COT 6405
ANLYSIS OF ALGORITHMS

Maximum Flow

Computer & Electrical Engineering and Computer Science Department
Florida Atlantic University

Spring 2017

Outline

- Flow networks
- Ford-Fulkerson method
- Edmonds-Karp algorithm
- Applications:
    - Maximum bipartite matching
    - Disjoint paths in directed and undirected graphs
- Extensions:
    - Circulations with demands
    - Circulations with demands and lower bounds
- Application:
    - Survey design
    - Natural disaster scenario

$\}$ • Reference: CLRS ch 26

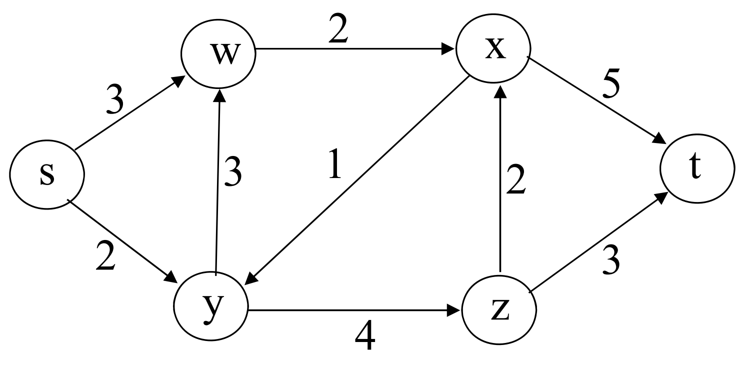
$\}$ • Reference: KT ch 9

CLRS = *Introduction to Algorithms*, 3rd edition, by T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, The MIT Press, 2009.
KT = *Algorithm Design*, J. Kleinberg and E. Tardos, Addison-Wesley Publishing Company, 2006.

Flow Networks, motivation

- Use of graphs to model *transportation networks*
  - edges carry some sort of traffic
  - nodes act as “switches”, passing traffic between edges
- Examples
  - highway system: edges are highways, nodes are interchanges
  - computer network: edges are links that carry packets, nodes are switches
  - fluid network: edges are pipes that carry liquid, and nodes are junctures where pipes are plugged together
- Such network models have several ingredients:
  - capacities of the edges: how much they can carry
  - source nodes: nodes that generate traffic
  - sink (or destination) nodes: nodes that “absorb” traffic
  - traffic transmitted across edges

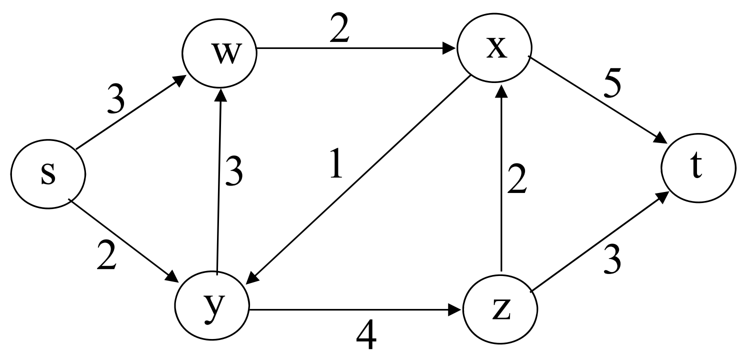
Flow Network, definition



Flow network:

- directed graph G = (V, E)
- each edge (u,v) has a capacity c(u,v) ≥ 0
- if (u,v) ∈ E, then (v, u) ∉ E
- if (u, v) ∉ E, then we define c(u, v) = 0
- no self-loops
- **source vertex** s, **sink vertex** t
- for each vertex v ∈ V, there is a path $s \rightsquigarrow v \rightsquigarrow t$

Flow network



- The graph of a flow network is connected

$$|E| \geq |V| - 1$$

Flow definition

Given:

- G is a flow network, with capacity function c
- s is the source, t is the sink

A **flow** is a function f : V $\times$ V $\rightarrow$ R, which satisfies the properties:

- **Capacity constraint**: for all u, v $\in$ V

$$0 \leq f(u,v) \leq c(u,v)$$

- **Flow conservation**: for all u $\in$ V-{s, t}

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

"flow in equals flow out"

when (u,v) $\notin$ E, f(u, v) = 0

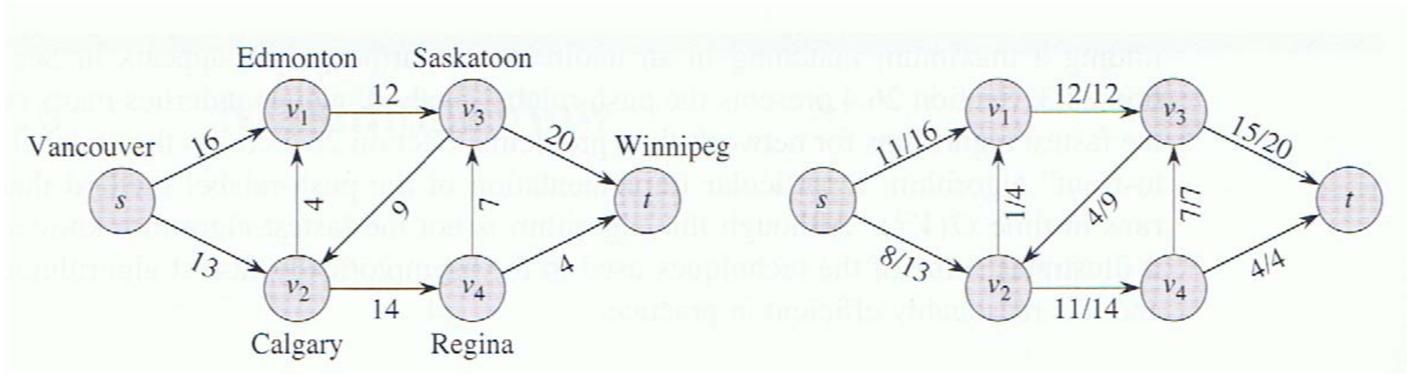
Maximum-flow problem

- f(u,v) – the flow from u to v
- The **value of a flow** is defined as:

$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$$

Maximum-flow problem: given a flow network G with source s and sink t, find a flow of maximum value.

Example

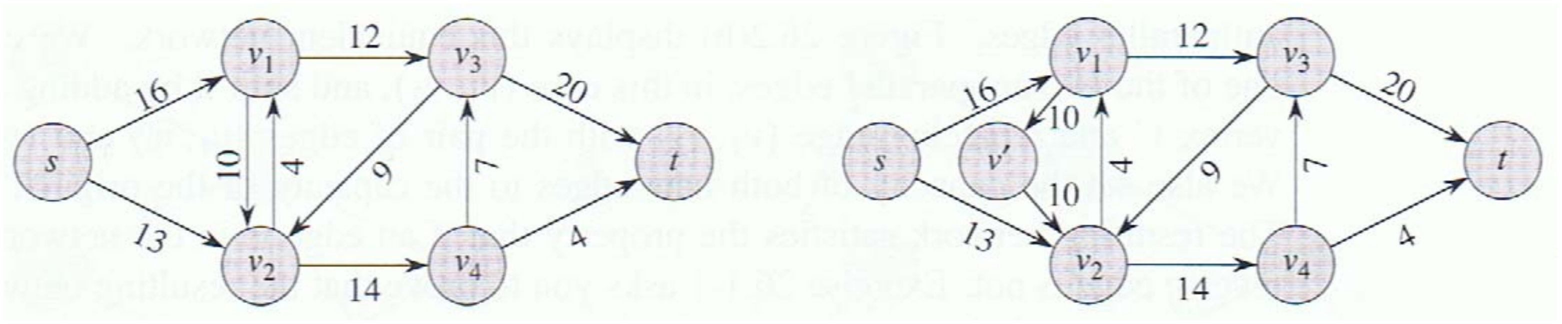


Flow network

A flow with value |f| = 19

- Lucky Puck Company ships hockey pucks from the factory, source city s to the warehouse, city t
- capacity c(u, v) – limit on the number of crates per day that can go from city u to city v
- objective: maximize the flow

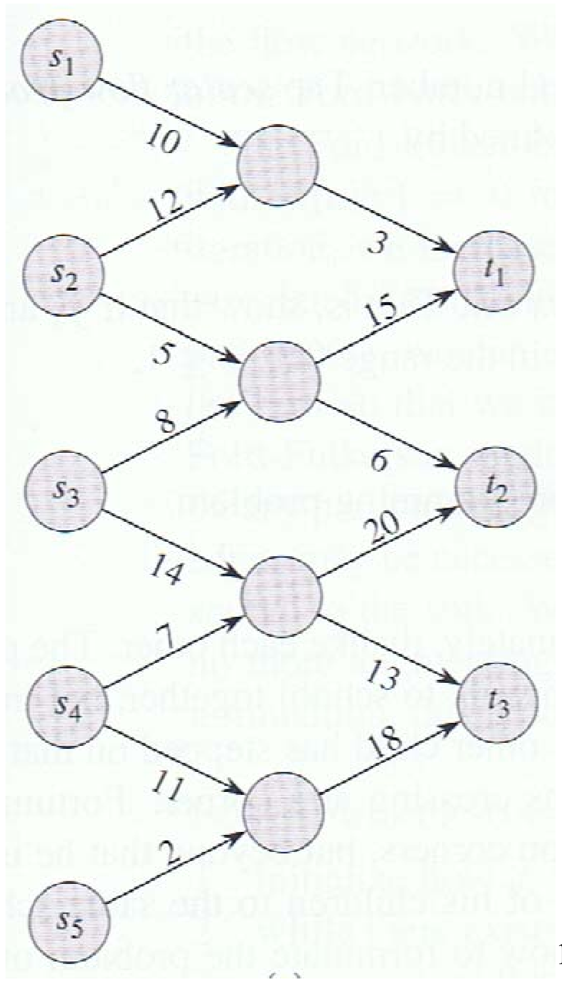
Modeling problems with antiparallel edges



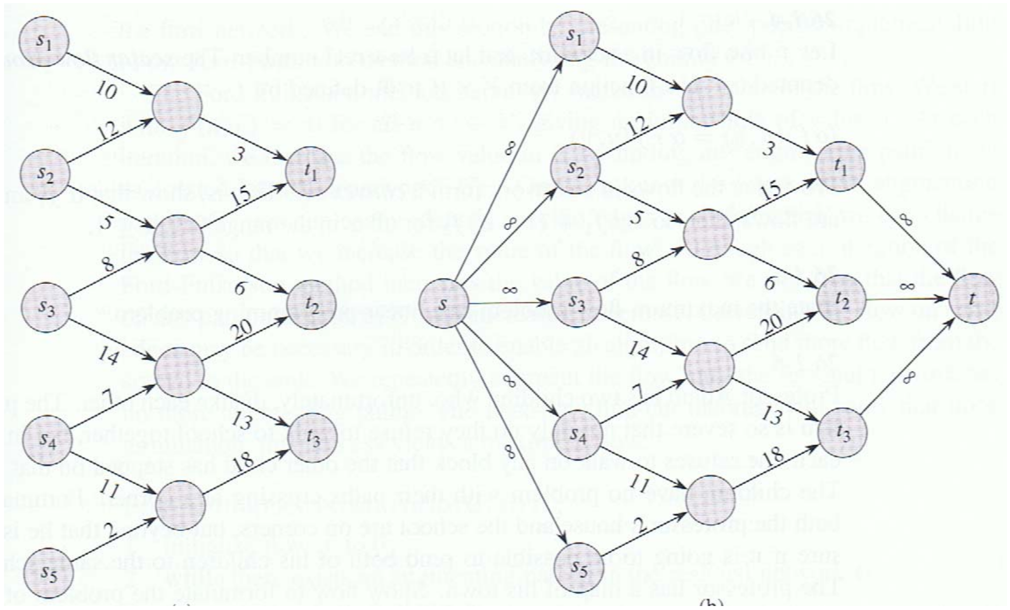
- Previous example: Lucky Puck leases additional space for 10 crates between cities v_1 and v_2
  - if $(v_1, v_2) \in E \Rightarrow (v_2, v_1) \notin E$
  - (v_1, v_2) and (v_2, v_1) are called ***antiparallel***
- Transform to an equivalent flow network w/o antiparallel edges
  - add a new vertex v'
  - replace (v_1, v_2) by two edges (v_1, v') and (v', v_2)

Networks with multiple sources and sinks

- A maximum-flow problem may have several sources and sinks
- The Lucky Puck Company may have a set of m factories $\{s_1, s_2, \ldots, s_m\}$ and a set of n warehouses $\{t_1, t_2, \ldots, t_n\}$



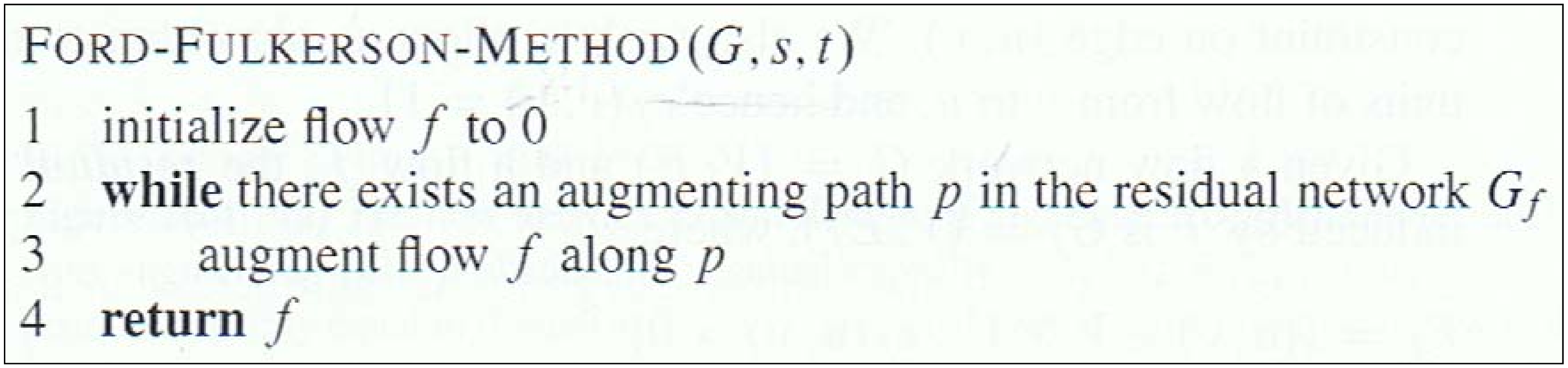
Converting a multiple-source, multiple-sink problem into an *equivalent* problem with 1 source and 1 sink



- add a **supersource** s and edges (s, s_i) with capacity $c(s,s_i)= \infty$ for all i=1,2,…, m
- add a **supersink** t and edges (t_i, t) with capacity $c(t_i,t)= \infty$ for all i=1,2,…, n

The Ford-Fulkerson method

- has several implementations with different RT
- Idea:
    - start with a flow of 0
    - at each iteration increase the flow by finding an “augmenting path” in the “residual network”
    - repeat until there are no more augmenting paths in the residual network



```
FORD-FULKERSON-METHOD$(G, s, t)$
initialize flow $f$ to 0
while there exists an augmenting path $p$ in the residual network $G_f$
    augment flow $f$ along $p$
return $f$
```

Three important concepts

- Residual network
- Augmenting paths
- Cuts

Residual networks

- Suppose we have a flow network $G = (V, E)$ with source s and sink t
- Let f be a flow in G, and let $u,v \in V$
- The *residual capacity* $c_f(u,v)$ is defined as:

$$c_f(u, v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u, v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

- Since $(u,v) \in E$ implies $(v,u) \notin E$, exactly one of the above cases applies.

Residual Networks

Given a flow network $G = (V,E)$ and a flow f, the ***residual network of G induced by f*** is $G_f = (V,E_f)$, where

$E_f = \{(u,v) \in V \times V: c_f(u,v) > 0\}$

- each edge of the residual network, called residual edge, can admit a flow > 0
- $|E_f| \leq 2\ |E|$
- residual network is similar to a flow network with capacities given by c_f
  - not a flow network, since it may contain both an edge (u,v) and its reversal (v,u)
- a flow in G_f satisfies the flow properties with respect to capacities c_f
  - a flow in G_f can be used as a roadmap for adding flow in G

Augmenting a flow in the flow network G

- let f be a flow in the flow network G
- let f ' be a flow in the corresponding residual network G_f
- then $f \uparrow f'$ is the **augmentation** of the flow f by f '

  $f \uparrow f' : V \times V \to R$

$$f \uparrow f' = \begin{cases} f(u,v) + f'(u,v) - f'(v,u) & \text{if } (u, v) \in E \\ 0 & \text{otherwise} \end{cases}$$

- f '(v,u) – pushing flow on the reverse edge in G_f is also known as *cancellation*

Flow augmentation

Lemma

Let G be a flow network

Let f be a flow in G

Let G_f be the residual network of G induced by f

Let f' be a flow in G_f

Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$

Augmenting paths

- *augmenting path* – is a simple path from s to t in the residual network G_f
- Let p be an augmenting path in G_f
- Then the *residual capacity* of p, denoted $c_f(p)$, is defined as:

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$$

- it represents the max amount by which we can increase the flow on each edge in the path p

Augmenting paths

Lemma:

Let $G = (V,E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p : V \times V \rightarrow R$ by:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

Then f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$.

Augmenting paths

Corollary: Let

- G be a flow network
- f be a flow in G
- p be an augmenting path in G_f
- f_p defined as in the previous lemma
- suppose that we augment f by f_p

Then $f \uparrow f_p$ is a flow in G with value:

$$| f \uparrow f_p | = | f | + | f_p |$$

Cuts of flow networks

- Ford-Fulkerson method repeatedly augments the flow along augmenting paths until it has found a max flow
- How do we know when the flow is maximum (when does the algorithm terminates)?
- Answer: max-flow min-cut theorem

A ***cut*** (S,T) of flow network G = (V,E) is a partition of V into S and T = V – S, such that $s \in S$ and $t \in T$.

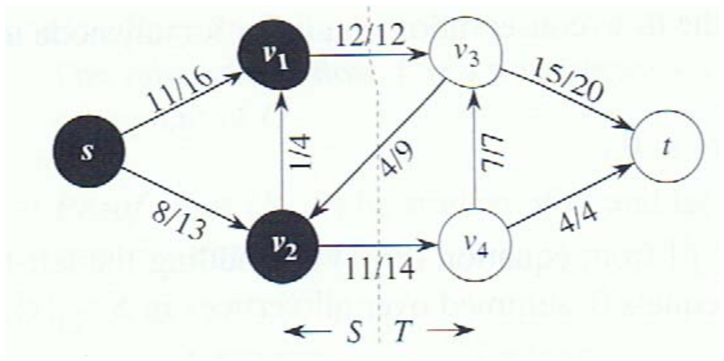
More on cuts …

- If f is a flow, then the ***net flow*** f(S,T) across the cut (S,T) is:

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

- The ***capacity*** of the cut (S,T) is:

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$



f(S,T) = ?
c(S,T) = ?

More on cuts …

Lemma: Let f be a flow in a flow network G, and let (S,T) be a cut of G. Then the *net flow* across (S,T) is f(S,T) = | f |.

Corollary: The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Proof:

$$|f| = f(S,T) = \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u)$$

$$\le \sum_{u\in S}\sum_{v\in T} f(u,v) \le \sum_{u\in S}\sum_{v\in T} c(u,v) = c(S,T)$$

Max-flow min-cut theorem

If f is a flow in a flow network $G = (V,E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G
2. the residual network G_f contains no augmenting paths
3. $| f | = c(S,T)$ for some cut (S,T) of G

Proof:

(1) $\Rightarrow$ (2) if G_f has an augmenting path p, then $f \uparrow f_p$ is a flow in G with value $> | f |$, thus f is not maximum.

Max-flow min-cut theorem, cont.

(2) $\Rightarrow$ (3) If G_f has no augmenting path:

Let S = {v $\in$ V: there is a path from s to v in G_f}

T = V – S

then (S,T) is a cut

Let u $\in$ S and v $\in$ T.

- if (u,v) $\in$ E, then f(u,v) = c(u,v) since otherwise (u,v) $\in$ E_f, then v $\in$ S
- if (v,u) $\in$ E, then f(v,u) = 0. Otherwise c_f(u,v) = f(v,u), then (u,v) $\in$ E_f, then v $\in$ S
- If neither (u,v) or (v,u) $\in$ E, then f(u,v) = f(v,u) = 0

$$f(S,T) = \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{v\in T}\sum_{u\in S} f(v,u)$$
$$= \sum_{u\in S}\sum_{v\in T} c(u,v) - \sum_{v\in T}\sum_{u\in S} 0 = c(S,T)$$

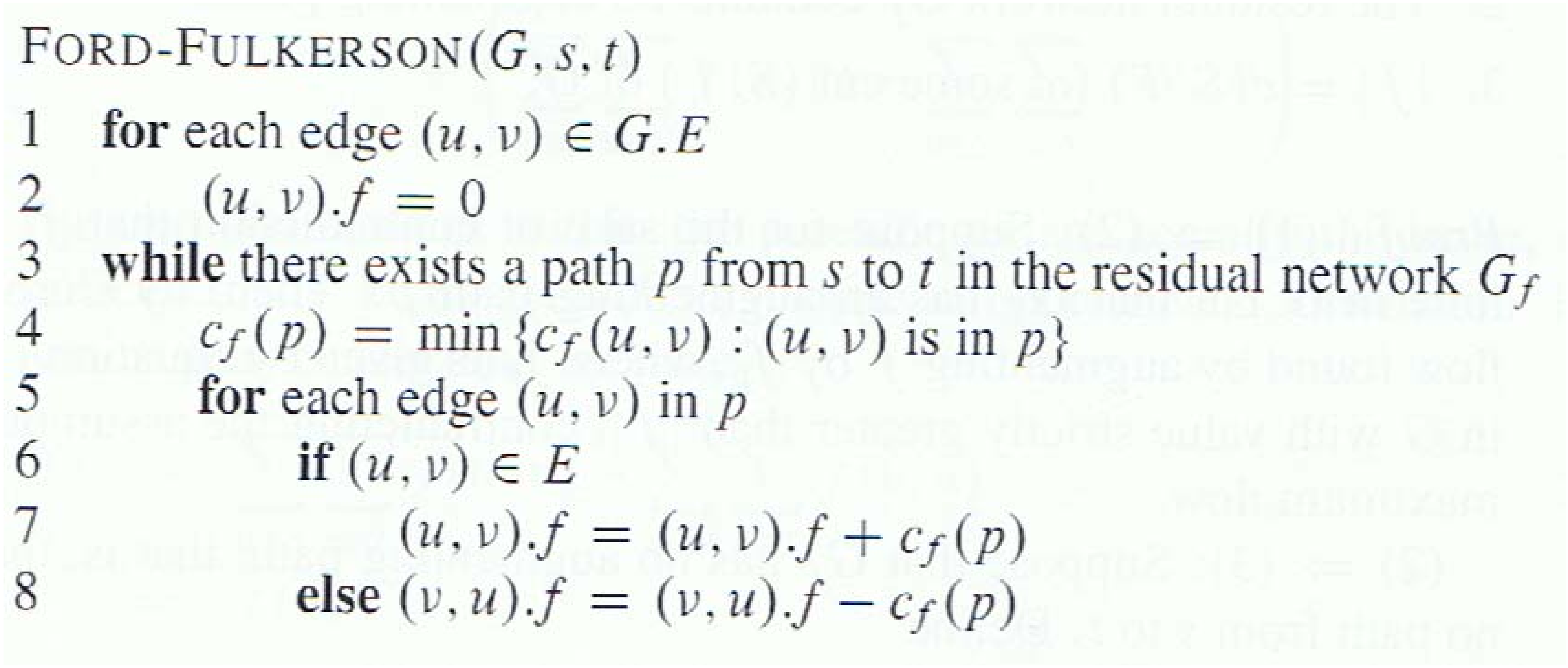
Therefore, | f | = c(S, T) for some cut (S, T) of G.

Max-flow min-cut theorem, cont.

(3) $\Rightarrow$ (1)

- from the corollary, $| f | \leq c(S,T)$ for every cut (S,T)
- $| f | = c(S,T)$ implies that f is a max flow.

Ford-Fulkerson algorithm



```
FORD-FULKERSON($G, s, t$)
for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path $p$ from $s$ to $t$ in the residual network $G_f$
    $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
    for each edge $(u, v)$ in $p$
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$
```

Analysis of Ford-Fulkerson

- RT depends on how we find the augmenting paths p
  - If we choose poorly, it may not even terminate when capacities are irrational numbers
- In practice, usually capacities are integer numbers
- If they are rational numbers, apply a scaling transformation to make them integral
- For integral capacities, the while loop executes at most |f*| times, where f* is the max flow
  - Flow increases by at least one unit each iteration

Analysis of Ford-Fulkerson

- How to find an augmenting path p?

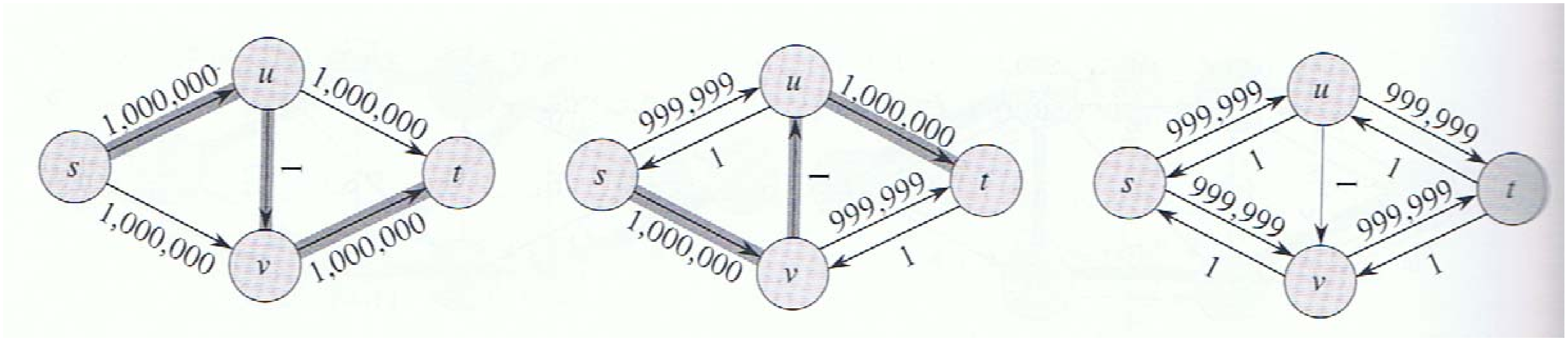
- Let G' = (V, E') – be the graph where we store the residual network G_f
  - $|E'| \leq 2|E|$
- Use Breadth-First-Search (BFS) or Depth-First-Search (DFS) to find an augmenting path between s and t
  - RT for BFS/DFS is O(V + E') = O(E)

Then the total RT of the Ford-Fulkerson is

RT = O(E |f*|)

Analysis of Ford-Fulkerson

- When | f* | is small and capacities are integral, the RT is good
- When | f* | is large, max-flow may converge slow



|f*| = 2,000,000

- The algorithm performs 2,000,000 augmentations, each increasing the flow by 1 unit

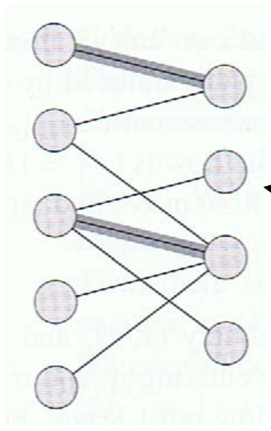
Edmonds-Karp algorithm

- Finds the augmenting path using BFS
- Then the augmenting path is a shortest-path (in terms of number of edges) from s to t in the residual network
- Edmonds-Karp has ***RT = O(VE^2)***

Maximum bipartite matching

- undirected graph G = (V,E)

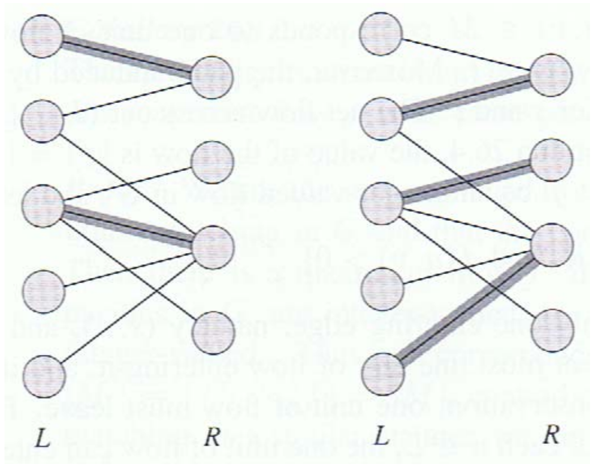
A ***matching*** is a subset of edges $M \subseteq E$ s.t. for all vertices $v \in V$, at most one edge of M is incident on v.



a matching with cardinality 2

- v is *matched* by the matching M if some edge in M is incident to v
- otherwise v is *unmatched*

Maximum bipartite matching



maximum matching with cardinality 3

- A ***maximum matching*** is a matching of maximum cardinality, such that for any other matching M', we have | M | ≥ | M'|.
- ***Bipartite graph*** is a graph in which the vertex set V is partitioned into V = L ∪ R, where L and R are disjoint and all the edges go between L and R.

  Assume that each vertex has at least one incident edge.

Maximum bipartite matching

Problem: given a bipartite graph G = (V,E), find a maximum matching.

Application example:
- L is a set of machines and R is a set of tasks
- edge (u,v) means that the machine $u \in L$ is capable of performing task $v \in R$
- maximum matching: maximize the number of tasks being executed

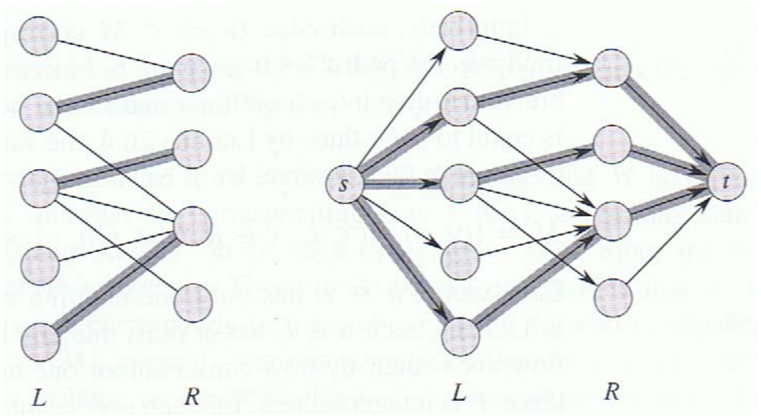
Finding a maximum bipartite matching

- construct a ***corresponding flow network*** G' = (V', E') from the bipartite graph G:

    $V' = V \cup \{s,t\}$

    $E' = \{(s,u) : u \in L\} \cup E \cup \{(v,t): v \in R\}$

    assign a capacity of 1 to each edge in E'



- each vertex in V has at least one incident edge $\Rightarrow |E| \geq |V| / 2$

    $|E'| = |E| + |V| \leq 3|E| \quad \Rightarrow \quad |E'| = \Theta(E)$

Finding a maximum bipartite matching

- we'll show that:
  - a matching in G corresponds to a flow in G'
  - then the maximum-flow corresponds to the maximum matching
- a flow f is ***integer-valued*** if f(u,v) is an integer for all $(u,v) \in V \times V$

Lemma: Let G be a bipartite graph and let G’ be the corresponding flow network.

- If M is a matching in G $\Rightarrow$ there is an integer-valued flow f in G’ with | f | = | M |

- Conversely, if f is an integer-valued flow in G’ $\Rightarrow$ there is a matching M in G with | M | = | f |

Proof:

- let M be a matching in G
  define f as follows:
  - if $(u,v) \in$ M then f(s,u) = f(u,v) = f(v,t) = 1
  - for all other edges $(u,v) \in$ E’ define f(u,v) = 0
  - then f satisfies the flow properties (capacity constraint and flow conservation) and | f | = | M |

Proof cont.

- let f be an integer-valued flow in G'

  - take $M = \{(u,v): u \in L, v \in R, \text{ and } f(u,v) > 0\}$

  at most one unit of flow can enter any $u \in L$

  - if a flow of 1 enters u then must be exactly one $v \in R$ s.t. $f(u,v)=1$

  - similarly, at most one unit can leave any node $v \in R$, thus at most one edge entering v

  - also $| M | = | f |$ = net flow across the cut

    $(L \cup \{s\}, R \cup \{t\})$

Integrality Theorem

If the capacity function c takes only integral values, then the max flow produced by the Ford-Fulkerson | f | is an integer. Moreover, for all vertices u and v, the value of f(u,v) is an integer.

Corollary: The cardinality of a maximum matching in a bipartite graph G equals the value of a maximum flow f in the corresponding flow network G'.

Finding the maximum bipartite matching

Algorithm

- create flow network G’
- run Ford-Fulkerson method to obtain max-flow f
- obtain max matching M from f

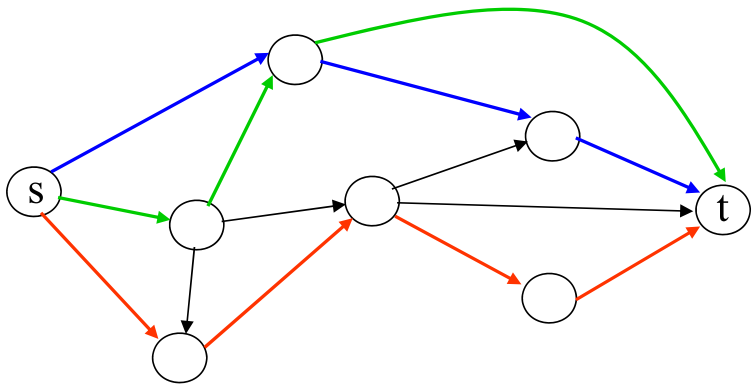
RT analysis:

- any matching $|M| \leq \min(L,R) = O(V)$
- the value of the maximum flow in G’ is O(V)
- finding a max matching takes O(VE), since $|E'| = \Theta(E)$

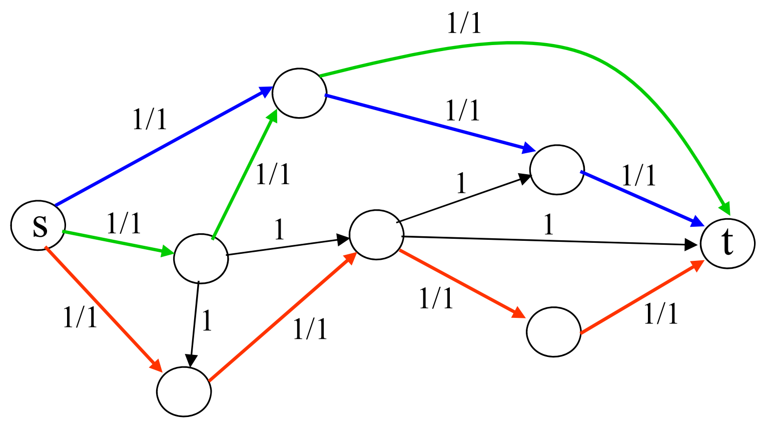
Disjoint paths in directed graphs

- A set of paths is *edge-disjoint* if no two paths share an edge, even though multiple paths may go through same vertices.

Directed Edge-Disjoint Paths Problem:
Given a directed graph G = (V,E) with two distinguished nodes s,t ∈ V, find a maximum number of edge-disjoint s-t paths in G.



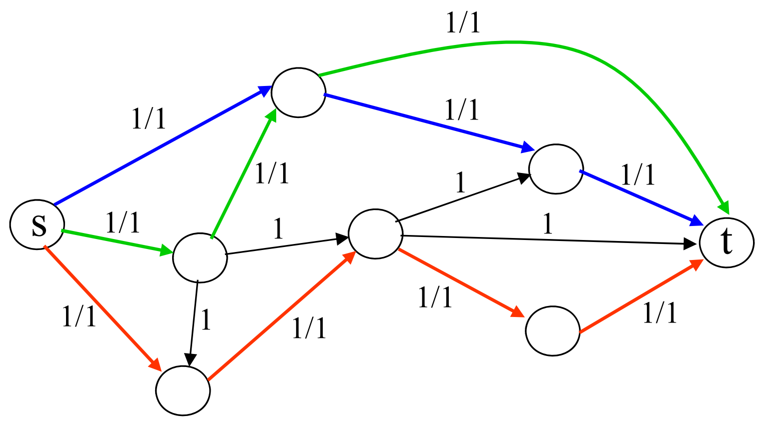
Algorithm



- Model the problem as a max-flow problem
- Flow network G = (V,E) where:
  - s is the source and t is the sink
  - each edge has capacity 1
- Find a max-flow with integral values, let’s say with value | f | = k
- Then there are k edge-disjoint paths between s and t
  - Can find the paths by tracking the edges with flow = 1 from s to t

Analysis

- If there are k edge-disjoint paths in the directed graph G from s to t, then the value of the max flow is at least k.



- If f is a 0-1 flow of value k, then the set of edges with value f(e) = 1 contains a set of k edge-disjoint paths.

- The max number of edge-disjoint paths in a directed graph G from s to t is k iff the max flow is k.

RT analysis

Ford-Fulkerson can be used to find a maximum set of edge-disjoint paths in O(VE) time

- The max flow value $| f^* | \leq | V |$
  since $\Sigma_{e \text{ out of } s}\, c_e \leq | V |$
- Ford-Fulkerson takes $O(| f^*| E) = O(VE)$

Extension: disjoint paths in undirected graphs

- issue: an edge (u,v) can mean an edge u $\rightarrow$ v or an edge v $\rightarrow$ u
- To build the flow network:
  - for each undirected edge (u,v) in G, add two directed edges (u,v) and (v,u)
  - remove the edges into s and out of t
  - apply the rule for antiparallel edges (slide 9)
- Then apply the same algorithm as for the directed graphs

Extension: disjoint paths in undirected graphs

- Transform the undirected graph G to a directed graph G':
  - replace each undirected edge (u,v) by two directed edges (u,v) and (v,u)
  - delete edges into s and out of t since they are not useful

- Issue: two paths P_1 and P_2 may be edge disjoint in the directed graph and yet share an edge in the undirected graph G
  - P_1 uses (u,v) and P_2 uses (v,u)

Extension: disjoint paths in undirected graphs

Property: *In any flow network, there is a max-flow f where for all opposite directed edges e = (u,v) and e' = (v,u), either f(e) = 0 or f(e') = 0. If the capacities of the flow-network are integral, then there is also an integral maximum flow.*

- consider a max-flow f, modify it to meet the condition
- assume e = (u,v) and e' = (v,u) have f(e) $\neq$ 0 and f(e') $\neq$ 0
- let δ = min (f(e), f(e'))
- decrease flow for both e and e' by δ
- then the resulting flow f' is feasible, has the same value as f, and f(e) or f(e') is 0

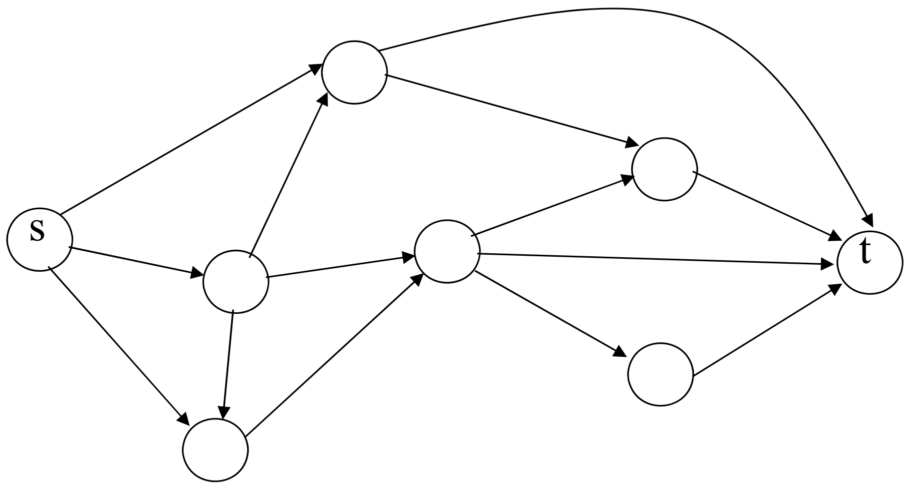
Algorithm

- Transform undirected graph to the corresponding directed graph (remove edges in s and out of t)
- Construct the flow network; apply the rule for antiparallel edges
- Compute the maximum flow
- Modify the flow such there will be no flow along antiparallel edges (note that the max-flow value remains unchanged)
- The flow will give us the edge-disjoint paths between s and t

$$RT = \Theta(VE)$$

Related question

- Finding paths that are not only edge-disjoint, but also node-disjoint (of course other than s and t)



Extensions to the max-flow problem

- Circulations with demands
- Circulations with demands and lower bounds

Circulations w/ demands

In this problem:

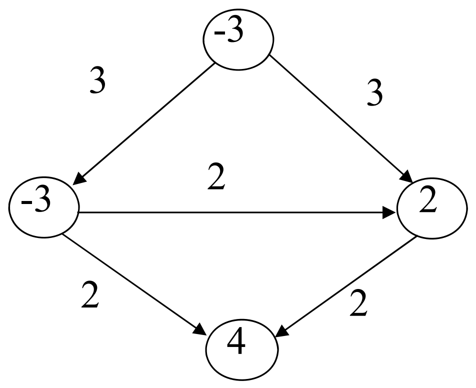
- Sources have fixed *supply* values
- Sinks have fixed *demands* value

Goal:

- Not to maximize the flow !!
- Find a solution to satisfy all the demands using the available supply (e.g. circulate flow from the nodes with supply to the nodes with demands)

This is a ***feasibility problem***

Circulations w/ demands



Flow network G:

- Has capacities on the edges
- Each node $v \in V$ has a demand d_v
  - if $d_v > 0$, then the node is a **sink**: node v has a **demand** of d_v for flow, meaning it has to receive d_v more units of flow than it sends out
  - if $d_v < 0$, then the node is a **source**: node v has a **supply** of $-d_v$ for flow, meaning it wishes to send out $-d_v$ units more flow than it receives
  - if $d_v = 0$, then node v is neither a source or a sink
- All capacities and demands are integers

Circulations w/ demands

A ***circulation*** with demands $\{d_v\}$ is a function f that assigns a nonnegative number to each edge s.t.:

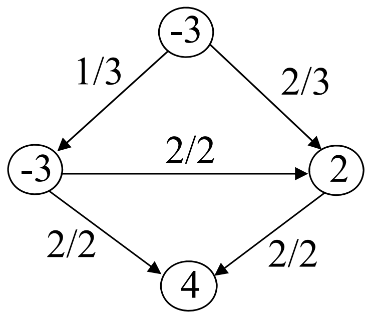
(i) *capacity condition*: for each $e \in E$, $0 \leq f(e) \leq c_e$

(ii) *demand conditions*: for each $v \in V$, $f^{in}(v)-f^{out}(v) = d_v$

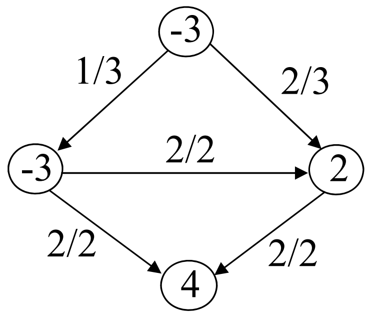
We have a ***feasibility problem***:

Find whether there exists a circulation that meets conditions (i) and (ii)

Example: flow shows a feasible circulation



Circulations w/ demands



- Condition that must hold in order for a feasible circulation to exist:

If there exists a feasible circulation with demands $\{d_v\}$, then $\Sigma_v \, d_v = 0$

proof:

suppose there is a feasible circulation f

$\Sigma_v \, d_v = \Sigma_v \, (f^{in}(v) - f^{out}(v))$

for each edge e = (u,v), f(e) is counted twice, as $f^{out}(u)$ and $f^{in}(v) \Rightarrow$ these two terms cancel out

then $\Sigma_v \, d_v = 0$

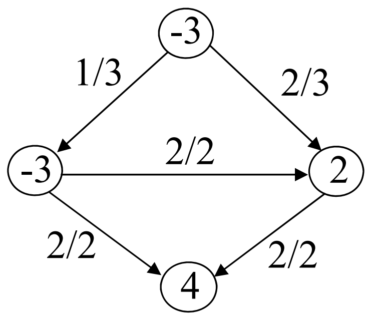
We can also write as:

$$\sum_{v:d_v>0} d_v = \sum_{v:d_v<0} -d_v = D$$

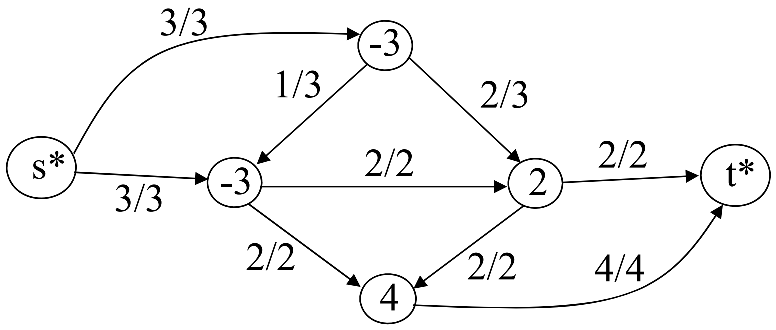
Algorithm for circulations

- Reduce the feasible circulation problem to a max-flow problem
- Let S – set with all the source nodes, T – set with all the sink nodes
- Create a graph G' from G:
  - add a "super-source" s*
  - add a "super-sink" t*
  - for each node $v \in T$, add an edge (v, t*) with capacity d_v
  - for each node $u \in S$, add an edge (s*,u) with capacity $-d_u$
- Find a max-flow in G'
- If the max-flow s*-t* in G' equals D, then G has a feasible demand circulation, given by the flow

Example



graph G



graph G'

- the flow in G' cannot be larger than D
- take a cut (A,B) with A = {s*}, then c(A,B) = D

Analysis

There is a feasible circulation with demands $\{d_v\}$ in G iff the max-flow s*-t* in G' is D. If all capacities and demands in G are integers, and there is a feasible-circulation, then the feasible circulation is integer-valued.

Proof:

• If there is a feasible circulation with demands $\{d_v\}$ in G, by sending a flow $-d_v$ on each edge (s*,v) and a flow d_v on each edge (v,t*) we get a flow in G' of value D $\Rightarrow$ max-flow

• If G' has a max-flow with value D, then every edge out of s*, into t* must be saturated with flow. If we remove these edges $\Rightarrow$ circulation f in G with $f^{in}(v) - f^{out}(v) = d_v$ for each node v

Circulations w/ demands and lower bounds

- to force flow to make use of certain edges, we can place *lower bounds* on edges

Problem: Given:
- a flow network $G = (V,E)$ with capacity c_e and lower bound l_e on each edge, $0 \leq l_e \leq c_e$ for each $e \in E$
- each node v has a demand d_v (positive or negative)
- all demands, capacities and lower bounds are integers

A *circulation* f must satisfy the conditions:
(i) capacity condition: for each $e \in E$, $l_e \leq f(e) \leq c_e$
(ii) demand conditions: for each $v \in V$, $f^{in}(v)-f^{out}(v) = d_v$

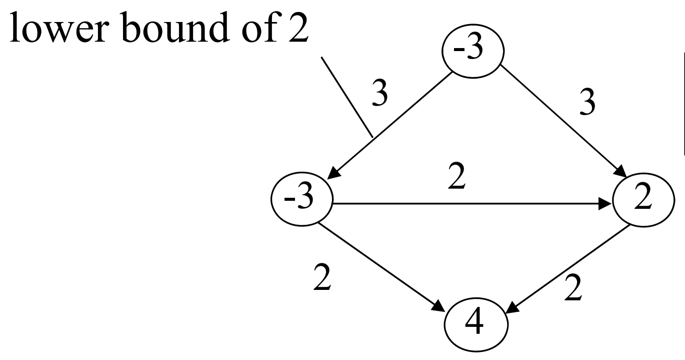
Find whether there is a *feasible circulation.*

Strategy

- reduce this to the problem of finding a circulation w/ demands but no lower bounds (which can be then reduced to the max-flow problem)
- start with an initial circulation f_0, s.t. $f_0(e) = \ell_e$
  - Capacity condition is satisfied
  - Demand condition may not be satisfied
- for any node v: $$L_v = f_0^{in}(v) - f_0^{out}(v) = \sum_{e\_into\_v} \ell_e - \sum_{e\_out\_of\_v} \ell_e$$
- if $L_v = d_v$, then the demand cond at v is satisfied
- otherwise, add a flow of f_1 on top of f_0 to make the demand condition true: $f_1^{in}(v) - f_1^{out}(v) = d_v - L_v$
- capacity left is $c_e - l_e$

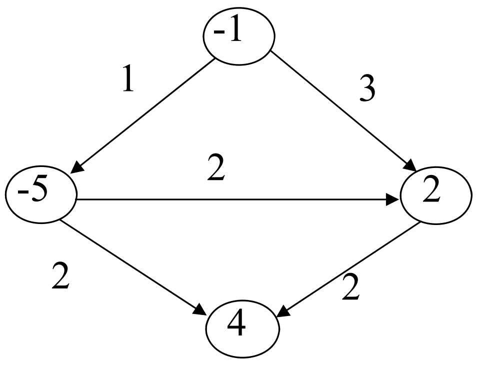
Construct a graph G'

- G' graph with capacities and demands, but ***without lower bounds***
- G' has same nodes and edges as G
- for any edge e, capacity is $c_e - \ell_e$
- demand of a node v is d_v-L_v



graph G

Eliminating a lower bound from an edge



graph G'

Algorithm & Analysis

- The problem of finding a feasible circulation with demands & lower bounds in G reduces to the problem of finding a feasible circulation with demands (and *without* lower bounds) in G'.
- Can be solved using Ford-Fulkerson in RT = O(VE)

Analysis

There is a feasible circulation in G iff there is a feasible circulation in G'. If all demands, capacities, and lower bounds are integers, and there is a feasible circulation, then there is a feasible circulation that is integer-valued.

Proof:

- Let f ' be a circulation in G'

  Define a circulation f in G by $f(e) = f'(e) + l_e$

  $f^{in}(v)-f^{out}(v) = \Sigma_{e \text{ into } v}(l_e + f'(e)) - \Sigma_{e \text{ out of } v}(l_e + f'(e)) = L_v + (d_v-L_v) = d_v$, thus it satisfies the demand condition

- Conversely, let f be a circulation in G

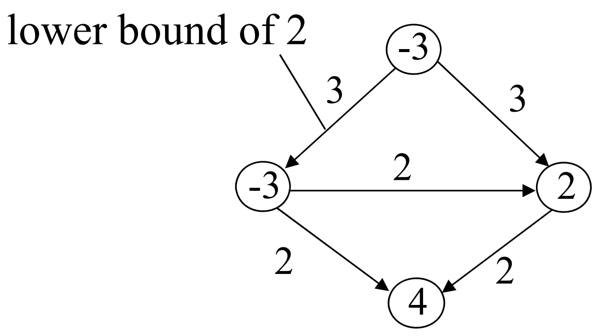
  Let f ' in G' be defined by $f'(e) = f(e) - l_e$

  f ' satisfies the capacity condition in G'

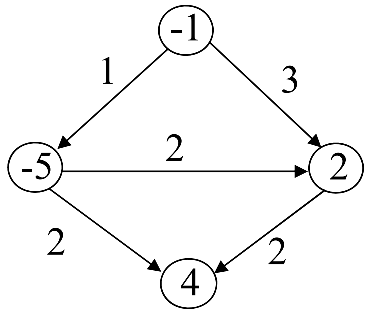
  f ' satisfies the demand condition:

  $(f')^{in}(v) - (f')^{out}(v) = \Sigma_{e \text{ into } v}(f(e) - l_e) - \Sigma_{e \text{ out of } v}(f(e) - l_e) = d_v - L_v$

Discussion on the previous example



graph G



graph G'

- Graph G' does not have a feasible circulation
  - a node with supply of 5 and only 4 units of capacity on its outgoing edges
- Therefore G does not have a feasible circulation as well

Survey Design

- A company wants to conduct a survey, asking a group of n customers which products they like

Guidelines for the survey:

- Each customer receives questions about a subset of the products
- A customer can only be asked about products that he has purchased
- Each customer i should be asked about a number of products between c_i and c_i'
- To collect sufficient data, between p_j and p_j' distinct customers must be asked about each product j

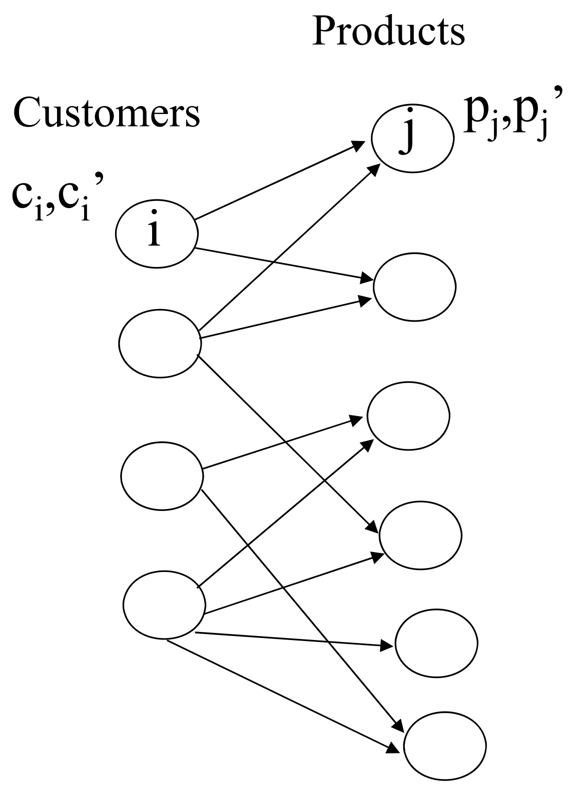
Survey Design Problem

Formalizing the problem

Input:

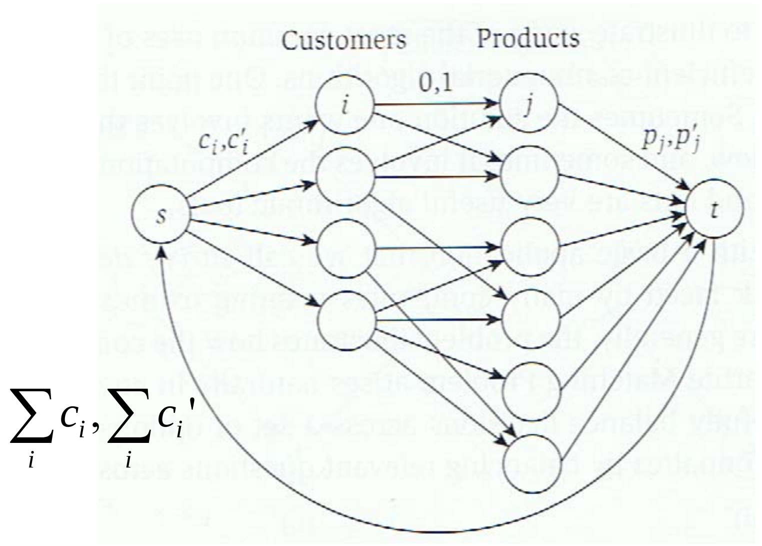
- bipartite graph G
  - nodes are the customers and the products
  - edge (i,j) means customer i has purchased before the product j
- each customer i can be asked between c_i, c_i' products
- between p_j and p_j' customers asked about each product j

Output: *decide if there is a way to design the survey to meet these conditions*



Algorithm

- Reduce the survey problem to a circulation problem w/ demands and lower bounds

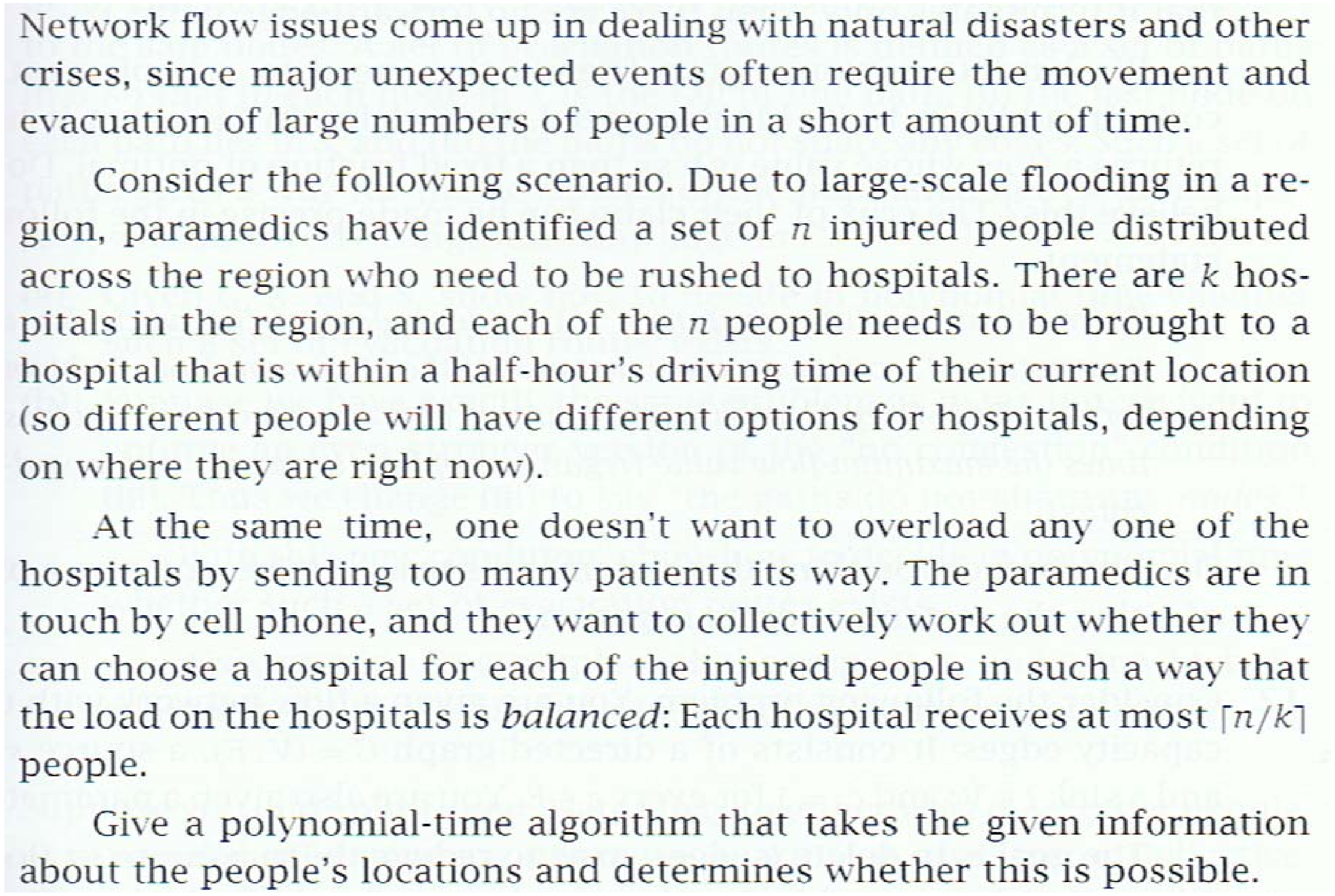


- construct a flow network G'
- values on each edge e are the lower bound and capacity constraints l_e, c_e
- flow on each edge (s,i) – number of questions for customer i
- flow on edge (j,t) – number of customers asked about product j
- flow on edge (t,s) – how many questions were asked
- all nodes have demand 0

Algorithm:

- construct network G'
- check whether it has a feasible circulation

Natural disaster scenario – problem



Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of n injured people distributed across the region who need to be rushed to hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location (so different people will have different options for hospitals, depending on where they are right now).

At the same time, one doesn't want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is *balanced*: Each hospital receives at most $\lceil n/k \rceil$ people.

Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.