• We assume each vertex $v$ in $V$ is on some path from $s$ to $t$, which implies that the graph is connected, that is $|E| \geq |V| - 1$
Flow Networks

- A flow in $G$ is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties:
  1. **Capacity constraint:** $\forall u, v \in V, \quad 0 \leq f(u, v) \leq c(u, v)$
  2. **Flow conservation:** $\forall u \in V - \{s, t\}, \quad \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$

- The non-negative quantity $f(u, v)$ is called the flow from vertex $u$ to vertex $v$.
- If $(u, v) \notin E$ then $f(u, v) = 0$.

Maximum flow problem

- The value of the flow is

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

that is, the total flow out of the source minus the total flow into the source (typically, the flow into the source is zero).

- **Maximum flow problem:** given a flow network $G$ with source $s$ and sink $t$, find a flow of maximum value.
Example

Figure 26.1 (a) A flow network $G = (V, E)$ for the Lucky Pack Company’s trucking problem. The Vancouver factory is the source $s$, and the Winnipeg warehouse is the sink $t$. The company ships packs through intermediate cities, but only $c(u, v)$ crates per day can go from city $u$ to city $v$. Each edge is labeled with its capacity. (b) A flow $f$ in $G$ with value $|f| = 19$. Each edge $(u, v)$ is labeled by $f(u, v)/c(u, v)$. The slash notation merely separates the flow and capacity; it does not indicate division.

Antiparallel Edges

Converting a network with antiparallel edges (a) to an equivalent one with no antiparallel edges (b).
One source, one sink

Ford-Fulkerson Method
Ford-Fulkerson Method

- The *Ford-Fulkerson* method is a framework into which one can implement more than one algorithm.

- We need to define
  - residual networks
  - augmenting paths
  - cuts

Ford-Fulkerson Method

- Start with a flow function $f(u,v) = 0$ on all pairs of vertices of $G$.
- Improve the flow iteratively by finding (at each iteration) an *augmenting path* (path from $s$ to $t$) over which the flow (and thus $f$) can be increased.
- When no augmenting paths can be found, we are done.
Ford-Fulkerson Method

Ford-Fulkerson-Method \((G, s, t)\)

1. initialize flow \(f\) to 0
2. while there exists an augmenting path \(p\) do
3. augment flow \(f\) along \(p\)
4. return \(f\)

We need to show that

- Augmentation is well-defined
- The process of successive augmentations will terminate in a finite number of steps (termination and time complexity)
- The flow returned is a maximal flow (optimality)
Residual Network

- Let $G = (V, E)$ be a flow network with source $s$, sink $t$ and capacity $c$
- Let $f$ be a flow in $G$ and let $u, v$ be vertices in $V$
- The residual capacity of $(u,v)$ is
  \[
  c_f(u,v) = \begin{cases} 
  c(u,v) - f(u,v) & \text{if } (u,v) \in E \\
  f(v,u) & \text{if } (v,u) \in E \\
  0 & \text{otherwise}
  \end{cases}
  \]
- The quantity $c_f(u,v)$ is the amount of additional flow that can be pushed from $u$ to $v$ before exceeding the capacity $c(u,v)$

Residual Network

- The residual network of $G$ induced by $f$ is $G_f(V, E_f)$, where
  \[
  E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}
  \]
- Each edge of the residual network, called residual edge, can admit a strictly positive flow
- Fact: Residual edges are either edges in $E$ or their reversals and thus $|E_f| \leq 2 |E|$
Residual Network

- Graph $G_f$ is similar to a flow network with capacities $c_f$ but it does not satisfy our definition because it contains antiparallel edges.
- Given a flow $f$ in $G$ and a flow $f'$ in $G_f$, we define the augmentation of $f$ by $f'$ as follows:

$$ (f \uparrow f')(u,v) = \begin{cases} 
  f(u,v) + f'(u,v) - f'(v,u) & \text{if } (u,v) \in E \\
  0 & \text{otherwise}
\end{cases} $$

- Increasing $f'$ on $(v,u)$ means decreasing the flow $f$ on $(u,v)$.

Example

![Diagram of residual networks](image)

Figure 26.4: (a) The flow network $G$ and flow $f$ of Figure 26.1(b). (b) The residual network $G_f$ with augmenting path $p$ shaded; its residual capacity is $c_f(p) = c_f(v_2, v_3) = 4$. Edges with residual capacity equal to 0, such as $(v_1, v_3)$, are not shown; a convention we follow in the remainder of this section. (c) The flow in $G$ that results from augmenting along path $p$ by its residual capacity 4. Edges carrying no flow, such as $(v_1, v_2)$, are labeled only by their capacity, another convention we follow throughout. (d) The residual network induced by the flow in (c).
Residual Networks

**Lemma.** Let $G = (E,V)$ be a flow network with source $s$, sink $t$, capacity $c$ and a flow $f$. Let $G_f$ be the residual network of $G$ induced by $f$, let $f'$ be a flow in $G_f$. Then the function $f \uparrow f'$ defined previously is a flow in $G$ with value

$$|f \uparrow f'| = |f| + |f'|$$

**Proof.** Omitted (lack of time)

Augmenting Paths

- Let $G = (V, E)$ be a flow network with a flow $f$
- An augmenting path $p$ is a simple path from $s$ to $t$ in the residual network $G_f$
- The residual capacity of the augmenting path $p$ is the maximum amount by which we can increase the flow on each edge in $p$
  $$c_f(p) = \min\{c_f(u, v) : (u,v) \text{ is on } p\}$$
Augmenting Paths

**Lemma.** Let $G = (V, E)$ be a flow network, $f$ a flow in $G$, and $p$ an augmenting path in $G_f$. Define $f_p : V \times V \to \mathbb{R}$ as follows

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

Then $f_p$ is a flow in $G_f$ with value $|f_p| = c_f(p) > 0$.

**Proof:** exercise.

---

Augmenting Paths

**Lemma.** Let $G = (V, E)$ be a flow network, $f$ a flow in $G$, and $p$ an augmenting path in $G_f$. Let $f_p$ be defined as in the previous Lemma, and suppose we augment $f$ by $f_p$. Then the function $f \uparrow f_p$ is a flow in $G$ with value

$$|f \uparrow f_p| = |f| + |f_p| > |f|$$

**Proof:** immediate from previous result.
Augmenting Paths

- Now we have a way, by the Ford-Fulkerson method, to construct a maximum flow in a flow network.
- Keep augmenting a starter flow until there is no augmenting path left.
- This is reasonable (if the scheme actually delivers a maximum flow) if all the capacities are integers.
- If the capacities are real numbers one may run into problems with the convergence of this scheme.

\[ f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \]

**Def.:** A cut \((S, T)\) of a flow network \(G = (V, E)\) is a partition of \(V\) into \(S\) and \(T = V - S\) s.t. \(s \in S\) and \(t \in T\).

**Def.:** If \(f\) is a flow and \((S, T)\) is a cut, the net flow \(f(S, T)\) across the cut \((S, T)\) is

\[ f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \]

**Def.:** The capacity of the cut \((S, T)\) is

\[ c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v) \]

Note that the capacity across the cut is computed only from edges \(S \rightarrow T\).

- A minimum cut of a network is a cut whose capacity is minimum over all cuts of the network.
Example

A cut \((S, T)\) where \(S = \{s, v_1, v_2\}\) and \(T = \{v_3, v_4, t\}\)
The flow across \(S, T\) is \(f(S, T) = 19\) and the capacity is \(c(S, T) = 26\)

Cuts

**Lemma.** Let \(f\) be a flow in a network \(G = (V, E)\), with source \(s\) and sink \(t\). Let \((S, T)\) be any cut of \(G\). Then the net flow across \((S, T)\) is \(f(S, T) = |f|\).

**Proof.** We can rewrite the flow-conservation condition for any node \(u \in V - \{s, t\}\) as follows

\[
\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = 0 \quad (1)
\]

Taking the def of \(|f|\) and adding the lhs of (1) summed over all vertices in \(S - \{s\}\) (which sums to zero), gives
Proof (contd)

\[
|f| = \sum_{v \in V'} f(s,v) - \sum_{v \in V'} f(v,s) + \sum_{u \in S - \{x\}} \left( \sum_{v \in V'} f(u,v) - \sum_{v \in V'} f(v,u) \right).
\]

\[
= \sum_{v \in V'} f(s,v) - \sum_{v \in V'} f(v,s) + \sum_{u \in S - \{x\}} f(u,v) - \sum_{u \in S - \{x\}} f(u,v).
\]

\[
= \sum_{v \in V'} \sum_{u \in S - \{x\}} f(u,v) - \sum_{v \in V'} \sum_{u \in S - \{x\}} f(v,u).
\]

Because \( V = S \cup T \) and \( S \cap T = \emptyset \), we can split each summation over \( V \) into summations over \( S \) and \( T \) to obtain

Proof (contd)

\[
|f| = \sum_{v \in S} \sum_{u \in S} f(u,v) + \sum_{v \in S} \sum_{u \in S} f(u,v) - \sum_{v \in S} \sum_{u \in S} f(v,u) - \sum_{v \in T} \sum_{u \in S} f(v,u).
\]

\[
= \sum_{v \in T} \sum_{u \in S} f(u,v) - \sum_{v \in T} \sum_{u \in S} f(v,u) + \left( \sum_{v \in S} \sum_{u \in S} f(u,v) - \sum_{v \in S} \sum_{u \in S} f(v,u) \right).
\]

The two summations within the parenthesis are actually the same, since for all vertices \( x, y \in V \), the term \( f(x, y) \) appears once in each summation. Hence, these summations cancel, and we have

\[
|f| = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) = f(S,T).
\]
Cuts

**Corollary.** The value of any flow $f$ in a flow network $G$ is bounded above by the capacity of any cut in $G$.

**Proof.** Let $(S, T)$ be any cut of $G$, $f$ any flow.

$$|f| = f(S, T)$$

$$= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

$$\leq \sum_{u \in S} \sum_{v \in T} f(u, v)$$

$$\leq \sum_{u \in S} \sum_{v \in T} c(u, v)$$

$$= c(S, T)$$

Max-flow min-cut theorem

- An immediate observation is that the maximum flow must be bounded by the capacity of a minimum cut

- This leads to the conjecture, possibly backed up by an algorithm and a proof, that the two quantities are the same
Max-flow min-cut theorem

**Theorem.** If \( f \) is a flow in a flow network \( G = (V, E) \) with source \( s \) and sink \( t \), the following three conditions are equivalent

1. \( f \) is a maximum flow in \( G \)
2. the residual network \( G_f \) contains no augmenting paths
3. \( |f| = c(S, T) \) for some cut \((S, T)\) of \( G \)

Max-flow min-cut theorem (proof)

(1) \( \rightarrow \) (2) By contradiction. Suppose that there is a max flow \( f \) in \( G \), but \( G_f \) has an augmenting path \( p \). Then, if we augment flow \( f \) by \( f_p \) (where \( f_p \) is defined in Slide 25), we obtain a flow in \( G \) with value strictly greater than \( |f| \), contradicting the assumption that \( f \) is a max flow.
Max-flow min-cut theorem (proof)

(2) → (3) Suppose $G_f$ has no augmenting path, i.e., there is no path from $s$ to $t$ in $G_f$. Define

$$S = \{ v \in V : \text{there exists path from } s \text{ to } v \text{ in } G_f \}$$

and $T = V - S$. The partition $(S,T)$ is a cut: we have $s$ in $S$, and $t$ not in $S$ because there is not path from $s$ to $t$ in $G_f$.

(2) → (3) continued.

Now consider a pair of vertices $u \in S$ and $v \in T$. If $(u,v) \in E$, we must have $f(u,v) = c(u,v)$, since otherwise $(u,v) \in E_f$, which would place $v$ in $S$.

If $(v,u) \in E$, we must have $f(v,u) = 0$, because otherwise $c_f(u,v) = f(v,u) > 0$ and we would have $(u,v) \in E_f$, which would place $v$ in $S$. If neither $(u,v)$ nor $(v,u)$ is in $E$, then $f(u,v) = f(v,u) = 0$. We have

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{v \in T} \sum_{u \in S} f(v,u)$$

$$= \sum_{u \in S} \sum_{v \in T} c(u,v) - \sum_{v \in T} \sum_{u \in S} 0$$

$$= c(S,T). \text{ Conclusion from } |f| = f(S,T) \text{ (slide 30)}.$$
Max-flow min-cut theorem (proof)

(3) → (1) By the Corollary, $|f| \leq c(S, T)$ for all cuts $(S, T)$. The condition $|f| = c(S, T)$ thus implies that $f$ must be a maximum flow.

Basic Ford-Fulkerson algorithm

We are now ready to state a more detailed variant of Ford-Fulkerson.

```plaintext
FORD-FULKERSON(G, s, t)
1  for each edge $(u, v) \in G.E$
2      $(u, v).f = 0$
3  while there exists a path $p$ from $s$ to $t$ in the residual network $G_f$
4      $c_f(p) = \min \{c_f(u, v) : (u, v) \text{ is in } p\}$
5      for each edge $(u, v)$ in $p$
6          if $(u, v) \in E$
7              $(u, v).f = (u, v).f + c_f(p)$
8          else $(v, u).f = (v, u).f - c_f(p)$
```
Residual network $G_f$

\[ f = f \uparrow f_p \]

Residual network $G_f$

\[ f = f \uparrow f_p \]

Max flow
Time-complexity analysis

• Lines 1-2 take $\Theta(|E|) = \Theta(m)$

• What is the worst case for lines 3-8?

• What if we have “no particular strategy” for how we choose the augmenting path, but assume that all capacities have integer values?

• Let $f^*$ denote the maximum flow found by the algorithm, with $|f^*|$ denoting its (integer) value

Time-complexity analysis

• It is conceivably that each pass of the while loop (lines 4-8) will increase the value of the flow by a single unit - but not less

• Since finding an augmenting path (or finding that one does not exists) might require examining all the edges of $G_f$, the total cost of the loop is $O(ml|f^*|) \ [\text{recall that the number of edges in } G_f \text{ is never more than double that of } G]$
Problem 1: the bound is tight

The example below shows that arbitrary strategies for finding the augmenting path should be avoided.

![flow network diagram](image)

Figure 26.6 (a) A flow network for which Ford-Fulkerson can take \( \Theta(|E|/f^*) \) time, where \( f^* \) is a maximum flow, shown here with \( |f^*| = 2,000,000 \). An augmenting path with residual capacity 1 is shown. (b) The resulting residual network. Another augmenting path with residual capacity 1 is shown. (c) The resulting residual network.

Problem 2: convergence

- In case the capacities are irrational, the process may not even produce a max flow after a finite time.
- Computers cannot explicitly represent irrational numbers - floating point numbers ARE rational - but other factors (floating point error and drift) may be just as bad.
Edmonds-Karp

Edmonds-Karp Heuristics

• Two heuristic by Edmonds and Karp [1972]
• The first
  *Always augment by a path of maximum residual capacity*
• The second
  *Always augment by a path of minimum length in the residual network* [where each edge had weight 1 → use BFS]
Edmonds-Karp Heuristics

- Under the assumption that the capacities are integers, the first heuristic requires $O(m \log |f^*|)$ augmenting steps, where $f^*$ is the max flow.
  
  **Proof**: omitted.

- The second does not require the restriction and results in a $O(nm^2)$ time complexity, thus independent of the size of the max flow.

- **Def**: Let $\delta_f(u,v)$ be the shortest-path distance from $u$ to $v$ in $G_f$ where each edge has unit distance (weight of 1).

Second Edmonds-Karp Heuristics

**Lemma.** If the second E-K heuristic is run on a flow network $G = (V, E)$ with source $s$ and sink $t$, then for all vertices $v \in V - \{s, t\}$, the shortest path distance $\delta_f(s,v)$ in the residual network $G_f$ increases monotonically with each flow augmentation.
Second Edmonds-Karp Heuristics

Proof: Suppose $\exists v \in V - \{s, t\}$ for which a flow augmentation cause the shortest path distance from $s$ to $v$ to decrease. Let $f$ be the flow just before the first augmentation that decreases some shortest path distance, $f'$ the flow just afterward. Let $v$ be the vertex with the minimum $\delta_f(s, v)$ whose distance was decreased by the augmentation, that is $\delta_f(s, v) < \delta_f(s, v)$.

Let $p = s \rightarrow u \rightarrow v$ be a shortest path from $s$ to $v$ in $G_{f''}$, so that $(u, v) \in E_f$, and $\delta_f''(s, u) = \delta_f'(s, v) - 1$. (1)

Because of how we chose $v$, the distance label of $u$ did not decrease, that is $\delta_f'(s, u) \geq \delta_f(s, u)$. (2)

Claim: $(u, v) \notin E_f$. Proof: Assume otherwise. Then

$$\delta_f'(s, v) \leq \delta_f'(s, u) + 1 \leq \delta_f(s, u) + 1 = \delta_f(s, v),$$

which contradicts the assumption that $\delta_f'(s, v) < \delta_f(s, v)$. (3)
Second Edmonds-Karp Heuristics

To have \((u,v) \in E_f'\), and \((u,v) \notin E_f\) the augmentation must have increased the flow from \(v\) to \(u\). Since the Edmonds-Karp always augments the flow along shortest paths, the shortest path from \(s\) to \(u\) in \(G_f\) has \((v, u)\) as its last edge. Therefore

\[
\delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_f(s, u) - 1 = \delta_f(s, v) - 2
\]

Which contradicts our assumption that \(\delta_f(s, v) < \delta_f(s, v)\). We conclude that such vertex \(v\) exists is incorrect.

---

**Second Edmonds-Karp Heuristics**

**Theorem.** If the second E-K heuristic is run on a flow network \(G = (V, E)\) with source \(s\) and sink \(t\), then the total number of flow augmentations performed is \(O(nm)\).
Proof

**Def.** An edge \((u, v)\) in a residual network \(G\) is *critical* on an augmenting path \(p\) if the residual capacity of \(p\) is the residual capacity of \((u, v): c_f(p) = c_f(u, v)\).

**Proof.** After augmentation of the flow along an augmenting path, any critical edge on the path disappears from the residual network. Also, at least one edge on each augmenting path must be critical. We will show that each of the \(|E|\) edges of \(G\) can become critical at most \(|V|/2\) times. Let \((u, v) \in E\).

Since augmenting paths are shortest paths by construction, when \((u, v)\) is critical for the first time, we have
\[
\delta_f(s, v) = \delta_f(s, u) + 1.
\]

**Proof (continued)**

Once the flow is augmented, \((u, v)\) disappears from the residual network. It cannot reappear on an augmenting path until after the flow from \(u\) to \(v\) is decreased, which happens only if \((v, u)\) is on an augmenting path. Let \(f'\) be the flow in \(G\) when this event occurs. Then
\[
\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2.
\]
Between two successive times of criticality, the distance from \(u\) to the source has increased by at least 2, being initially at least 0.

The intermediate vertices on a shortest path from \(s\) to \(u\) cannot contain \(s, u\) or \(t\) since \((u, v)\) on the critical path implies \(u \neq t\).

Thus, until \(u\) becomes unreachable from \(s\), if ever, its distance from \(s\) is, at most \(|V| - 2\).
Proof (continued)

After the first time \((u,v)\) becomes critical, \((u,v)\) can become critical at most \((|V| - 2)/2 = |V|/2 - 1\) times more, for a total of \(|V|/2\) time. Since \(O(|E|)\) pairs of vertices can have an edge between them in a residual graph, the total number of critical edges during the entire execution of the algorithm is \(O(|V||E|) = O(nm)\). Each augmentation has, by definition, at least one critical edge. The bound on the number of augmentations follows.

Conclusions

- Using BFS to find the shortest paths we obtain a time complexity for Edmonds-Karp of \(O(nm^2)\)
- FYI: Push-relabel algorithms achieve \(O(n^3)\) [see textbook]
- Even faster algorithms have been invented
Bipartite matching

Maximum Bipartite Matching

• **Def.** We say graph $G=(V,E)$ is *bipartite* if $V$ can be partitioned into two disjoint sets $L$ and $R$ such that $(u,v) \in E \Rightarrow (u \in L \text{ and } v \in R) \text{ or } (u \in R \text{ and } v \in L)$.

• A *matching* is a subset $M \subseteq E$ such that for all $v \in V$ at most one edge of $M$ is incident on $v$.

• A node $v \in V$ is *matched* by the matching $M$ if some edge in $M$ is incident on $v$, otherwise $v$ is *unmatched*.

• A *maximum matching* is a matching of maximum cardinality, i.e., a matching $M$ such that for any other matching $M'$, $|M| \geq |M'|$. 
Example

The left matching is not maximal; the right one is.

Maximum Bipartite Matching

- We first transform a bipartite matching problem into a maximal flow problem

- Then we will apply the Ford-Fulkerson method to produce a maximum matching in an undirected bipartite graph $G$
Maximum Bipartite Matching

Given $G = (V, E)$, we construct a flow network $G' = (V', E')$.
Add new vertices $s$ and $t$, $V' = V \cup \{s, t\}$,
$$E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, (u, v) \in E\} \cup \{(v, t) : v \in R\}$$
Each edge is assigned unit capacity.
Maximum Bipartite Matching

**Def.** A flow $f$ on a flow network $G = (V, E)$ is *integer-valued* if $f(u,v)$ is an integer for all $(u,v) \in V \times V$.

**Lemma.** Let $G = (V, E)$ be a bipartite graph with partition $V = L \cup R$, and let $G' = (V', E')$ be its corresponding flow network. If $M$ is a matching in $G$, then there is an integer-valued flow $f$ in $G'$ with value $|f| = |M|$. Conversely, if $f$ is an integer-values flow in $G'$, then there is a matching $M$ in $G$ with cardinality $|M| = |f|$.

**Proof.** 1) Given be a matching $M$, construct integer-valued flow $f$ as follows: if $(u,v) \in M$, set $f(s, u) = f(u, v) = f(v, t) = 1$. For all other edges in $E'$, $f(u, v) = 0$. Easy to verify that $f$ satisfies capacity constraints and flow conservation. All the paths are of the form $s \rightarrow u \rightarrow v \rightarrow t$, carrying one unit of flow, thus the flow across the cut $(L,R)$ is equal to $|M|$, thus $|f| = |M|$. 
Maximum Bipartite Matching

Proof. 2) Given integer-valued flow \( f \), produce matching \( M \).
Let \( M = \{(u, v) : u \in L, v \in R, \text{ and } f(u, v) > 0\} \).
Each vertex \( u \) in \( L \) has exactly one entering edge \((s, u)\) with capacity one, thus it has at most one unit of flow entering, and by flow conservation, leaving \( u \). Since \( f \) is integer-valued, for each \( u \) in \( L \), one unit of flow can enter on at most one edge and can leave on at most one edge. Thus, one unit of flow enters \( u \) if and only if there is exactly one vertex \( v \) in \( R \) such that \( f(u, v) = 1 \) and at most one edge leaving each \( u \) carries positive flow. A symmetric arguments applies to each \( v \). Thus \( M \) is matching.

Maximum Bipartite Matching

Proof (2 continued).

To show that \(|M| = |f|\), observe that every matched vertex \( u \) in \( L \), we have \( f(s, u) = 1 \) and for every edge \((u, v)\) in \( E-M \), we have \( f(u, v) = 0 \). Consequently, \( f(L \cup \{s\}, R \cup \{t\}) \), the net flow across the cut \((L \cup \{s\}, R \cup \{t\})\) is equal to \(|M|\). Applying the Lemma on Slide 31, \(|f| = f(L \cup \{s\}, R \cup \{t\}) = |M|\).
Matching

**Theorem (Integrality theorem).** If the capacity function $c$ takes only integral values, then the maximum flow $f$ produced by Ford-Fulkerson has the property the $|f|$ is integer valued. Moreover, for all vertices $u$ and $v$, the value of $f(u, v)$ is an integer.

**Proof.** By induction on the number of iterations. The initial flow $f$ satisfies the integrality properties since it vanishes identically. The first augmentation leads to an augmenting flow with integer values, so the sum is an integer-valued flow. By induction, the conclusion follows (Exercise).

Matching

**Corollary.** The cardinality of a maximum matching $M$ in a bipartite graph $G$ is the value of a maximum flow $f$ in the corresponding flow network $G'$.

**Proof.** Assume $M$ is a maximum matching in $G$, and that the corresponding flow $f$ in $G'$ is not maximum. Then there is a maximum flow $f'$ in $G'$ s.t. $|f'| > |f|$. Since the capacities in $G'$ are integer valued, the previous theorem guarantees that $f'$ is integer valued. Thus $f'$ corresponds to a matching $M'$ in $G$ with cardinality $|M'| = |f'| > |f| = |M|$. Contradiction. In a similar manner we can show that if $f$ is a max flow on $G'$, then its corresponding matching is a max matching on $G$. 
Time complexity

Note that any matching $M$ in $G$ has cardinality at most $\min(|L|, |R|) = O(|V|)$. The value of the maximum flow $f$ in $G'$ is thus $O(|V|)$. The first analysis of Ford-Fulkerson gave a time bound of $O(m|f^*|) = O(nm)$ for the construction of the flow and thus the matching.

FYI: Fastest known algorithm for bipartite matching is by Hopcroft and Karp $O(n^{5/2}m)$

Summary

<table>
<thead>
<tr>
<th></th>
<th>Iterations</th>
<th>Cost each iteration</th>
<th>Overall time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford-Fulkerson</td>
<td>$</td>
<td>f^*</td>
<td>$</td>
</tr>
<tr>
<td>Edmonds-Karp 1</td>
<td>$m \log</td>
<td>f^*</td>
<td>$</td>
</tr>
<tr>
<td>Edmonds-Karp 2</td>
<td>$nm$</td>
<td>$m$</td>
<td>$nm^2$</td>
</tr>
<tr>
<td>Matching (using FF)</td>
<td>$</td>
<td>f^*</td>
<td>= n$</td>
</tr>
</tbody>
</table>
Reading assignment

• Chapter 26, “Maximum Flow”