Lecture 7: Graph algorithms III

- Network flows
- Bipartite matchings
Sources for this lecture

- The presentation of network flow theory in the textbook Dasgupta et al., Section 7, builds on general Linear Programming theory, which we have not covered.
- Hence, the presentation here follows P. Kaski’s lectures at the former course T-79.5203 Graph Theory.
- There the textbook was:
- Two general references on network flows are:
7.1 Network flows

- Let $G$ be a directed graph with a nonnegative capacity $c(e)$ assigned to each edge $e \in E(G)$, and let $s, t \in V(G)$ be two distinct vertices.

- The quadruple $N = (G, c, s, t)$ is a flow network with source $s$ and sink $t$. 
Flows

- A flow on a flow network $N = (G, c, s, t)$ is a mapping $f : E(G) \rightarrow \mathbb{R}$ that satisfies conditions:

  (F1) $0 \leq f(e) \leq c(e)$ for each edge $e \in E(G)$; and

  (F2) for each vertex $v \neq s, t$, we have that

  $$\sum_{e : e^+ = v} f(e) = \sum_{e : e^- = v} f(e),$$

  where $e^-$ and $e^+$ denote the start and end vertex of $e$.

- Condition (F1) requires that the flow is feasible, i.e., is nonnegative and does not exceed the capacity of an edge.

- Condition (F2) requires flow conservation, i.e., that the flow entering any vertex $v \neq s, t$ is equal to the flow leaving $v$. 

Value of a flow, maximum flow

- The value \( w(f) \) of a flow \( f \) on \( N \) is the net flow leaving the source \( s \) (equivalently, the net flow entering the sink), i.e.,
  \[
  w(f) := \sum_{e^- = s} f(e) - \sum_{e^+ = s} f(e) = \sum_{e^+ = t} f(e) - \sum_{e^- = t} f(e).
  \]

- A flow \( f \) is a maximum flow if \( w(f) \geq w(f') \) for all flows \( f' \) on \( N \).

- The maximum flow problem is to determine a maximum flow for a given flow network.
Cut, capacity of a cut

- A cut of a flow network $N = (G, c, s, t)$ is a partition of the vertex set $V(G)$ into two disjoint sets $S, T$ such that $s \in S$ and $t \in T$.
- The capacity of a cut $(S, T)$ is
  
  $$c(S, T) := \sum_{\begin{subarray}{c} e^- \in S, \\ e^+ \in T \end{subarray}} c(e).$$

- Lemma 1 For any flow $f$ and any cut $(S, T)$, we have
  
  $$w(f) = \sum_{\begin{subarray}{c} e^- \in S, \\ e^+ \in T \end{subarray}} f(e) - \sum_{\begin{subarray}{c} e^- \in T, \\ e^+ \in S \end{subarray}} f(e) \leq c(S, T).$$
Proof. By (F2) we have

\[ w(f) = \sum_{e^{-}=s} f(e) - \sum_{e^{+}=s} f(e) = \sum_{\forall v \in S} \left( \sum_{e^{-}=v} f(e) - \sum_{e^{+}=v} f(e) \right). \]

The last sum can be partitioned into sums over edges with (i) both endpoints in \( S \) and (ii) exactly one endpoint in \( S \):

\[ w(f) = \underbrace{\sum_{e^{-} \in S, \ e^{+} \in S} f(e)}_{(i)} - \underbrace{\sum_{e^{-} \in S, \ e^{+} \notin S} f(e)}_{(ii)} + \underbrace{\sum_{e^{-} \in S, \ e^{+} \in T} f(e)}_{(i)} - \underbrace{\sum_{e^{-} \in T, \ e^{+} \in S} f(e)}_{(ii)} \]

\[ = \sum_{e^{-} \in S, \ e^{+} \in T} f(e) - \sum_{e^{-} \in T, \ e^{+} \in S} f(e) \]

\[ \leq c(S, T), \]

where the last inequality follows from (F1).
Augmenting paths

- Let $G$ be a digraph and let $u, v \in V(G)$.
- An undirected path $P$ from $u$ to $v$ is an acyclic subgraph of $G$ such that the underlying graph of $P$ is a path with endpoints $u, v$.
- An edge in $P$ is either a forward edge (directed from $u$ to $v$) or a backward edge (directed from $v$ to $u$).
- Let $N = (G, c, s, t)$ be a flow network with flow $f$. An undirected path $P$ from $u$ to $v$ is ($f$-) incrementing if
  - (I1) $f(e) < c(e)$ for every forward edge $e \in E(P)$; and
  - (I2) $f(e) > 0$ for every backward edge $e \in E(P)$.
- An $f$-incrementing path from $s$ to $t$ is an ($f$-) augmenting path.
Let $P$ be an $f$-augmenting path.

The residual capacity of a forward edge $e \in E(P)$ is $c(e) - f(e)$; the residual capacity of a backward edge $e \in E(P)$ is $f(e)$.

The residual capacity of $P$ is the minimum of the residual capacities of the edges in $P$.

Let $d$ be the residual capacity of $P$. Clearly, $d > 0$. Define

$$f'(e) := \begin{cases} 
  f(e) + d & \text{if } e \text{ is a forward edge in } P; \\
  f(e) - d & \text{if } e \text{ is a backward edge in } P; \\
  f(e) & \text{otherwise, i.e., } e \notin E(P).
\end{cases}$$

It is straightforward to check that $f'$ satisfies (F1) and (F2), i.e., is a flow. Moreover, $w(f') = w(f) + d$.

Thus, whenever an $f$-augmenting path exists, there exists a flow $f'$ on $N$ with larger value.
7.2 Three theorems of Ford and Fulkerson (1956)

**Theorem 2 (Augmenting Paths)** A flow $f$ on a flow network $N = (G, c, s, t)$ is a maximum flow if and only if there exists no $f$-augmenting path.

*Proof.*

$(\Rightarrow)$ If there exists an $f$-augmenting path with residual capacity $d$, then $f$ is not a maximum flow because the flow $f'$ defined earlier satisfies $w(f') = w(f) + d > w(f)$.

$(\Leftarrow)$ Let $S$ be the set of all vertices reachable from $s$ by an $f$-incrementing path. Since there exists no $f$-augmenting path, $t \notin S$. Put $T := V(G) - S$. Clearly, $(S, T)$ is a cut of $N$. We show that $w(f) = c(S, T)$ to establish that $f$ is a maximum flow.
Proof: (continued) Lemma 1 applied to the cut \((S, T)\) gives:

\[
\begin{align*}
w(f) &= \sum_{e^- \in S, \ e^+ \in T} f(e) - \sum_{e^- \in T, \ e^+ \in S} f(e) \leq \sum_{e^- \in S, \ e^+ \in T} c(e) = c(S, T).
\end{align*}
\]

Suppose that \(w(f) < c(S, T)\). Then, there exists an edge \(e\) that satisfies either (i) \(e^- \in S, \ e^+ \in T\), and \(f(e) < c(e)\); or (ii) \(e^- \in T, \ e^+ \in S\), and \(f(e) > 0\).

Suppose case (i) occurs. By definition of \(S\), \(e^-\) is reachable from \(s\) by an \(f\)-incrementing path \(P\). But then \(e^+\) is reachable from \(s\) by the \(f\)-incrementing path \(P + e\), which contradicts the definition of \(S\). For case (ii) we obtain a similar contradiction, so \(w(f) = c(S, T)\).

By Lemma 1, any flow \(f'\) on \(N\) satisfies \(w(f') \leq c(S, T)\). Thus, \(f\) is a maximum flow.
**Theorem 3 (Max-Flow Min-Cut)** In any flow network, the maximum value of a flow is equal to the minimum capacity of a cut.

*Proof.* Lemma 1 shows that the capacity of any cut \((S, T)\) is an upper bound on the value of a flow. The proof of Theorem 2 shows that the maximum flow on a network reaches this bound for a particular cut \((S, T)\).
Theorem 4 (Integral Flows) Let $N = (G, c, s, t)$ be a flow network in which all capacities $c(e)$ are integers. Then, there exists a maximum flow on $N$ such that all values $f(e)$ are integers.

Proof. Starting from the all zero flow $f_0$, repeatedly augment the flow using an augmenting path until no augmenting path exists (in which case the flow is a maximum flow by Theorem 1). Clearly, the residual capacity in each augmentation is an integer. Thus, there are at most $\sum_e c(e)$ augmentations and the resulting maximum flow is integral.
The Ford-Fulkerson labelling algorithm

- The proof of the augmenting paths theorem suggests a straightforward algorithm for computing maximum flows: starting with the all zero flow, repeatedly augment the flow until no augmenting path exists; output the flow and halt.

- This is essentially the labelling algorithm of Ford and Fulkerson (1957); see [Jun, p. 159–160] for pseudocode.

- There are two problems with the above approach:
  1. For general real-valued edge capacities, the algorithm may not halt at all (and converge to a flow that is not a maximum flow; see e.g. [Ahu, p. 205–206] or [For, p. 21]).
  2. Even when all capacities are integral, the algorithm may require time proportional to $\sum_e c(e)$ with a poor choice of augmenting paths, which is very inefficient.
Example. In the flow network below, $2n$ augmenting steps are required to reach the maximum flow of value $2n$ if we use alternately the augmenting paths $sa, ab, bt$ and $sb, ba, at$. 
Edmonds and Karp (1972) gave a simple modification to the repeated augmentation algorithm (which ensures termination also in the general real-valued case):

In each augmenting step, use a shortest possible augmenting path (i.e. an augmenting path with the least number of edges).

Theorem 5 The Edmonds-Karp algorithm performs at most $O(n(G)e(G))$ augmenting steps.

Proof. See [Jun, Theorem 6.2.1].
Remarks

- A shortest augmenting path can be located in time $O(e(G))$ using BFS.
- Thus, the Edmonds-Karp algorithm computes a maximum flow in time $O(n(G)e(G)^2)$ (assuming that the arithmetic is constant-time).
- More efficient maximum flow algorithms exist, e.g.:
  - Dinic (1970): $O(n(G)^2 e(G))$
    - For unit costs, $O(\min\{n(G)^{2/3}, e(G)^{1/2}\} e(G))$
  - Orlin + King, Rao & Tarjan (2013): $O(n(G)e(G))$
    - For sparse networks, i.e. when $e(G) = O(n(G))$, $O(n(G)^2 / \log n(G))$
7.3 Zero-one flows and bipartite matchings

- A zero-one flow in a flow network $N$ is a flow $f$ with $f(e) \in \{0, 1\}$ for all $e \in E(G)$.
- Zero-one flows occur in many combinatorial applications of flow theory.
- We apply zero-one flows and the max-flow min-cut theorem to the important topic of bipartite matchings.
Matchings in graphs

- Let $G$ be a graph. A matching $M$ in $G$ is a set of nonloop edges with no shared endpoints, i.e. a 1-to-1 pairing among some subset of the vertices.
- Let $M$ be a matching in $G$. A vertex of $G$ is covered by $M$ if it is incident to an edge in $M$; otherwise the vertex is uncovered by $M$.
- A matching in $G$ is maximal if it is not a subset of a matching of larger cardinality. A maximum matching is a matching of maximum size among all matchings in $G$.
- A matching that covers all vertices in $G$ is a perfect matching.
Bipartite matchings

Recall that a graph $H = (V, E)$ is bipartite if the vertex set $V$ can be partitioned into two disjoint sets, $X$ and $Y$, so that every edge in $E$ has one end in $X$ and the other in $Y$.

For $A \subseteq X$, denote by $\Gamma(A) \subseteq Y$ the set of all vertices in $Y$ that have at least one neighbour in $A$.

**Theorem 5 (P. Hall 1935)** A bipartite graph $H = (X \cup Y, E)$ has a matching that covers every vertex in $X$ if and only if for all $A \subseteq X$ it holds that $|\Gamma(A)| \geq |A|$.

The requirement that $|\Gamma(A)| \geq |A|$ for all $A \subseteq X$ is called **Hall’s condition**.
Proof. Necessity of Hall’s condition is immediate. Indeed, if there exists an $A \subseteq X$ with $|\Gamma(A)| < |A|$, then no matching $M \subseteq E$ can cover all vertices in $A$ (and hence, all vertices in $X$) because there are too few vertices in $\Gamma(A)$ to match to vertices in $A$.

To establish sufficiency of Hall’s condition, we employ network flow theory.

First, let us transform the bipartite graph $H$ into a flow network. Construct the following flow network $N = (G, c, s, t)$. Include in $G$ all vertices in $H$; add two new vertices $s, t$. For each edge $\{u, v\}$ in $H$, with $u \in X$ and $v \in Y$, add the directed edge $(u, v)$ to $G$. Furthermore, add all edges of the form $(s, u), (v, t)$ to $G$, where $u \in X$ and $v \in Y$. Each edge has capacity one.
Next, let us relate matchings in $H$ with flows in $N$.

We first claim that $N$ has a flow of value $|X|$ if and only if there exists a matching in $H$ that covers every vertex in $X$:

$(\Leftarrow)$ Let $M$ be a matching that covers every vertex in $X$. Put $f(e) = 1$ for every edge $e \in M$, and put $f(s, u) = 1$ for each covered vertex $u \in X$. Similarly, put $f(v, t) = 1$ for each covered vertex $v \in Y$. For all other edges $e$ in $G$, put $f(e) = 0$. Clearly, $f$ is a flow on $N$ with $w(f) = |X|$.

$(\Rightarrow)$ A flow $f$ on $N$ with $w(f) = |X|$ is clearly a maximum flow on $N$. (Indeed, the cut with $S = \{s\}$ has capacity $|X|$.) By the integral flow theorem, we can assume $f$ to be a zero-one flow. Thus, we obtain a matching $M$ that covers all vertices in $X$ by simply including all edges $e = (u, v)$ with $u \in X$, $v \in Y$ and $f(e) = 1$ into $M$. 
Suppose now that $H$ does not have any matching that covers all vertices in $X$. (That is, a maximum flow on $N$ has value less than $|X|$.) We show that then there is some $A \subseteq X$ for which Hall’s condition fails, i.e. $|\Gamma(A)| < |A|$.

By the max-flow min-cut theorem, there exists a cut $(S, T)$ of $N$ with $c(S, T) < |X|$. We propose to choose $A := X \cap S$.

Firstly, by the construction of $N$,

$$c(S, T) \geq |\Gamma(X \cap S) \cap T| + |X \cap T| + |Y \cap S|.$$ 

Secondly,

$$|\Gamma(X \cap S)| \leq |\Gamma(X \cap S) \cap T| + |Y \cap S|.$$ 

Combining these inequalities, we obtain

$$|X \cap T| \leq c(S, T) - |\Gamma(X \cap S)| < |X| - |\Gamma(X \cap S)|,$$ 

i.e.

$$|\Gamma(X \cap S)| < |X| - |X \cap T| = |X \cap S|.$$ 

Thus, Hall’s condition fails for $A = X \cap S$. 
Algorithms for bipartite matchings

- The given proof of P. Hall’s theorem suggests also a natural approach to *finding* a maximum matching in a bipartite graph $H$:
  1. Construct the corresponding flow network $N$.
  2. By the integral flow theorem $N$ has a zero-one max flow.
  3. Apply any network flow algorithm to determine such.
  4. Read the corresponding matching off the edges of $H$.

- There are also other, more direct approaches to this important special case of network flows. One such algorithm will be discussed at the tutorials.