ALGORITHM DESIGN NETWORK FLOWS

by

David Toman

University of Waterloo

List of Slides

1
2 Network Flows
3 Network Flows (cont.)
4 Solution(s)
5 Linear Programming
6 Augmentation
7 Augmentation (cont.)
8 Complexity of Augmentation
9 Summary

Network Flows

Def: A *flow* for a given directed graph $G = (V, E)$ with positive (integer) *capacities* $c(e)$ assigned to edges is a function $f : E \to N$ such that

- $f(e) \leq c(e)$ for all $e \in E$
- for each node (with the exception of u and v the sums of incoming and outgoing flows are equal.
- there is no incoming/outgoing flows into/from u/v.

Network Flows (cont.)

PROBLEM ("max network flow"):

Input: a directed graph $G = (V, E)$,
positive (integer) edge values (*edge capacities*) $c(e)$,
two nodes $u, v \in V$ (the *source* and the *sink*).

Output: a flow f for G with maximum value.

Solution(s)

- linear programming

  $\Rightarrow$ Simplex/Elliptical method (works for rationals)
  $\Rightarrow$ HARD for integer solutions

- specialized algorithms

  $\Rightarrow$ integral solutions in PTIME

  1. augmentation approach
  2. “push” methods

Linear Programming

IDEA: formulate as an *constraint optimization problem*

$\Rightarrow$ linear programming techniques

$$\max \sum_{(u,w)\in E} f((u,w))$$

with respect to:

- $f((x,y)) \leq c((x,y))$ for $(x,y) \in E$,
- $\sum_{(x,y)\in E} f((x,y)) = \sum_{(y,z)\in E} f((y,z))$ for $y \in N - \{u,v\}$,
- $\sum_{(v,y)\in E} f((v,y)) = \sum_{(y,u)\in E} f((y,u)) = 0$

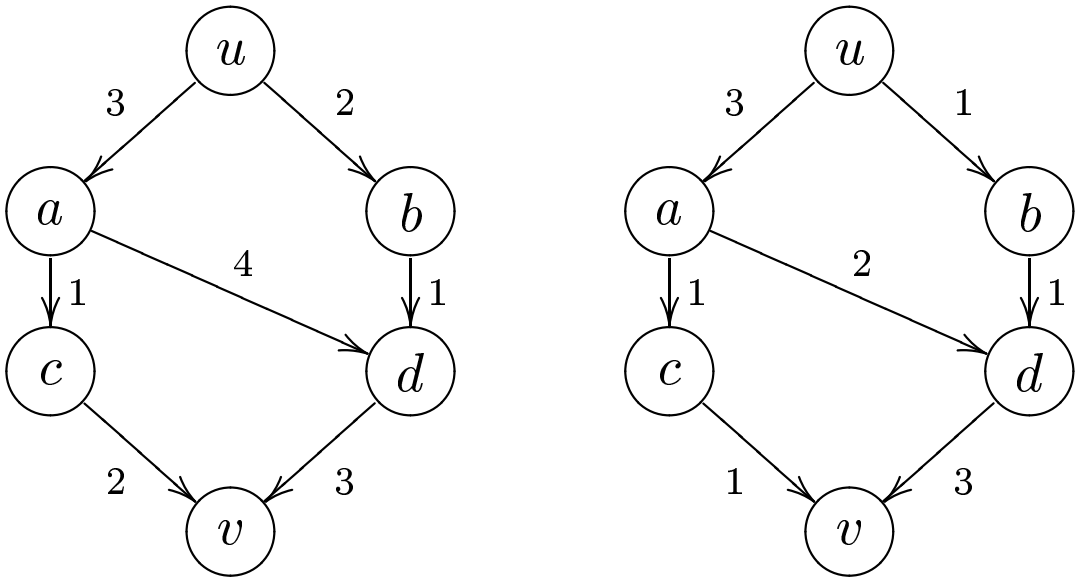
Augmentation

IDEA: find $u - v$ paths in G
$\Rightarrow$ add their "value" to the flow (F)
$\Rightarrow$ "subtract" their value **carefully** from G

Algorithm (Ford-Fulkerson):

$F := G$ with edge labels set to 0
while G contains a path p from u to v
  let x be the minimal edge on p
    add flow p with value x to F
    augment G using p and x

Augmentation (cont.)



Paths (explored in this order):

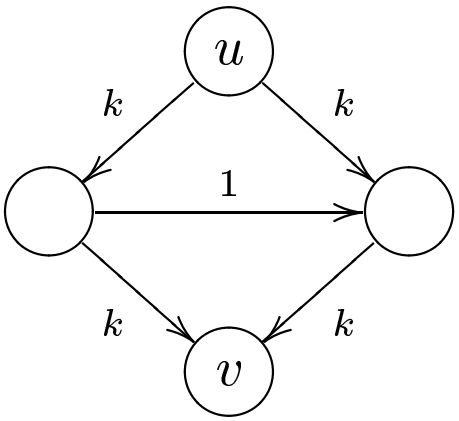
1. $u - a - c - v$, value 1
2. $u - b - d - v$, value 1
3. $u - a - d - v$, value 2 (note: only 2 units left in G!)

What if we started with $u - a - d - v$ (value 3)?

$\Rightarrow$ need to introduce "back-flows" in the residual graph

Complexity of Augmentation

Assume (worst case) that the algorithm always picks a path through the middle edge:



$\Rightarrow$ it needs k augmentation steps

$\Rightarrow$ $O(k \cdot |E|)$ — *pseudo*-polynomial

$\Rightarrow$ pick the "fattest" path

only $O(|E| \log k)$ augmentations only $O(|V||E| \log k)$ total

Summary

- Applications:
  - $\Rightarrow$ Traffic (transportation) problems
  - $\Rightarrow$ Network Design
  - $\Rightarrow$ Tournaments
- min-cut max-flow duality
  - $\Rightarrow$ many other graph problems