Lecture 9. Network Flows.

Dmitri Shmelkin

Math Modeling Competence Center at Moscow Research Center of Huawei
Shmelkin.Dmitri@huawei.com

May 19, 2023

Overview

1. Network Flow Definition
2. Network Flow and Network Cut
3. Ford-Fulkerson Method
4. Edmonds-Karp Algorithm
5. Bipartite Graph Matching

What is a Network?

• In previous lectures we learned graphs, directed and undirected, weighted and non-weighted. In this lecture we deal with directed graphs $G(V, E)$ and we use notation $(u, v), u, v \in V$ for a directed edge (or arrow) that may belong on not belong to E. So we use the matrix presentation of the graph, for simplicity of further notation. In fact, when implementing algorithms, a more convenient adjacency list data is typically used, but we will not need it here.

Definition 1

A (flow) Network is weighted directed graph, $G(V, E, c)$ such that $c : E \to \mathbb{R}_{\geq 0}$ is a *capacity* function.

Examples of networks are numerous and mostly intuitive:

• Pipe network with water in houses and other liquids in different enterprises with capacity of a pipe equal to the surface of its cross section

• Electrical circuits with capacity being the maximal electrical current that an interconnection can carry without damage

• City road network with c being the number of lanes in a directed road

• Communication network, such as IP, c is the bandwidth measured in Mbps

What is a Network Flow?

Definition 2

Let $G(V, E, c)$ be a network and let $s, t \in V, s \neq t$ be a *source* and a *sink* vertices, that is, both outgoing degree of t and incoming degree of s are equal to 0. A map $f : E \to \mathbb{R}$ is called a flow from s to t if constraints hold:
Non-negativity and Capacity Constraint: $\forall (u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$
Flow Conservation: $\forall v \in V \setminus \{s, t\}$ holds $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,w)\in E} f(v, w)$.

Remark 1: Flow Conservation is also known as Kirchhoff equations.
Remark 2: In fact, the condition that s is a source and t is a sink can be ensured for arbitraty vertices u, v by adding additional vertices s with a unique arrow (s, u) such that $c(s, u) = \infty$ and t with $c(v, t) = \infty$.
Remark 3. Assume that both (u, v) and (v, u) belong to E, f is a flow, and, say $f(u, v) \leq f(v, u)$. Set $g(u, v) = 0, g(v, u) = f(v, u) - f(u, v)$, and $g(e) = f(e)$, $e \neq (u, v), e \neq (v, u)$. Then g is a flow, by Definition 2.

Definition 3

The throughput $|f|$ of a flow f is $|f| = \sum_{(s,v)\in E} f(s, v)$.

Reduced Flow

Definition 4

For a flow, f, let $G_f(V_f, E_{f>0}) \subseteq G$ be the subgraph with $E_{f>0} = \{e \in E | f(e) > 0\}$. The flow is called *reduced* if G_f is a DAG.

Lemma 5

For any flow f there is a reduced flow, f_{red} such that $|f| = |f_{red}|$.

Proof: We proceed as in Remark 3 after Definition 2. Assume $C \subseteq E_{f>0}$ is a cycle. Let $m_C = MIN(f(e), e \in C)$. Then set $f_1(e) = f(e) - m_C$ for $e \in C$ and $f_1(e) = f(e)$, otherwise. Then f_1 fits Definition 2, if f does. Since s is a sink, none of edges going out of s is in C, hence, $|f_1| = |f|$. We have: $E_{f_1>0} \subsetneq E_{f>0}$ and $E_{f_1>0}$ got rid of at least one of cycles in $E_{f>0}$. Therefore, we may apply the same procedure again to get $f_2, f_3, \cdots$ and finally at some step get a reduced flow. $\square$

Path model of a flow

Definition 2 introduces network flows as solutions of a system of Kirchhoff equation and capacity and positivity inequalities, this is an **implicit edge model**.

Definition 6

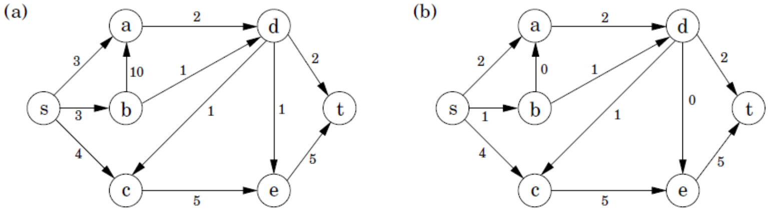
Let $P = (s = u_0, u_1, \ldots, u_k = t)$ be a directed path and also a set of edges $P = \{(u_0, u_1), (u_1, u_2), \ldots, (u_{k-1}, u_k)\}$. Set $m(P) = MIN(c(e), e \in P)$. For $0 \leq \lambda \leq m(P)$, let $f_{\lambda,P}(e) = \lambda$, if $e \in P$ and $f_{\lambda,P}(e) = 0$, otherwise. One can check Definition 2 to see that $f_{\lambda,P}$ is a flow, reduced, if all u_i are distinct.

Theorem 7

For every reduced flow f holds $f = \sum_{i=1}^{t} f_{\lambda_i,P_i}$ for some paths P_i and some λ_i such that $0 < \lambda_i \leq m(P_i), i = 1, \cdots, t$.

Remark: We postpone the proof of Theorem 7 that introduces a **Path Model** of all (reduced) flows. It allows for linearization of non-linear problem of path finding by reducing it to finding flows in the edge model, which is a linear programming problem on a convex polytope.

Example of a Network Flow



- On figure a), a network with capacity values is drawn;
- Figure b) shows a flow f on this network.
- f is reduced and holds: $f = f_{2,(s,a,d,t)} + f_{4,(s,c,e,t)} + f_{1,(s,b,d,c,e,t)}$.

Network Cut

Recall a (slightly updated) definition from Lecture 7:

Definition 8

A network cut is a decomposition of V into $S \subseteq V$ and $T = V \setminus S$ such that $s \in S, t \in T$. The *capacity of the cut* is $c(S,T) = \sum_{(u,v)\in E, u\in S, v\in T} c(u,v)$. For a network flow, f, the *flow across the cut*:

$$f(S,T) = \sum_{(u,v)\in E, u\in S, v\in T} f(u,v) - \sum_{(v,u)\in E, u\in S, v\in T} f(v,u). \tag{1}$$

Lemma 9

For every flow f from s to t and every cut (S,T) holds $f(S,T) = |f|$.

Corollary 10

For every flow holds: $|f| = \sum_{(v,t)\in E} f(v,t)$.

Proof: Apply Lemma 9 to $S = V \setminus \{t\}$, $T = \{t\}$ and the fact that t is a sink. $\square$

Proof of Lemma 9

Using $S = \{s\} \sqcup S \setminus \{s\}$ we have: $\sum_{(u,v)\in E, u\in S} f(u,v) - \sum_{(v,u)\in E, u\in S} f(v,u) =$

$$= |f| + \sum_{(u,v)\in E, u\in S\setminus\{s\}} f(u,v) - \sum_{(v,u)\in E, u\in S\setminus\{s\}} f(v,u) = |f|, \tag{2}$$

using Kirchhoff equations for $u \in S \setminus \{s\}$. Using $V = S \sqcup T$, we get from (2):

$$|f| = [\sum_{(u,v)\in E, u\in S, v\in T} f(u,v) - \sum_{(v,u)\in E, u\in S, v\in T} f(v,u)] + [\sum_{(u,v)\in E, u\in S, v\in S} f(u,v) - \sum_{(v,u)\in E, u\in S, v\in S} f(v,u)] = [f(S,T)] + [0], \tag{3}$$

where we use Definition 8 of $f(S,T)$ for the first term, and obvious equality $\sum_{(u,v)\in E, u\in S, v\in S} f(u,v) = \sum_{(v,u)\in E, u\in S, v\in S} f(v,u)$, for the second one. $\square$

Proof of Theorem 7

Lemma 11

For every flow f from s to t with $|f| > 0$, t is reachable from s in $E_{f>0}$.

Proof: Assume that t is not reachable. Let S be the set of vertices in V, reachable from s in $E_{f>0}$, $T = V \setminus S$. Then by Lemma 9 $f(S, T) > 0$, hence, there is at least one $(u, v) \in E$ such that $u \in S, v \in T$ and $f(u, v) > 0$. In other words, $e \in E_{f>0}$, hence, v is reachable from s in $E_{f>0}$, contradiction. □

Recall that **Theorem 7** claims that every flow is a sum of path flows. To prove it, we take our reduced flow and, using Lemma 11, find a path P from s to t in $E_{f>0}$ and consider $f_1 = f - f_{m(P),P}$, which is another flow with $|f_1| = |f| - m(P)$. Besides decreasing the throughput, we notice that $E_{f_1>0} \subsetneq E_{f>0}$, because $m(P) = f(e)$ for at least one edge in P, so $f_1(e) = 0$. Then we pass from f_1 to f_2 etc., and in at most $|E_{f>0}|$ steps we will get a flow f_{t+1} with $|f_{t+1}| = 0$. However, since f is a reduced flow, and a subgraph of a DAG is DAG, all $f_1, \ldots, f_{t+1}$ are also reduced. Hence $f_{t+1} = 0$. □

Remark: The statement 7 holds for every flow, but for a non-reduced one, besides paths from s to t, the summands include circular flows over cyclic paths. These summands stand for f_{t+1} above.

Corollary 12

For every flow f from s to t and every cut (S, T) holds $|f| \leq c(S, T)$.

Proof: By Lemma 9, $|f| = f(S, T)$, By Definitions 2 and 8, $f(S, T) =$

$$= \sum_{(u,v)\in E, u\in S, v\in T} f(u, v) - \sum_{(u,v)\in E, u\in S, v\in T} f(v, u) \leq \sum_{(u,v)\in E, u\in S, v\in T} f(u, v) \leq (4)$$

$$\leq \sum_{(u,v)\in E, u\in S, v\in T} c(u, v). \quad \square$$

Corollary 13

If for some flow, f, and some cut, (S, T) holds $|f| = c(S, T)$, then the flow has the maximal possible throughput and the cut has the minimal capacity.

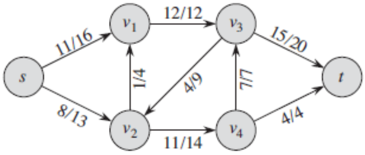
Remark: The sense of Corollary is that that the problems of maximal throughput flow and of the minimal cut of the graph are dual to each other.

Residual Network

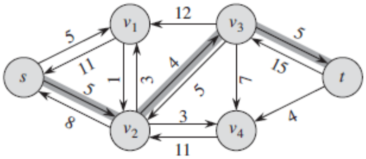
In this section we will introduce Ford-Fulkerson method including the existance theorem of a flow and a cut like in Corollary 13 and also a method of constructing these. We start with their crucial definition:

Definition 14

Let f be a flow from s to t in a network $G(V, E, c)$. Residual network G_f has edge list E_f and capacity $c_f : E_f \to \mathbb{R}$. E_f includes the edges $(u, v) \in E$ such that $f(u, v) < c(u, v)\}$ with $c_f(u, v) = c(u, v) - f(u, v)$ and the vertex pairs (v, u) such that $(u, v) \in E$ and $f(u, v) > 0$, with capacity $c_f(v, u) = f(u, v)$.



Network capacity and Flow



Residual network and a path on it

Augmenting flows

Lemma 15

Let f_1 be a reduced flow from s to t in the residual network G_f built for a flow from s to t in G. Define $f_2 = f + f_1 : E \to \mathbb{R}$ as $f_2(u,v) = f(u,v) + f_1(u,v)$ if $f_1(u,v) > 0$, $f_2(u,v) = f(u,v) - f_1(v,u)$ if $f_1(v,u) > 0$, and $f_2(e) = f(e)$ for other edges. Then f_2 is a flow in G with thoroughput $|f_2| = |f| + |f_1|$.

Proof: The reducedness condition makes the definition sense, because in particular, for any $(u,v) \in E$ either $f_1(u,v) = 0$ or $f_1(v,u) = 0$. Checking Definition 14 for both cases, we see that $f_1(u,v) \le c_f(u,v) = c(u,v) - f(u,v)$ implies $f(u,v) + f_1(u,v) \le c(u,v)$, for $(u,v) \in E$. And if $f_1(v,u) > 0$, then, $f_1(v,u) \le c_f(v,u) = f(u,v)$, hence $f(u,v) - f_1(v,u) \ge 0$, so the capacity constraints are fulfilled. Let $f'(e) = f_1(e)$, for $e \in E$, and $f'(u,v) = -f_1(v,u)$ if $(v,u) \in E_f \setminus E$. Then f' fits Kirchhoff equations, together with f_1, and for the same reason as we gave in Remark 3 to Definition 2. So both f and f' are defined on E and fit Kirchhoff equations, so f_2 does. The throughput equality is deduced similarly. $\square$

Augmenting path

As an augmenting flow in Lemma 15 it is standard to consider $f_{P,m(P)}$ for a path P from s to t in the residual network G_f. If such a path exists, it means, if t is reachable from s in G_f, then by the Lemma, we augment the throughput from $|f|$ to $|f| + m(P)$. Such a path is called **augmenting**. Does it always exist?

Theorem 16

(Ford-Fulkerson) For a flow f from s to t the following conditions are equivalent:
1. f has maximal throughput
2. t is not reachable from s in G_f
3. $|f| = c(S, T)$ for a network cut (S, T). This cut has the minimal capacity.

Proof: $3 \Rightarrow 1$ follows from Corollary 13. $1 \Rightarrow 2$ follows from Corollary 15. Let's prove $2 \Rightarrow 3$ for S defined as all vertices reachable from s in G_f. Indeed, $t \in T = V \setminus S$. Assume that $u \in S, v \in T$. Then $(u, v) \notin E_f$ otherwise v would be reachable. Hence, if $(u, v) \in E$ then we must have $f(u, v) = c(u, v)$ and if $(v, u) \in E$, then $f(v, u)$ must equal 0. Then, by formula (1), $f(S, T) = c(S, T)$ and by Lemma 9, $|f| = f(S, T) = c(S, T)$. □

Ford-Fulkerson Method and Integer Networks

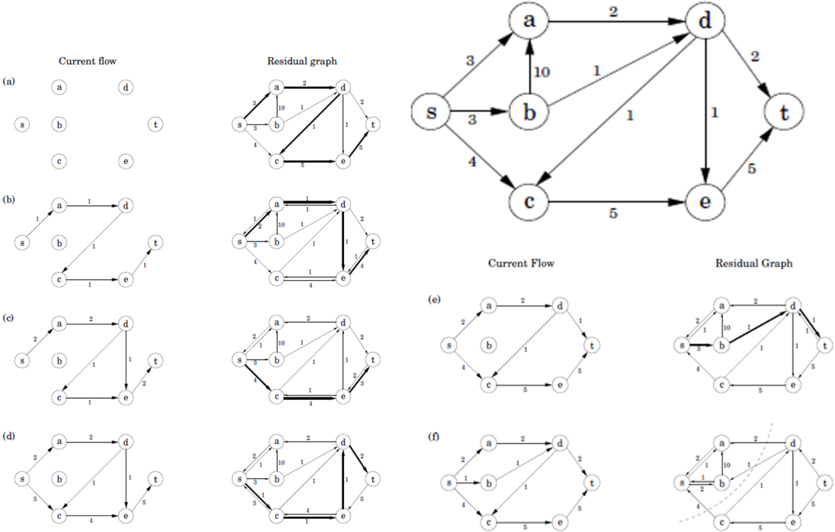
Theorem 16 implies that a method exists to solve the maximum flow problem, by searching for an augmenting path in the residual network G_f for a flow f, which is originally set as 0 flow, and after finding an augmenting path, P, f is replaced with $f + f_{P.m(P)}$. How to select an augmenting path may be decided in different way, therefore, we talk about Ford-Fulkerson *method*. However, in general Theorem 16 dosn't guarantee that there will be finitely many augmenting steps and the throughput will not increase as $1 + 1/2 + 1/4 + \ldots$. Still, in some important particular case this is guaranteed:

Theorem 17

If all edge capacities are integer numbers, $c : E \to \mathbb{Z}_{\geq 0}$, then the maximal flow has integer values and is obtained by the Ford-Fulkerson method with any path search algorithm in a finite number of steps.

Proof: Applying induction on the step, we prove that $m(P)$ is integer for any augmenting path, hence, c_f is integer, hence, every intermediate f is also integer. Therefore, the throughput of every intermediate flow is integer, hence, every augmentation increases the throughput for at least 1 and the maximum is reached in a finite number of steps.

Ford-Fulkerson Method Example



Edmonds-Karp Algorithm

• The algorithm proposed by Edmonds-Karp is just the Ford-Fulkerson method with **BFS-Traverse** used to find the path using the minimal number of edges.
• Recall that BFS-Traverse was introduced in Lecture 7 and it has $O(|V| + O(|E|)$ complexity in general and in this particular case as well, because $|E_f| \leq 2|E|$.
• We may assume the underlying non-directed graph connected, hence, $|V| \in O(|E|)$, so the complexity of BFS-Traverse in G_f for any f is $O(|E|)$.
• Notice that for an augmentation path the throughput gain is $m(P)$, which is equal to $c_f(u, v)$ for some $u, v \in V$ and either $(u, v) \in E$ or $(v, u) \in E$. We call such edge (u, v) or reverse edge (v, u) *critical* for an augmentation path in E_f.
• The number of times BFS-Traverse is called is bounded by $2|E|K$, where K is the maximal number of times the edge may be critical.
• Let $\delta_f(v)$ be the shortest path distance from s to $v \in V$ in E_f. Since the shortest path does not visit a vertex twice, we have $\delta_f(v) < |V|$ for any v and f.

Lemma 18

For a flow f, let the shortest path P from s to t in E_f augments the flow to g. Then for every $v \in V$, $\delta_f(v) \leq \delta_g(v)$.

Proof of Lemma 18 and Edmonds-Karp Theorem

- Assume the contrary and let $v \in V$ has the minimal $\delta_g(v)$ out of all $v \in V$ such that $\delta_f(v) > \delta_g(v)$. Let $s \to \ldots \to u \to v$ be the shortest path in E_g of length $\delta_g(v)$. Then $\delta_g(u) = \delta_g(v) - 1$, hence, $\delta_f(u) \leq \delta_g(u)$ thanks to the choice of v.
- If $(u, v) \in E_f$, then $\delta_f(v) \leq \delta_f(u) + 1 \leq \delta_g(u) + 1 = \delta_g(v)$, a contradiction.
- So, $(u, v) \notin E_f$ and $(u, v) \in E_g$ by our choice of u. Then P contains (v, u) as a critical edge (if $(v, u) \in E$, then $g(v, u) = c(v, u)$ else $g(u, v) = 0$).
- Recall that P is a shortest path in E_f, hence, $\delta_f(u) = \delta_f(v) + 1$.
- Hence, $\delta_f(v) = \delta_f(u) - 1 \leq \delta_g(u) - 1 = \delta_g(v) - 2 < \delta_g(v)$, contradiction. $\square$

Theorem 19

For each $(u, v) \in E$ this edge and reverse edge (v, u) become critical at most $|V|$ times together. Consequently, this algorithm finds the maximal flow with $O(|V||E|^2)$ complexity.

Proof of Theorem 19

• First of all, for $(u, v) \in E$ we notice that at any state of E_f either $(u, v) \in E_f$ or $(v, u) \in E_f$ or both, hence, $|\delta_f(u) - \delta_f(v)| \leq 1$.
• When (u, v) is critical in E_f, then it belongs to the shortest path in E_f, hence, $\delta_f(v) = \delta_f(u) + 1$ and after that (u, v) leaves E_f.
• The cases when (u, v) and (v, u) become critical come alternately.
• So assume that in E_f, (u, v) is critical and next time, in E_g, (v, u) is critical. By Lemma 18, $\delta_g(u) \geq \delta_f(u)$, hence, $\delta_g(u) - \delta_f(u) \geq 2$.
• So after (u, v) critical then (v, u) critical the distance from s to u jumps on at least 2, and as we noticed above, every distance has $|V|$ as an upper limit. So the pair (u, v) and (v, u) together may become critical at most $|V|$ times. $\square$

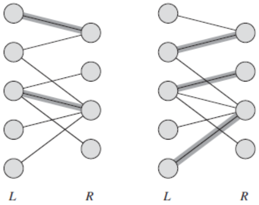
Remark: The proof of Theorem 19 seems to leave much place for improvement for the complexity upper bound. However, in [Galil, 1981] it has been given an example of a network such that Edmonds-Karp algorithms uses $\Theta(|V||E|^2)$ complexity.

Bipartite graph matching

Definition 20

An undirected graph $G(V, E)$ is called *bipartite*, if for some cut, $V = L \sqcup R$ all edges connect a vertex from L to a vertex from R. Use notation such that for $(u, v) \in E$ it is assumed that $u \in L$ and $v \in R$. A *matching* is a set $M \subseteq E$ such that for any $(u, v), (x, y) \in M$ holds $u \neq x, v \neq y$.

The problem to be solved is finding a **maximal matching**, that is, a matching $M \subseteq E$ of maximal size $|M|$. The below picture shows that it is not the same as a matching M that does not allow for adding more $e \in E \setminus M$.

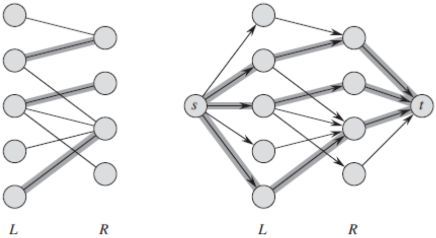


Reducing the maximal matching to maximal flow

For a bipartite graph $G(V = L \sqcup R, E)$ consider a network $\tilde{G}(\tilde{V}, \tilde{E}, c)$ as follows. $\tilde{V} = V \cup \{s, t\}$, $\tilde{E}$ consists of arrows $u \to v$ for every edge $(u, v) \in E$ and additionally, $\{s \to l, r \to t|, l \in L, r \in R\}$. The capacity of each edge in $\tilde{E}$ is 1.

Lemma 21

For every matching $M \subseteq E$ there is an integer valued flow f in $\tilde{G}$ with $|f| = |M|$. Conversely, for an integer valued flow f in $\tilde{G}$ there is a matching M, $|M| = |f|$.



Reducing the maximal matching to maximal flow

Formal proof of Lemma 21: For a matching $M \in E$ and $(u, v) \in M$ we set $f(s, u) = f(u, v) = f(v, t) = 1$, and $f(x, y) = 0$, otherwise. Then the matching condition implies that both incoming and outgoing traffic for u and v is 1, and 0, for $v \in V$ not involved in M. So the Kirchhoff equations and capacity limits are fulfilled, and the throughput is $|M|$. Otherwise, if f is an integer valued flow, then $f(e) = 0$ or $f(e) = 1$, for $e \in \tilde{E}$. Kirchhoff equations imply that the arrows $u \to v, u \in L, v \in R$ with $f(u, v) = 1$ may not share vertices. $\square$

Corollary 22

The maximal matching corresponds to the maximal flow in $\tilde{E}$.

Proof: By Theorem 17, the maximal flow in $\tilde{E}$ is integer valued. $\square$

Theorem 23

Ford-Fulkerson method with any augmentation path search algorithm solves the maximal matching problem in $O(|E||V|)$.

Proof: Since $|M| \leq |V|$, by Lemma 21 the maximal flow has thorughout $|f| \leq |V|$. So Ford-Fulkerson does at most $|V|$ steps. $\square$

References

Z.Galil, *On theoretical efficiency of various network flow algorithms*, *Theoretical Computer Science* **14** (1981), 103 -111.