Maximum Flow

Chapter 26

Flow Graph

- A common scenario is to use a graph to represent a “flow network” and use it to answer questions about material flows
- Flow is the rate that material moves through the network
- Each directed edge is a conduit for the material with some stated capacity
- Vertices are connection points but do not collect material
  - Flow into a vertex must equal the flow leaving the vertex, flow conservation

Sample Networks

Network	Nodes	Arcs	Flow
communication	telephone exchanges, computers, satellites	cables, fiber optics, microwave relays	voice, video, packets
circuits	gates, registers, processors	wires	current
mechanical	joints	rods, beams, springs	heat, energy
hydraulic	reservoirs, pumping stations, lakes	pipelines	fluid, oil
financial	stocks, companies	transactions	money
transportation	airports, rail yards, street intersections	highways, railbeds, airway routes	freight, vehicles, passengers
chemical	sites	bonds	energy

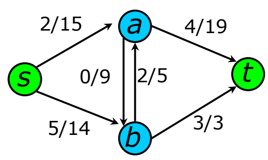
Flow Concepts

- Source vertex s
  - where material is produced
- Sink vertex t
  - where material is consumed
- For all other vertices – what goes in must go out
  - Flow conservation
- **Goal: determine maximum rate of material flow from source to sink**

Formal Max Flow Problem

– Graph $G=(V,E)$ – a **flow network**
  - Directed, each edge has **capacity** $c(u,v) \geq 0$
  - Two special vertices: ***source*** *s*, and ***sink*** *t*
  - For any other vertex *v*, there is a path $s \to \ldots \to v \to \ldots \to t$

– **Flow** – a function $f : V \times V \to \boldsymbol{R}$
  - *Capacity constraint*: For all $u, v \in V$: $f(u,v) \leq c(u,v)$
  - *Skew symmetry*: For all $u, v \in V$: $f(u,v) = -f(v,u)$
  - *Flow conservation:* For all $u \in V - \{s, t\}$:

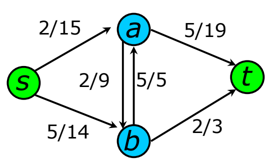


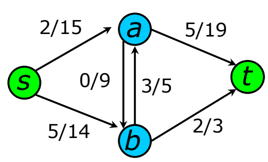
$$\sum_{v \in V} f(u,v) = f(u,V) = 0, \text{ or}$$

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

Cancellation of flows

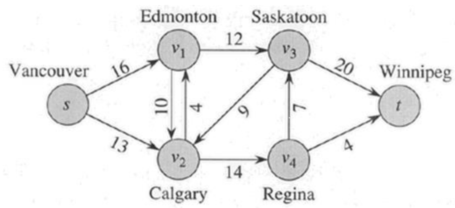
- We would like to avoid two positive flows in opposite directions between the same pair of vertices
  – Such flows *cancel* (maybe partially) each other due to skew symmetry



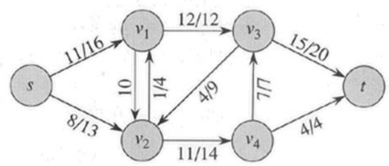


Max Flow

- We want to find a flow of maximum value from the source to the sink
  - Denoted by |f|



Lucky Puck Distribution Network



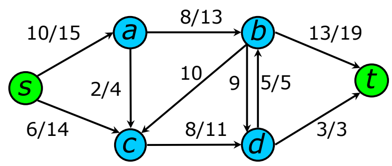
Max Flow, |f| = 19
Or is it?
Best we can do?

Ford-Fulkerson method

- Contains several algorithms:
  - Residue networks
  - Augmenting paths
    - Find a path *p* from *s* to *t* (**augmenting path**), such that there is some value *x* > 0, and for each edge *(u,v)* in *p* we can add *x* units of flow
      - $f(u,v) + x \leq c(u,v)$

```
FORD-FULKERSON-METHOD(G, s, t)
initialize flow f to 0
while there exists an augmenting path p
    do augment flow f along p
return f
```

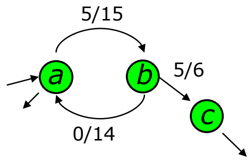
Augmenting Path?



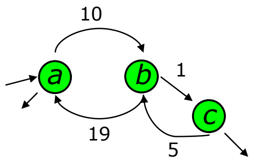
Residual Network

- To find augmenting path we can find any path in the ***residual network***:
  - Residual capacities: $c_f(u,v) = c(u,v) – f(u,v)$
    - i.e. the actual capacity minus the net flow from u to v
    - Net flow may be negative
  - Residual network: $G_f = (V,E_f)$, where
    $E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$
  - Observation – edges in E_f are either edges in E or their reversals: $|E_f| \leq 2|E|$

Sub-graph With c(u,v) and f(u,v)

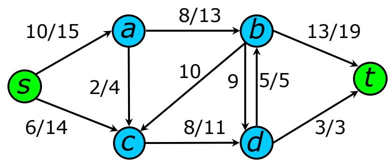


Residual Sub-Graph



Residual Graph

- Compute the residual graph of the graph with the following flow:



Residual Capacity and Augmenting Path

- Finding an Augmenting Path
  - Find a path from *s* to *t* in the residual graph
  - The r*esidual capacity* of a path *p* in G_f:
    $c_f(p) = \min\{c_f(u,v): (u,v) \text{ is in } p\}$
    - i.e. find the minimum capacity along *p*
  - Doing augmentation: for all (*u,v*) in *p*, we just add this $c_f(p)$ to $f(u,v)$ (and subtract it from $f(v,u)$)
  - Resulting flow is a valid flow with a larger value.

Residual network and augmenting path

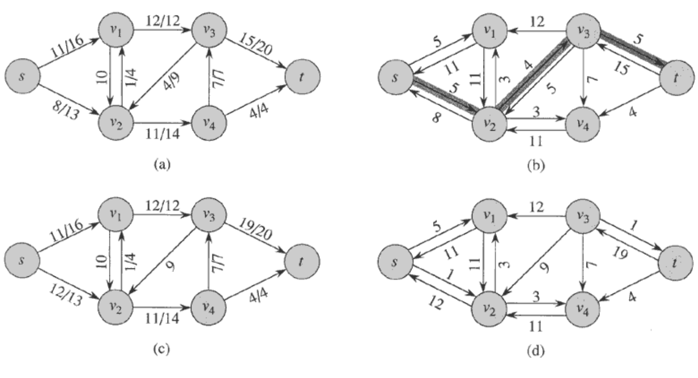


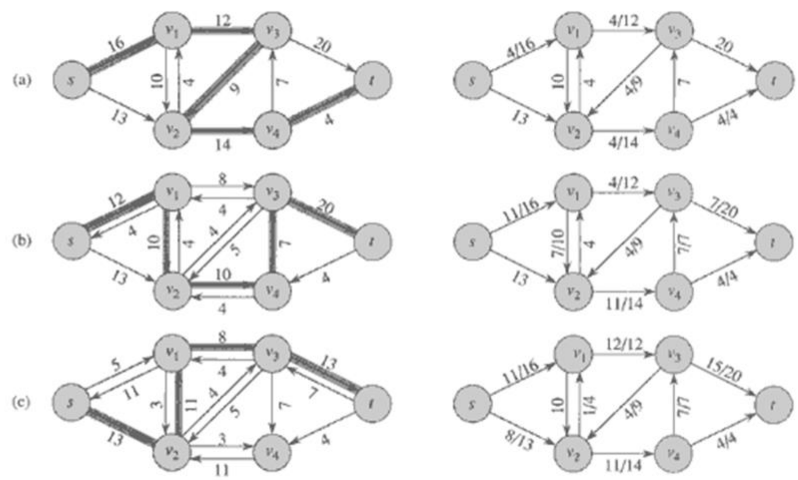
Figure 26.3 **(a)** The flow network G and flow f of Figure 26.1(b). **(b)** The residual network G_f with augmenting path p shaded; its residual capacity is $c_f(p) = c(v_2, v_3) = 4$. **(c)** The flow in G that results from augmenting along path p by its residual capacity 4. **(d)** The residual network induced by the flow in (c).

The Ford-Fulkerson method

```
Ford-Fulkerson(G,s,t)
1 for each edge (u,v) in G.E do
2    f(u,v) ← f(v,u) ← 0
3 while there exists a path p from s to t in residual
   network G_f do
4    c_f = min{c_f(u,v): (u,v) is in p}
5    for each edge (u,v) in p do
6        f(u,v) ← f(u,v) + c_f
7        f(v,u) ← -f(u,v)
8 return f
```

The algorithms based on this method differ in how they choose p in step 3.
If chosen poorly the algorithm might not terminate.

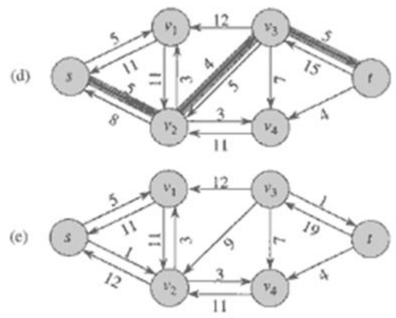
Execution of Ford-Fulkerson (1)

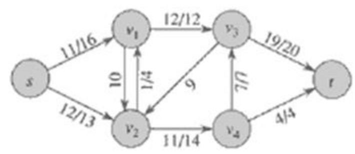


Left Side = Residual Graph

Right Side = Augmented Flow

Execution of Ford-Fulkerson (2)



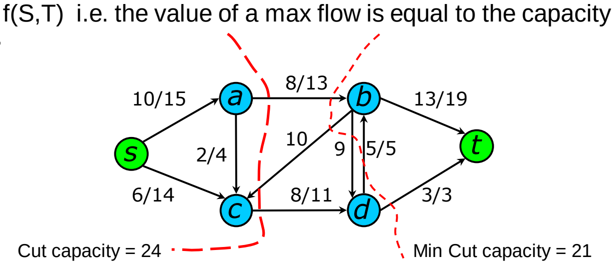


Left Side = Residual Graph

Right Side = Augmented Flow

Cuts

- Does the method find the minimum flow?
  - Yes, if we get to the point where the residual graph has no path from *s* to *t*
  - A **cut** is a partition of V into S and T = V – S, such that *s* ∈ S and *t* ∈ T
  - The **net flow** (f(S,T)) through the cut is the sum of flows f(u,v), where *s* ∈ S and *t* ∈ T
    - Includes negative flows back from T to S
  - The **capacity** (c(S,T)) of the cut is the sum of capacities c(u,v), where *s* ∈ S and *t* ∈ T
    - The sum of positive capacities
  - **Minimum cut** – a cut with the smallest capacity of all cuts.
    |f|= f(S,T) i.e. the value of a max flow is equal to the capacity of a min cut.



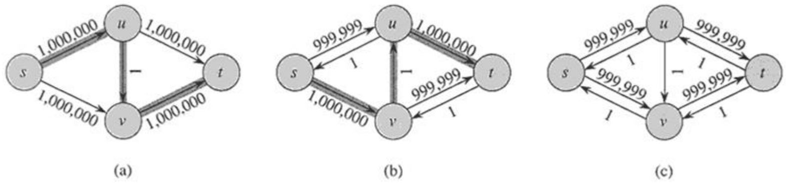
Max Flow / Min Cut Theorem

1. Since $|f| \leq c(S,T)$ for all cuts of (S,T) then if $|f| = c(S,T)$ then c(S,T) must be the min cut of G
2. This implies that f is a maximum flow of G
3. This implies that the residual network G_f contains no augmenting paths.
   - If there were augmenting paths this would contradict that we found the maximum flow of G

- 1→2→3→1 … and from 2→3 we have that the Ford Fulkerson method finds the maximum flow if the residual graph has no augmenting paths.

Worst Case Running Time

- Assuming integer flow
- Each augmentation increases the value of the flow by some positive amount.
- Augmentation can be done in O(E).
- Total worst-case running time O(E|f*|), where f* is the max-flow found by the algorithm.
- Example of worst case:



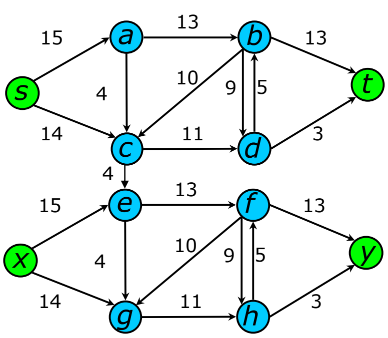
Augmenting path of 1 | Resulting Residual Network | Resulting Residual Network

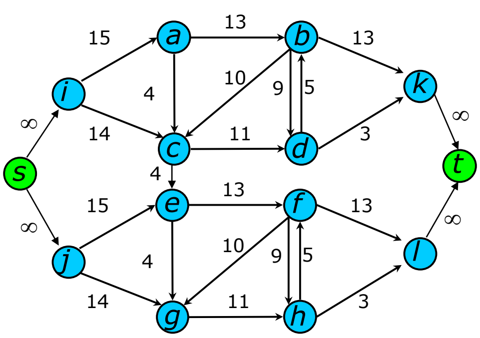
Edmonds Karp

- Take **shortest path** (in terms of number of edges) as an augmenting path – Edmonds-Karp algorithm
  - How do we find such a shortest path?
  - Running time $O(VE^2)$, because the number of augmentations is $O(VE)$
  - Skipping the proof here
  - Even better method: push-relabel, $O(V^2E)$ runtime

Multiple Sources or Sinks

- What if you have a problem with more than one source and more than one sink?
- Modify the graph to create a single supersource and supersink



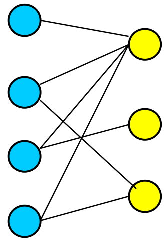


Application – Bipartite Matching

- Example – given a community with *n* men and *m* women
- Assume we have a way to determine which couples (man/woman) are compatible for marriage
  - E.g. (Joe, Susan) or (Fred, Susan) but not (Frank, Susan)
- Problem: Maximize the number of marriages
  - No polygamy allowed
  - Can solve this problem by creating a flow network out of a bipartite graph

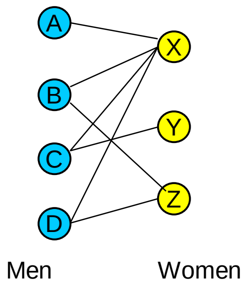
Bipartite Graph

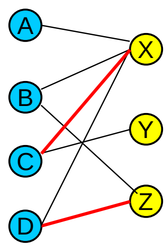
- A bipartite graph is an undirected graph G=(V,E) in which V can be partitioned into two sets V_1 and V_2 such that (u,v) $\in$ E implies either u $\in$ V_1 and v $\in$ V_{12} or vice versa.
- That is, all edges go between the two sets V_1 and V_2 and not within V_1 and V_2.



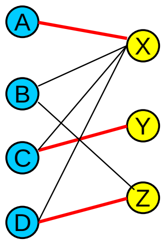
Model for Matching Problem

- Men on leftmost set, women on rightmost set, edges if they are compatible





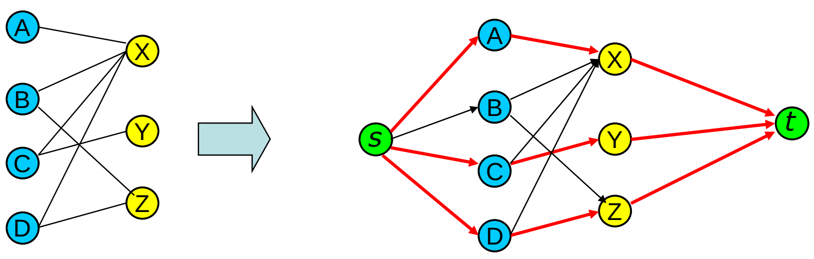
A matching



Optimal matching

Solution Using Max Flow

- Add a supersouce, supersink, make each undirected edge directed with a flow of 1



Since the input is 1, flow conservation prevents multiple matchings