MATH 566: Lecture 18 (10/17/2024)

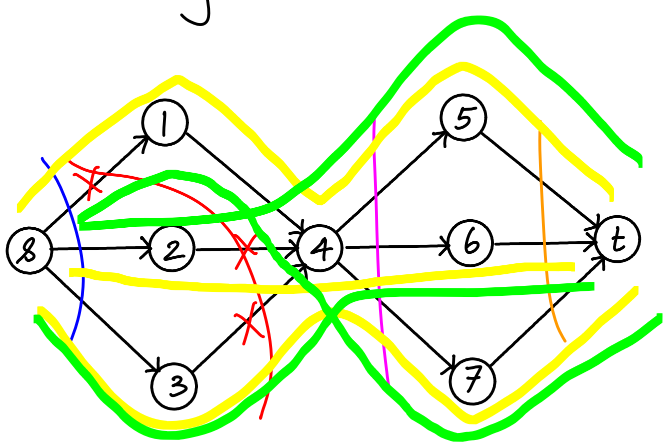
Today: * application of MFMC in network reliability
* augmenting path algorithm

Applications of MFMC in Network Reliability

In a communication network, we might want to have more than one s-t path to send info/packets. We want to count the number of arc-disjoint s-t paths in G. Same scenario arises in road networks too.

Q. What is the maximum number of arc-disjoint paths from s to t?

Def Two s-t paths are **arc-disjoint** if they do not share any arcs. They could share nodes.



There are 3 arc-disjoint paths (2 sets of 3 such paths are highlighted). Several min-cuts are also shown.

Theorem The maximum number of arc-disjoint s-t paths in a directed network G is equal to the minimum number of arcs up on whose removal there is no directed s-t path.

Proof Set $u_{ij} = 1 \ \forall (i,j) \in A$, and apply MFMC theorem! $\square$

It is much harder to prove this result from first principles, or using other techniques. But it follows directly from the application of MFMC!

We consider also a stronger notion of independence of paths — being node-disjoint.

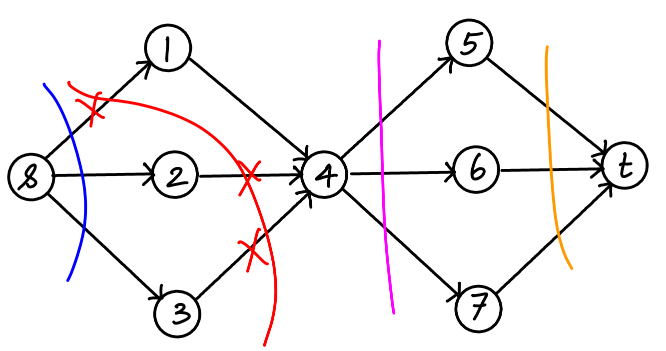
Node-disjoint Paths

Def Two s-t paths are **node-disjoint** if the only nodes they share are s and t.

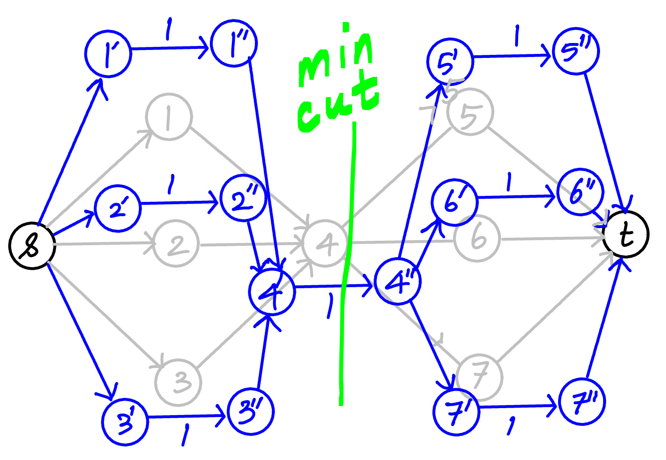
There are no groups of node-disjoint s-t paths in the previous example except for isolated s-t paths. Notice that each s-t path goes through node 4.

Theorem Let G have ~~not~~ no (s,t) arc. The maximum number of node-disjoint s-t paths is equal to the minimum number of nodes whose removal leaves no directed s-t paths.

Proof Can use node-splitting. Removing node i is then equivalent to removing arc (i', i''). Set $u_{i',i''} = 1$, and apply MFMC. Set $u_{ij} = \infty$ for all other arcs (i,j). □



Here, the min # nodes whose removal disconnects s and t is 1 — remove node 4. Equivalently, the # node-disjoint s-t paths is also 1.



We now return to algorithms for max flow. We first describe polynomial implementations of the augmenting path algorithm.

The *Generic Augmenting Path Algorithm* (Ford-Fulkerson)

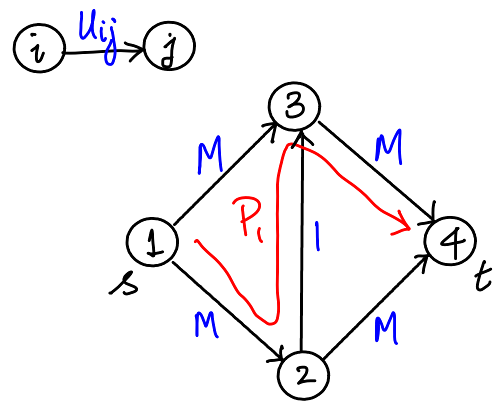
Assume $l_{ij} = 0 \ \forall (i,j) \in A$.

begin
  $\bar{x} := \bar{0}$;
  initialize $G(\bar{x})$;
  while $G(\bar{x})$ has a path P from s to t **do**
    augment $\delta(P)$ units of flow along P;
    update $\bar{x}$, $G(\bar{x})$;
  end_while
end_begin

We saw this algorithm terminates in $O(nU)$ augmentations. In the worst case, we perform updates on every arc following each augmentation. Hence the algorithm runs in $O(mnU)$ time.

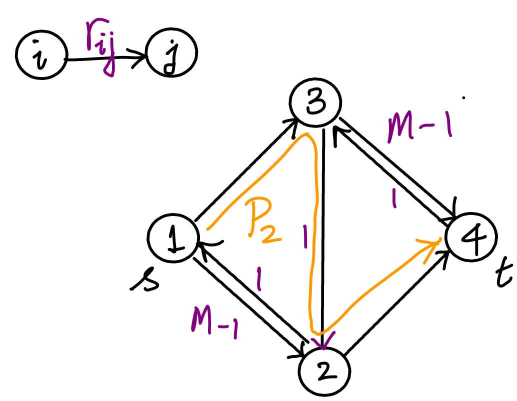
A pathological example

Consider the network on the right.



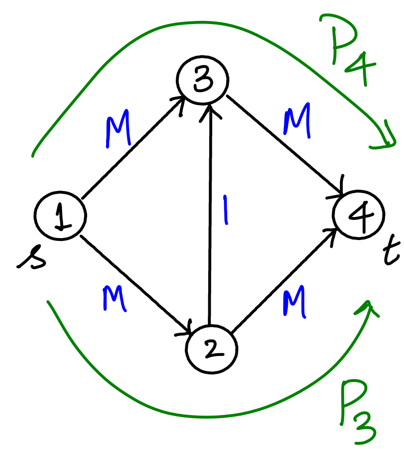
We start with $\bar{x} = \bar{0}$, and can choose $P_1 = 1\text{-}2\text{-}3\text{-}4$, with $\delta(P_1) = 1$, and augment along P_1.

Then we can choose $P_2 = 1\text{-}3\text{-}2\text{-}4$ with $\delta(P_2) = 1$, and augment. Then pick P_1, and alternately P_2, and so on. In this way, we will find the max-flow in $2M$ augmentations.



But we could have found the max flow in just 2 augmentations by choosing $P_3 = 1\text{-}2\text{-}4$, $\delta(P_3) = M$ and $P_4 = 1\text{-}3\text{-}4$, $\delta(P_4) = M$.

Also, if data (i.e., u_{ij}'s) are irrational, the generic algo might converge to a suboptimal solution.



We need to select augmenting paths carefully!

1. Largest augmenting path algo: Select augmenting path P with largest $\delta(P)$.
2. Shortest augmenting path algo: Select augmenting path P with smallest number of arcs.

We will look at option 2 first.

Shortest Augmenting Path (SAP) Algorithm

* Use distance labels $d(i)$; time for each augmentation: $O(n)$
* total number of augmentations : $O(mn)$

  Total complexity : $O(n^2 m)$.

Def (Distance labels)

A set of distance labels $d: N \to \mathbb{Z}^n_{\geq 0}$ is **valid** with respect to flow $\bar{x}$ if

(1) $d(t) = 0$, and

(2) $d(i) \leq d(j) + 1 \quad \forall (i,j) \in G(\bar{x})$.

Property 7.1 If the distance labels are valid, then $d(i)$ is a **lower bound** on the length (in terms of # arcs) of the SP from i to t in $G(\bar{x})$.

Proof We add the validity conditions for all arcs in any i-t path P, say, with k arcs.

$$\underbrace{i \to i_2 \to i_3 \to \cdots \to i_k \to t}_{k \text{ arcs}}$$

We get
$$d(i) \le d(t) + k = k \quad (\text{as } d(t)=0).$$

This result holds for all i-t paths, and hence $d(i)$ is a lower bound. $\square$

Property 7.2 If $d(s) \geq n$, there is no directed s-t path in $G(\bar{x})$.

Intuition for $d(i)$: Think of $d(i)$ as the height above ground level that node i has to be raised for flow to happen "freely". Node t is at ground level ($d(t)=0$), and s need not be raised above level $n-1$ from the ground.

We now look for candidate arcs in $G(\bar{x})$ that could be part of augmenting path(s). Recall the notion of admissible arcs in search (BFS/DFS) — we redefine admissibility here.

Def An arc $(i,j) \in G(\bar{x})$ is **admissible** if $d(i) = d(j) + 1$.

A path from s to t consisting of only admissible arcs is an **admissible path**.