CS624 - Analysis of Algorithms

Max Flow

November 9, 2019

The problem

- $G = (V, E)$ is a directed graph with two distinguished vertices, s (the "source") and t (the "sink" or "target"). We also assume that
  1. There is at least one path from s to t. In fact, even more is true: for each node u in the graph, there is at least one path from s through u to t.
  2. There are no "antiparallel" edges. That is, if $e = (u, v)$ is an edge from vertex u to vertex v, then there is no edge from v to u. (there could be a *path* from v to u.)

The problem

- This last item ("no antiparallel edges") is not a big deal.
- In fact, 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.

Definition

A *flow network* is a directed graph $G = (V, E)$ with two distinguished vertices s and t such that

- for each vertex u there is at least one path from s through u to t, and
- there are no antiparallel edges in the graph.

Flows

Definition

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

- We denote the assignment by $f : E \to \{r \in \mathbb{R} : r \geq 0\}$. Thus, 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.
- We think of $f(e)$ as the amount of some substance passing over edge e per unit time (current or water for example).
- We might also think of a transportation network like a railroad lines. And then for each edge e, $f(e)$ represents the amount of something that was being transported over e in some unit of time

A conservation property

- Remember we are transporting some substance from s to t.
- So it must be that the function f satisfies the following constraint:
  Conservation: At every node u in the graph other than nodes s and t, the flow into u must exactly equal the flow out of u.
- This constraint is really a sort of conservation property – it says that the amount being transported is conserved at each vertex.
- We need to be able to express this constraint mathematically. To do this, it is most convenient to extend the function f to all pairs of nodes (u, v), even of (u, v) is not an edge in the graph.

Two Ways To Define The conservation property

extend f so it remains non-negative, We extend the function f so that if (u, v) is not an edge, then we simply set $f(u, v) = 0$. With that notation, we can write our constraint as follows: 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.

extend f so that it can also be negative. if (u, v) is a (directed) edge in the graph, then we know already that (v, u) is not. We define $f(v, u) = -f(u, v)$. And for all other pairs (x, y) such that neither (x, y) nor (y, x) is an edge, we define $f(x, y) = f(y, x) = 0$.
The idea here is that we can think of a flow along (u, v) as a "negative" flow along (v, u).

A conservation property

- With this notation, we can write our constraint more simply: for each node u different from s and t,

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

- Both of these conventions are completely equivalent.
- The first convention has the advantage that in the beginning, anyway, it can seem somewhat easier to understand what is going on.
- The second one has the advantage that the mathematics turns out to be much simpler.
- We have to pick one. Both are widely used. We are going to use the second one.

Definition of Flow

Definition

A flow f on the graph G with source and target vertices s and t is any function mapping pairs of vertices of G to $\mathbb{R}$ such that it 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 a conservation property. For all vertices v other than s and t, $\sum_{u \in V} f(u, v) = 0$

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* $|f|$ of the flow is the flow out of the source. To be precise,

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

Notice 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 $\overline{X}$ 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 to be

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

Lemma

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

Proof

First, note that

$$\sum_{\substack{v \in X \\ w \in V}} f(v,w) = \sum_{w \in V} f(s,w) + \sum_{\substack{v \in X \setminus \{s\} \\ w \in V}} f(v,w)$$

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

The last sum in the first line vanishes because v is neither s nor t, and so we can apply the conservation property:

$$\sum_{\substack{v \in X \setminus \{s\} \\ w \in V}} f(v,w) = \sum_{v \in X \setminus \{s\}} \sum_{w \in V} f(v,w)$$

$$= \sum_{v \in X \setminus \{s\}} 0 = 0$$

because (since v is neither s nor t) the inner sum vanishes by the conservation property)

Proof

Using this, we see then that

$$\begin{aligned} f(X, \overline{X}) &= \sum_{\substack{v \in X \\ w \in \overline{X}}} f(v, w) \\ &= \sum_{\substack{v \in X \\ w \in V}} f(v, w) - \sum_{\substack{v \in X \\ w \in X}} f(v, w) \\ &= |f| - 0 \\ &= |f| \end{aligned}$$

where here the second sum in the second line vanishes because of skew-symmetry.

The capacity of an edge

- To make our problem realistic, we assume that each edge has a maximum capacity that it can carry.
- Therefore, the flow along that edge can be no more than the capacity of the edge.
- So there is a *capacity function* $c(e)$ which assigns to each edge a non-negative real number.
- Therefore, we insist that for each edge e,

$$0 \leq f(e) \leq c(e)$$

- If (v, w) is a (directed) edge, we also use the notation $c(v, w)$ 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 $c(v, w) = 0$.

The capacity of an edge

- We can also talk about the *capacity of a cut*: if $(X, \overline{X})$ is a cut, we define its capacity $c(X, \overline{X})$ by

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

- It can be shown that if $(X, \overline{X})$ is a cut, then

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

or equivalently,

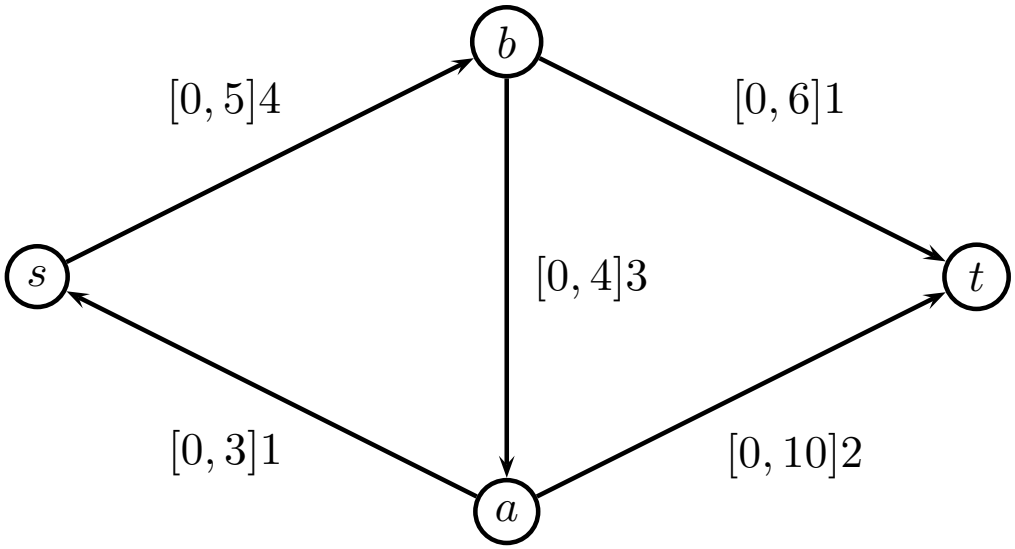
$$|f| \leq c(X, \overline{X})$$

The actual statement of the problem

- Given a flow network with a capacity function, we want to find the flow respecting that capacity and having the greatest possible value.
- We call such a flow a *maximum flow*. We're trying to find a maximum flow.

Representation of flows and capacities on a flow network

- We are actually going to represent capacities as *pairs* of numbers – one will represent the capacity in the direction of the flow, and the other will represent the capacity in the opposite direction.
- Of course the way we have set things up, the "capacity in the opposite direction" will be 0.
- We will nevertheless find this notation very useful as we g on.



Representation of flows and capacities on a flow network

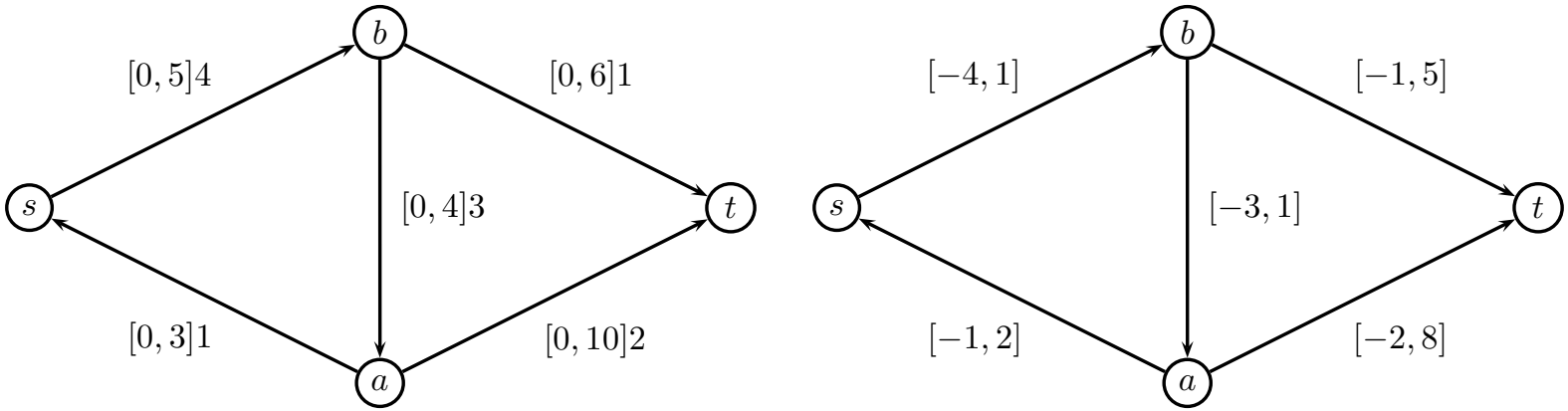
- The two numbers in square brackets represent the capacity.
- The first number is the capacity in the direction *against* the direction of the edge, and the second number is the capacity in the direction *in* the direction of the edge.
- The number after the square brackets is the actual flow.
- Of course it has to be between the two capacities, so in this case it's always positive, which means that the flow is in the direction of the corresponding edge.

Representation of flows and capacities on a flow network

- Notice also that the flow from node a to node s is "backwards".
- Of course, if we were really interested in transporting material from s to t, we wouldn't be sending back from a to s.
- But this *is* a legal flow, nonetheless, because it respects the capacity constraints.
- Finally, notice that we really do have a flow here. That is, the conservation condition is satisfied at the two nodes a and b.
- And you should be able to answer this question: What is the *value* of this flow?

The residual graph for a flow

- Given a flow on a network we can ask ourselves this question: how much could the flow on each edge change, in either direction?
- The answer for the case of the figure above is shown below.
- It is called the *residual graph* of the flow.
- Again the two numbers in brackets represent allowable values for the flow *against* and *in the direction of* the edge. Here, though, one of them can actually be negative.



The residual graph for a flow

- Where do those numbers come from? It's simple: we just take the original flow on that edge and subtract it from the two original capacities of that edge.
- So for instance the edge $s \rightarrow b$ has capacities $[0, 5]$ and flow 4.
- We subtract 4 from 0 and 5 and get $[-4, 1]$.
- Why do we say this is the range of allowable flows along that edge? Because if we take any value between -4 and 1 and add it to the original flow (which is 4), we get a number between 0 and 5, which are the capacities of that edge.
- To allow for this possibility, we are going to say that an edge in the residual graph can go in either direction.

The residual graph for a flow

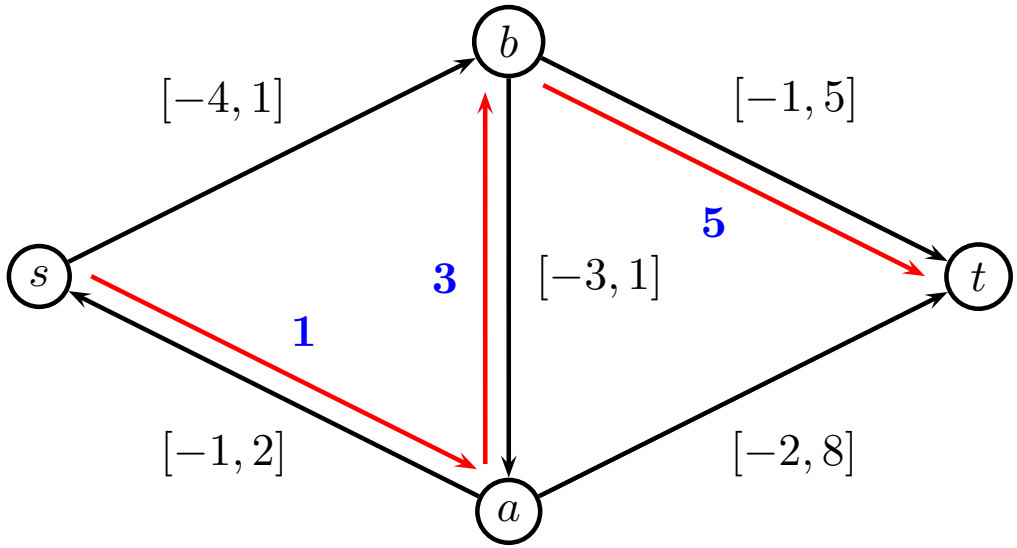
- The edges that we draw are there to show which of the two numbers we use. So if we introduce an edge from $s \rightarrow a$, that edge can carry a flow anywhere from 0 to 1.
- If we introduce an edge from $a \rightarrow s$, that edge can carry a flow anywhere from 0 to 2.
- It's important to realize that this is just something we do in the residual graph. Any edge in the original graph must be one of the edges that was there in the first place.
- We say that a flow g in the residual graph is *allowable* iff for each edge e, $g(e)$ lies in the interval defined by the pair of numbers in the residual graph corresponding to that edge.
- 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.

The residual graph for a flow

- What we need to do is to start with a flow f. Then form the residual graph for that flow.
- Then find a flow g in the residual graph such that $f + g$ has a greater value than f. And then just continue this process. That's the idea, anyway.
- Note that we could (in principle, anyway) have a flow in the residual graph whose value on the edge $a \rightarrow s$ was -1.
- That would actually be a good choice to make (if we could), because that corresponds to a flow in the opposite direction $s \rightarrow a$, and that's what we're really looking for.
- We want a flow that takes as much *from* s and transports it *to* t.

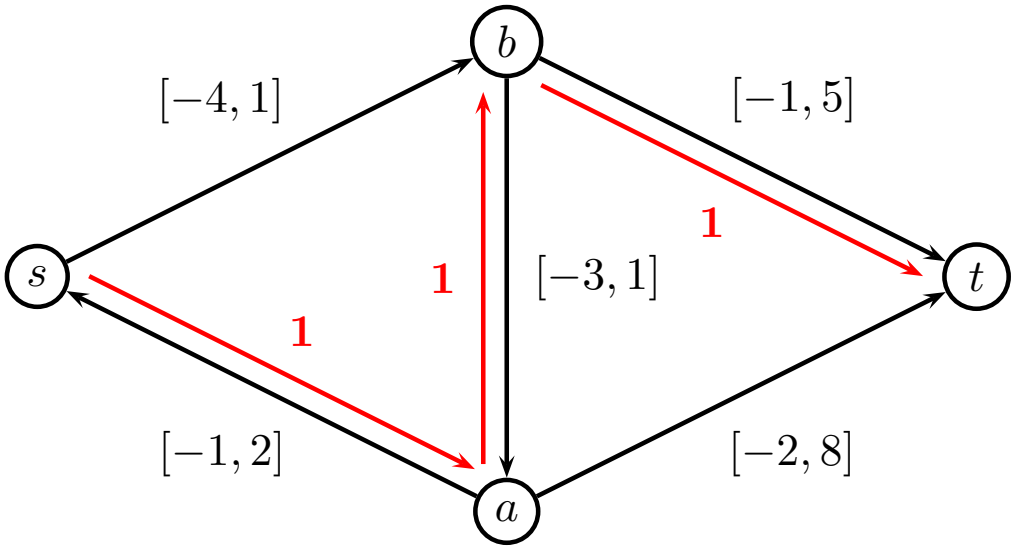
Augmenting paths

- 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.
- Suppose $s = v_0 \to v_1 \to \cdots \to v_n = t$ is a path from s to t.
- Each edge on that path can carry an amount determined by the residual capacity constraints. So for instance, let us 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 an example, not necessarily the best path to pick.
- For any such path, we let r be the minimum possible flow along the edges of that path. In this case, the minimum is $r = \min\{1, 3, 5\} = 1$
- We call this the *residual flow* along that path.
- That path with its residual flow gives the path below.
- We call this path with its associated flow values an *augmenting path*.



Augmenting paths

- We actually need to make one more restriction.
- It's simple, but it's a key constraint. We define an augmenting path as follows:

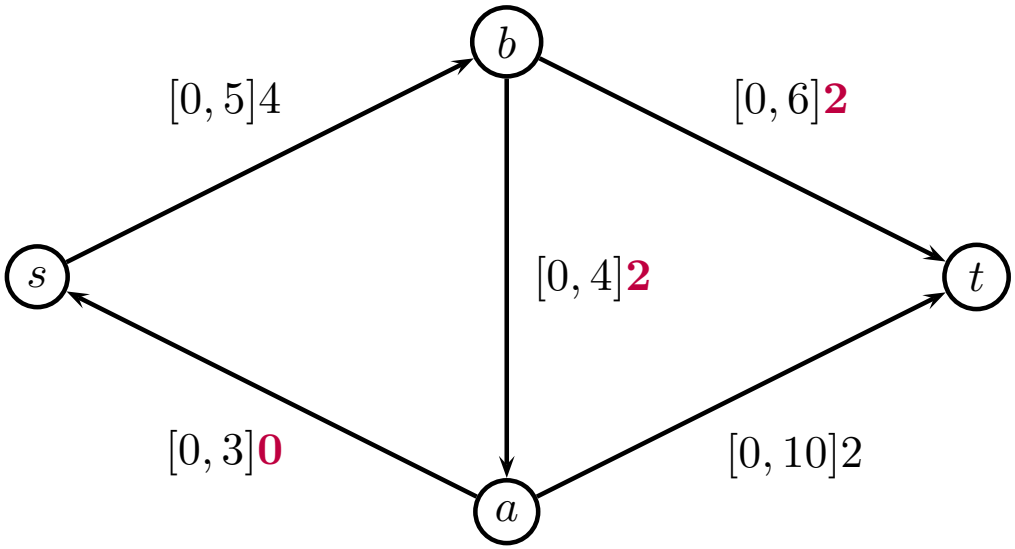
Definition

An *augmenting path* is a path from s to t with a positive residual flow.

- That is, the flow along the path has to be > 0. If the flow along the path is 0, it's not 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 figure, 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

- The second question actually has a pretty curious answer.
- But the first question can be answered right away, based on a remarkable theorem due to Ford and Fulkerson, who laid the foundations for this whole theory and wrote the original book on the subject.

Theorem (The max-flow min-cut theorem)

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

(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 (Cont.)

(3) $\Longrightarrow$ (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.

An 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.

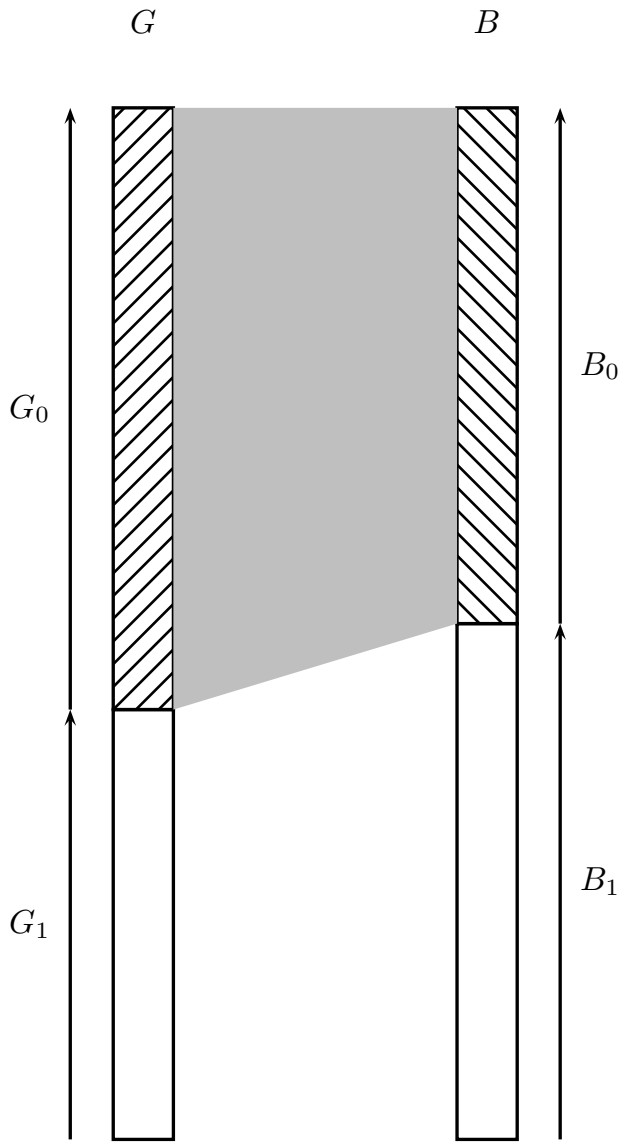
Lemma

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

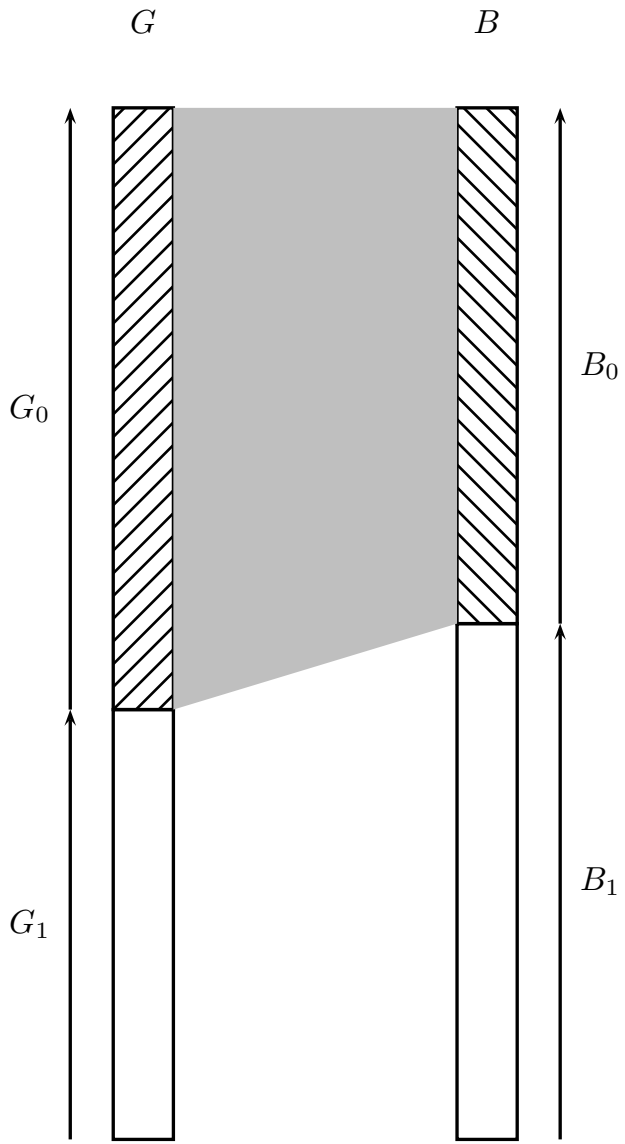
- Suppose B_0 is a set of r boys. Let G_0 be the set of girls that those boys like.
- Equivalently, G_0 is the set of girls that like any of those boys.
- We need to show that G_0 contains at least r girls.
  Let us set

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



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