15.082J and 6.855J and ESD.78J
November 2, 2010

Network Flow Duality and Applications of Network Flows

Overview of Lecture

- Applications of network flows
  - shortest paths
  - maximum flow
  - the assignment problem
  - minimum cost flows

- Linear programming duality in network flows and applications of dual network flow problems

- **Applications of network flows**
  - **shortest paths**
  - **maximum flow**
  - **the assignment problem**
  - **minimum cost flows**

Most reliable paths

Let p_{ij} be the probability that an arc is working, and that all arcs are independent.

The probability that a path P is working is $\prod_{(i,j)\in P} p_{ij}$

What is the most reliable path from s to t, that is the one that maximizes the probability of working?

$$\begin{array}{ll} \max & \prod_{(i,j)\in P} p_{ij} \\ s.t. & P \in \mathcal{P}(s,t) \end{array}$$

$$\begin{array}{ll} \min & \prod_{(i,j)\in P} 1/p_{ij} \\ s.t. & P \in \mathcal{P}(s,t) \end{array}$$

$$\begin{array}{ll} \min & \log\left(\prod_{(i,j)\in P} 1/p_{ij}\right) = \sum_{(i,j)\in P} \log\left(1/p_{ij}\right) \\ s.t. & P \in \mathcal{P}(s,t) \end{array}$$

Let $c_{ij} = \log 1/p_{ij}$

Dynamic Shortest Paths

Suppose that the time it takes to travel in arc (i, j) depends on when one starts. (e.g., rush hour vs. other hours in road networks.)

Let $c_{ij}(t)$ be the time it takes to travel in (i, j) starting at time t. What is the minimum time it takes to travel from node 1 to node n starting at 7 AM?

Start time

	7	7:10	7:20	7:30	7:40	7:50	…
(1,2)	20	30	30	20	…	…	…
(1,3)	10	10	10	10	…	…	…
(2,3)	20	20	20	20	…	…	…
(3,4)	10	20	20	10	…	…	…
…							

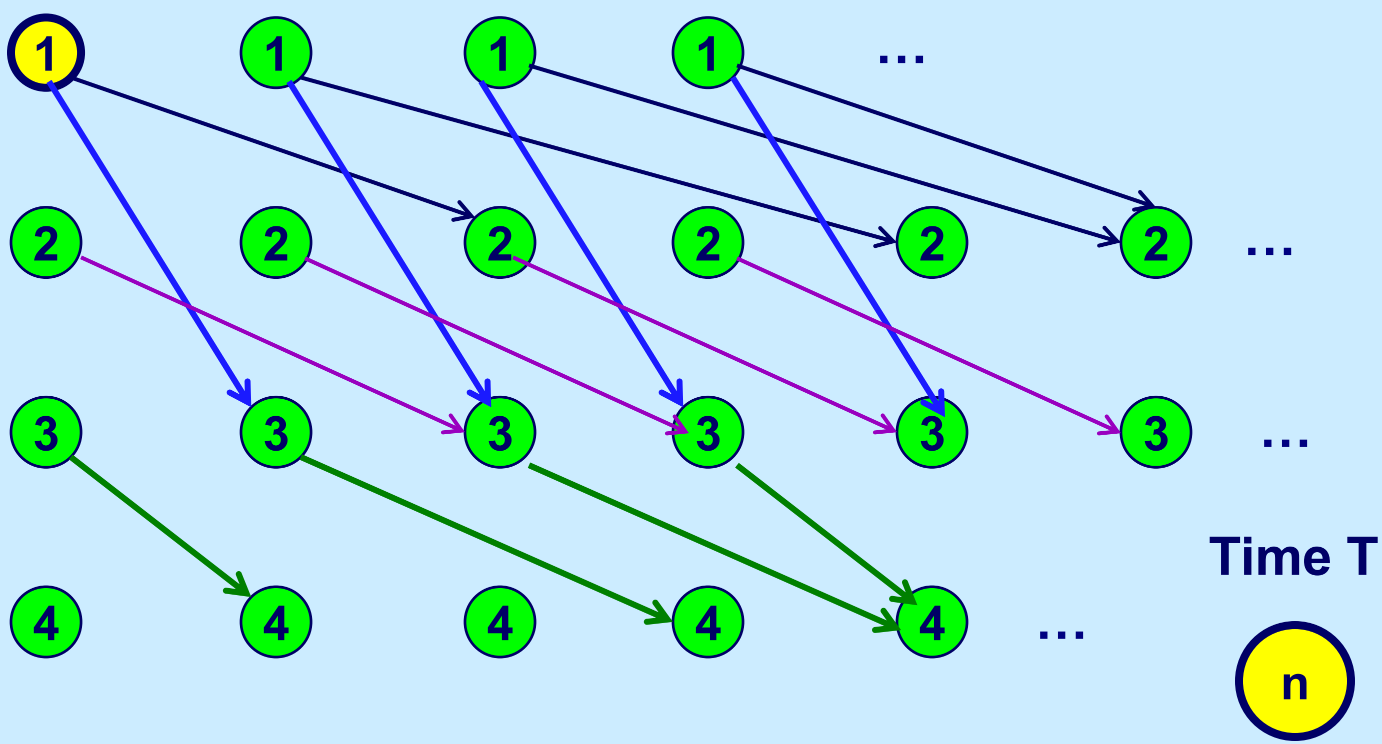
arc

travel time in minutes

The time expanded network

(1,2)	20	30	30	20	…	…
(1,3)	10	10	10	10	…	…
(2,3)	20	20	20	20	…	…
(3,4)	10	20	20	10	…	…
	7	7:10	7:20	7:30	7:40	7:50

What is the minimum time T such that there is a path from node 1 at 7 AM to node n at time T?

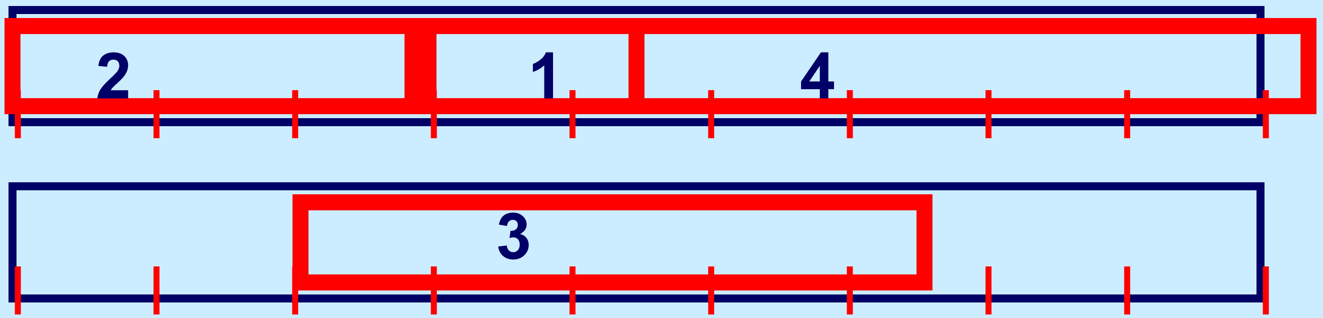


- **Applications of network flows**
  - **shortest paths**
  - **maximum flows**
  - **the assignment problem**
  - **minimum cost flows**

App. 6.4 Scheduling on Uniform Parallel Machines

Job(j)	1	2	3	4
Processing Time	1.5	3	4.5	5
Release Time	2	0	2	4
Due Date	5	4	7	9

Suppose there are 2 parallel machines. Is there a feasible schedule?

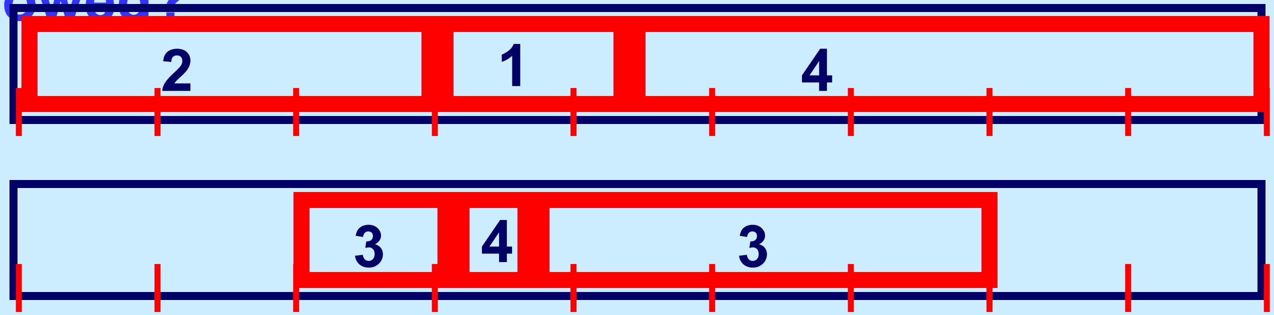


The best (infeasible) schedule without preemption.

A feasible schedule with preemption

Job(j)	1	2	3	4
Processing Time	1.5	3	4.5	5
Release Time	2	0	2	4
Due Date	5	4	7	9

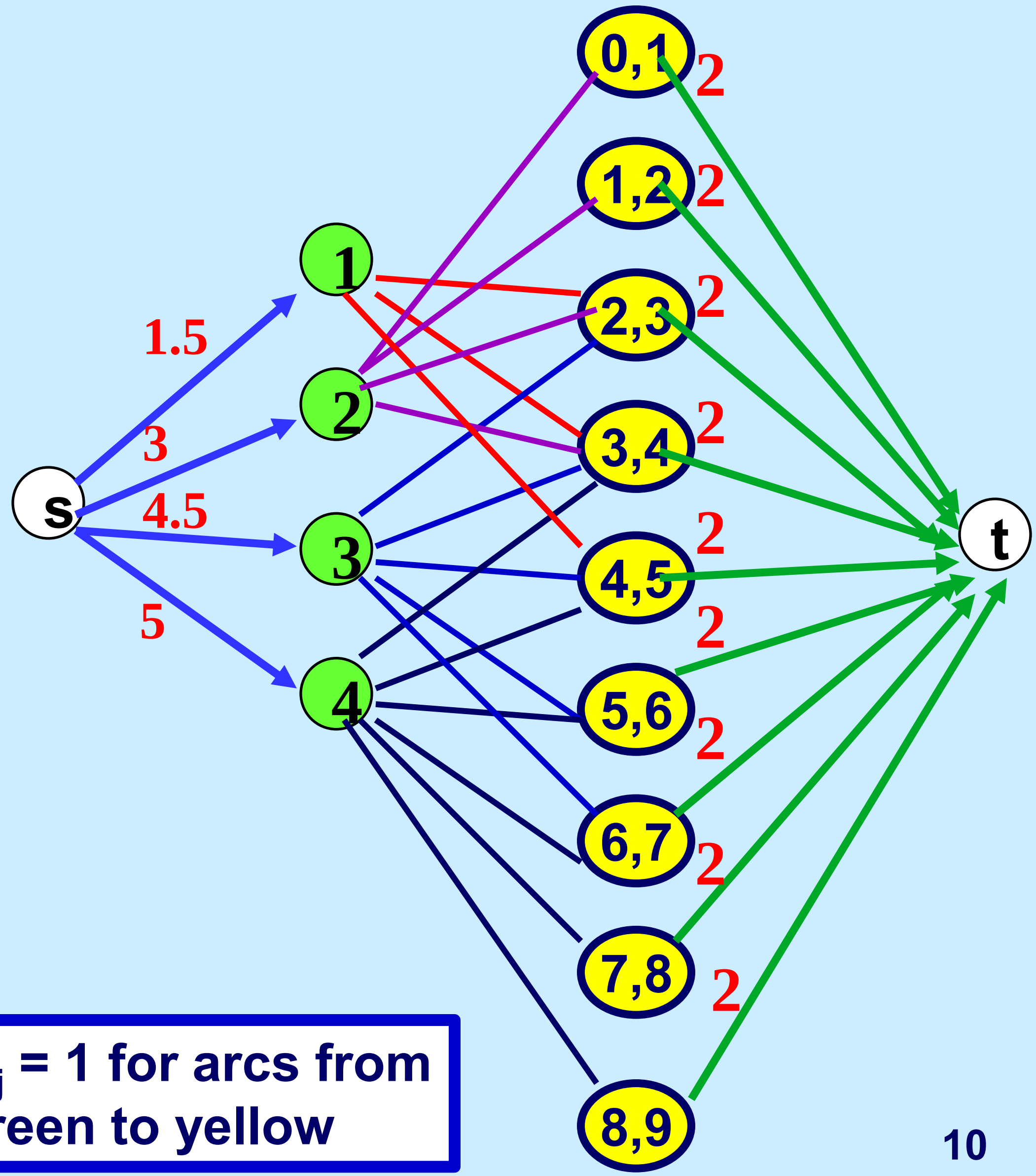
Preemption permits a job to be split into two or more parts and scheduled on the same or different machines, but not two machines at the same time. How can one find a feasible schedule when preemption is allowed?



Transformation into a maximum flow problem

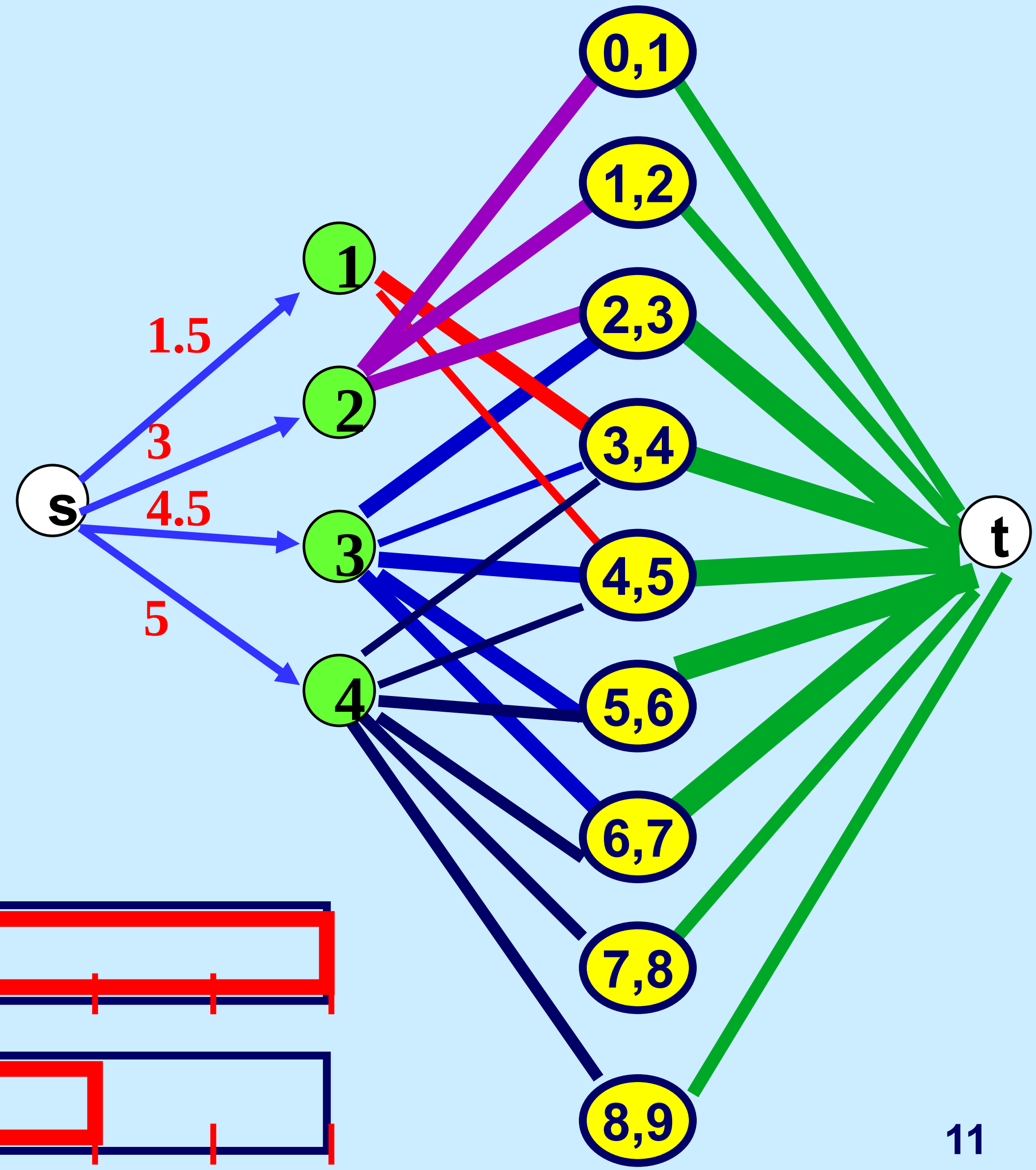
Time allocation is the thing that is flowing.

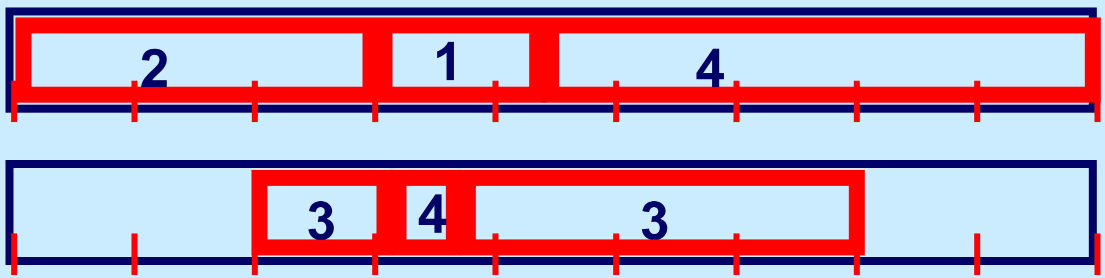
- **Job j must have p(j) units of time allocated to it.**
- **Job j can be scheduled for at most one time unit in any period.**
- **If there are m machines, then at most m different jobs may be scheduled in any period.**



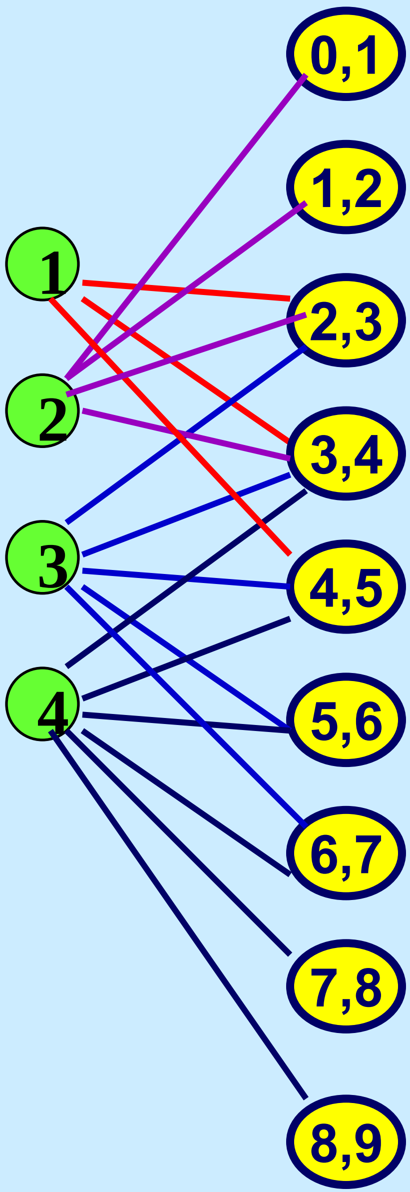
u_{ij} = 1 for arcs from green to yellow

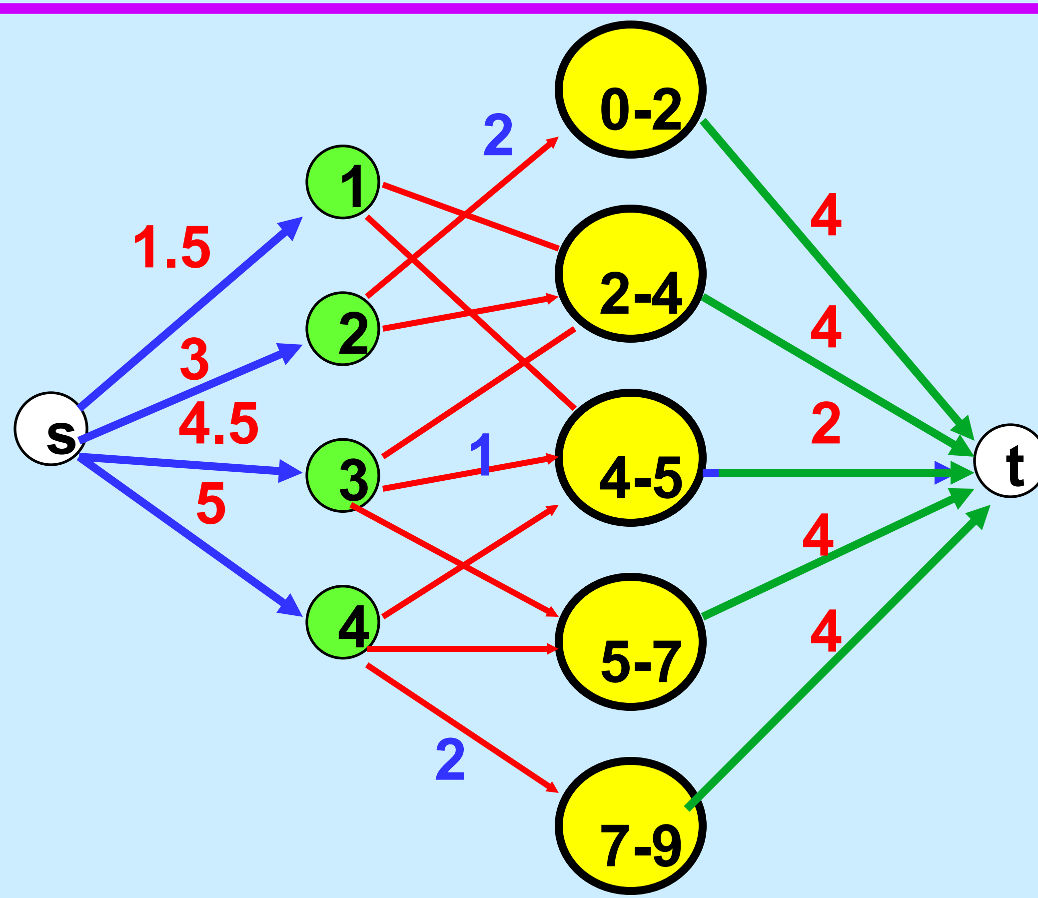
The optimal allocation and flow





A more efficient transformation





Merge two adjacent period nodes if the same set of tasks can be scheduled in both periods. The number of nodes after merging is less than 2 |J|, where J = set of jobs.

Sports Writer Problem

	Games Won	Games Left
Bos	82	8
NY	77	8
Balt	80	8
Tor	79	8
Tamp	74	9

	Bos	NY	Balt	Tor	Tamp
Bos	--	1	4	1	2
NY	1	--	0	3	4
Balt	4	0	--	1	0
Tor	1	3	1	0	3
Tamp	2	4	0	3	0

Games remaining

Has Tampa already been eliminated from winning in this hypothetical season finale?

Data assuming that Tampa wins all its remaining games.

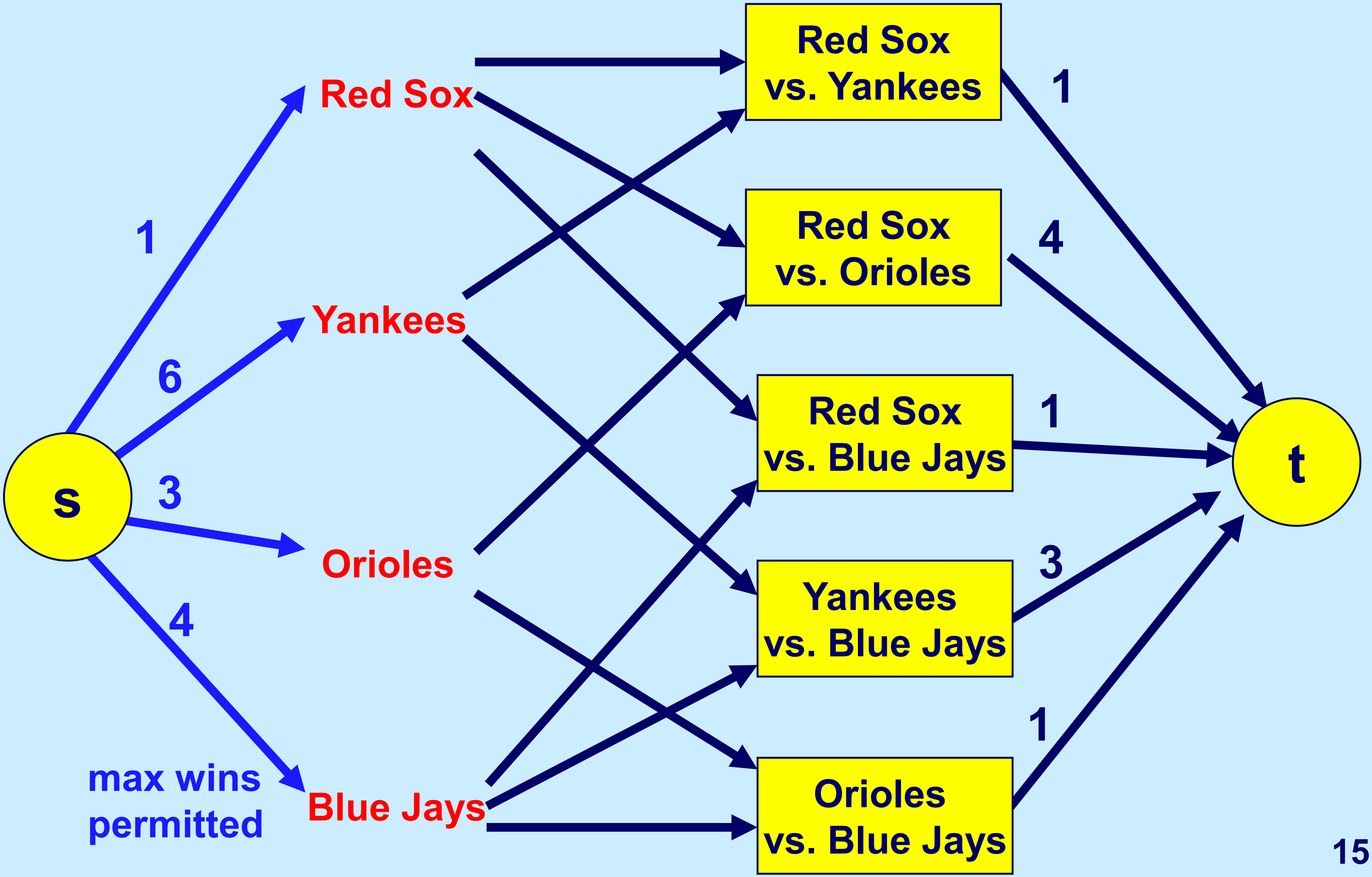
	Games Won	Games Left
Bos	82	6
NY	77	4
Balt	80	6
Tor	79	5
Tamp	83	0

	Bos	NY	Balt	Tor
Bos	--	1	4	1
NY	1	--	0	3
Balt	4	0	--	1
Tor	1	3	1	--

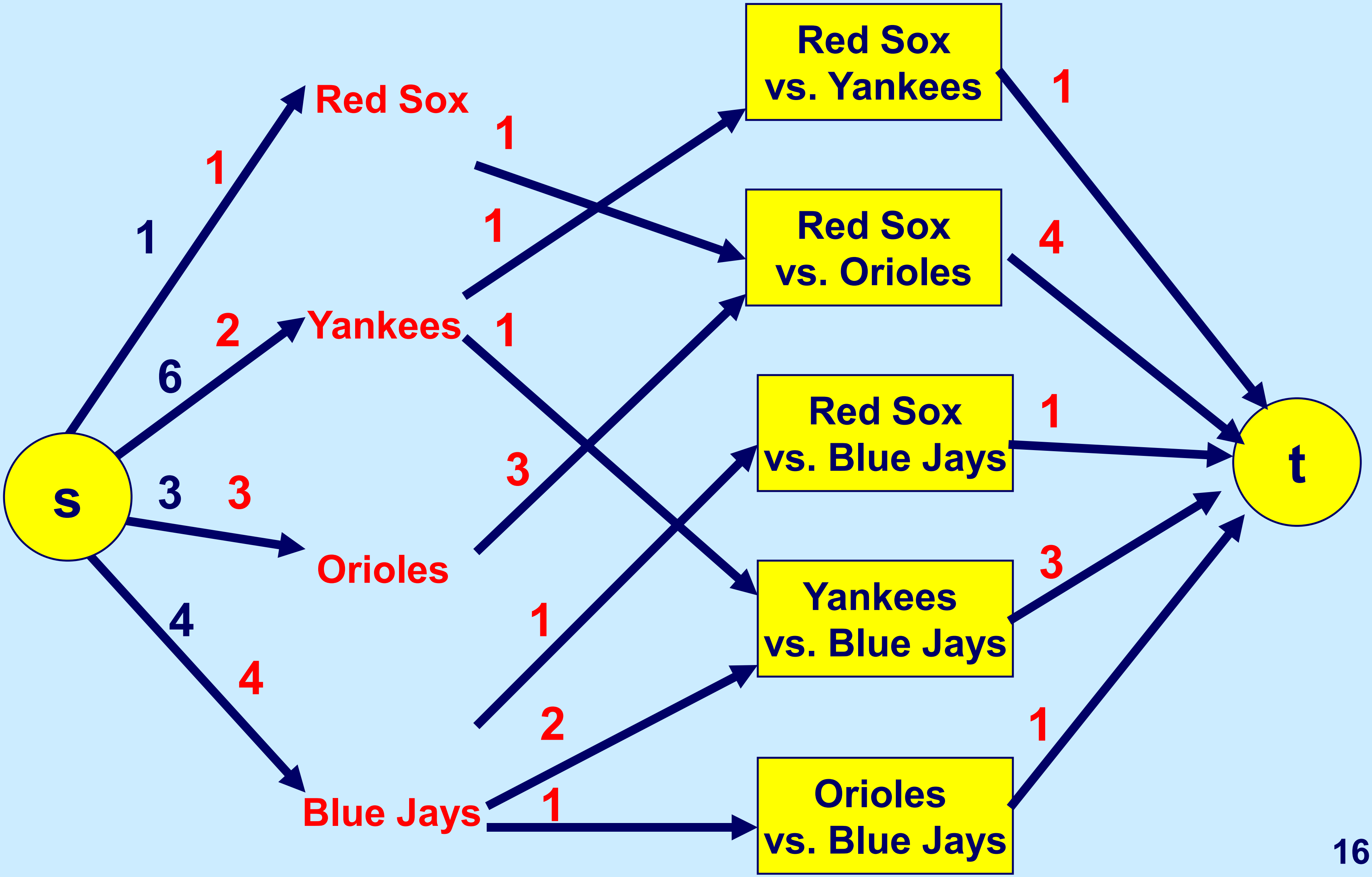
Games remaining

Question: can the remaining games be played so that no team wins more than 83 games?

Flow on (i,j) is interpreted as games won.



A Maximum Flow



- **Applications of network flows**
  - shortest paths
  - maximum flows
  - the assignment problem
  - minimum cost flows

The Assignment Problem

n persons

n tasks

u_{ij} = utility of assigning person i to task j

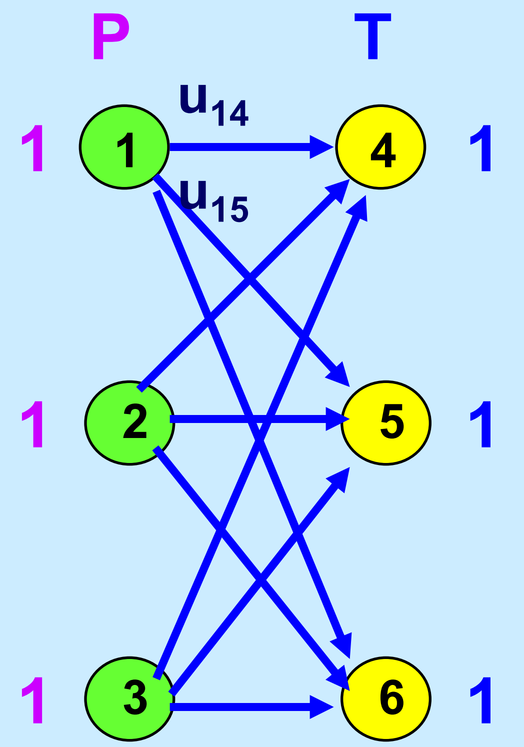
- Maximize the sum of the utilities
- Each person gets assigned to a task
- Each task has one person assigned to it.

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j$$

$$x_j \in \{0,1\} \qquad \forall j$$

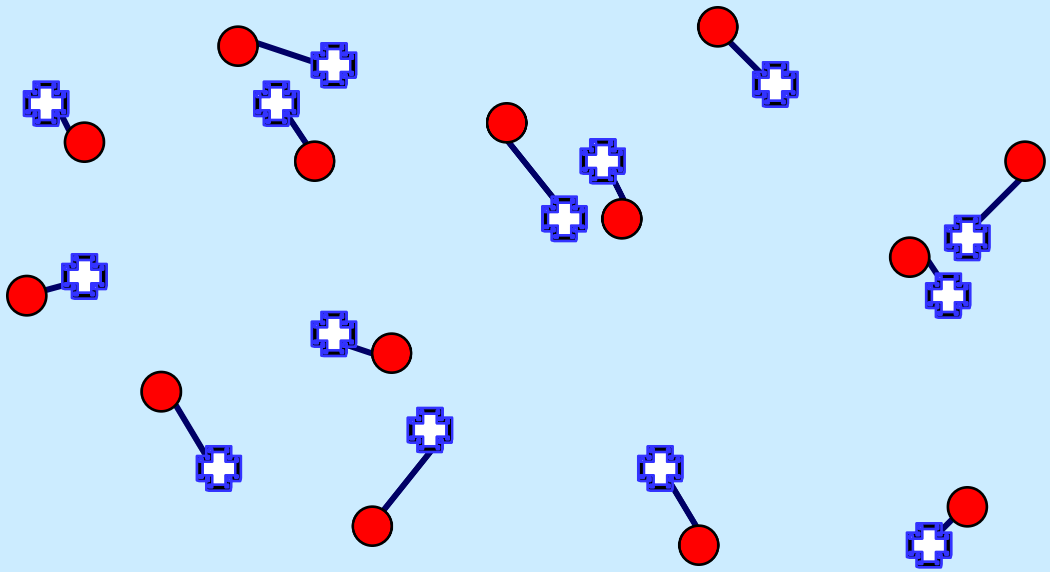


Identifying Moving Targets in Space

Suppose that there are moving targets in space.

You can identify each target as a pixel on a radar screen.

Given two successive pictures, identify how the targets have moved.

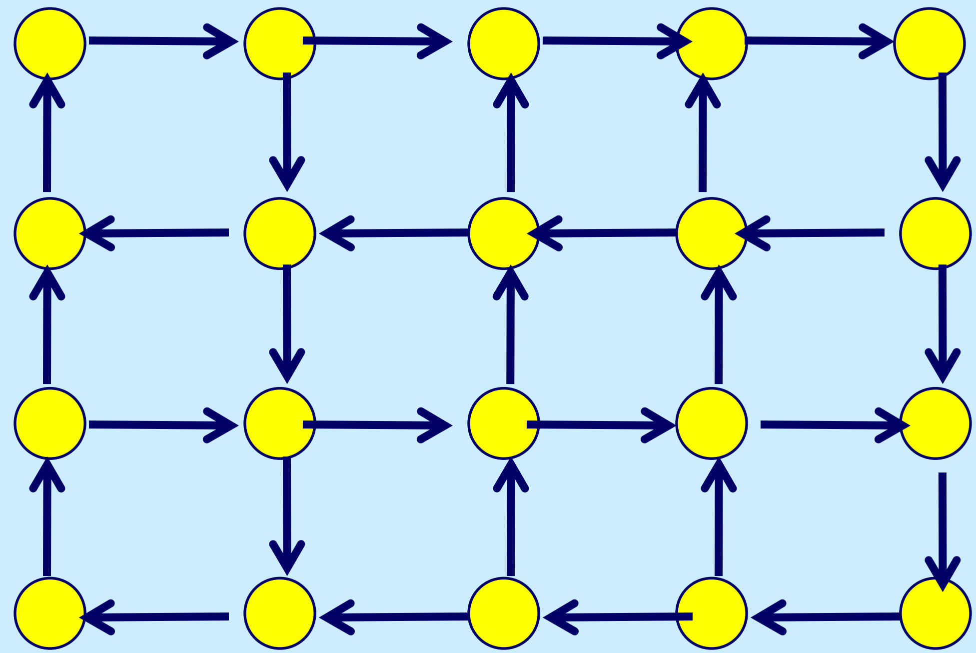


This is an efficient way of tracking items.

- **Applications of network flows**
  - **shortest paths**
  - **maximum flows**
  - **the assignment problem**
  - **minimum cost flows**

Chinese Postman Problem (directed version)

A postman (using a postal truck) wants to visit every city street at least once while minimizing the total travel time.



x_{ij} = number of times that the postman traverses arc (i, j)

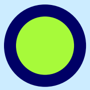
c_{ij} = length of (i, j)

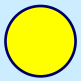
$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij}$$

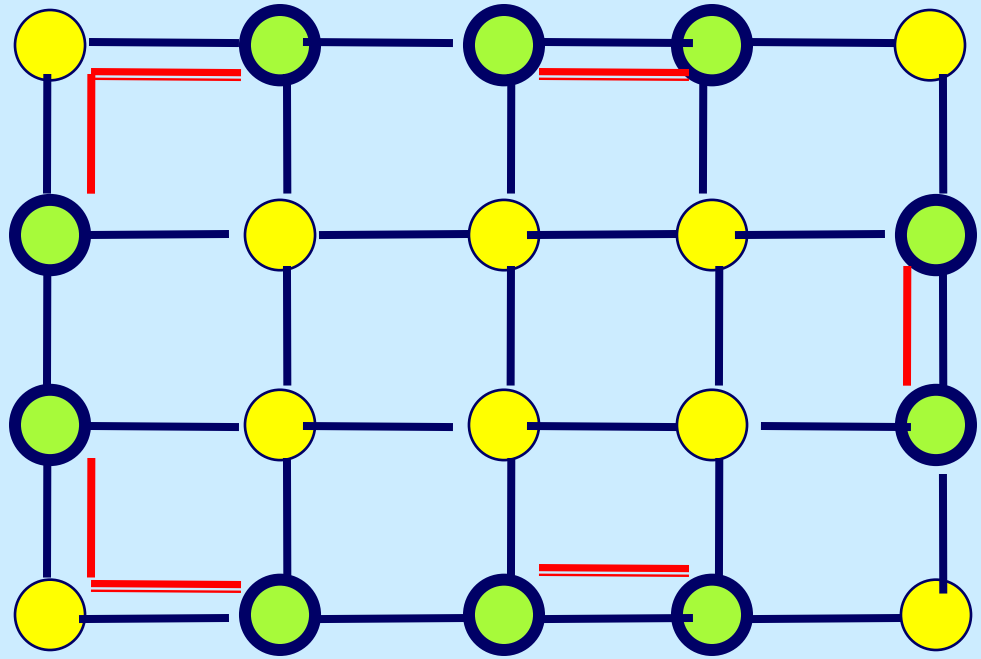
$$s.t. \quad \sum_j x_{ij} - \sum_k x_{ki} = 0 \quad \forall i \in N$$

$$x_{ij} \geq 1 \qquad \forall (i,j) \in A$$

Chinese Postman Problem (undirected)


odd degree


even degree



d_{ij} = minimum length of a path from node i to node j.

Compute d_{ij} for all nodes i, j of odd degree.

Add paths joining nodes of odd degree so as to minimize the total d-length. (This is a nonbipartite matching problem.)

Duality for Network Flow Problems

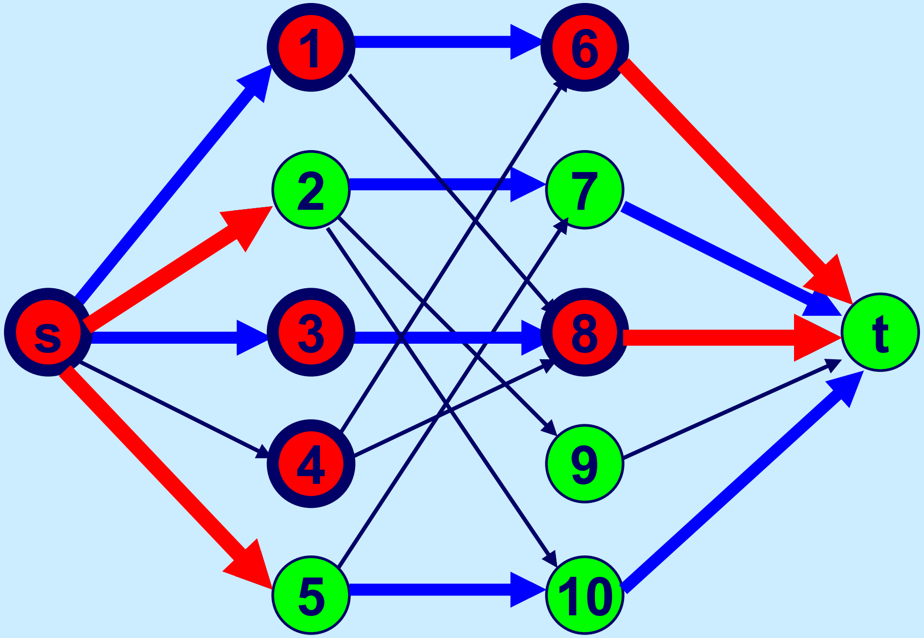
Each network flow problem has a corresponding problem called the “dual”.

Rest of lecture:

- Review of max flow min cut theorem
- Weak Duality in linear programming
- Strong Duality in linear programming
- Duality for shortest paths plus applications
- Duality for min cost flow plus applications

Duality for the max flow problem

***Theorem*. (Weak duality). If x is any s-t flow and if (S, T) is any s-t cut, then the flow out of s is at most the capacity of the cut (S,T).**



In this example, the capacities of all arcs is 1.

The max flow consists of the thick arcs.

If S = red nodes, then min cut (S, N\S).

***Theorem*. (Max-flow Min-Cut). The maximum flow value is the minimum capacity of a cut.**

Weak Duality in Linear Programming

$$\begin{array}{ll} \max & cx \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

Primal Problem

Weak Duality Theorem. Suppose that x is feasible for the primal problem and that π is feasible for the dual problem. Then cx ≤ πb.

$$\begin{array}{ll} \min & \pi b \\ s.t. & \pi A \geq c \end{array}$$

Dual Problem

Proof

$$\pi A \geq c \text{ and } x \geq 0 \Rightarrow \pi A x \geq cx$$

$$Ax = b \Rightarrow \pi A x = \pi b$$

Therefore, $\pi b \geq cx.$

Strong Duality in Linear Programming

$$\begin{aligned} \max \quad & cx \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Primal Problem

$$\begin{aligned} \min \quad & \pi b \\ s.t. \quad & \pi A \geq c \end{aligned}$$

Dual Problem

Strong Duality Theorem.

Suppose that the primal problem has an optimal solution x*.

Then the dual problem also has an optimal solution, say π*, and the two optimum objective values are the same. That is, cx* = π*b.

Note: it is not obvious that max-flow min-cut is a special case of LP duality.

Duality for the shortest path problem

Let G = (N, A) be a network, and let c_{ij} be the length or cost of arc (i, j). The shortest path problem is to find the path of shortest length from node 1 to node n.

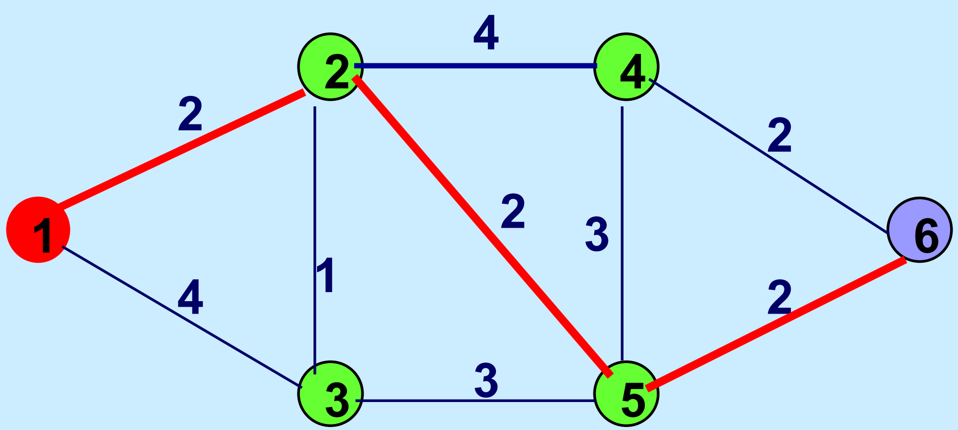
We say that a distance vector d( ) is dual feasible for the shortest path problem if

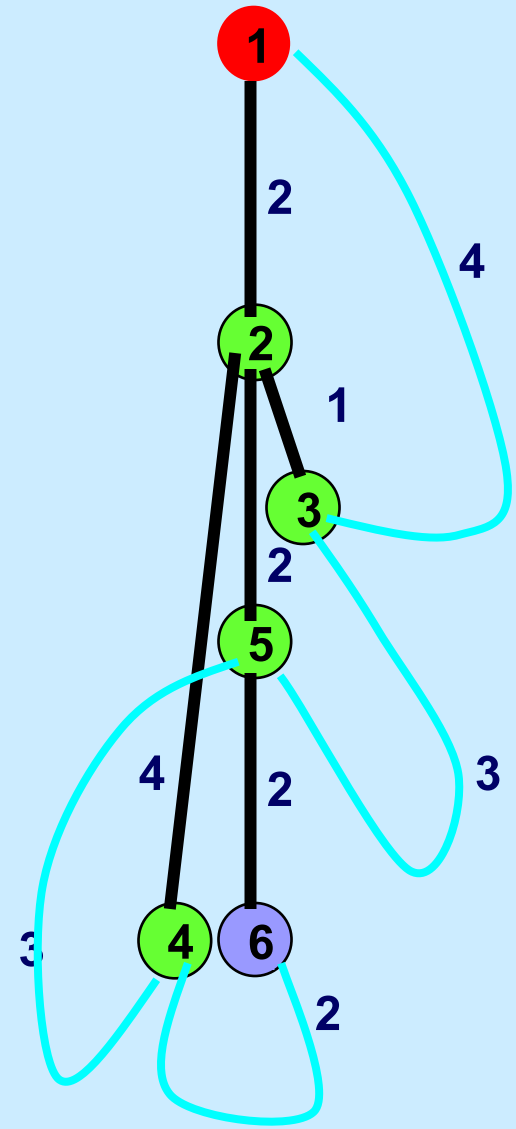
1. $d(1) = 0$
2. $d(j) \leq d(i) + c_{ij}$ for all $(i, j) \in A$.

The dual shortest path problem is to maximize d(n) subject to the vector d( ) being dual feasible.

Duality Theorem for Shortest Path Problem

Let G = (N, A) be a network, and let c_{ij} be the length or cost of arc (i, j). If there is no negative cost cycle, then the minimum length of a path from node 1 to node n is the maximum value of d(n) subject to d( ) being dual feasible.





Application: Optimum Paragraph Layout

T_eX optimally decomposes paragraphs by selecting the breakpoints for each line optimally. It has a subroutine that computes the attractiveness F(i,j) of a line that begins at word i and ends at word j-1. How can one use F(i,j) to create a shortest path problem whose solution will solve the paragraph problem?

The paragraph layout problem can be modeled as a shortest path problem or the dual of a shortest path problem.

Application: Optimum Paragraph Layout

Let $d^*(j)$ be the value of laying out words 1 to j-1 most attractively. d^* can be computed as follows.

$$\begin{array}{lll} \min & d(n+1) & \\ \text{s.t} & d(j) \geq d(i) + F(i, j) \ \forall\ (i,j) \in A & \quad d(i) \leq d(j) - F(i, j) \\ & d(1) = 0 & \end{array}$$

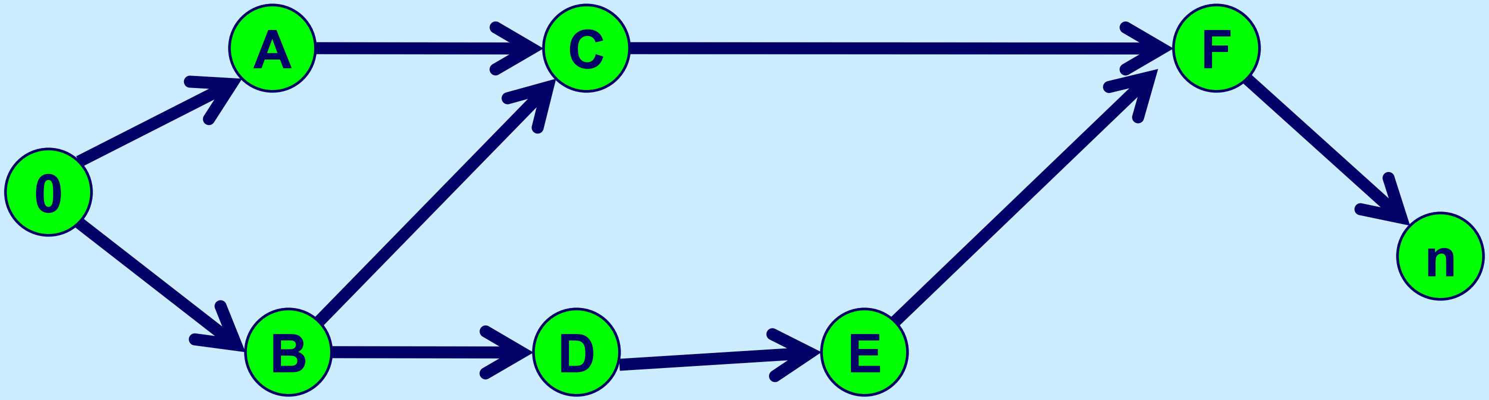
For any feasible vector d, d(j) is an upper bound on the beauty of laying out words 1 to j – 1.

The most accurate upper bound gives the optimum beauty.

There is a close connection to dynamic programming.

Application: project scheduling

Activity	Predecessor	Duration
A (Train workers)	None	6
B (Purchase raw materials)	None	9
C (Make subassembly 1)	A, B	8
D (Make subassembly 2)	B	7
E (Inspect subassembly 2)	D	10
F (Assemble subassemblies)	C, E	12



Application: project scheduling

Activity	Predecessor	Duration
A (Train workers)	None	6
B (Purchase raw materials)	None	9
C (Make subassembly 1)	A, B	8

Let s(i) be the start time of task i.

Let f(i) be the finish time of task i.

Let p(i) be the processing time of task i.

minimize $f(n)$

subject to $s(0) = 0$

$f(j) = s(j) + p(j)$ for all $j \neq 0$ or n

$s(j) \geq f(i)$ if i precedes j.

Corresponds to a longest path problem. We can make it a shortest path problem by letting $g(j) = - f(j)$ for all j.

Project scheduling with just-in-time delivery

Suppose that for some tasks i and j, task j must be started within h(i, j) time units of task i finishing.

Let s(i) be the start time of task i.

Let f(i) be the finish time of task i.

Let p(i) be the processing time of task i.

minimize $\quad f(n)$

subject to $\quad s(0) = 0$

$f(j) = s(j) + p(j) \quad$ for all $j \neq 0$ or n

$s(j) \geq f(i)$ if i precedes j

$s(j) \leq f(i) + h(i, j) \qquad$ or $\quad f(i) \geq s(j) - h(i, j)$

Duality for minimum cost flows

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ s.t. \quad & \sum_j x_{ij} - \sum_k x_{ki} = b_i \quad \forall i \in N \\ & x_{ij} \geq 0 \qquad\qquad \forall (i,j) \in A \end{aligned}$$

Uncapacitated min cost flow problem

$$\begin{aligned} \max \quad & \sum_{i\in N} \pi_i b_i \\ s.t. \quad & c_{ij} - \pi_i + \pi_j \geq 0 \qquad \forall (i,j) \in A \\ & \pi_i - \pi_j \leq c_{ij} \end{aligned}$$

Dual of the uncapacitated MCF problem.

Theorem. Suppose that x is feasible for the uncapacitated MCF problem, and π is feasible for the dual problem. Then cx ≥ πb.

If one of the problems has a finite optimum, then so does the other, and the two values are the same.

More on Duality

1. **One can solve the dual problem using an algorithm for solving the uncapacitated MCF problem.**

2. **Any linear programming problem in which every constraint is either a lower bound on a variable or an upper bound on a variable or of the form "$y_i - y_j \leq c_{ij}$" is the dual of a minimum cost flow problem.**

Maximum Weight Closure of a Graph

Let G = (N, A). Let w_i be the weight of node i.

A subset S ⊆ N is called a closure if there are no arcs leaving the subset. That is, if i ∈ S and if (i, j) ∈ A, then j ∈ S.

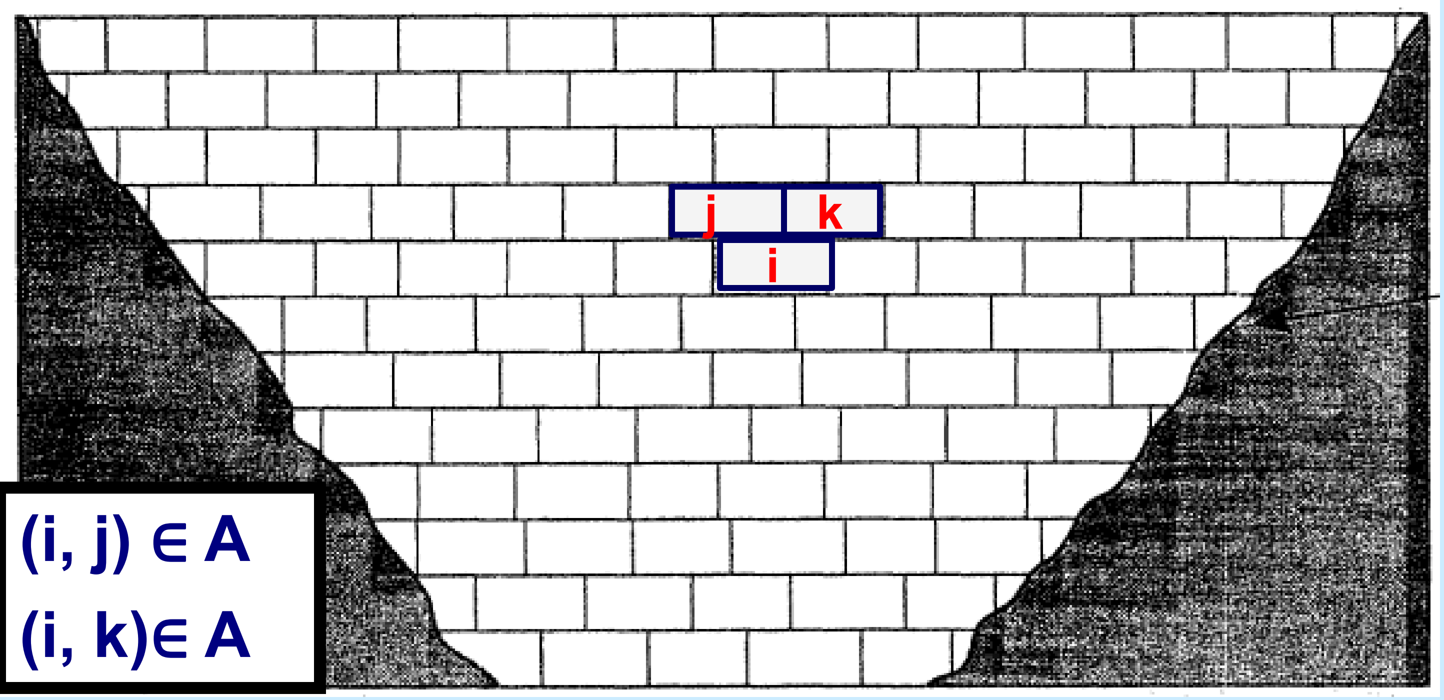
The maximum weight closure problem is to find a closure of maximum weight. It is the dual of a minimum cost flow problem.

$$
\begin{array}{lll}
\max & \sum_{i \in N} w_i y_i & \\
s.t. & y_i - y_j \le 0 & \forall (i,j) \in A \\
 & 0 \le y_i \le 1 & \forall i \in N
\end{array}
$$

Open Pit Mining

Suppose an open pit mine is subdivided into blocks. We create a graph G = (N, A) as follows:

1. There is a node for each block
2. If block j must be removed before block i, then (i, j) ∈ A.
3. The net revenue from block i is w_i.



(i, j) ∈ A

(i, k)∈ A

Special case of the closure problem.

Project management with "crashing"

Suppose that one can reduce the time at which task j is completed for each j. The cost of reducing the time for task i is c_i per unit of time.

What is the least cost schedule that completes all tasks by time T?

Let s(i) be the start time of task i.
Let f(i) be the finish time of task i.
Let p(i) be the original processing time of task i.

$$\begin{aligned}
\min \quad & \textstyle\sum_i c_i \big(p(i) + s(i) - f(i)\big) \\
s.t. \quad & f(j) - s(i) \le 0 \qquad \forall (i, j) \in A \\
& f(i) - s(i) \le p(i) \quad \forall i \in N \\
& s(0) = 0; \qquad f(n) \le T
\end{aligned}$$

The above LP is the dual of a minimum cost flow problem.

Summary

There are hundreds of direct applications of the minimum cost flow problem or its dual.

Even more common, min cost flow problems arise as subproblems of a larger and more complex problem. We will see more of this in a few lectures from now.

Next Lecture: the simplex algorithm for the min cost flow problem.