Maximum Flow

CS 624 — Analysis of Algorithms

April 30, 2024

UMass Boston

Flow Networks

Definition (Flow Network)

A **flow network** is a directed graph $G = (V, E)$ with two distinguished vertices:

- s — the "source", and
- t — the "sink"

such that the following properties hold:

1. For every node $u \in V$, there is a path from s to t through u. Equivalently, for every node $u \in V$ there is a path from s to u and a path from u to t.
2. There are no "antiparallel" edges. That is, if $(u, v) \in E$ then $(v, u) \notin E$. (But there could be a path from v to u.)

Flow Networks

The “no antiparallel edges” condition is not a big deal.

If we have a graph with edges (u, v) and (v, u), we can just introduce a new vertex w and replace the edge (v, u) by the two edges (v, w) and (w, u). It won’t change anything.

Flows

A flow in the flow network G as above is an assignment of a non-negative real number to each edge of the graph.

The assignment is denoted by $f : E \to \mathbb{R}^{\geq 0}$.
That is, for each edge e, $f(e)$ is a non-negative real number.

If $e = (u, v)$, then we may (by an abuse of notation) also write $f(u, v)$ for $f(e)$ — they will mean the same thing.

Examples

- If G represents a system of circuits or pipes, then $f(e)$ might represent the amount of current or water passing over edge e per unit of time.
- If G represents a transportation network like a railroad line, then $f(e)$ might represent weight being transported over e per unit time.

Conservation

Additionally, a flow must satisfy the following **conservation** property:

Conservation

At every node u except for the nodes s and t, the flow **into** u must exactly equal the flow **out of** u.

There are two ways to define the conservation property mathematically.

Conservation v1: f remains non-negative

We extend the function f so that if (u, v) is not an edge, then we simply set $f(u, v) = 0$.

Then **conservation** is: For each node u different from s and t:

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

This is in fact the way our text writes this constraint.

Conservation v2: extend f with negative flow

If (u, v) is a (directed) edge in the graph, then (v, u) is not. We define

$$f(v, u) = -f(u, v) \qquad \text{when } (u, v) \in \boldsymbol{E}$$

For all x and y not joined by an edge in either direction, we define

$$f(x, y) = f(y, x) = 0 \qquad \text{when } (x, y) \notin \boldsymbol{E} \text{ and } (y, x) \notin \boldsymbol{E}$$

We can think of a flow along (u, v) as a "negative" flow along (v, u).
Then **conservation** is: For each node u different from s and t:

$$\sum_{v \in V} f(u, v) = 0$$

We will use this formulation of flow and conservation.

Definition of Flow

Definition (Flow)

A **flow** f on the flow network $G = (V, E)$ with source s and target t is any function $f : V \to \mathbb{R}$ that satisfies the following three conditions:

1. f **lives on edges:** If u and v are not joined by an edge in either direction, then $f(u, v) = 0$.
2. f **is skew-symmetric:** For all vertices u and v, $f(u, v) = -f(v, u)$.
3. f **satisfies the conservation property:** For all vertices u other than s and t, $\sum_{v \in V} f(u, v) = 0$.

Flows can be added

Definition

If f and g are two flows, we define their sum $f+g$ in the obvious way:

$$(f+g)(u,v) = f(u,v) + g(u,v)$$

Definition

If f is a flow, the **value of the flow**, written $|f|$, is the flow out of the source. To be precise,

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

Note that the flow out of s is equal to the total flow into t.

Cuts

Definition

A **cut** in G is a partition of the vertex set V into two sets X and $\overline{X} = V - X$ such that $s \in X$ and $t \in \overline{X}$.

If f is a flow and $(X, \overline{X})$ is a cut, the **flow across the cut** is defined as

$$f(X, \overline{X}) = \sum_{v \in X, w \in \overline{X}} f(v, w)$$

Cuts and Flows

Lemma

If f is a flow and $(X, \overline{X})$ is a cut, then the flow across the cut is equal to the flow value $|f|$.

Proof.

First consider the following sum (note the bounds!):

$$\sum_{\substack{v \in X \\ w \in V}} f(v,w) = \sum_{w \in V} f(s,w) + \sum_{\substack{v \in X - \{s\} \\ w \in V}} f(v,w) \qquad \text{split sum}$$

$$= |f| + \sum_{v \in (X - \{s\})} \sum_{w \in V} f(v,w) \qquad \text{by def. of flow value; unnest sum}$$

$$= |f| + \sum_{v \in (X - \{s\})} 0 \qquad \text{by conservation } (x \neq s \text{ and } x \neq t)$$

$$= |f| + 0 = |f|$$

Cuts and Flows (Proof Continued)

Proof (continued).

Using this, we see then that

$$
\begin{aligned}
f(X, \overline{X}) &= \sum_{\substack{v \in X \\ w \in \overline{X}}} f(v, w) && \text{by def. of flow across cut} \\
&= \sum_{\substack{v \in X \\ w \in V}} f(v, w) - \sum_{\substack{v \in X \\ w \in X}} f(v, w) && \text{split sum} \\
&= |f| - \frac{1}{2} \sum_{\substack{v \in X \\ w \in X}} (f(v, w) + f(w, v)) && \text{by def. of flow value; sum twice + combine} \\
&= |f| + 0 = |f| && \text{by skew-symmetry}
\end{aligned}
$$

□

Capacity

Definition (Capacity)

Let $G = (V, E)$ be a flow network. A **capacity** function $c : E \to \mathbb{R}^{\geq 0}$ represents the maximum flow that each edge can carry.

If $(v, w) \in E$, we also write $c(v, w)$ for the capacity of that edge.

If (v, w) is not an edge (and in particular, if it is a "reverse edge"), then we set $c(v, w) = 0$.

We want to optimize flow for a network with **limited capacity**. That is, the flow along an edge can be no more than the capacity of the edge:

$$0 \leq f(e) \leq c(e) \qquad \text{for every } e \in E$$

Cut Capacity

Definition

If $(X, \overline{X})$ is a cut, then the **capacity of the cut** is defined as

$$c(X, \overline{X}) = \sum_{\substack{v \in X \\ w \in \overline{X}}} c(v, w)$$

It can be shown that if $(X, \overline{X})$ is a cut and f is a legal flow, then

$$f(X, \overline{X}) \leq c(X, \overline{X})$$

or equivalently,

$$|f| \leq c(X, \overline{X})$$

Maximum Flow Problem

Maximum Flow Problem

Given a flow network with a capacity function, we want to find the flow respecting that capacity and having the greatest possible value.

Such a flow is called a **maximum flow**.

How can we compute the maximum flow?

Representation of Flows and Capacities

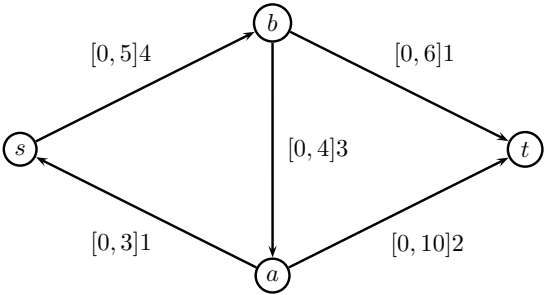
We will label each edge of the flow network with a label of the form

$$[c', c]f$$

with the following components:

- c' is the capacity in the **opposite direction** of the edge. (?)
- c is the capacity in the **direction** of the edge.
- f is the flow in the direction of the edge.
  It must satisfy $c' \leq f \leq c$.

Representation of Flows and Capacities

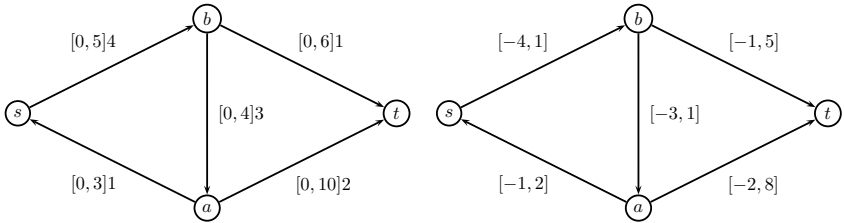


- This graph represents a legal flow. (Check.)
- Notice that the flow from node a to node s is "counter-productive".
- What is the *value* of this flow?

Residual Graph

Given a flow on a network, how much could the flow on each edge change, in either direction?

The **residual graph** answers this question. Its capacities are calculated by subtracting the flow from the original capacities.



The residual capacity on $a \to s$ is $[-1, 2]$. That means we could increase the flow by 1 *backwards* across that edge.

Allowable Flow on Residual Graph

A flow g is **allowable** on a residual graph if for each edge e, $g(e)$ is in the capacity interval labeling that edge e.

It should be clear that if f is the flow in the original graph, and g is any allowable flow in the residual graph, then $f + g$ is a flow which satisfies the original capacity constraints.

Strategy for Max Flow

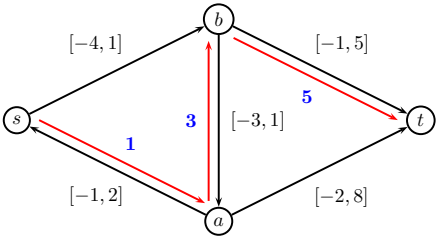
Strategy:

- Start with a flow f. Form the residual graph for that flow.
- Then find an allowable flow g in the residual graph such that $f + g$ has a greater value than f.
- Repeat this process until the maximum flow is found.

Augmenting Paths

How do we find an allowable flow in the residual graph?

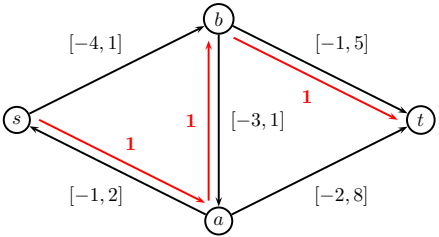
- The easiest kind of flow to find is a "path" from s to t where each edge carries the same amount. (Backwards traversals are allowed!)
- For instance, consider the "path" $s \to a \to b \to t$ in our residual graph.
- The maximum amount on each of these edges is shown here:



Augmenting Paths

- This is just one example, not necessarily the best path to pick.
- For any such path, the **residual flow** r is the minimum of those maximum possible flows along that path's edges.

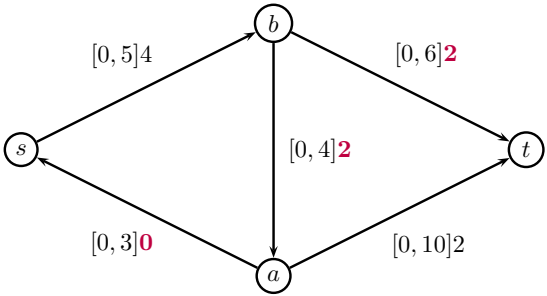
  In this case, $r = \min\{1, 3, 5\} = 1$.
- If $r > 0$, the path with its flow values is an **augmenting path**.



Augmenting Paths

That augmenting path with its associated residual flow is itself a flow. Let's call it g.

If we add this residual flow g to our original flow f, we get the new flow $f + g$ shown below.



(In that graph above, the purple amounts are the amounts that have changed.)

Augmenting Paths

There are two important things to note:

1. We still have an allowable flow in our graph. That's because the adjustment we made was within the bounds allowed by the "residual" computation. Note that while the augmenting path had two edges "in the wrong direction", the final graph does not.
2. The value of the flow in the new path is greater than the original value. In fact, we have $|f + g| = |f| + |g| = |f| + r > |f|$.

Augmenting Paths Algorithm

So we have the beginnings of an algorithm here:

- Start with any allowable flow.
- From it, produce the residual graph.
- Use that residual graph to find any augmenting path.
- Add the augmenting path to the original flow to produce a new flow. The new flow will still be allowable and will have a greater value.
- Repeat this whole process until nothing more can be done.

Does the Algorithm Really Work?

There are two questions that come up immediately:

1. Is it possible that we might have a non-maximum flow f, but nevertheless there was no augmenting path for f? If there was, then of course the algorithm would fail to yield a maximum flow.
2. Is it possible that the algorithm fails in some other way? For instance, maybe it never ends.

The Max-Flow Min-Cut Theorem

Theorem (The Max-Flow Min-Cut Theorem, by Ford and Fulkerson)

If $G = (V, E)$ is a flow network with capacity function c and source and sink nodes s and t, the following statements are equivalent:

1. *f is a maximum flow.*
2. *There is no augmenting path for f.*
3. *There is some cut $(X, \overline{X})$ for which $|f| = c(X, \overline{X})$.*

Proof of Max-Flow Min-Cut Theorem

(1) $\Longrightarrow$ (2). If there is an augmenting path p for f, then we can increase the flow value.

(2) $\Longrightarrow$ (3). Suppose there is no augmenting path for f. Let X be the set of vertices reachable from s on a path on which each edge has positive residual value. Trivially, X includes s, and by our assumption, X does not include t. Thus, $(X, \overline{X})$ is a cut. Further, if (v, w) is an edge in the original graph that crosses the cut (i.e., with $v \in X$ and $w \in \overline{X}$), we must have $f(v, w) = c(v, w)$, since otherwise w would also be in X. Thus, we have

$$|f| = \sum_{\substack{v \in X \\ w \in \overline{X}}} f(v, w) = \sum_{\substack{v \in X \\ w \in \overline{X}}} c(v, w) = c(X, \overline{X})$$

Proof of Max-Flow Min-Cut Theorem (Continued)

(3) $\implies$ (1). We know that in any case, $|f| \leq c(X, \overline{X})$ (See exercise in class notes). The fact that with this cut they are equal means that there is no flow with a greater value than f.

- Note that as a consequence of this theorem, we know that the value of the maximum flow in the network is equal to minimum capacity of any cut. That's where the theorem gets its name.
- So this answers the first of our questions: if f is not maximum, we know that there will be at least one augmenting flow, so the algorithm can make progress.

□

Termination and correctness

- We need to know if the algorithm is guaranteed to terminate. Here's where things get complex.
- First of all, let us suppose that all the capacities of the network are integers. In that case, there is something we can say, provided we make two very natural assumptions:
- We assume that we start with a flow that is identically 0. (We'll always assume this in any case; it's the natural way to start.)
- We assume that when we pick an augmenting path, we make the flow along that path as large as possible. That is, we let it be the minimum of the residual values of each edge on the flow.

Termination and correctness

- In this case, the augmenting flow value will of necessity be an integer, and so we will always be increasing the value of the flow by an integer each time we pick an augmenting path.
- Since the maximum flow value is finite, this process has to stop.
- This also shows that the maximum flow is an integral flow–the flow along each edge is an integer.
- However, there are many flow networks that arise in practice in which the capacities are not necessarily integers.
- This is not really a problem. If they are rational numbers, then we can always reason in exactly the same way by using as our "minimal unit" the fraction which is 1 over the least common denominator of all the capacities. Exactly the same argument then works.

Termination and correctness

- The problem of irrational numbers is peculiar, though. Ford and Fulkerson actually gave an example where, by picking the augmenting paths in a particularly stupid manner, even though the flow along these augmenting paths was chosen to be as big as possible, the following would happen:
- The process would not terminate. Of course the values of the successive flows would keep increasing, but they would always be less than the maximum possible value.
- It's even worse than that. Because the values of the successive flows keep increasing, they have a limit. But this limit is strictly less than the value of the maximum flow in the network.

Termination and correctness

- You can find the example in their book. It's amusing, at least.
- We can always assume that our capacities are rational, and so our algorithm – however we pick the augmenting paths – will definitely terminate and give the maximum flow.
- However, there still is the question of efficiency: how well can we pick the augmenting paths so that we have to repeat the iterations of the algorithm as few times as possible?
- This question has been considered over many years, and successively better algorithms have been produced.
- Our textbook goes into this a little but we will stop here.

Application: the “Marriage Problem”

- Suppose we have two finite sets G (“girls”) and B (“boys”).
- Each girl likes one or more boys, and each boy likes one or more girls.
- Let’s assume that if a girl and a boy both like each other, they would be happy to marry each other.
- The problem then is: Is there a way of marrying all the girls and all the boys to someone they each like? If so, we say that the marriage problem is *solvable*.
- Let’s assume that there are n girls and n boys.

The “Marriage Problem”

- It’s not so simple to decide the question.
- For instance, suppose all the girls liked one boy (and he liked all the girls), or all the boys liked one girl (and she liked all the boys).
- There will be a lot of unhappy people in this case.
- To make the wording in what follows less cumbersome, let us agree that when we say that “a girl g likes a boy b”, we also mean that the boy b likes the girl g.
- The following theorem gives a necessary and sufficient condition for the problem to be solvable:

Theorem (Hall’s “Marriage Theorem”)

The marriage problem is solvable iff for each r (with $1 \leq r \leq n$), every set of r girls likes at least r boys

The “Marriage Problem”

- The necessary and sufficient condition provided by this theorem seems one-sided.
- Presumably one could instead assume that each set of r boys likes at least r girls.
- It turns out that they are equivalent, and we state that as a lemma now, because we will need this fact in the proof of the theorem below.

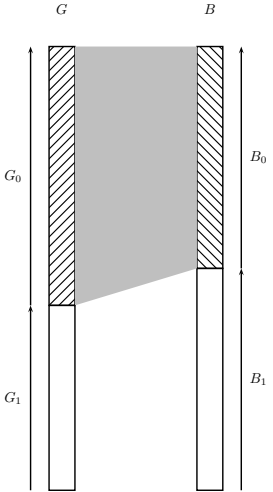
Lemma

If for each r (with $1 \leq r \leq n$), every set of r girls likes at least r boys, then also every set of r boys likes at least r girls.

Proof of Lemma

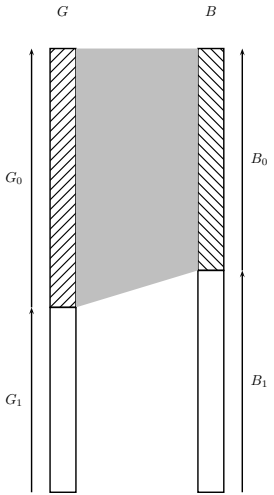
- Suppose B_0 is a set of r boys. Let G_0 be the set of girls that those boys like.
- Equivalently, G_0 is the set of girls that like any of those boys.
- We need to show that G_0 contains at least r girls.
  Let us set

$$B_1 = B \setminus B_0$$
$$G_1 = G \setminus G_0$$



Proof of Lemma

- For notational convenience, let us define g_0 to be the size of the set G_0, and similarly for g_1, b_0, and b_1.
- The set of girls G_1 likes only boys in B_1. (That's by definition; any girl who likes a boy in B_0 is automatically in G_0.)
- Therefore by the assumption of the lemma, we must have $g_1 \leq b_1$. (It might be that g_1 is strictly less than b_1, but that's OK also.)
- Therefore it follows that $g_0 \geq b_0$, and that's what we needed to prove.



Proof of Theorem

- If the problem is solvable, then certainly every group of r girls likes at least r boys, since each one likes at least the one she will marry, and possibly others as well, and none of those boys is going to marry more than one girl, so there are at least r boys liked by the r girls.
- Thus, we have to prove the other direction: if each group of r girls likes at least r boys, then the problem is solvable.
- We can model this as a graph problem: the nodes in our graph will be the union of the sets G and B.
- There is an (undirected) edge between a node $g \in G$ and $b \in B$ iff they like each other.
- This is an example of what is called a bipartite graph. We want to know if there is a map m from G to B such that the map is 1-1 and onto and such that $m(g) = b$ only if there is an edge between g and b.

Proof of Theorem – The Max Flow Problem

- First, we make the graph a directed graph. Every edge is between a girl and a boy. We make the edge a directed edge from the girl to the boy.
- We introduce two new nodes, s and t.
- We introduce an edge from s to each girl, and we introduce an edge from each boy to t.
- The capacity of each flow is simply 1.
- The question is then to find the maximum flow from s to t.
- It is clear that the marriage problem is solvable iff the maximum flow has value n. Certainly the maximum flow can't be greater than n.
- So to show that the marriage problem is solvable, it is enough to show that the maximum flow has value $\geq n$.

Proof of Theorem – The Max Flow Problem

- The max-flow min-cut theorem tells us that the maximum flow will have value n iff the cut of minimum capacity has capacity n.
- Thus it is enough to show that the capacity of every cut is $\geq n$.
- We are assuming that every group of r girls likes at least r boys.
- So say $(X, \overline{X})$ is a cut.
- Let us divide this into a number of different cases.

Proof of Theorem – The Different Cases

Case 1: $X = \{s\}$. In this case, $c(X, \overline{X})$ is just the sum of the capacities of the n edges from s to G. And this is n.

Case 2: $\overline{X} = \{t\}$. In this case $c(X, \overline{X})$ is just the sum of the capacities of the n edges from B to t. And this is also n.

Case 3: $X \subseteq \{s\} \cup G$. ($\overline{X}$ includes all of B.) Let $G_0 = X \cap G$. (G_0 might be all of G, but doesn't have to be.)
Say the number of elements of G_0 is r. $c(X, \overline{X})$ counts the following edges:

- The edges leaving G_0. By our assumption, the number of these edges is $\geq r$
- The edges entering $G \setminus G_0$. The number of these edges is just $n - r$.

So $c(X, \overline{X}) \geq n$.

Proof of Theorem – The Different Cases

Case 4: $\overline{X} \subseteq B \cup \{t\}$. (X includes all of G.) Let $\overline{X}_0 = \overline{X} \cap B$. ($\overline{X}_0$ might be all of B, but it doesn't have to be.
Say the number of elements of $\overline{X}_0$ is r. $c(X, \overline{X})$ counts the following edges:

- The edges entering $\overline{X}_0$—and by our assumption and the result of the lemma, the number of these edges is $\geq r$.
- The edges leaving $B \setminus \overline{X}$. The number of these edges is just $n - r$.

So $c(X, \overline{X}) \geq n$.

Proof of Theorem – The Different Cases

Case 5: $\overline{X} \cap G \neq \varnothing$ and $\overline{X} \cap B \neq \varnothing$. Set

$$\overline{X}_0 = \overline{X} \cap G$$
$$\overline{X}_1 = \overline{X} \cap B$$

Say the number of elements of $\overline{X}_1$ is r. Let A be the set of elements of G which have edges leaving them and arriving at elements of $\overline{X}_1$. There are at least r elements of A.
Let us set

$$A_0 = A \cap X$$
$$A_1 = A \cap \overline{X} = A \setminus A_0$$

Let the number of elements of A_0 be a_0, and the number of elements of A_1 be a_1. So we know that $a_0 + a_1 \geq r$.

Proof of Theorem – The Different Cases

$c(X, \overline{X})$ counts the following edges:

- The edges from $B \setminus \overline{X}_1$ to t. The number of these edges is $n - r$.
- The edges entering $\overline{X}_1$ that come from A_0. Since each element of A_0 has at least one edge leaving it and arriving at $\overline{X_1}$, there are at least a_1 such edges.
- The edges leaving s and entering A_1. There are exactly a_1 such edges.

Thus $c(X, \overline{X}) \geq (n - r) + a_1 + a_0 = (n - r) + r = n$

Proof of Theorem – Case 5