Lecture notes for Feb 22, 2023
Maximum flows and minimum cuts

Chun-Hung Liu

February 22, 2023

Recall:

- A *network* is a 4-tuple (D, s, t, c), where D is a digraph, s, t are distinct vertices of D and $c : E(D) \to \mathbb{R}_{\geq 0}$. (Here $\mathbb{R}_{\geq 0}$ denotes the set of all nonnegative real numbers.) The vertex s is called the *source* and t is called the *sink*.
- A *flow* in a network (D, s, t, c) is a function $f : E(D) \to \mathbb{R}_{\geq 0}$.
- A flow f is *feasible* if it satisfies
  - (Capacity condition:) $0 \leq f(e) \leq c(e)$ for every $e \in E(D)$, and
  - (Conservation condition:) $\sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e)$ for every $v \in V(D) - \{s, t\}$.
- The *value* of a flow f is $\sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e)$. That is, the value is the "net amount" flowing out from the source. We denote the value of f by $\text{val}(f)$.
- If f is a feasible flow with maximum value, then f is called a *maximum flow*.

The following lemma shows that for every subset S of $V(D)$ containing the source s but not the sink t, the "net amount" of a flow flowing out from S is the same as the value. That is, the value of a flow equals the "net amount" of any such set S. In particular, if we choose $S = V(D) - \{s\}$, then we know $\delta^+(S) = \delta^-(t)$ and $\delta^-(S) = \delta^+(t)$ (except for loops incident with t), so $\text{val}(f) = \sum_{e \in \delta^+(S)} f(e) - \sum_{e \in \delta^-(S)} f(e) = \sum_{e \in \delta^-(t)} f(e) - \sum_{e \in \delta^+(t)} f(e)$.

Lemma 1 *Let (D,s,t,c) be a network, and let f be a feasible flow. For every $S \subseteq V(D)$ with $s \in S$ and $t \notin S$, we have* $\sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) = \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) = \mathrm{val}(f)$.

Proof. Let $W = \sum_{v\in S}(\sum_{e\in\delta^+(v)} f(e) - \sum_{e\in\delta^-(v)} f(e))$. Since f satisfies the conservation condition, $W = (\sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e)) + \sum_{v\in S-\{s\}} 0 = \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) = \mathrm{val}(f)$.

Now we compute W in another way. For every edge e of D whose both ends not in S, it contributes 0 in W. For each edge e of D whose both ends in S, it also contributes 0 in W, since it contributes $f(e)$ for its tail and $-f(e)$ for its head. For each edge e of D in $\delta^+(S)$, it contributes $f(e)$ in W. For each edge e of D in $\delta^-(S)$, it contributes $-f(e)$ in W. Therefore, $W = \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e)$. ■

Definition:

- A *cut* in a network (D,s,t,c) is an ordered partition (S,T) of $V(D)$ of two parts such that $s \in S$ and $t \in T$.
- The *capacity* of a cut (S,T), denoted by $\mathrm{cap}(S)$, is $\sum_{e\in\delta^+(S)} c(e)$. Note that we do not care about $\delta^-(S)$ when we define the capacity of a cut.

The following is called the "weak duality theorem" which shows one side of the Max-Flow-Min-Cut Theorem.

Lemma 2 *Let (D,s,t,c) be a network. Let (S,T) be a cut and let f be a feasible flow. Then* $\mathrm{val}(f) \leq \mathrm{cap}(S)$.

Proof. Recall that $f(e) \geq 0$ for every $e \in E(D)$ by the definition of a flow. By Lemma 1, $\mathrm{val}(f) = \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) \leq \sum_{e\in\delta^+(S)} f(e) \leq \sum_{e\in\delta^+(S)} c(e) = \mathrm{cap}(S)$, where the last inequality follows from the capacity condition. ■

To prove the other side of the Max-Flow-Min-Cut Theorem, we need the following notions.

Definition: Let (D,s,t,c) be a network, and let f be a feasible flow.

- The *residue graph (with respect to f)*, denoted by $R(f)$, is the digraph with vertex-set $V(D)$ constructed by the following procedure:

  - for each edge $e = (x, y)$ of D, if $f(e) > 0$, then create an edge (y, x) in $R(f)$, and
  - for each edge $e = (x, y)$ of D, if $f(e) < c(e)$, then create an edge (x, y) in $R(f)$.

  (Note that if an edge e of D satisfies $0 < f(e) < c(e)$, then e creates 2 edges with different directions in $R(f)$.)

- An *f-augmenting path* is a directed path in $R(f)$ from s to t.
- For an f-augmenting path P and every edge $e = (u, v) \in E(P)$,
  - if $(u, v) \in E(D)$, then we know $f(e) < c(e)$, and we define $\epsilon(e) = c(e) - f(e)$ and call e a *forward edge*, and
  - if $(v, u) \in E(D)$, then we know $f(e) > 0$, and we define $\epsilon(e) = f(e)$ and call e a *backward edge*.
- We define the *residue* of an f-augmenting path P to be $\min_{e \in E(P)} \epsilon(e)$. (Notice that the residue of any f-augmenting path is always positive.)

Intuitively, if we want to send more amount in a flow from s to t along P, then for every forward edge e, we can send more amount (up to $\epsilon(e)$) on this edge without exceeding the capacity on this edge; for every backward edge e, we can pull back some amount (up to $\epsilon(e)$) on this edge without making the amount sending on this edge negative.

Lemma 3 *Let (D, s, t, c) be a network, and let f be a feasible flow. If there exists an f-augmenting path, then f is not a maximum flow.*

Proof. Let ϵ be the residue of P. Define f' to be a function on $E(D)$ such that

- for every forward edge e on P, define $f'(e) = f(e) + \epsilon$,
- for every backward edge e on P, define $f'(e) = f(e) - \epsilon$, and
- for every edge $e \in E(D) - E(P)$, define $f'(e) = f(e)$.

It is straight forward to verify that f' is a feasible flow in (D, s, t, c).

Since s is incident with exactly one edge in P, $\text{val}(f') = \text{val}(f) + \epsilon > \text{val}(f)$. So f is not a maximum flow. ■

We call the flow f' in the proof of Lemma 3 the *flow obtained from f by augmenting on P*.

================================

Ford-Fulkerson Algorithm
Input: A network (D, s, t, c).
Output: A function $f : E(D) \to \mathbb{R}_{\geq 0}$ and a subset S of $V(D)$ with $s \in S$ such that f is a flow with $\text{val}(f) = \text{cap}(S)$.
Procedure:

Step 0: Set f to be the 0-function.

Step 1: Construct $R(f)$.

Step 2: Set $S = \{v \in V(D) :$ there exists a directed path in $R(f)$ from s to $v\}$. If $t \in S$, then find such a directed path P, and set f to be the flow obtained from f by augmenting on P and repeat Step 1; otherwise, output the function f and the set S and stop.

================================

Lemma 4 (Ford, Fulkerson) *Let f and S be the output of the algorithm. Then* $\text{val}(f) = \text{cap}(S)$.

Proof. Let $B = V(D) - S$. Note that $s \in S$, since s is a directed path from s to s. And $t \notin S$, so $t \in B$. So (S, B) is a cut in (D, s, t, c).

Note that in $R(f)$, there exists no edge whose tail in S and head in B, for otherwise we should include the head of this edge into S. That is, for every edge e of D with one end in S and one end in B, we know

- if the tail of e is in S (i.e. $e \in \delta^+(S)$), then $f(e) = c(e)$, and
- if the head of e is in S (i.e. $e \in \delta^-(S)$), then $f(e) = 0$.

Therefore, the "net amount" flowing out from S is $\sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) = \sum_{e\in\delta^+(S)} c(e) - 0 = \text{cap}(S)$. By Lemma 1, $\text{val}(f) = \sum_{e\in\delta^+(S)} f(e) - \sum_{e\in\delta^-(S)} f(e) = \text{cap}(S)$. ■

Theorem 5 (Max-Flow-Min-Cut-Theorem) *Let (D, s, t, c) be a network. Then*

$$\max_f \operatorname{val}(f) = \min_{(S,T)} \operatorname{cap}(S),$$

where the first maximum is over all feasible flows and the second minimum is over all cuts (S, T).

Proof. By Lemma 2, $\max_f \operatorname{val}(f) \leq \min_{(S,T)} \operatorname{cap}(S)$. By Lemma 4, $\max_f \operatorname{val}(f) \geq \min_{(S,T)} \operatorname{cap}(S)$. This proves the theorem. ■

Notice that if the capacity of each edge is an integer, then the residue of each augmenting path is integral and is at least 1. Since $\delta^+(s)$ is finite, if the capacity of each edge is an integer, then Ford-Fulkerson algorithm must stop by repeating Step 1 a finite number of times and output an integral function f. Hence we get the following important theorem.

Theorem 6 *Let (D, s, t, c) be a network such that $c(e) \in \mathbb{Z}$ for every $e \in E(D)$. Then there exists a maximum flow f such that* $\operatorname{val}(f) = \min_{(S,T)} \operatorname{cap}(S)$ *and $f(e) \in \mathbb{Z}$ for every $e \in E(D)$.*

However, it is unclear whether Step 1 in the above procedure will be repeated only a finite number of times when the capacity is not an integral function. So the Ford-Fulkerson algorithm might not stop. Indeed, Ford and Fulkerson provided an example showing that the above procedure will run forever, if we do not choose the augmenting paths carefully. Edmonds and Karp showed that if we always do augmenting on the shortest augmenting path, then the above procedure will stop in finitely number of steps.

================================

Edmonds-Karp Algorithm
Input: A network (D, s, t, c).
Output: A function $f : E(D) \to \mathbb{R}_{\geq 0}$ and a subset S of $V(D)$ with $s \in S$ such that f is a flow with $\operatorname{val}(f) = \operatorname{cap}(S)$.
Procedure:

Step 0: Set f to be the 0-function.

Step 1: Construct $R(f)$.

Step 2: Do BFS on $R(f)$ starting from s to find the set $S = \{v \in V(D) :$ there exists a directed path in $R(f)$ from s to $v\}$. If $t \in S$, then the BFS finds a shortest path P (in terms of the number of edges) in $R(f)$ from s to t, set f to be the flow obtained from f by augmenting on P, and repeat Step 1; otherwise, output f and S and terminate the algorithm.

==============================

Lemma 7 *Let $P_1, P_2, \ldots$ be the paths found in in Step 2, where they are found in the order listed. Then for every $i \geq 1$,*

1. *$|E(P_i)| \leq |E(P_{i+1})|$, and*
2. *for any k with $1 \leq k < i$, if P_i and P_k traverse some edge of D in different direction, then $|E(P_i)| \geq |E(P_k)| + 2$.*

Proof. For each j, let f_j be the flow such that P_j is found in $R(f_j)$.

We first prove Statement 1. Let G_1 be the graph obtained from $P_i \cup P_{i+1}$ by duplicating each edge appearing in both P_i and P_{i+1} and deleting each edge reversed in different directions in P_i and P_{i+1}. It is easy to see that in G_1, s has in-degree 0 and out-degree 2, t has in-degree 2 and out-degree 0, and every other vertex in G_1 has in-degree equal to out-degree. By adding two edges from t to s to G_1, we obtain an Eulerian circuit. By deleting the two added edges from t to s, we obtain two edge-disjoint paths Q_1, Q_2 from s to t in G_1 such that $Q_1 \cup Q_2 = G_1$. Since the edges traversed in different directions in P_i and P_{i+1} are deleted in G_1, all edges of G_1 contained in $R(f_i)$. So both Q_1 and Q_2 are candidates of P_i. Since P_i is chosen to be shortest, $|E(Q_1)| \geq |E(P_i)|$ and $|E(Q_2)| \geq |E(P_i)|$. Then $|E(P_{i+1})| \geq |E(G_1)| - |E(P_i)| = |E(Q_1)| + |E(Q_2)| - |E(P_i)| \geq |E(P_i)|$.

Now we prove Statement 2. By Statement 1, we may assume that for every $k+1 \leq j \leq i-1$, P_i and P_j do not traverse edges of D in different direction. Let G_2 be the graph obtained from $P_k \cup P_i$ by duplicating each edge appearing in both P_k and P_i and deleting each edge reversed in different directions in P_k and P_i. It is easy to see that in G_2, s has in-degree 0 and out-degree 2, t has in-degree 2 and out-degree 0, and every other vertex in G_2 has in-degree equal to out-degree. By adding two edges from t to s to G_2, we obtain an Eulerian circuit. By deleting the two added edges from t to s, we obtain two edge-disjoint paths Q_1', Q_2' from s to t in G_2 such that $Q_1' \cup Q_2' = G_2$. Since the edges traversed in different directions in P_k and P_i are deleted in G_2, and P_i and P_j do not traverse edges in different directions for all $k+1 \leq j \leq i-1$, we know that all edges of G_2 contained in $R(f_k)$. So both Q_1' and Q_2' are candidates of P_k. Since P_k is chosen to be shortest, $|E(Q_1')| \geq |E(P_k)|$ and $|E(Q_2')| \geq |E(P_k)|$. So $|E(G_2)| = |E(Q_1')| + |E(Q_2')| \geq 2|E(P_k)|$. On the other hand, since P_k and P_i traverse some edge in different direction, at least one edge in $P_i \cup P_k$ is deleted when constructing G_2. It implies that $|E(G_2)| = |E(Q_1')| + |E(Q_2')| \leq |E(P_k)| + |E(P_i)| - 2$ since the at least one edge in each P_k and P_i is deleted. So $2|E(P_k)| \leq |E(P_k)| + |E(P_i)| - 2$. That is, $|E(P_i)| \geq |E(P_k)| + 2$. ■

Theorem 8 (Edmonds, Karp) *A maximum flow and a minimum cut can be found in time $O(|V(D)||E(D)|^2)$.*

Proof. The correctness of the algorithm follows the same one as for Ford-Fulkerson algorithm. So it suffices to prove the time complexity.

Let $f_1, f_2, ..., f_k$ (some k) be the flow constructed in the algorithm, appearing in the order listed. For each i, let P_i be the f_i-augmenting path found in $R(f_i)$ in the algorithm, and let e_i be the edge in P_i that defines the number ϵ so that $\text{val}(f_{i+1}) = \text{val}(f_i) + \epsilon$.

Let $W = \{i \geq 1 : |E(P_{i+1})| \geq |E(P_i)| + 1\}$. Note that $|W| \leq |V(D)| - 1$.

If there exist $i_1 < i_2$ such that $|E(P_j)| = |E(P_{i_1})|$ for every $i_1 \leq j \leq i_2$, then by Statement 2 in Lemma 7, for any j_1, j_2 with $i_1 \leq j_1 < j_2 \leq i_2$, P_{j_1} and P_{j_2} do not traverse some edge in different directions, so $e_{j_1} \neq e_{j_2}$. Hence $i_2 - i_1 \leq |E(D)|$ for any $i_1, i_2 \in W$ satisfying $j \notin W$ for every $i_1 < j < i_2$. So $k \leq |V(D)||E(D)|$.

Note that each P_i can be found in time $O(|V(D)| + |E(D)|)$ by BFS and each $R(f_i)$ can be constructed in time $O(|V(D)|+|E(D)|)$. So the entire algorithm runs in time $O(|V(D)||E(D)|(|V(D)| + |E(D)|)) = O(|E(D)|^2|V(D)|)$ (we may assume $|V(D)| \leq |E(D)|$ since we may assume the underlying digraph is connected). ■

The maximum flow problem is one of the central research problems in graph algorithms. The currently most efficient deterministic exact algorithm for finding a maximum flow was provided by King, Rao and Tarjan (1994) with time complexity $O(nm \log_{2+\frac{m}{n \log n}} n)$ and provided by Goldberg and Rao (1998) with time complexity $O(\min\{\sqrt{m}, n^{2/3}\} \cdot m \log(\frac{n^2}{m}) \log c_M)$, where $n = |V(D)|$, $m = |E(D)|$, and $c_M = \max_{e \in E(D)} c(e)$. Recently, Chen, Kyng, Liu, Peng, Gutenberg and Sachdeva (2022) proved that a maximum flow of a network with integral capacity can be found in time $O(m^{1+o(1)} \log c_M)$ with high probability.