CSC 373 Lecture 13

- Ford Fulkerson and augmenting paths
- Ford Fulkerson as a local search algorithm
- 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. In doing so we follows the convention of allowing negative flows. While intuitively this may not seem so natural, it does simplify the development.
- A flow network (more suggestive to say a capacity network) is as follows: $F = (G,s,t,c)$ where $G = (V,E)$ is a "bidirectional graph", s (the source) and t (the terminal) are nodes in V, and c is a non negative real valued function on the edges.

What is a flow?

- A flow f is a real valued function on the edges satisfying the following properties:
  1) $f(e) <= c(e)$ for all edges
  (capacity constraint)
  2) $f(u,v) = - f(v,u)$ (skew symmetry)
  3) for all nodes u (except for s and t), the sum of flows into (or out of) u is zero. (Flow conservation). Note: this is the "flow in = flow out" constraint for the convention of only having non negative flows.

The max flow problem

- The goal of the max flow problem 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 in to 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 $val(f) = |f|$ 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 don’t view it as a general “paradigm” in the way I view (say) greedy and DP algorithms.

A flow f and its residual graph

- Given any flow f for a flow network $F = (G,s,t,c)$, we can define the **residual graph** $G(f) = (V,E(f))$ where $E(f)$ is the set if all edges e having **positive residual capacity** ; i.e. residual capacity of e wrt $f = c\_f\ (e) = c(e)-f(e) > 0$.
- Note that $c(e)-f(e) >= 0$ for all edges by the capacity constraint. Also note that with our convention of negative flows, even a zero capacity edge (in G) can have residual capacity.
- The basic concept underlying Ford Fulkerson is that of 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'.

The residual capacity of an augmenting path

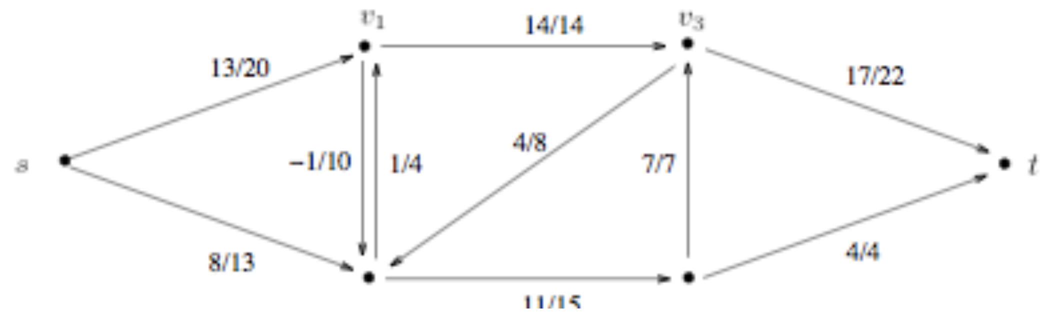
- Given an augmenting path pi in G(f), we can define its residual capacity as *c_f (pi)* (wrt *f*) to be the *min{c_f (e) | e in the path pi}.*
- Note that the residual capacity of an augmenting path is itself >0 since every edge in the path has positive residual capacity. How would you compute an augmenting path of maximum residual capacity?

Using the flow on an augmenting path (wrt a flow *f*) to improve the flow

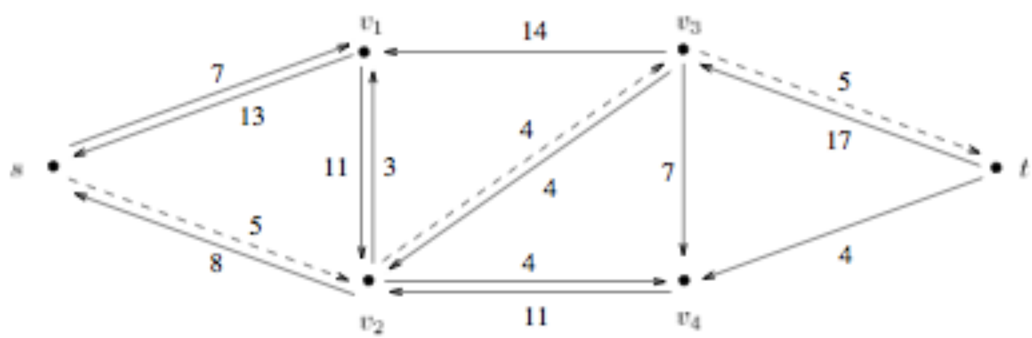
- We can think of an augmenting path as defining a flow *f_pi* (in the "residual network):
  f_pi(u,v) = *c_f (pi)* if *(u,v)* is on *pi*
  f_pi(u,v) = - *c _f (pi)* if *(v,u)* in on *pi*
  f_pi(u,v) = *0* otherwise
- Claim : *f' = f + f_pi* is a flow in *F* and *val(f') > val(f)*

The Ford Fulkerson scheme

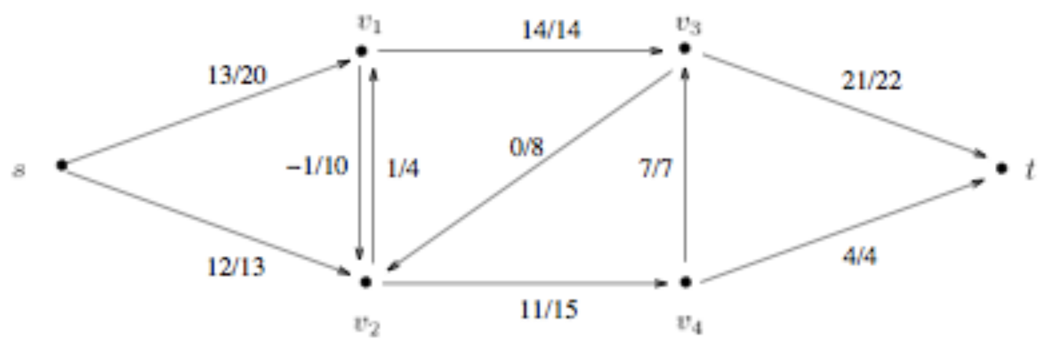
- The format if the Ford Fulkerson scheme is:
  f := 0 ; G(f) = G %initialize
  While there is an augmenting path in *G(f)*
  Choose an augmenting path pi
  f':= f+f_pi; f:=f' % Note this also changes *G(f)*
  End While
- I call this a "scheme" rather than a well specified algorithm since we have not said how one chooses an augmenting path (as there can be many such paths)



THE RESIDUAL GRAPH G_f WITH AUGMENTING PATH π:



FLOW NETWORK $\mathcal{F}$ WITH FLOW f':



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 neighbourhood of a (partial) solution *S*, which we will denote as *Nbhd(S)*

The local search meta-algorithm

- Initialize *S*
  While there is a "better solution" *S'* in *Nbhd(S)*
  S := S'
  End While
- Here "better" can mean different things. For a search problem, it can mean "closer" to being feasible (in some sense); 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. Namely, we are using a trivial initial solution and defining the local neighbourhood of a flow *f* to be all flows *f'* defined by adding the flow of an augmenting path *f_pi* to *f*.

Many issues concerning local search

- 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)?

The local search issues for Ford the 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.
- Upon termination how good is the flow?
  Answer:The flow is an optimal flow

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 the result 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 (resp.the flow *f(S,T)* across) a cut as the sum of all capacities (resp. the sum of all flows) for edges *(u,v)* with *u* in *S* and *v* in *T*

Max flow-min cut continued

One basic fact that intuitively should be clear is that *f(S,T) <= 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) <= c(S,T)* for every cut.

The Theorem

- The following are equivalent:
- 1) f is a max flow
  2) There are no augmenting paths wrt flow f; that is, no s-t path in $G(f)$
  3) $val(f) = c(S,T)$ for some cut (S,T) ; hence this cut (S,T) must be a min (capacity) cut since $val(f) <= c(S,T)$ for all cuts.
- Hence the name max flow (=) min cut

The proof

- 1) => 2) If there is an augmenting path (wrt *f* then f can be increased and hence not optimal
- 2) => 3) Consider all the nodes *S* 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 and *c(S,T) =* val(f) since *f(u,v) = c(u,v)* for all edges *(u,v)* with *u* in *S, v* in *T*
- 3) =>1) Let *f'* be an arbitrary flow. We know *val(f') <=c(S,T)* for any *cut (S,T)* and hence *val(f') <= val(f)* for the cut constructed in 2).