CSC 373: Algorithm Design and Analysis
Lecture 11

Allan Borodin

February 1, 2013

Lecture 11: Announcements and Outline

Announcements

- Term test 1 in tutorals
- I will entertain more assignment questions in Monday lecture

Today's outline

- Review the max-flow min-cut theorem
- Ways to choose an augmenting path so as to ensure polynomial time termination
- Immediate applications of the max-flow problem

The Ford-Fulkerson scheme

The Ford-Fulkerson scheme

1: /* *Initialize* */
2: $f := 0$
3: $G_f := G$
4: **while** there is an augmenting path π in G_f **do**
5: $\quad f := f + f_\pi$ /* *Note this also changes* G_f */
6: **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)

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 local neighbourhood can be say exponential size as long as we can efficiently search for a "better" solution in the 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.

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. ⇒ 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.

A consequence of the max-flow min-cut theorem

Corollary

If all capacities are integral (or rational), then any implementation of the Ford-Fulkerson algorithm will terminate with an optimal integral max flow.

Rational capacities

Why does the claim about integral capacities imply the same for rational capacities?

The runtime of Ford-Fulkerson

Observation

Each augmenting path has residual capacity at least one.

- The max-flow min-cut theorem along with the above observation ensures that with integral capacities, Ford-Fulkerson must always terminate and the number of iterations is at most:

$$C = \text{the sum of edge capacities leaving } s.$$

Notes

- There are bad ways to choose augmenting paths such that with irrational capacities, the Ford-Fulkerson algorithm will not terminate.
- However, even with integral capacities, there are bad ways to choose augmenting paths so that the Ford-Fulkerson algorithm does not terminate in polynomial time.

Bad example for naive Ford-Fulkerson

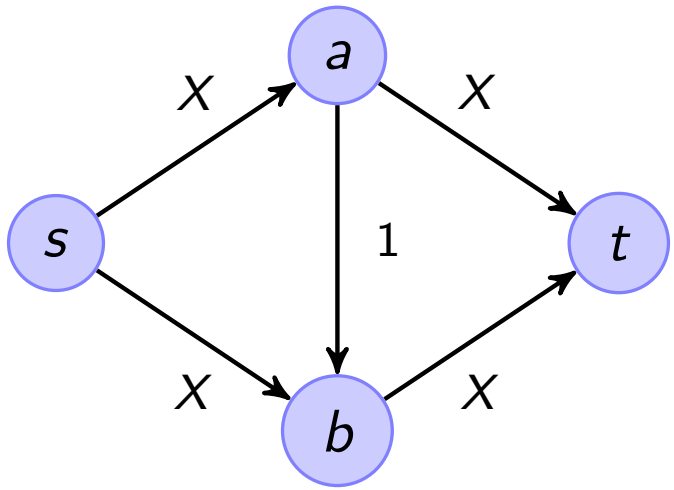


Figure: The numbers denote the capacities of the edges.

- The max-flow is clearly $2X$.
- A naive Ford-Fulkerson algorithm could alternate between
  - pushing a 1 unit flow along the augmenting path $s \to a \to b \to t$
  - pushing a 1 unit flow along the augmenting path $s \to b \to a \to t$
- This would resul in $2X$ iterations, which is exponential if say X is given in binary.

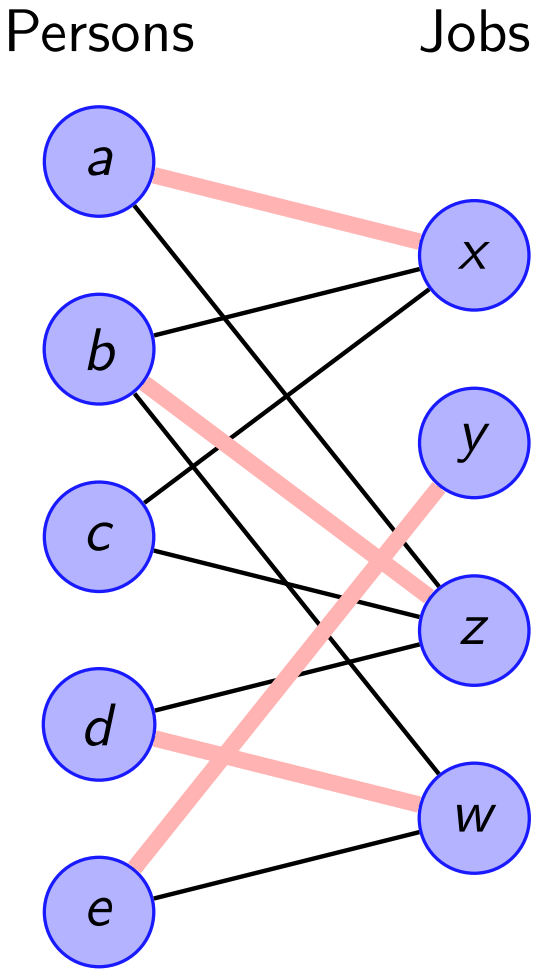
Some ways to achieve polynomial time

- Choose an augmenting path having shortest distance: This is the Edmonds-Karp method and can be found in CLRS. It has running time $O(nm^2)$, where $n = |V|$ and $m = |E|$.
- There is a "weakly polynomial time" algorithm in KT
  - Here the number of arithmetic operations depends on the length of the integral capacities.
  - It follows that always choosing the largest capacity augmenting path is at least weakly polynomial time.
- There is a history of max flow algorithms leading to a recent $O(mn)$ time algorithm (see http://tinyurl.com/bczkdfz).
- Although not the fastest, next lecture I will present Dinitz's algorithm which has runtime $O(n^2m)$.
  - A shortest augmenting-path method based on the concept of a blocking flow in the leveled graph.
  - Has an additional advantage (i.e. an improved bipartite matching bound) beyond the somewhat better running time of Edmonds-Karp.

An application of max-flow: the maximum bipartite matching problem

The maximum bipartite matching problem

- Given a bipartite graph $G = (V, E)$ where
  - $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \varnothing$
  - $E \subseteq V_1 \times V_2$
- **Goal:** Find a maximum size matching.



- We do not know any efficient DP or greedy optimal algorithm for solving this problem.
- But we can efficiently reduce this problem to the max-flow problem.

The reduction

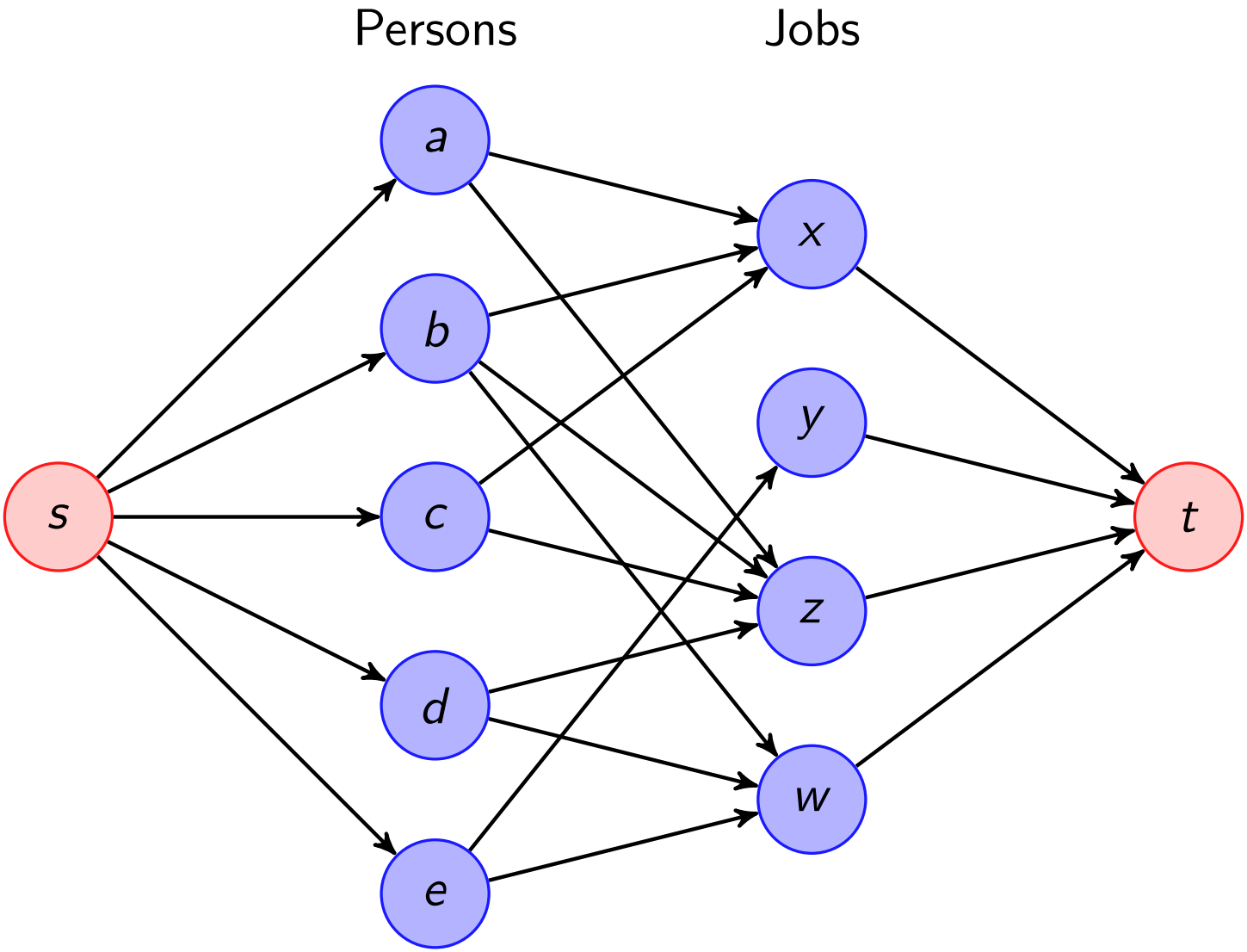


Figure: Assign every edge of the network flow a capacity 1.

The reduction preserves solutions

Claims

1. Every matching M in G gives rise to an integral flow f_M in the newly constructed network flow F_G with $val(f_M) = |M|$
2. Conversely every integral flow f in F_G gives rise to a matching M_f in G with $|M_f| = val(f)$.

- Time complexity: $O(mn)$ using any Ford Fulkerson algorithm since the max flow in at most n and all capacities are integral.
- Dinitz's algorithm can be used to obtain a runtime $O(m\sqrt{n})$.