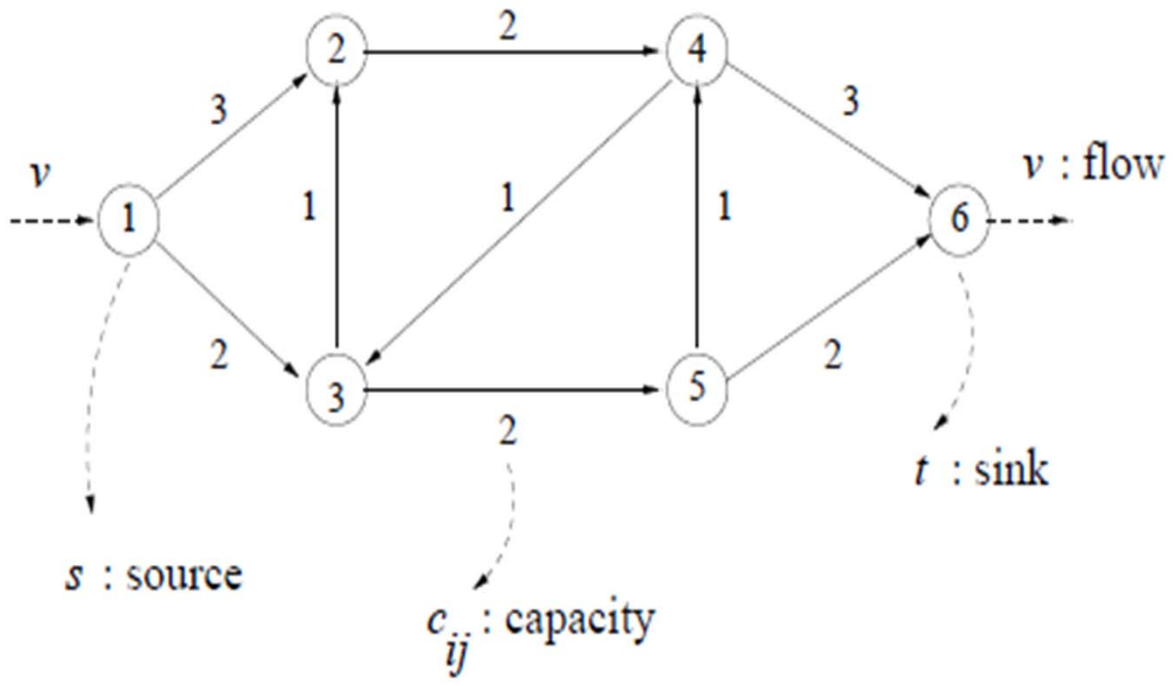
Lecture 4: Network Flows

- In this chapter, we study “classical" network flow theory, max-flow min-cut theorem, minimum cost flows, conditions for the existence of feasible circulations, “out-of-kilter" method of Minty and Fulkerson.

- Network flow problems are linear programs with the particularly useful property that they possess optimal solutions in integers. This permits a number of interesting and important combinatorial problems to be formulated and solved as network flow problems.

Maximal Flow Problem

- How big can ν be?



References

- L. R. Ford, Jr. and D. R. Fulkerson, “*Maximal Flow Through a Network*,” Canad. J. Math. 8(1956)399-404.

- P. Elias, A. Feinstein, and C. E. Shannon, “*Note on Maximum Flow Through a Network*," IRE Trans. on Information Theory, IT-2(1956)117-119.

Problem Formulation

- ◇ Suppose that each arc (i, j) of a directed graph G has assigned to it a nonnegative number c_{ij}, the *capacity* of (i, j).
- ◇ Consider the problem of finding a maximal flow from a *source* node s to a *sink* node t, Let

$$x_{ij} = \text{the amount of flow through arc } (i, j).$$

Then

$$0 \leq x_{ij} \leq c_{ij}, \quad (2.1)$$

Problem Formulation

A *conservation law* is observed at each of the nodes other than s or t. That is, what goes out of node i,

$$\sum_j x_{ij},$$

must be equal to what comes in,

$$\sum_j x_{ji},$$

Problem Formulation

So we have

$$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v, & i = s \\ 0, & i \neq s, t \\ v, & i = t. \end{cases} \quad (2.2)$$

- ◇ We call any set of numbers $x = (x_{ij})$ which satisfy (2.1) and (2.2) a *feasible flow*, or simply a *flow*, and v is its *value*. The problem of finding a maximum value flow from s to t is a linear program in which the objective is to maximize v subject to constraints (2.1) and (2.2).

LP Model for Maximal Flow Problem

$$\max \ v$$

s. t.

(flow conservation law) $$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v, & i = s, \\ 0, & i \neq s, t \\ v, & i = t, \end{cases}$$

(capacity bound) $$0 \leq x_{ij} \leq c_{ij}$$

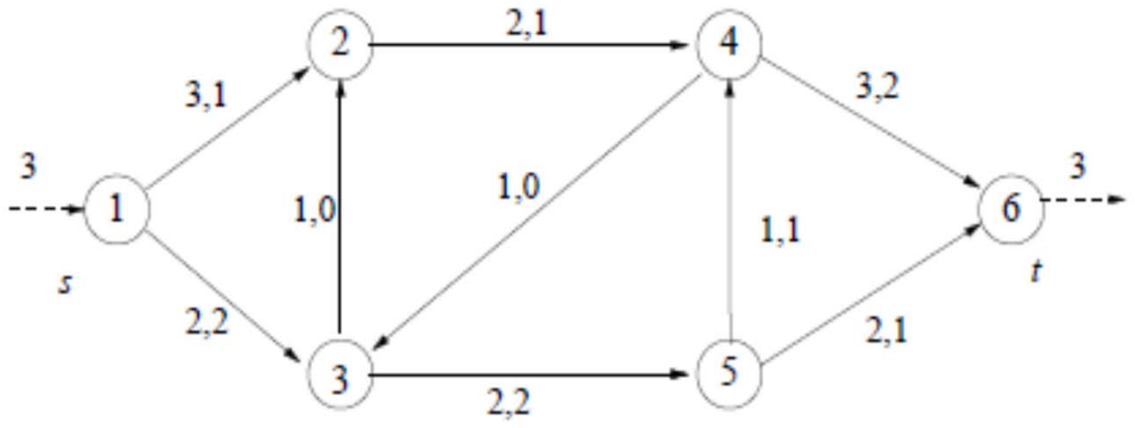
<u>Question:</u> How many variables?

How many constraints?

Applications

- Telecommunication capacity planning
- Traffic network directing
- Fleet shipment

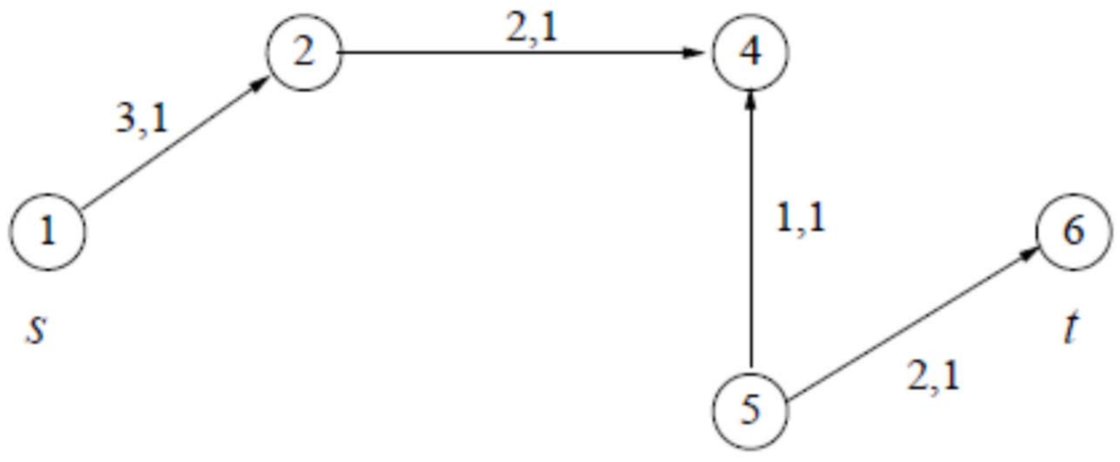
Feasible Flow – An Example



<u>Question:</u> Is it maximal?

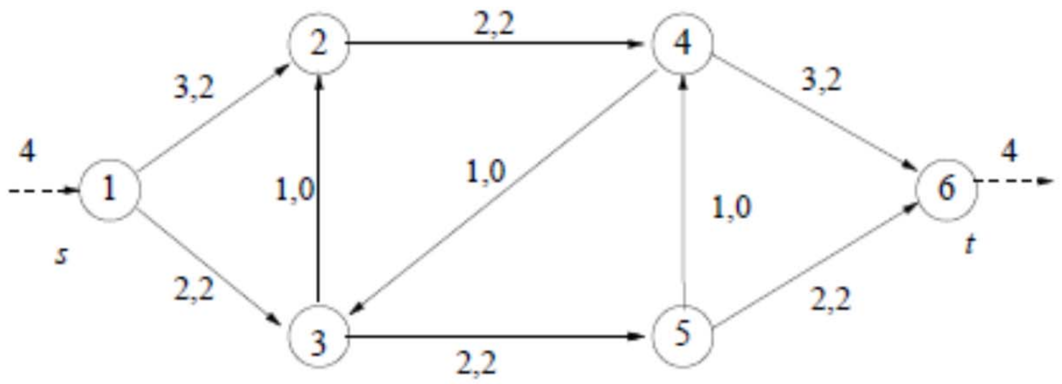
Why?

Augmenting Path



Augmenting Path

Augmented Flow



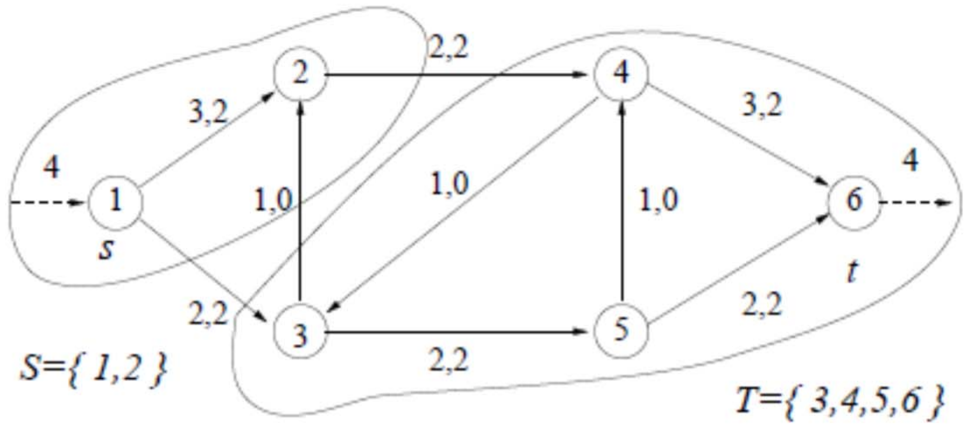
Augmented Flow

Question: Is it feasible?

Is it maximal?

why?

Optimality Condition



(S, T) : an (s, t)-cutset with capacity

$$c(S,T) = \sum_{i \in S} \sum_{j \in T} c_{ij} = 2 + 2 = 4$$

$\Rightarrow$ No feasible flow can be greater than 4!!

Definitions

- Let P be a directed path from s to t. An arc (i, j) in P is said to be a *forward* arc if it is directed from s toward t and *backward* otherwise. P is said to be a *flow augmenting path* with respect to a given flow $x = (x_{ij})$ if $x_{ij} < c_{ij}$ for each forward arc (i, j) and $x_{ij} > 0$ for each backward arc in P.

- An (s, t)-cutset is identified by a pair (S, T) of complementary subsets of nodes, with $s \in S$ and $t \in T$. The *capacity* of the cutset(S, T) is defined as

$$c(S,T) = \sum_{i \in S} \sum_{j \in T} c_{ij}.$$

Observations

- ◇ The value of any (s,t)-flow cannot exceed the capacity of any (s,t)-cutset.
- ◇ When a maximal flow is found, there exists an (s,t) cutset such that each arc (i,j) is *saturated*, i.e., $x_{ij} = c_{ij}$, if $i \in S, j \in T$ and *void*, i.e., $x_{ij} = 0$, if $i \in T, j \in S$.

Major Results

Theorem 2.1 (*Augmenting Path Theorem*) A flow is maximal if and only if it admits no augmenting path from s to t.

Theorem 2.2 (*Integral Flow Theorem*) If all arc capacities are integers there is a maximal flow which is integral.

Theorem 2.3 (*Max-Flow Min-Cut Theorem*) The maximum value of an (s, t)-flow is equal to the minimum capacity of an (s, t)-cutset.

How to Prove Theorems 2.1 – 2.3?

Lemma 1: The value of any (s,t)-flow $\leq$ the capacity of any (s,t)-cutset.

Proof: Sum up (2.2) for all nodes in S

$$\begin{aligned}
v &= \sum_{i\in S}\Big(\sum_j x_{ij} - \sum_j x_{ji}\Big) \\
&= \sum_{i\in S}\left[\sum_{j\in S}(x_{ij}-x_{ji}) + \sum_{j\in T}(x_{ij}-x_{ji})\right] \\
&= \sum_{i\in S}\sum_{j\in S}(x_{ij}-x_{ji}) + \sum_{i\in S}\sum_{j\in T}(x_{ij}-x_{ji}) \\
&= \sum_{i\in S}\sum_{j\in T}(x_{ij}-x_{ji}) \quad (2.3) \\
&\leq \sum_{i\in S}\sum_{j\in T}(c_{ij}-0) \\
&= c(S,T)
\end{aligned}$$

A Corollary for Optimality Condition

Corollary 1: If v is an (s,t)-flow and (S,T) is an (s,t)-cutset with $v = c(S,T)$, then v is maximal and $c(S,T)$ is minimal

Proof of Theorem 2.1

If an augmenting path exists, then (clearly) the flow is not maximal.

Suppose $x = (x_{ij})$ is a flow that admits no augmenting path. We let $S = \{j|$ there is an augmenting path from s *to* $j\}$ and $T = S^C$. Then the definitions of augmenting path and S, T implies that, $\forall\ i \in S$ and $j \in T$,

$x_{ij} = c_{ij}$ and $x_{ji} = 0$.

Hence, by (2.3), we have

$$v = \sum_{i \in S} \sum_{j \in T} (c_{ij} - 0) = c(S, T)$$

It follows from Corollary 1 that x is a maximal flow.

Proof of Theorem 2.2

Let us start with a zero-flow $x_{ij}^0 = 0, \forall i, j$.

If x^0 is not maximal, then it admits an augmenting path (Thm 2.1). Because each capacity is an integer, we have an integral flow x^1 whose value exceeds that of x^0. If x^1 is not maximal, repeat the reasoning. Because the capacity is finite, we arrive eventually at an integral flow that admits no augmenting path and hence is maximal.

Proof of Theorem 2.3

- This proof is a little bit complicated. We need to prove that every network actually admits a maximal flow.
- Where can we find that information?
- If we cannot get the information on the primal side, then it is logic to check the dual side.

Dual LP Problem

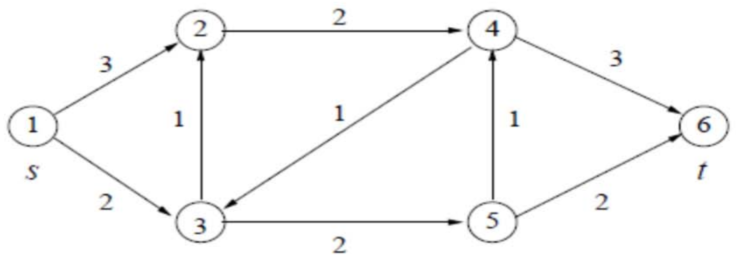
$$
\text{(P)}\quad
\begin{aligned}
&\max\ v\\
&\text{s. t.}\\
&\textstyle\sum_j x_{ji} - \sum_j x_{ij} + v &&= 0,\ i = s\\
&\textstyle\sum_j x_{ji} - \sum_j x_{ij} &&= 0,\ i \neq s, t\\
&\textstyle\sum_j x_{ji} - \sum_j x_{ij} - v &&= 0,\ i = t\\
&x_{ij} \leq c_{ij}\\
&x_{ij} \geq 0
\end{aligned}
$$

$$
\text{(D)}\quad
\begin{aligned}
&\min \textstyle\sum c_{ij} w_{ij}\\
&\text{s. t.}\\
&u_j - u_i + w_{ij} &&\geq 0\\
&u_s - u_t &&\geq 1\\
&\qquad\qquad w_{ij} &&\geq 0\\
&u_i : \text{unrestricted}
\end{aligned}
$$

<u>Question</u>: What is the meaning of (D)?

Example

Example:



	max v s. t.										
u_1	$-x_{12}$	$-x_{13}$								$+v$	$= 0$
u_2	x_{12}		$-x_{24}$	$+x_{32}$							$= 0$
u_3		x_{13}		$-x_{32}$	$-x_{35}$	$+x_{43}$					$= 0$
u_4			x_{24}			$-x_{43}$	$-x_{46}$	$+x_{54}$			$= 0$
u_5					x_{35}			$-x_{54}$	$-x_{56}$		$= 0$
u_6							x_{46}		$+x_{56}$	$-v$	$= 0$
w_{12}	x_{12}										≤ 3
w_{13}		x_{13}									≤ 2
w_{24}			x_{24}								≤ 2
w_{32}				x_{32}							≤ 1
w_{35}					x_{35}						≤ 2
w_{43}						x_{43}					≤ 1
w_{46}							x_{46}				≤ 3
w_{54}								x_{54}			≤ 1
w_{56}									x_{56}		≤ 2
	$x_{12},$	$x_{13},$	$x_{24},$	$x_{32},$	$x_{35},$	$x_{43},$	$x_{46},$	$x_{54},$	x_{56}		≥ 0

Observation 1:

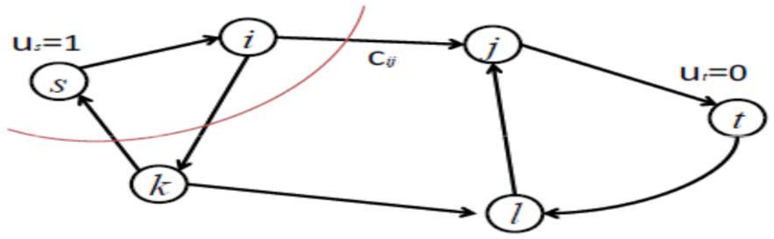
Lemma 2: Any (s,t)-cutset (S,T) produces feasible solution whose dual objective value $= c(S,T)$.

Proof: Let

$$\begin{aligned} u_i &= 1, \quad \text{if} \quad i \in S \\ &= 0, \quad \text{if} \quad i \in T \\ w_{ij} &= 1, \quad \text{if} \quad i \in S, j \in T \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

Everything follows correctly!

Observation 2: Minimum Cut Set Problem



Minimize

$$\sum_{i,j} c_{ij} w_{ij}$$

$$\begin{aligned} \text{s.t.} \qquad u_j - u_i + w_{ij} &\geq 0 \\ u_s &= 1 \\ u_t &= 0 \\ u_i, u_j, w_{ij} &\in \{0, 1\} \end{aligned}$$

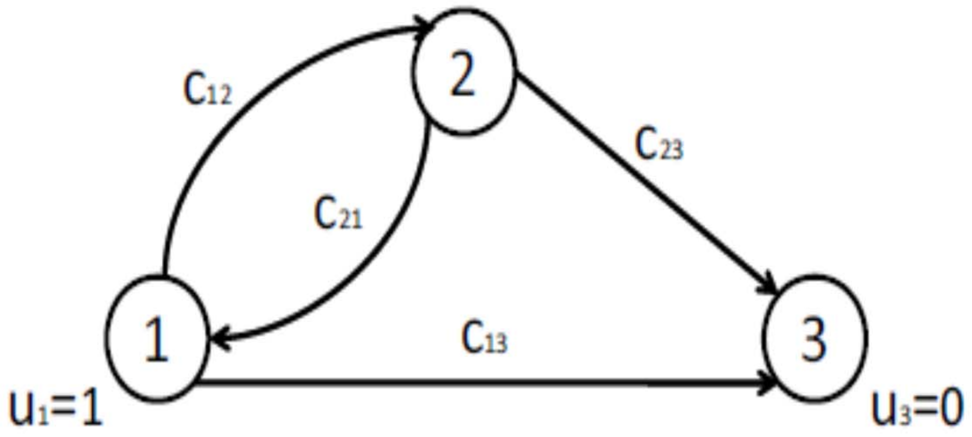
Note that (1) when $u_i, u_j \in \{0, 1\}$

$$\begin{matrix} u_s = 1 \\ u_t = 0 \end{matrix} \Leftrightarrow u_s - u_t \geq 1$$

(2) when $w_{ij} \geq 0$, since $c_{ij} > 0$

$0 \leq u_i \leq 1$ can be attained at an optimal solution.

Example



$$\begin{cases} u_2 - \cancelto{1}{u_1} + w_{12} \geq 0, \quad \cancelto{1}{u_1} - u_2 + w_{21} \geq 0 \\ \cancelto{0}{u_3} - \cancelto{1}{u_1} + w_{13} \geq 0, \; \cancelto{0}{u_3} - u_2 + w_{23} \geq 0 \end{cases}$$

Example

Note that:

$$c_{12}, c_{13}, c_{21}, c_{23} > 0; w_{12}, w_{13}, w_{21}, w_{23} \geq 0$$

In an optimal solution,

$$\left.\begin{array}{l} u_2 > 1 \text{ is not necessary} \\ u_2 < 0 \text{ is not necessary} \end{array}\right\} \Rightarrow \begin{array}{l} 0 \leq u_2 \leq 1 \\ w_{13} = 1, w_{21} = 0 \end{array}$$

$$\begin{aligned} \sum c_{ij} w_{ij} &= (1-u_2)c_{12} + u_2 c_{23} + (1 \times c_{13}) + (0 \times c_{21}) \\ &\geq (1-u_2)min\{c_{12}, c_{23}\} + u_2 min\{c_{12}, c_{23}\} + c_{13} \\ &= min\{c_{12}, c_{23}\} + c_{13} \end{aligned}$$

When: $\left.\begin{array}{l} c_{12} > c_{23}, u_2 = 1 \\ c_{23} > c_{12}, u_2 = 0 \end{array}\right\}$ The best of 2 cases.

$$c_{12} = c_{23}, u_2 = 0 \ or \ 1$$

Observation 3:

Lemma 3: There is a dual optimal solution corresponding to an (s, t)-cutset.

Proof: Since the constraint for node t in (P) is redundant, we may assume that $u_t = 0$. Then we may assume $u_s = 1$, because there is no reason for u_s to be greater (we can alwayse scale it back to 1). Notice that $c_{ij} > 0$ and $w_{ij} \geq 0$, for each arc (i, j), a dual optimal solution can always be constructed in the way such that $w_{ij} = 1$ if and only if $u_i = 1$ and $u_j = 0$ (Why?). Then let

$$S = \{i | u_i = 1\},$$
$$T = \{j | u_j = 0\}.$$

The capacity of the cutset (S, T) is exactly equal to the value of the optimal dual solution.

Proof of Theorem 2.3

By Lemma 3 and Lemma 2, we know the dual problem (D) in facts finds a minimum capacity (s, t)-cutset. But (P) is feasible and bounded, the fundamental theorem of LP says value(P)=value(D), which proves Theorem 2.3.

More Observations

(P) — maximal flow

(D) — minimal capacity cutset

(P)+(D) = ?

By complementary slackness theorem:
For optimal solutions:

$$(c_{ij} - x_{ij})w_{ij} = 0$$
$$(u_j - u_i + w_{ij})x_{ij} = 0$$

Let u_i:= node potential.

- If the potential at $i > j$ $(u_i > u_j)$, then $w_{ij} > 0$, and hence $x_{ij} = c_{ij}$, i.e., the flow in (i, j) is saturated.
- If $u_i < u_j$, then $u_j - u_i + w_{ij} > 0$, and hence $x_{ij} = 0$, i.e., no flow in (i, j).
- If $u_i = u_j$, then there may or may not be positive flow in (i, j).

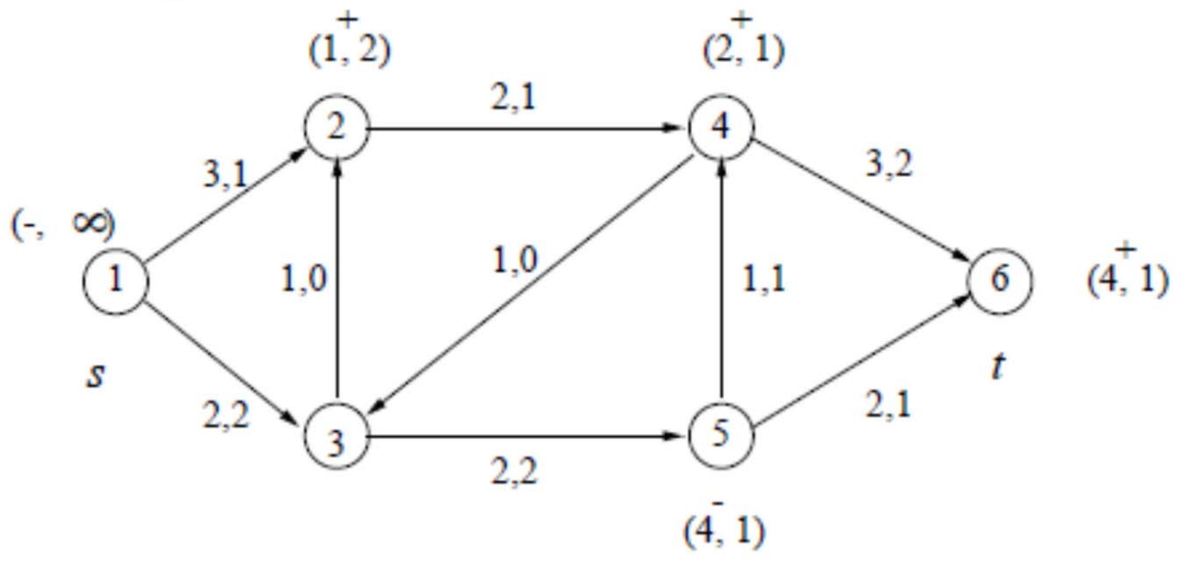
Maximal Flow Algorithm

- The key is to find a flow augmenting path.
- What kind of flow augmenting path?

Finding a flow augmenting path

Labeling Procedure

Example:



A labeling scheme for augmenting path

- ◇ A label (i^+, δ_j) indicates that there exists an augmenting path with capacity δ_j from the source to the node j in question, and that (i, j) is the last arc in this path.

- ◇ A label (i^-, δ_j) indicates that (j, i) is the last arc in the path, i.e., (j, i) will be a backward arc if the path is extended to the sink t.

- ◇ Initially only the source node s is labeled with the special label $(-, \infty)$. Additional nodes are labeled in one of two ways:

Labeling Scheme

If node i is labeled and there is an arc (i, j) for which $x_{ij} < c_{ij}$, then the unlabeled node j can be given the label (i^+, δ_j), where

$$\delta_j = min\{\delta_i, c_{ij} - x_{ij}\}.$$

If node i is labeled and there is an arc (j, i) for which $x_{ji} > 0$, then the unlabeled node j can be given the label (i^-, δ_j), where

$$\delta_j = min\{\delta_i, x_{ji}\}.$$

Labeling Scheme

- ◇ If node t is labeled, then an augmenting path has been found and the flow value can be augmented by δ_t. Otherwise, no augmenting path exists.

- ◇ A minimum capacity cutset (S, T) is constructed by letting S contain all labeled nodes and T contain all unlabeled nodes.

- ◇ A labeled node is either "scanned" or "unscanned." A node is scanned by examining all incident arcs and applying labels to previously unlabeled adjacent nodes, according to the rules given above.

Maximal Flow Algorithm

- *Step 0(Start)* Let $x = (x_{ij})$ be any integral feasible flow, possibly the zero flow. Give node s the permanent label $(-, \infty)$.

- *Step 1(Labeling and Scanning)*

(1.1) If all labeled nodes have been scanned, go to Step 3.

(1.2) Find a labeled but unscanned node i and scan it as follows: For each arc (i, j), if $x_{ij} < c_{ij}$ and j is unlabeled, give j the label (i^+, δ_j), where

$$\delta_j = min\{c_{ij} - x_{ij}, \delta_i\}.$$

For each arc (j, i), if $x_{ji} > 0$ and j is unlabeled, give j the label (i^-, δ_j), where

$$\delta_j = min\{\delta_i, x_{ji}\}.$$

(1.3) If node t has been labeled, go to Step 2; otherwise go to Step 1.1.

Maximal Flow Algorithm

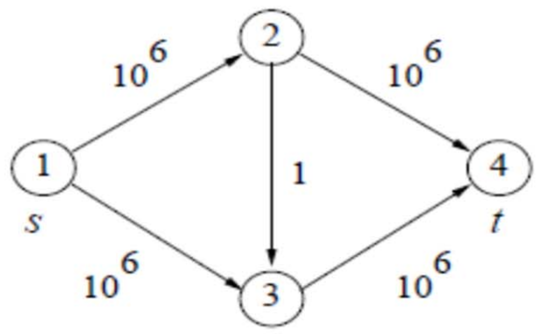
- ◇ *Step 2(Augmentation)* Starting at node t, use the index labels to construct an augmenting path. (The label on node t indicates the second-to-last node in the path, the label on that node indicates the third-to-last node, and so on.) Augment the flow by increasing and decreasing the arc flows by δ_t, as indicated by the superscripts on the index labels. Erase all labels, except the label on node s. Go to Step 1.

- ◇ *Step 3(Construction of Minimal Cut)* The existing flow is maximal. A cutset of minimum capacity is obtained by placing all labeled nodes in S and all unlabeled nodes in T. The computation is completed.

Note: the algorithm is $O(mv)$ in complexity.

Efficiency of the Maximal Flow Algorithm

Example:



How many augmentations are needed ?
2 or 2 millions?

- Any ideas?
  - first labeled, first scan?
  - augmenting path has fewest arcs?
  - augmenting path allows most increase in flow?

Maximum capacity flow augmenting path

- The problem of finding a maximum capacity flow augmenting path is evidently quite similar to the problem of finding a shortest path, or, more precisely, a path in which the minimum arc length is maximum.

Maximum capacity flow augmenting path

Shortest Path Approach

c_{ij}: capacity of arc (i, j)

$(c_{ij} = 0$, if arc (i, j) does not exist)

$\bar{c}_{ij} := \max\{c_{ij} - x_{ij}, x_{ji}\}$

$u_i :=$ the capacity of a max-flow augmenting path from node s to node i.

Bellman's Equation:

$$u_s = +\infty$$

$$u_i = \max_k \min\{u_k, \bar{c}_{ki}\}, \ i \neq s$$

Dijkstra-like algorithm: $O(n^2)$

Augmenting path with fewest arcs

Let

$$\delta_i^{(k)} = \text{the minimum number of arcs in an augmenting path from } s \text{ to } i \text{ after } k \text{ flow augmentations.}$$

and

$$\tau_i^{(k)} = \text{the minimum number of arcs in an augmenting path from } i \text{ to } t \text{ after } k \text{ flow augmentations.}$$

Augmenting path with fewest arcs

- **Lemma 4.2** If each flow augmentation is made along an augmenting path with a minimum number of arcs, then

$$\delta_i^{(k+1)} \geq \delta_i^{(k)}$$

and

$$\tau_i^{(k+1)} \geq \tau_i^{(k)}$$

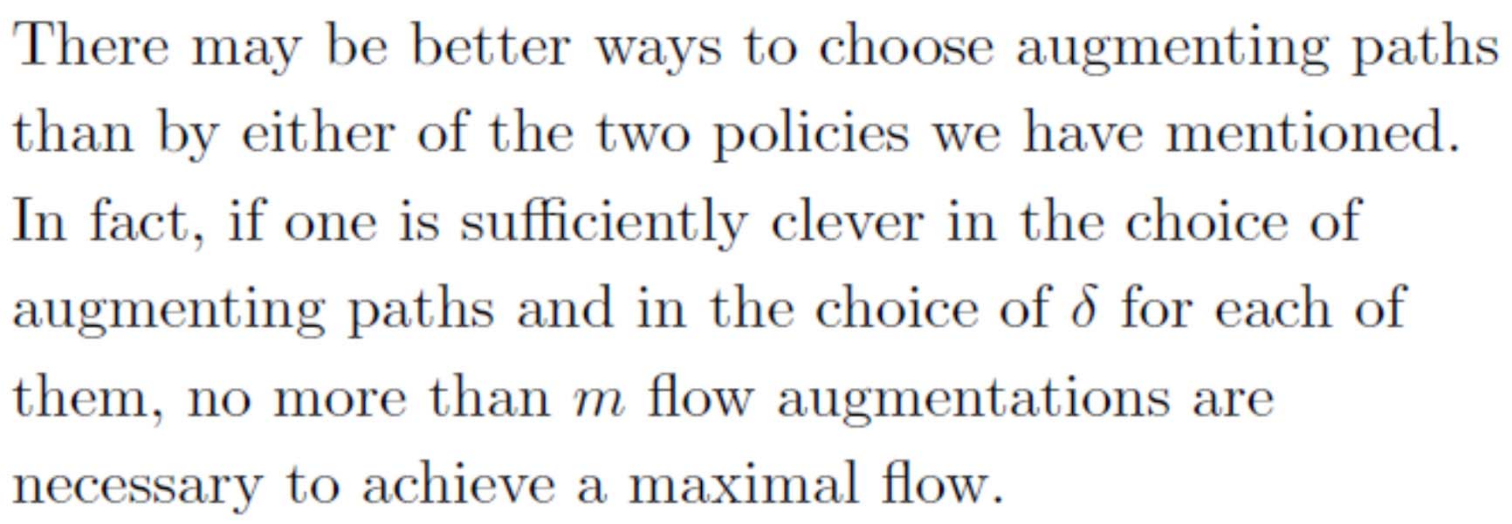
for all i, k.

Improved Efficiency

◇ **Theorem 4.1** *(Edmonds and Karp)* If each flow augmentation is made along an augmenting path with a minimum number of arcs, then a maximal flow is obtained after no more than $mn/2 \leq (n^3 - n^2)/2$ augmentations, where m is the number of arcs in the network and n is the number of nodes.

- The bound can be further reduced to “*mn*/4”.
- Complexity = $O(m^2n)$.
- J. Edmonds and R. M. Karp, “*Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*," JACM, 19(1972) 248-264.

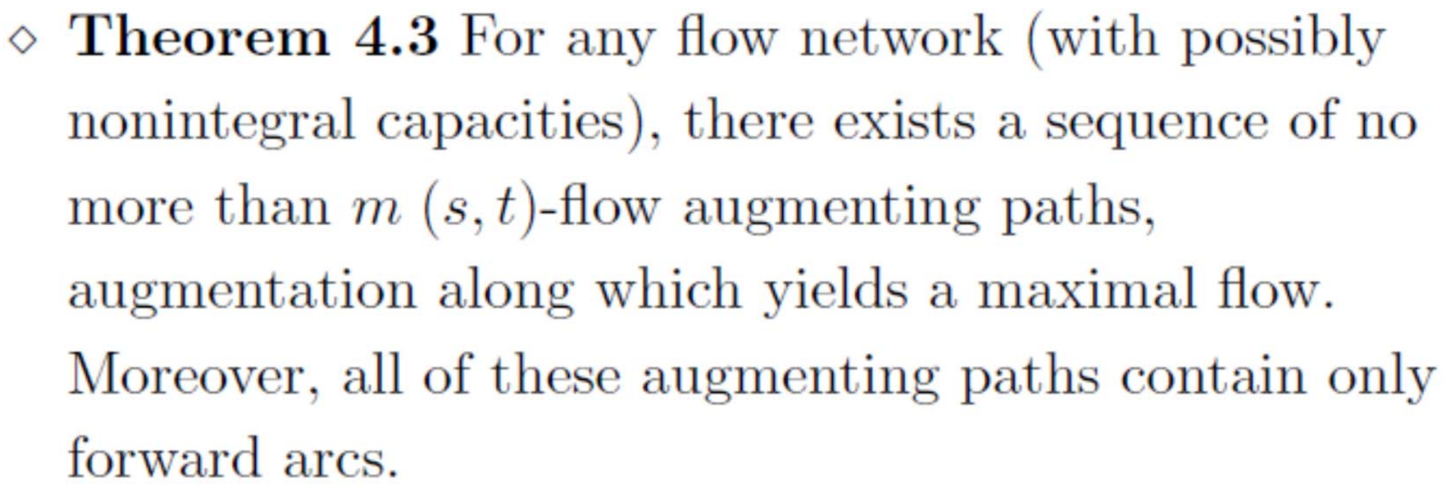
Any more improvement?



There may be better ways to choose augmenting paths than by either of the two policies we have mentioned. In fact, if one is sufficiently clever in the choice of augmenting paths and in the choice of δ for each of them, no more than m flow augmentations are necessary to achieve a maximal flow.

- Flow decomposition

Improved Efficiency

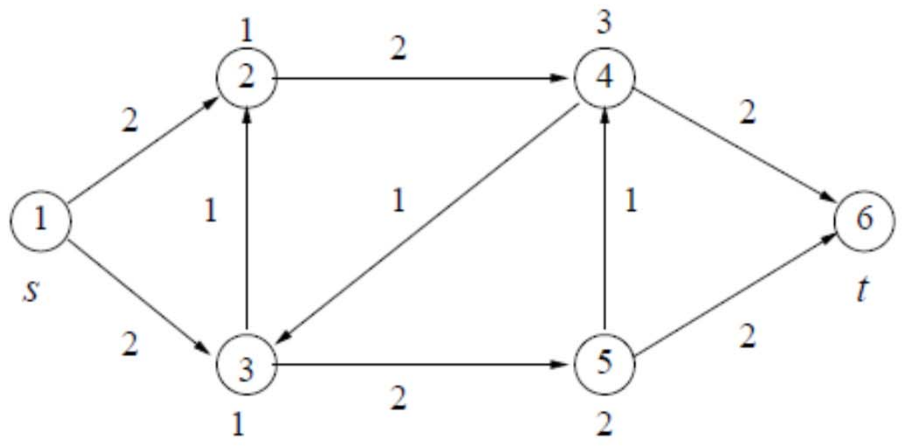


◇ **Theorem 4.3** For any flow network (with possibly nonintegral capacities), there exists a sequence of no more than m (s, t)-flow augmenting paths, augmentation along which yields a maximal flow. Moreover, all of these augmenting paths contain only forward arcs.

- Complexity = *O(m^2)*.
- How to realize such augmenting paths?

Implications of Max-Flow Min-Cut Theorem

- Max-Flow with node capacity constraints



- More applicable for real applications

Max-Flow with node capacity constraints

- Let us consider a flow network in which there are arc capacities $c_{ij} \geq 0$ and, in addition, node capacities $c_i \geq 0$. Flows are required to satisfy not only the conservation conditions and arc constraints $(0 \leq x_{ij} \leq c_{ij})$ but also the node constraints,

$$\sum_j x_{ij} \leq c_i, \quad i \neq s, t.$$

  That is, the outflow (and hence the inflow) at any interior nodes does not exceed the capacity of the node.

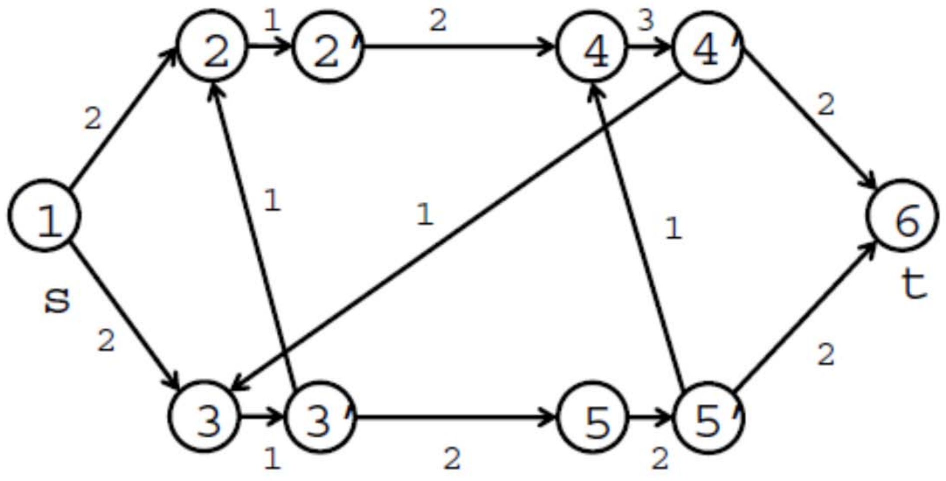
- For a node having node capacities as well as arc capacities, we define an (s, t)-*cut* as a set of arcs and nodes such that any path from s to t uses at least one member of the set. The *capacity* of a cut is the sum of the capacities of its members.

Generalized Max-Flow Min-Cut Theorem

- **Theorem 5.1** *(Generalized Max-Flow Min-Cut Theorem)* In a network having node capacities as well as arc capacities, the maximum value of an (s, t)-flow is equal to the minimum capacity of an (s, t)-cut. Moreover, if all capacities are integers, there is a maximal flow that is integral.

Proof of Theorem 5.1

- Basic idea



- Formal proof – Homework exercise

Implications to Graph Theory

A celebrated result of graph theory is a theorem of K. Menger.

A digraph G is said to be *k-connected from s to t* if for any set C of k-1 nodes missing s and t there is a directed path from s to t missing C. In other words, it is not possible to disconnect s from t by removing any fewer than k nodes.

Two (s, t) paths are said to be *independent* if they have no nodes in common except s and t.

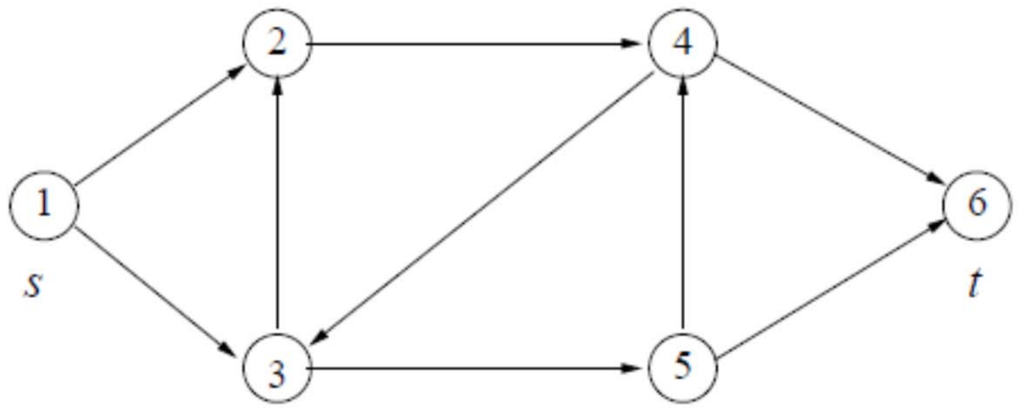
Observation

- Lemma:

  A directed graph has *k* independent (*s*-*t*) paths, then it is *k*-connected from *s* to *t*.

Menger's Theorem

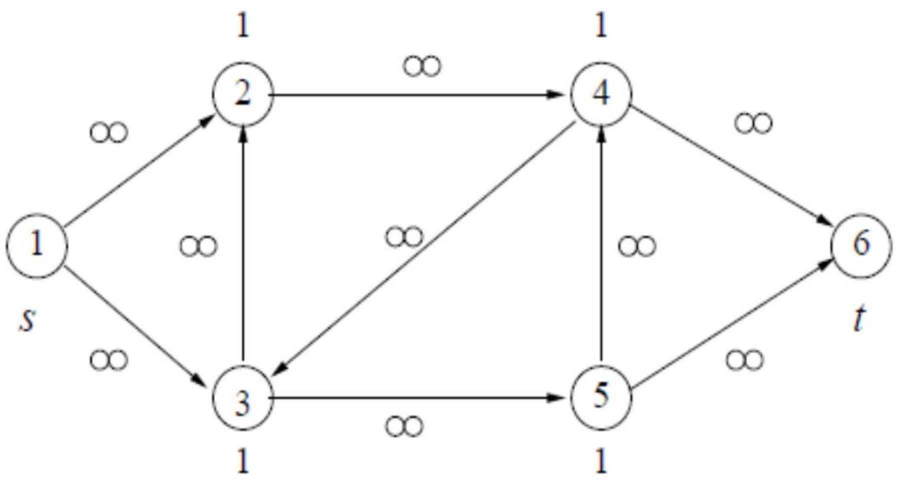
(# of independent (s, t) paths)



- How many independent (s, t) paths?
- k-connected digraph (k=?)
- Conjecture: Digraph G is k-connected from s to t, then G admits k independent (s, t) paths?

Other Implications of Max-Flow Min-Cut Theorem

- ◇ **Theorem 5.2** *(Menger)* If digraph G is k-connected from s to t and does not contain arc (s, t), then G admits k independent directed paths from s to t.



Proof of Menger’s Theorem

- Assign each node a capacity of one and each arc an infinite capacity.
- Because G does not contain arc (s, t), the minimum cut capacity is finite.
- From the definition of k-connectivity of G, we know the minimum cut capacity is at least k.
- From Theorem 5.1, there is an integral flow of value at least k.
- From the capacity assignment, such flow yields at least k pairwise independent (s, t) paths.

Other Implications

Network flows theory also yields a good deal of information about the structure of undirected graphs.

- **Theorem 5.3** The maximum number of arc-disjoint (s, t) paths in an undirected graph G is equal to the minimum number of arcs in an (s, t)-cutset.

Min-Cut Formulation

- Generalized knapsack problem – Provisioning problem

◇ A more sophisticated view of the knapsack can be taken. Suppose there are n items to choose from, where item j costs $c_j > 0$ dollars. Also suppose there are m sets of items, $S_1, S_2, \ldots, S_m$, that are known to confer special benefits. If all of the items in set S_i are chosen, then a benefit of $b_i > 0$ dollars is gained. The sets are arbitrary and need not be related in any particular way, e.g., a given item may be contained in several different sets.

◇ There is no restriction on the number of items that can be purchased, i.e., there is not limiting knapsack. Our objective is simply to maximize the net benefit, i.e., total benefit gained minus total cost of items purchased.

Min-Cut Formulation

- Min-cut formulation

$$\begin{cases} \min & \sum_{i,j} c_{ij} w_{ij} \\ \text{subject to} & \\ & u_j - u_i + w_{ij} \geq 0 \quad (6.1) \\ & u_s - u_t \geq 1 \\ & w_{ij} \geq 0 \\ & u_i \ \text{unrestricted.} \end{cases}$$

- How to formulate the provisioning problem in min-cut form?

Min-Cut Formulation

Step 1: A simple model

Define

$$\begin{aligned} v_j &= 1, \quad \text{if item } j \text{ is purchased} \\ &= 0, \quad \text{otherwise} \\ u_i &= 1, \quad \text{if all items in } S_i \text{ are purchased} \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

$$\max \quad Z = \textstyle\sum_i b_i u_i - \sum_j c_j v_j \tag{6.2}$$

$$\text{s.t.} \quad v_j - u_i \geq 0, \ \forall\ i, j \text{ such that } j \in S_i \tag{6.3}$$

$$u_i, v_j \in \{0, 1\}$$

Min-Cut Formulation

Step 2: Adding new variables

$$w_i \leftarrow 1 - u_i \qquad \text{(min-cut)}$$
$$z_j \leftarrow v_j$$

$$\begin{array}{lll}
\min & Z^1 = \sum_i b_i w_i + \sum_j c_j z_j & (6.4) \\
\text{s.t.} & v_j - u_i \geq 0, \quad j \in S_i & \\
 & u_i + w_i \geq 1, \quad i = 1, \ldots, m & (6.5) \\
 & -v_j + z_j \geq 0, \ j = 1, \ldots, n & (6.6) \\
 & u_i, \ v_j, \ w_i, \ z_j \in \{0, 1\} &
\end{array}$$

Min-Cut Formulation

Property 1: If $(\bar{u}, \bar{v})$ is feasible to the original problem, then $(\bar{u}, \bar{v}, \bar{w} = 1 - \bar{u}, \bar{z} = \bar{v})$ is feasible to the new problem.

Property 2:

$$\begin{aligned} Z^1 &= \textstyle\sum_i b_i \bar{w}_i + \sum_j c_j \bar{z}_j = \sum_i b_i(1 - \bar{u}_i) - \sum_j c_j \bar{v}_j \\ &= \textstyle\sum_i b_i - Z \end{aligned}$$

Property 3: If $(\bar{u}, \bar{v}, \bar{w}, \bar{z})$ is optimal to the new problem, then

$$\begin{aligned} &\bar{w}_i = 1 - \bar{u}_i && (\text{because } b_i > 0 \ \& \ (6.5)) \\ &\bar{z}_j = \bar{v}_j && (\text{because } c_j > 0 \ \& \ (6.6)) \end{aligned}$$

Hence $\bar{u}, \bar{v}$ is feasible to the original problem and $Z^1 = \sum_i b_i - Z$. This means $(\bar{u}, \bar{v})$ is optimal to the original problem.

Min-Cut Formulation

Step 3: Min-Cut formulation

Add new variables u_0 and v_{n+1} and w_{ij} $(i = 1, ..., m;\ j = 1, ..., n)$ and let K be a large number.

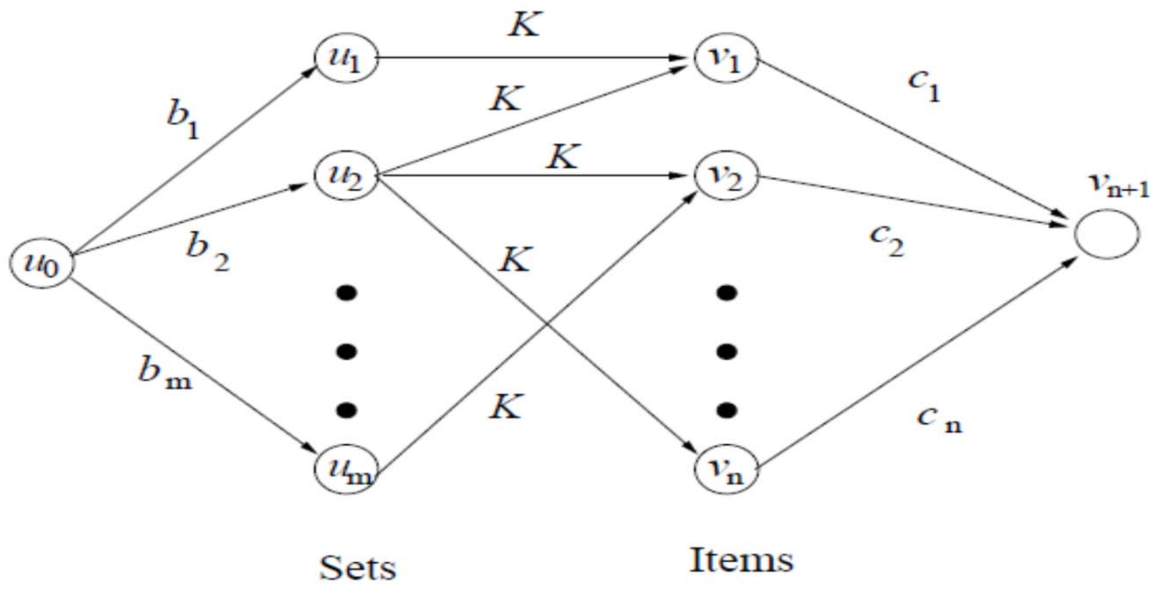
$$
\begin{aligned}
\text{min} \quad & z = \textstyle\sum_i b_i w_i + \sum_j c_j z_j + \sum_{ij} K w_{ij} \\
\text{s.t.} \quad & v_j - u_i + w_{ij} \geq 0, \quad \forall j \in S_i \\
& u_i - u_0 + w_i \geq 0, \quad i = 1, \ldots, m \\
& v_{n+1} - v_j + z_j \geq 0, \quad j = 1, \ldots, n \\
& u_0 - v_{n+1} \geq 1, \\
& u_i,\ v_j,\ w_i,\ z_j,\ w_{ij} \in \{0, 1\}
\end{aligned}
\tag{6.7}
$$

Min-Cut Formulation

Property 4: when $K \to \infty$, in an optimal solution, $u_0 = 1$, $v_{n+1} = 0$, and $w_{ij} = 0$.

Property 5: (6.7) is a min-cut problem except the integer-variable restrictions, which can be ignored and replaced by the non-negativity restrictions.

Property 6: Network for provisioning problem.



u_0
u_1
u_2
u_m
v_1
v_2
v_n
v_{n+1}
b_1
b_2
b_m
K
K
K
K
K
c_1
c_2
c_n
Sets
Items