CSC 373: Algorithm Design and Analysis
Lecture 10

Allan Borodin

January 30, 2013

Lecture 10: Announcements and Outline

Announcements

- Assignments due Friday
- Term test on Monday

Today's outline

- We will review:
  - Ford Fulkerson and augmenting paths (with figures now fixed).
  - Ford Fulkerson as a local search algorithm
- Discuss cuts and the max-flow min-cut theorem

Flow networks

- I will be following our old CSC364 lecture notes for the basic definitions and results concerning the computation of max flows.
- We follow the convention of allowing negative flows. While intuitively this may not seem so natural, it does simplify the development.
- The DPV and KT texts use the perhaps more standard convention of just having non-negative flows.

Definition

A flow network (more suggestive to say a capacity network) is a tuple $F = (G, s, t, c)$ where

- $G = (V, E)$ is a "bidirectional graph"
- the source s and the terminal t are nodes in V
- the capacity $c : E \to \mathbb{R}^{\geq 0}$

What is a flow?

A flow is a function $f : E \to \mathbb{R}$ satisfying the following properties:

1. Capacity constraint: for all $(u, v) \in E$,

$$f(u, v) \leq c(u, v)$$

2. Skew symmetry: for all $(u, v) \in E$,

$$f(u, v) = -f(v, u)$$

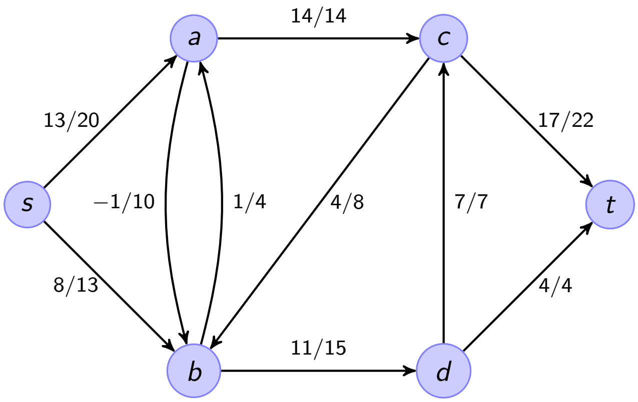
3. Flow conservation: for all nodes u (except for s and t),

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

Note

Condition 3 is the "flow in = flow out" constraint if we were using the convention of only having non-negative flows.

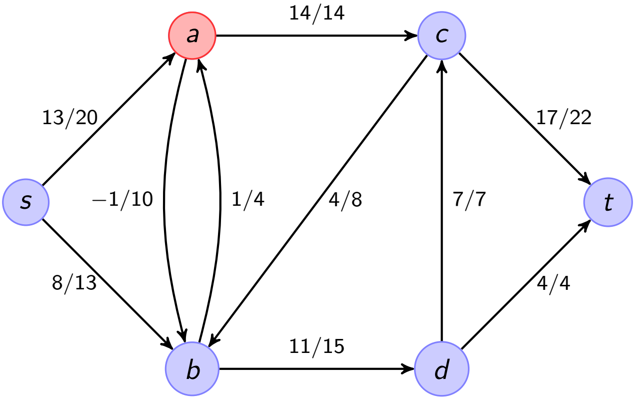
An example



The notation x/y on an edge (u, v) means

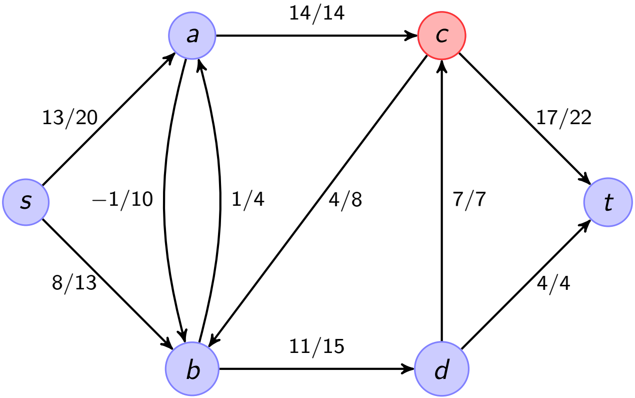
- x is the flow, i.e. $x = f(u, v)$
- y is the capacity, i.e. $x = c(u, v)$

An example of flow conservation



- For node a: $\quad f(a,s) + f(a,b) + f(a,c) = -13 + (-1) + 14 = 0$

An example of flow conservation



- For node a: $f(a,s) + f(a,b) + f(a,c) = -13 + (-1) + 14 = 0$
- For node c:

$$f(c,a) + f(c,b) + f(c,d) + f(c,t) = -14 + 4 + (-7) + 17 = 0$$

The max flow problem

The max flow problem

Given a network flow, the goal is to find a valid flow that maximizes the flow out of the source node s.

- As we will see this is also equivalent to maximizing the flow into the terminal node t. (This should not be surprising as flow conservation dictates that no flow is being stored in the other nodes.)
- We let $val(f)$ denote the flow out of the source s for a given flow f.
- We will study the Ford-Fulkerson augmenting path scheme for computing an optimal flow.
- I am calling it a "scheme" as there are many ways to instantiate this scheme although I dont view it as a general "paradigm" in the way I view (say) greedy and DP algorithms.

So why study Ford-Fulkerson?

- Why do we study the Ford-Fulkerson scheme if it is not a very generic algorithmic approach?
- As in DPV text, max flow problem can also be represented as a linear program (LP) and all LPs can be solved in polynomial time.
- I view Ford-Fulkerson and augmenting paths as an important example of a local search algorithm although unlike most local search algorithms we obtain an optimal solution.
- The topic of max flow (and various generalizations) is important because of its immediate application and many applications of max flow type problems to other problems (e.g. max bipartite matching).
  - That is many problems can be polynomial time transformed/reduced to max flow (or one of its generalizations).
  - One might refer to all these applications as “flow based methods”.

A flow f and its residual graph

- Given any flow f for a flow network $F = (G, s, t, c)$, we define the residual graph $G_f = (V, E_f)$, where
  - V is the set of vertices of the original flow network F
  - E_f is the set of all edges e having positive residual capacity

$$c_f(e) = c(e) - f(e) > 0.$$

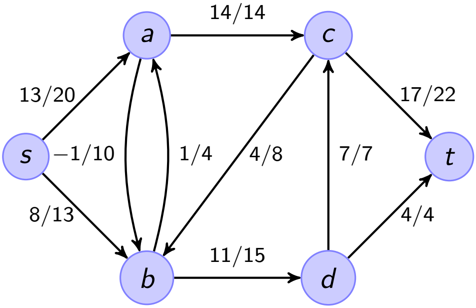
- Note that $c(e) - f(e) \geq 0$ for all edges by the capacity constraint.

Note

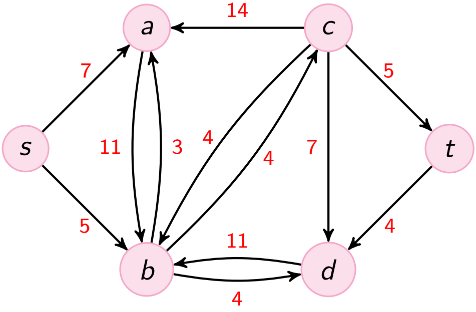
With our convention of negative flows, even a zero capacity edge (in G) can have residual capacity.

- The basic concept underlying the Ford-Fulkerson algorithm is an **augmenting path** which is an s-t path in G_f.
  - Such a path can be used to augment the current flow f to derive a better flow f'.

An example of a residual graph



The previous network flow



The residual graph

The residual capacity of an augmenting path

- Given an augmenting path π in G_f, we define its residual capacity $c_f(\pi)$ to be the

$$\min\{c(e) - c_f(e) \,|\, e \in \pi\}$$

- **Note:** the residual capacity of an augmenting path is itself is greater than 0 since every edge in the path has positive residual capacity.
- **Question:** How would we compute an augmenting path of maximum residual capacity?

Using an augmenting path to improve the flow

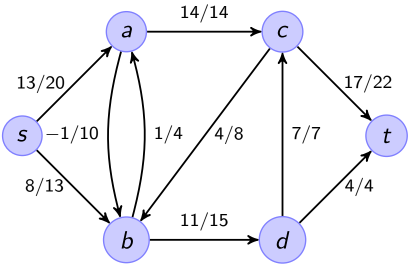
- We can think of an augmenting path as defining a flow f_π (in the "residual network"):

$$f_\pi(u,v) = \begin{cases} c_f(\pi) & \text{if } (u,v) \in \pi \\ -c_f(\pi) & \text{if } (v,u) \in \pi \\ 0 & \text{otherwise} \end{cases}$$

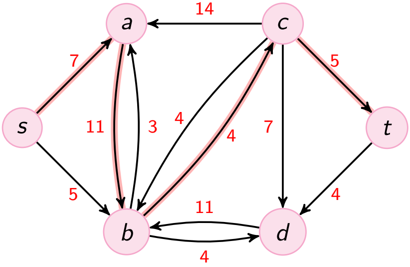
Claim

$f' = f + f_\pi$ is a flow in F and $val(f') > val(f)$

Deriving a better flow using an augmenting path

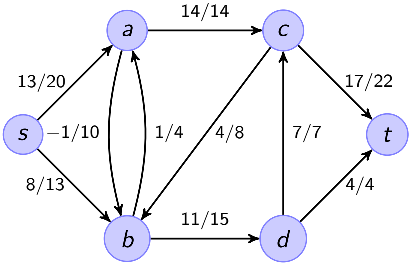


The original network flow

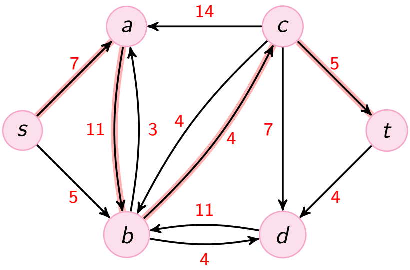


An augmenting path π with $c_f(\pi) = 4$

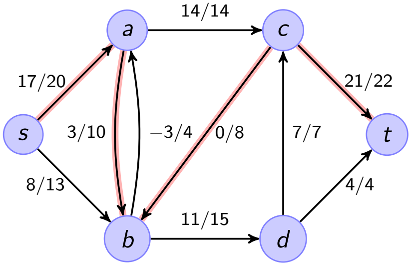
Deriving a better flow using an augmenting path



The original network flow

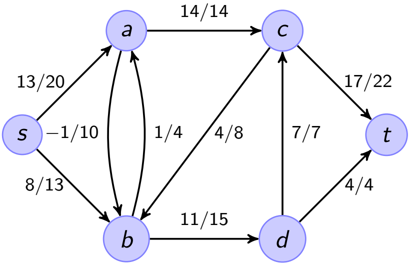


An augmenting path π with $c_f(\pi) = 4$

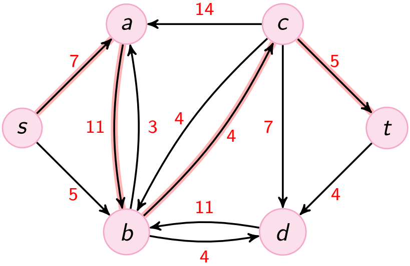


The updated flow whose value = 25

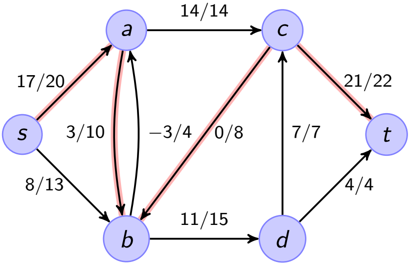
Deriving a better flow using an augmenting path



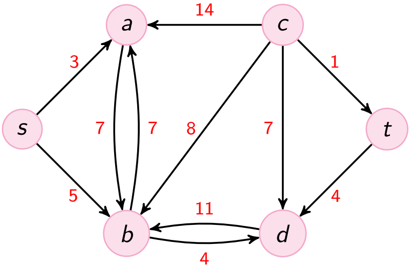
The original network flow



An augmenting path π with $c_f(\pi) = 4$



The updated flow whose value $= 25$



Updated res. graph with no aug. path

The Ford-Fulkerson scheme

The Ford-Fulkerson scheme

/ Initialize */*
$f := 0$
$G_f := G$
while there is an augmenting path π in G_f **do**
    $f := f + f_\pi$ */* Note this also changes G_f */*
end while

Note

I call this a “scheme” rather than an algorithm since we haven’t said how one chooses an augmenting path (as there can be many such paths)

Ford Fulkerson as a local search

- Local search is one of the most popular approaches for solving search and optimization problems.
- Local search is often considerd to be a "heuristic" since local search algorithms are often not analyzed but seem to often produce good results.
- For both search (i.e finding any feasible solution) and optimization, local search algorithms define some local neighborhood of a (partial) solution S, which we will denote as $Nbhd(S)$

The local search meta-algorithm

The local search meta-algorithm

Initialize S
while there is a "better" solution $S' \in Nbhd(S)$ **do**
    $S := S'$
end while

- Here "better" can mean different things.
  - For a search problem, it can mean "closer" to being feasible.
  - For an optimization problem it usually means being an improved solution.
- There are many variations of local search and we will study local search later but for now we just wish to observe the sense in which Ford-Fulkerson can be seen as a local search algorithm.
  - We start with a trivial initial solution.
  - We define the local neighbourhood of a flow f to be all flows f defined by adding the flow of an augmenting path f_π to f.

Many issues concerning local search algorithms

- How do we choose an initial solution?
- How do we define the local neighbourhood and how do we choose an $S' \in Nbhd(S)$?
- Can we guarantee that a local search algorithm will terminate? And if so, how fast will the algorithm terminate?
- Upon termination how good is the local optimum that results from a local search optimization?
- How can we escape from a local optimum (assuming it is not optimal)?

Local search issues for the Ford-Fulkerson scheme

- Does it matter how we choose an augmenting path for termination and speed of termination?

- That is, does it matter how we are choosing the $S' \in Nbhd(S)$?
  - **Answer:** YES, it matters and there are good ways to choose augmenting paths so that the algorithm is poly time.
  - Note that the Nbhd(S) here can be of exponential size but that is not a problem as long as we can efficiently search for solutions in the local neighbourhood.

- Upon termination how good is the flow?
  - **Answer:** The flow is an optimal flow. This will be proved by the max-flow min-cut theorem.
  - Note that this is unusual in that for most local search applications a local optimum is usually not a global optimum.

The max-flow min-cut theorem

- We will accept some basic facts and look at the proof of the max-flow min-cut theorem as presented in our old CSC 364 notes.
- Amongst the consequences of this theorem, we obtain that

> If any implementation of the Ford Fulkerson scheme terminates, then the resulting flow is an optimal flow.

- A cut (really an s-t cut) in a flow network is a partition (S, T) of the nodes such that $s \in S$ and $t \in T$.
- We define the capacity $c(S, T)$ of a cut as

$$\sum_{u \in S \text{ and } v \in T} c(u, v)$$

- We define the flow $f(S, T)$ across a cut as

$$\sum_{u \in S \text{ and } v \in T} f(u, v)$$

Max-flow min-cut continued

Some easy facts

- One basic fact that intuitively should be clear is that

$$f(S, T) \leq c(S, T)$$

  for all cuts (S, T) (by the capacity constraint for each edge).

- And it should also be intuitively clear that $f(S, T) = val(f)$ for any cut (S, T) (by flow conservation at each node).

- Hence for any flow f, $val(f) \leq c(S, T)$ for every cut (S, T).

The max-flow min-cut theorem

The following three statements are equivalent:

1. f is a max-flow
2. There are no augmenting paths w.r.t. flow f (i.e. no s-t path in G_f)
3. There exists some cut (S, T) satisfying $val(f) = c(S, T)$
   - Hence this cut (S, T) must be a min (capacity) cut since $val(f) \leq c(S, T)$ for all cuts.

Note

The name follows from the fact that the value of a max-flow $=$ the capacity of a min-cut

The proof outline

1 $\Rightarrow$ 2 If there is an augmenting path (w.r.t. f), then f can be increased and hence not optimal.

The proof outline

1. $\Rightarrow$ 2. If there is an augmenting path (w.r.t. f), then f can be increased and hence not optimal.

2. $\Rightarrow$ 3. Consider the set S of all the nodes reachable from s in the residual graph G_f.
   - Note that t cannot be in S and hence $(S, T) = (S, V - S)$ is a cut.
   - We also have $c(S, T) = val(f)$ since $f(u, v) = c(u, v)$ for all edges (u, v) with $u \in S$ and $v \in T$.

The proof outline

1 $\Rightarrow$ 2 If there is an augmenting path (w.r.t. f), then f can be increased and hence not optimal.

2 $\Rightarrow$ 3 Consider the set S of all the nodes reachable from s in the residual graph G_f.

- Note that t cannot be in S and hence $(S, T) = (S, V - S)$ is a cut.
- We also have $c(S, T) = val(f)$ since $f(u, v) = c(u, v)$ for all edges (u, v) with $u \in S$ and $v \in T$.

3 $\Rightarrow$ 1 Let f' be an arbitrary flow. We know $val(f') \leq c(S, T)$ for any cut (S, T) and hence $val(f') \leq val(f)$ for the cut constructed in 2.