

15.053

April 5, 2007

- **The Minimum Cost Flow Problem**

To do.

- Add an extra slide showing that the flows for the baseball elimination problem and how it leads to show that Tampa can win.

- Add the matching problem from the previous lecture and match it up to the 0-1 matrix form. Then give the duality result in terms of eliminating the three popular stars.
- check out who wants a piece of candy.

Quotes of the day

**A process cannot be understood by stopping it.
Understanding must move with the flow of the
process, must join it and flow with it.**

-- Frank Herbert

**No question is so difficult to answer as that to
which the answer is obvious.**

-- George Bernard Shaw

taken from www.flowgo.com

An Application

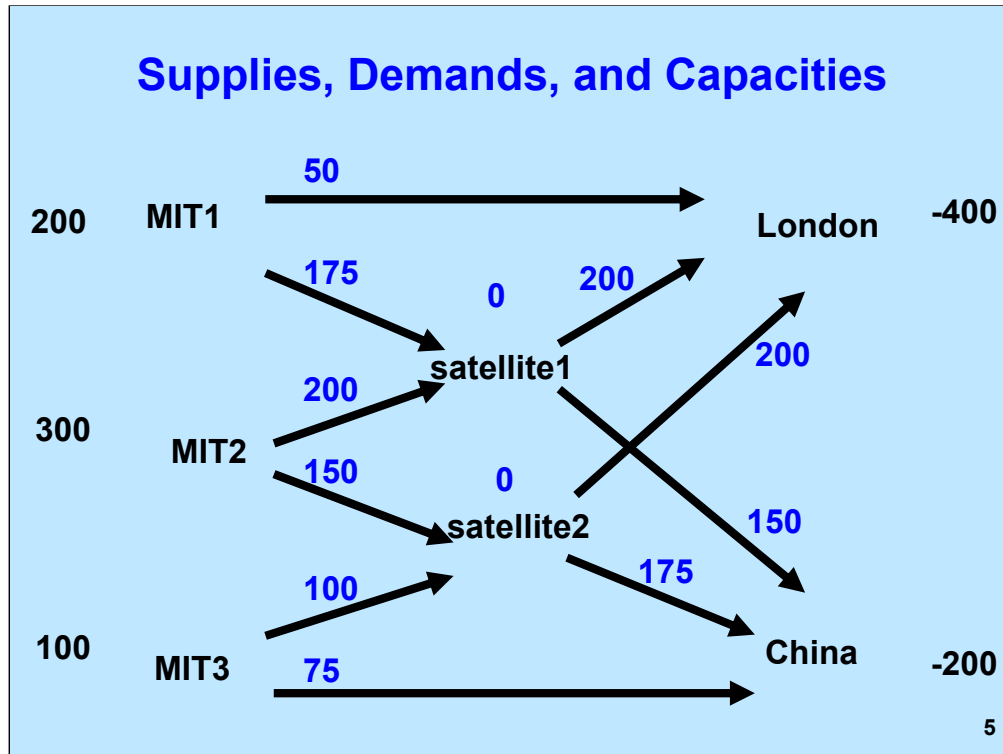
The year is 2010, and there is incredible demand for the 15.053 live podcasts. MIT has set up three sites to handle the incredible load.

The major demand for the podcasts are in London and China. There are some direct links from each of the three MIT sites, and the podcasts can be sent through two intermediate satellite dishes as well.

Each node has a supply (or a demand) indicating how much should be shipped from (or to) the node. Each link (arc) has a unit cost of shipping flow, and a capacity on how much can be sent per second. What is the cheapest way of handling the required load?

4

While the application to 15.053 podcasting is fanciful, this application is of great importance on the web. For example, the demand on CNN servers is very high. In order to meet this demand, it helps to have servers throughout the world with the content from CNN. This would be challenging and difficult for CNN to manage, and so it is handled by Akamai, a company founded by MIT students and faculty.



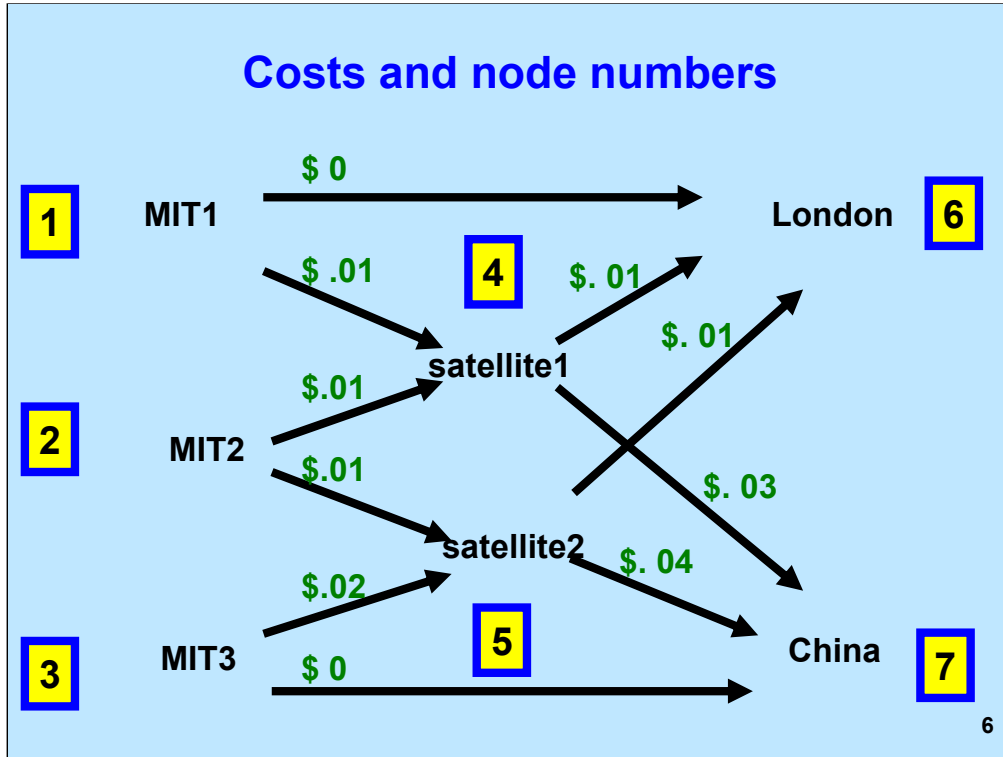
Usually, one lists the capacities on the same diagram as the costs. But students have found this too much data to absorb in a single slide. And so we have one slide with supplies, demands and capacities. The next slide will give the arc costs.

The node numbers are the supplies and demands. A positive number indicates a supply. A negative number indicates a demand. In a network flow problem, we have this constraint:

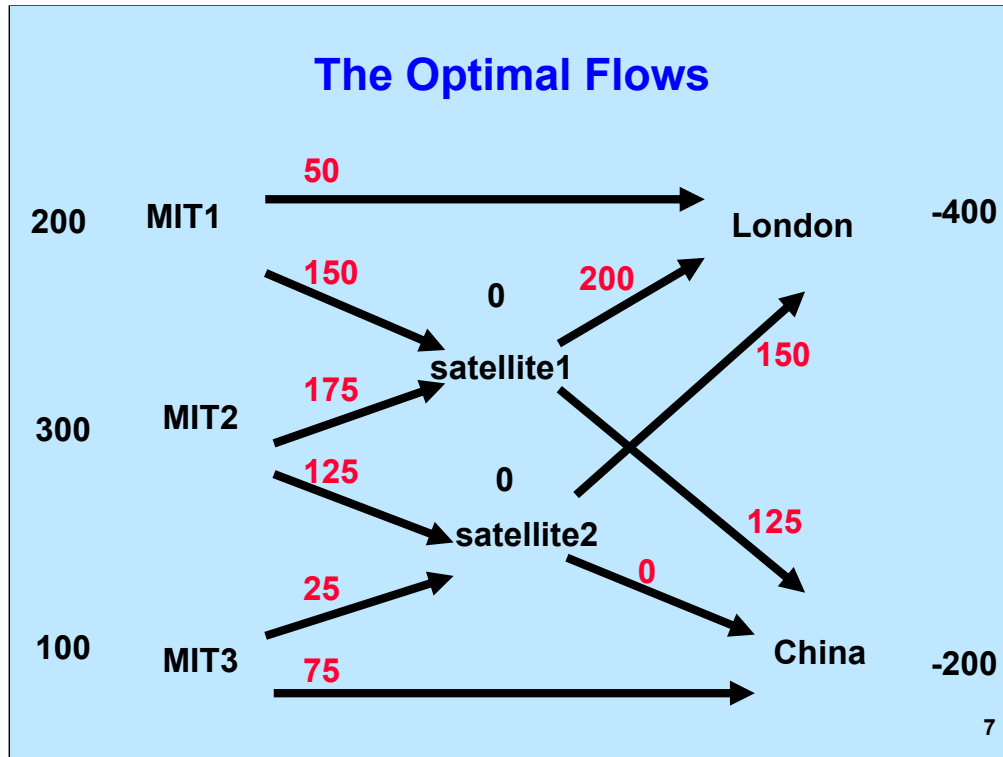
The total flow leaving a node minus the total flow entering the node is the “supply” at the node.

In this case, the total flow leaving the MIT node must be exactly 200. The total flow entering the London node (with the London Bridge) is exactly 400.

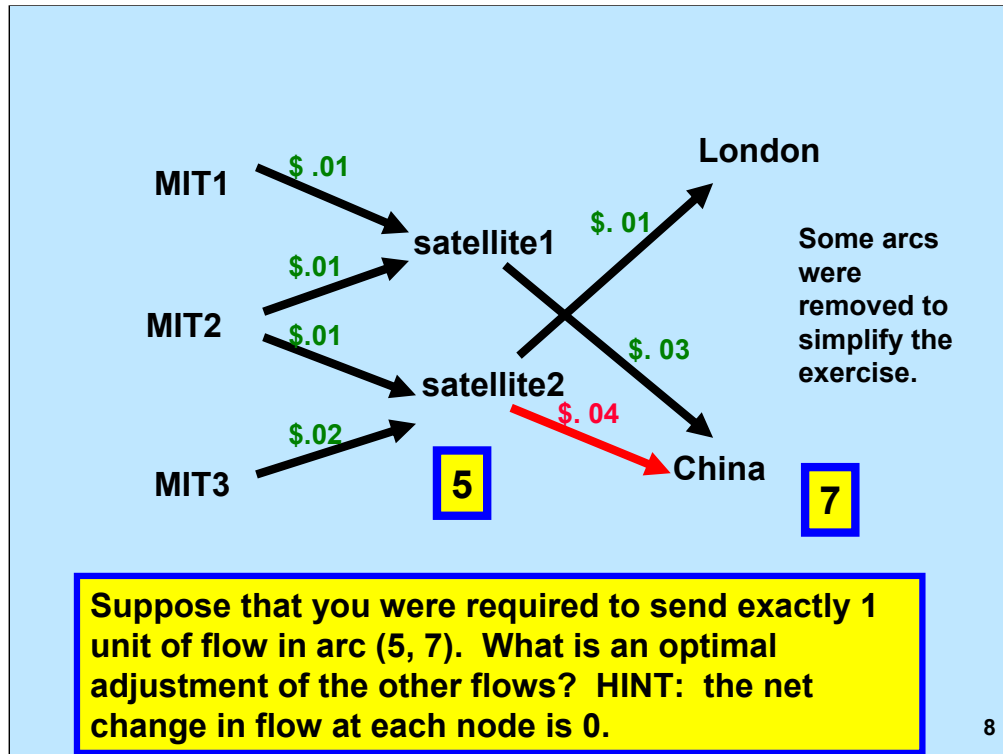
In practical cases, there are examples where the total flow leaving a node can be less than the supply. We can deal with such cases through a transformation into an equivalent problem in which the above constraint on supply/demand is satisfied with equality.



Here are the costs for the problem. they are not particularly realistic, but are just intended to illustrate a min cost network flow problem.



We found the optimal flows by modeling the problem on Excel and then using Excel Solver. Of course, all of the constraints are satisfied. But notice that the optimal solution has integer flows. This is not a coincidence. Optimal basic solutions to the min cost flow problem are integer valued so long as the supplies, demands, and capacities are all integer valued. (This can be proved.)



This slide gives a preview of what is to come. In order to increase the flow in (5, 7) by 1 unit, one needs to adjust the flows in other basic arcs. What this slide demonstrates is that the arcs on which flow changes form an undirected path P from node 5 to node 7. If we consider P union (5, 7), we get a cycle.

In network flow problems, it is quite common that flows adjust on cycles.

Formulation as a Linear Program

Let x_{ij} be the amount of flow sent from node i to node j .

Minimize **The cost of sending flow**

subject to **The amount shipped out of a node minus the amount shipped in to the node is the supply.**

$0 \leq x_{ij} \leq u_{ij}$ for all arcs (i, j) in A

- $G = (N, A)$
- usually n nodes, and m arcs
- arc costs c_{ij}
- arc capacities u_{ij}
- node supply/demands b_i

$$\sum_{i=1}^n b_i = 0$$

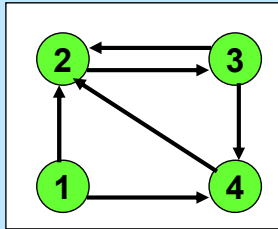
9

Here we model the min cost network flow problem as a linear program. The generic version of the min cost flow problem is given two slides later.

In words, the objective function is to minimize the total cost of sending flow.

The constraints are exactly what are given on this slide. There is a supply/demand constraint for each node as well as lower and upper bounds on arc flows.

Formulating a min cost flow problem



- x_{ij} = flow in (i, j)
- arc costs c_{ij}
- arc capacities u_{ij}
- node supply/demands b_i

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

x_{12}	x_{14}	x_{23}	x_{32}	x_{34}	x_{42}
----------	----------	----------	----------	----------	----------

RHS

=
=
=
=

$$0 \leq x_{ij} \leq u_{ij}$$

for all arcs
 $(i, j) \in A$

On the columns of network flow problems

- If we consider the flow constraints only, there is exactly one 1 and exactly one -1 in each column. All other elements are 0.
- If you ever see a constraint matrix with exactly one 1 and exactly one -1 in each column, with all other elements being 0, then it is a minimum cost flow problem.

A generic version of the min cost flow problem

Let x_{ij} be the flow on arc (i, j)

Minimize the cost of sending flow

s.t. (Flow out of i - Flow into i) = b_i

$$0 \leq x_{ij} \leq u_{ij}$$

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

s.t. $\sum_{j=1}^n x_{ij} - \sum_{k=1}^n x_{ki} = b_i$ for all i

$$0 \leq x_{ij} \leq u_{ij} \text{ for all } (i,j)$$

Overview of what comes next

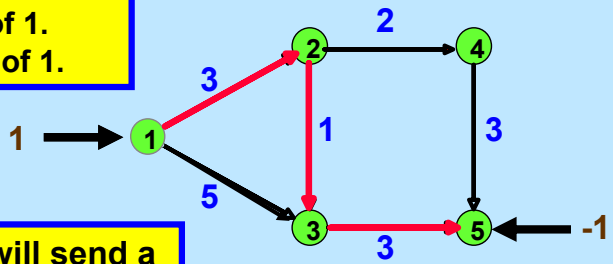
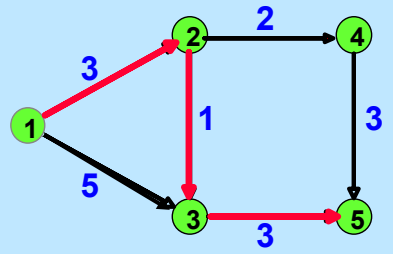
- shortest path and max flow are special cases

The shortest path problem with nonnegative arc lengths

Find the shortest path from node 1 to node 5.

**Translation to flow problem:
Node 1 has a supply of 1.
Node 5 has a demand of 1.**

The optimal solution will send a flow of 1 unit along the shortest path from node 1 to node 5.



When we described the shortest path problem, we did not view it as a “flow problem.” However, the shortest path from node s to node t can be viewed as finding the cheapest way of sending one unit of flow from node s to node t .

What I just wrote is not entirely true, but it is mostly true. It is true whenever all arc lengths are non-negative.

Actually, it is true even if arc lengths can be negative so long as there is no directed cycle with a negative length.

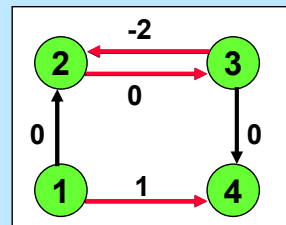
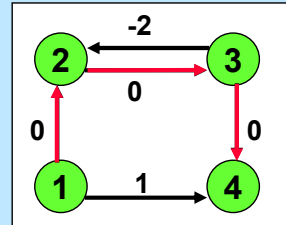
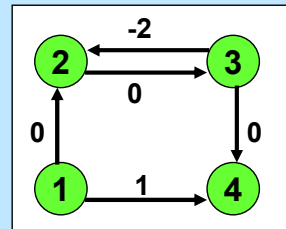
The example on the next slide illustrates the point.

The shortest path problem

The transformation does not work if arcs are permitted to have a negative length, even if all arc capacities are 1.

The shortest path from node 1 to node 4 is 1-2-3-4.

Here is the min cost flow satisfying a supply of 1 at node 1 and a demand of 1 at node 4.

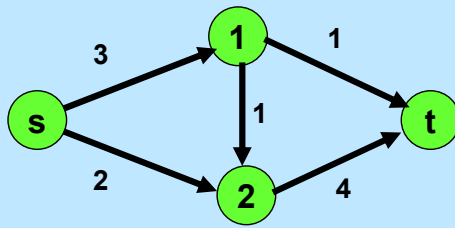


15

The graph on this slide has only two directed paths from node 1 to node 4: the path 1-4, and the path 1-2-3-4. The shortest of these paths is the path 1-2-3-4 with a total length of 0.

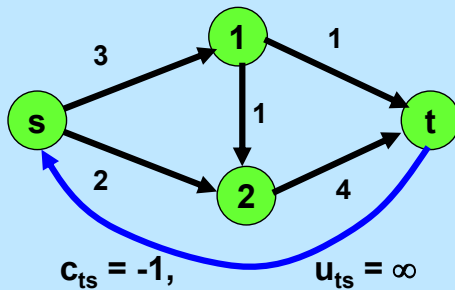
However, if we try to solve the min cost flow problem, something strange happens. Rather than obtaining the shortest path from 1 to 4, we obtain the path 1-4 together with the cycle 2-3-2. This flow has a cost of -1, which is less than the cost of either of the two paths from 1 to 4. (We are assuming that all capacities are 1 on this network.) The presence of a negative cost cycle implies that the min cost flow may not equal the shortest path.

The Maximum Flow Problem



To convert a max flow into a min cost flow, add an arc (t, s).

$c_{ts} = -1$.
 $c_{ij} = 0$ for all other arcs.
 $b_i = 0$ for all nodes.



opt solution: send as much flow from s to t as possible, and then send it back via (t, s).

16

The max flow problem is easily represented as a min cost flow problem. all one needs to do is to append the arc (t, s) to the max flow problem and give it a cost of -1 and a capacity that is larger than the largest possible flow from s to t. (It is sufficient to let the capacity be the sum of the capacities of the arcs entering node t, which in this case is 18.)

Let P1 denote the max flow problem and let P2 be the resulting min cost flow problem.

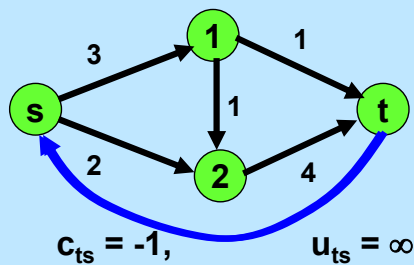
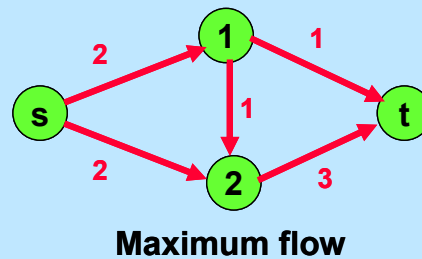
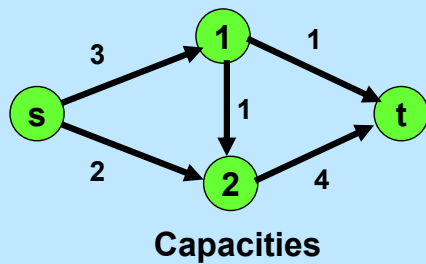
Theorem. The max s-t flow v^* in P1 is equal to the negative of the minimum cost z^* in P2.

To see that the theorem is true, let v^* be the max flow from s to t in P1. Then one can create a flow whose cost is $-v^*$ in P2 by copying the max flow and then by sending a flow of v^* in arc (t, s). This shows that $\text{negative } v^* \leq -z^*$.

Conversely, suppose consider the flow in P2 whose cost is z^* . If we delete (t, s), the flow is $-z^*$. We conclude that $v^* \geq -z^*$.

So, $v^* = -z^*$.

The Maximum Flow Problem



Min Cost Flow

17

The max flow problem is easily represented as a min cost flow problem. all one needs to do is to append the arc (t, s) to the max flow problem and give it a cost of -1 and a capacity that is larger than the largest possible flow from s to t . (It is sufficient to let the capacity be the sum of the capacities of the arcs entering node t , which in this case is 18.)

Let $P1$ denote the max flow problem and let $P2$ be the resulting min cost flow problem.

Theorem. The max s - t flow v^* in $P1$ is equal to the negative of the minimum cost z^* in $P2$.

To see that the theorem is true, let v^* be the max flow from s to t in $P1$. Then one can create a flow whose cost is $-v^*$ in $P2$ by copying the max flow and then by sending a flow of v^* in arc (t, s) . This shows that negative $v^* \leq -z^*$.

Conversely, suppose consider the flow in $P2$ whose cost is z^* . If we delete (t, s) , the flow is $-z^*$. We conclude that $v^* \geq -z^*$.

So, $v^* = -z^*$.

A Remarkable Theorem. (Integrality Theorem)

If the supplies, demands, and capacities of a minimum cost flow problem are all integral, then every basic feasible solution is integer valued. Therefore, the simplex method will provide an integer optimal solution.

Note: Most linear programs can have fractional solutions.

$$x + y = 1, \quad x - y = 0. \quad \text{Unique solution } (.5, .5)$$

18

Here is the result we stated earlier in this lecture. All min cost flow problem have integer solutions provided that the supplies, demands, and capacities are integer valued. Note that we do not need to assume that the costs are integer valued.

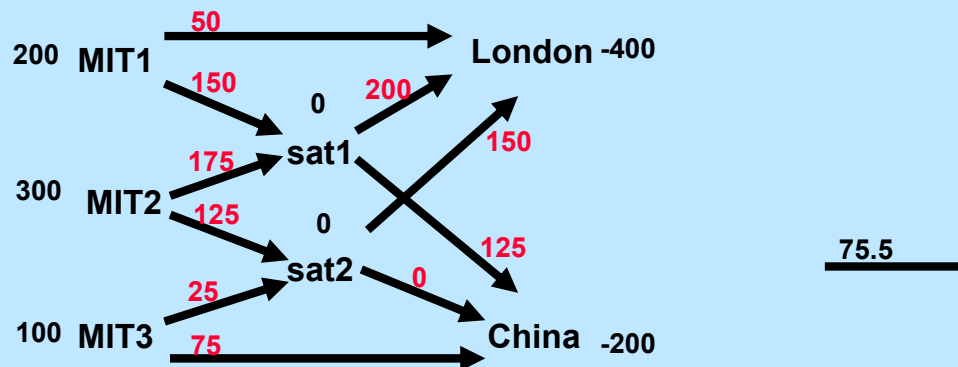
This result is more striking considering that the unique solution to the simple set of two equations given on the slide has a fractional solution.

But note the two exceptions. First of all, if supplies, demands, or capacities are fractional, it is possible that no optimal solution will be integer valued. We illustrate this on the next slide.

Also, even if all supplies, demands, and capacities are integer valued, it does not mean that all optimal flows are integer valued. It means that at least one optimal flow (assuming that an optimal flow exists) is integer valued. This is because all basic feasible flows will be integer valued.

We illustrate that there can also be an optimal fractional flow two slides from now.

The Optimal Flows (déjà vu)



Changing the supplies of the first two nodes to 200.5 and 299.5 will result in fractional flows.

Changing the capacity of the lowest arc to 75.5 will result in a fractional flow.

19

This is the problem we considered earlier, and the flows are integer valued. But if the supply at MIT is fractional, then no feasible flow will be integer valued.

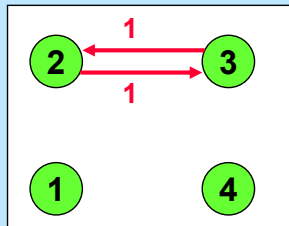
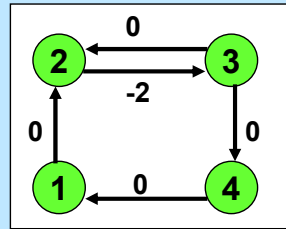
Also, if we change the capacity of the lowest arc in the graph from 75 to 75.5, then the optimal flow in that arc will be 75.5 (the constraint is tight in the optimal solution). In this case, the optimal flow will be non-integral.

**Basic Feasible Flows will be integer valued,
but not other flows.**

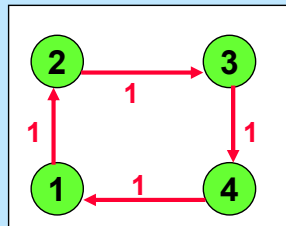
A network with arc costs.

Suppose $u_{ij} = 1$ for all (i, j)

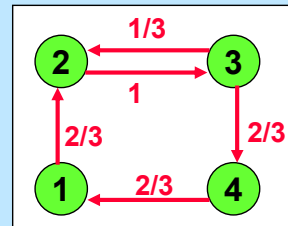
Suppose $b(i) = 0$ for all i .



Optimal Flow 1
cost = -2



Optimal Flow 2
cost = -2



Optimal Flow 3
cost = -2

In this case, the first two optimal flows are integer valued. They are both basic feasible solutions.

The third flow is obtained by taking $1/3$ of the first optimal flow and $2/3$ of the second optimal flow. Given how the flow was obtained, it is hardly surprising that it is non-integral.

Some Information on the Min Cost Flow Problem

- **Reference text: Network Flows: Theory, Algorithms, and Applications by Ahuja, Magnanti, and Orlin [1993]**
- **15.082J/6.885J: Network Optimization**
- **Polynomial time simplex algorithm (Orlin [1997])**
- **Basic feasible solutions of a minimum cost flow problem are integer valued (assuming that the data is integer valued)**
- **Very efficient solution techniques in practice**

21

In 1997, I solved a long standing open problem. I proved that there is a variant of the primal simplex algorithm for the min cost flow problem that runs in time polynomial in the number of nodes and arcs of the network. The number of pivots is at most $n^3m \log n$. Interestingly, the running time is only a slightly larger number.

It is

$n^3m \log^2 n$.

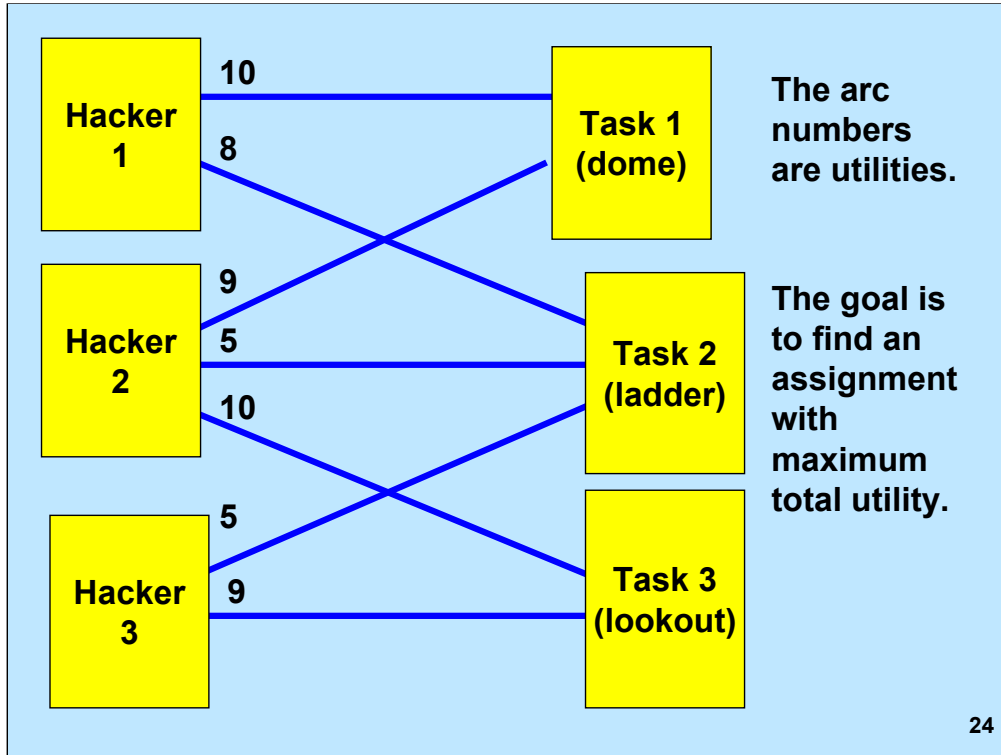
Since the simplex algorithm is so efficient in general, many researchers have asked whether it can be made polynomial time if one uses a clever rule. No one knows if there is an implementation of the simplex algorithm that runs in polynomial time for arbitrary linear programs.

Many researchers have conjectured that if one chooses the entering variable randomly among those with negative reduced cost, then the simplex algorithm will run in polynomial time on average. No one knows whether this conjecture is true.

Mental Break

An assignment problem

- **Three MIT hackers have decided to make the great dome look like R2D2, in honor of the hack from 5/17/99.**
- **Tasks.**
 - Putting the sheets on the great dome
 - Ladder holder
 - Lookout
 - Objective: find the optimal allocation of persons to tasks.
 - What is the optimal assignment of hackers to tasks.



I don't believe that these people would really be hackers. But it struck me as a funny idea when I made up this slide.

In any case, this problem does not seem like a flow problem. It is formulated as an "assignment problem."

An LP formulation

- Let x_{ij} = proportion of time that hacker i is assigned to task j .

x_{11}	x_{12}	x_{21}	x_{22}	x_{23}	x_{32}	x_{33}	RHS	
								Hacker 1
								Hacker 2
								Hacker 3
								Task 1
								Task 2
								Task 3

25

We will see that the assignment problem is a flow problem by first relaxing the problem and permitting fractional assignments of hackers to tasks. The fraction that hacker i is assigned to task j can be viewed as the proportion of time that hacker i spends on task j .

Once we permit fractions, we can express the fractional assignment problem as a linear program.

Transforming to a Min Cost Flow Problem

- **Multiply the last three constraints by -1.**
 - The supply of the hackers is 1
 - The “supply” of the tasks is -1.
- **Solutions are integer valued.**

26

The point is that the linear program is a network flow problem in disguise. Because of that, there will be an optimal integer solution.

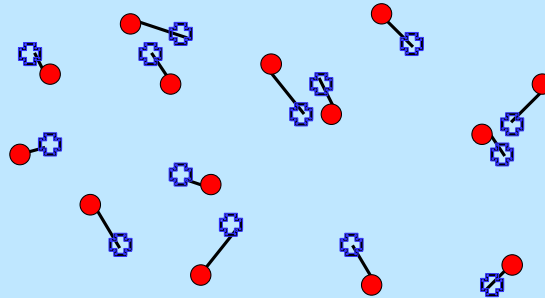
So, remarkably, permitting fractions and solving the assignment problem as an LP did not end up creating fractional solutions. The LP solver will give us integer valued solutions.

An Application of the Assignment Problem

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 may be the most efficient way of tracking items.

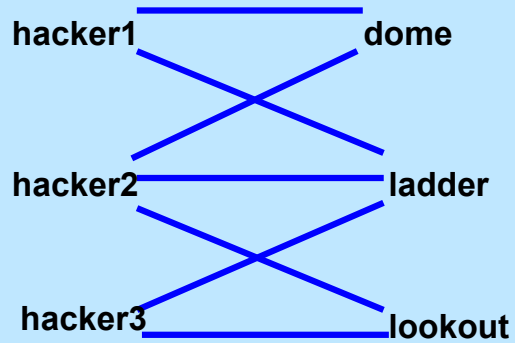
27

Here is an interesting application of the assignment problem to tracking. If one tries to track 100 objects in space, one takes a series of snapshots of the objects. But each object looks the same in each snapshot since it looks like a “pixel.” The only way to distinguish objects is by their location on the radar screen.

A natural method for matching pixels from one snapshot to the next is to assign each pixel in the first snapshot to the closest pixel in the second snapshot. But this often results in two pixels of the first snapshot being assigned to the same pixel in the next snapshot.

An alternative and more effective approach is to concurrently match all pixels by solving an assignment problem. This too is imperfect, but it leads to many fewer errors than the nearest neighbor assignment. The algorithms correct errors by looking at subsequent snapshots, but error correcting is time consuming. So, any algorithm that leads to fewer errors in the first place can be much more effective in practice.

A special case: maximize the number of persons who can get assigned.



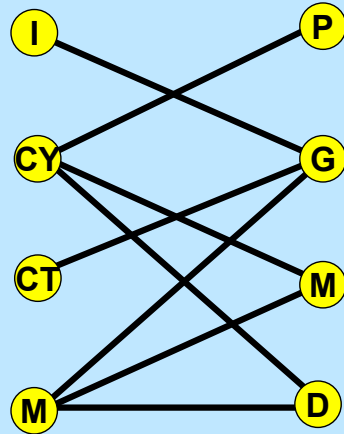
The LP has the same formulation as before except that each cost coefficient is 1.

28

If we maximize the number of persons assigned, we have the max matching problem, which was discussed in the previous lecture. Recall that our previous example of the max matching problem was to match four female stars of Grey's Anatomy to four of its male stars.

You will recall that the duality result for the max flow problem could be stated as "max flow = min cut". There is also a duality result for the maximum matching problem. We will show the duality result for the matching problem next. One of the reasons that I like this duality result is that it is nice to display visually.

Transforming Matching to a 0-1 Matrix Problem

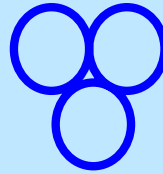


	P	G	M	D
I	0	1	0	0
CY	1	0	1	1
CT	0	1	0	0
M	0	1	1	1

The adjacency matrix

Matching and its dual problem

	P	G	M	D
I	0	1	0	0
CY	1	0	1	1
CT	0	1	0	0
M	0	1	1	1



Matching Problem

What is the maximum number of 1's, no two of which are on the same line (row or column)?

30

The maximum number of 1's no two of which are on a line is 3.

The 1's corresponds to the matching in the previous lecture.

Matching and its dual problem

	P	G	M	D
I	0	1	0	0
CY	1	0	1	1
CT	0	1	0	0
M	0	1	1	1

A Duality Theorem.

The minimum number of lines to cover all of the 1's of a matrix is equal to the maximum number of 1's no two of which are on a line.

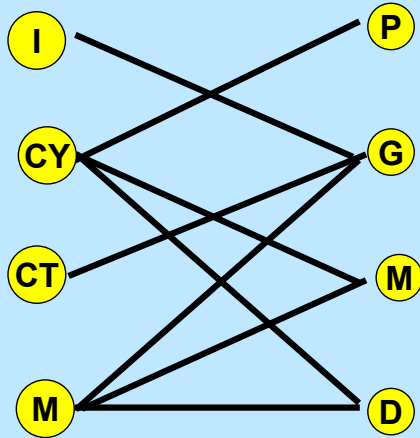
Line cover problem.

What is the minimum number of lines (rows or columns) that contain all of the 1's?

31

A row corresponds to one of the women. A column corresponds to one of the men. The minimum number of lines containing all of the ones corresponds to the minimum number of people such that every feasible pair includes one of these people.

Deleting three people eliminates all relationships.



Therefore, at most three couples can be matched.

If we eliminate one person from the problem, this can reduce the maximum size of a matching by 1. If we eliminate three people from the problem, this could possibly reduce the maximum size of a matching by 3. In this case, we eliminate Christina, Meredith, and George, and there is no feasible pair left.

Baseball Elimination 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

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

<http://riot.ieor.berkeley.edu/~baseball/>

33

In professional sports, one likes to know when a team is guaranteed to win regardless of the outcomes of the remaining games and when a team is guaranteed to be eliminated from winning regardless of the outcome of the remaining games.

It is relatively easy to determine whether a team T is guaranteed to win regardless of subsequent games. All one needs to do is to hypothesize that the team T loses all remaining games and its closest competitor wins all remaining games. If T wins under these totally adverse circumstances, then T is guaranteed to win under all circumstances.

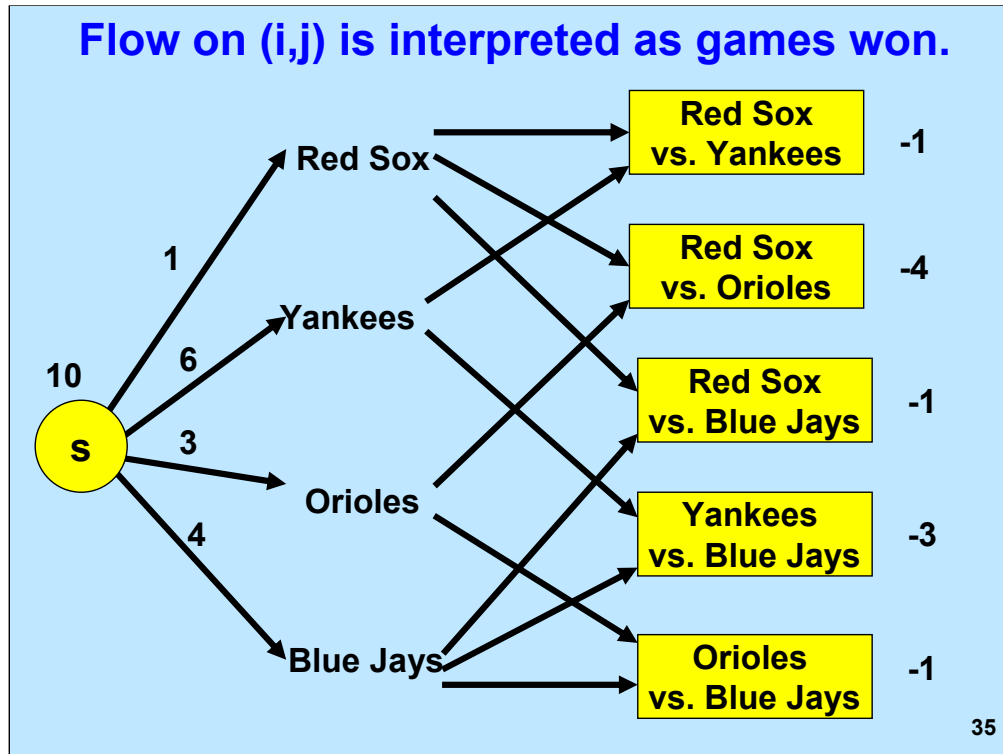
Surprisingly, it is not so easy to determine if a team T is guaranteed to be eliminated from winning. One starts in the same manner and presumes that T wins all of their games. But now suppose that T has 2 or more close competitors. One cannot assume that each of these competitors loses all of their games because some of the games may be against each other. So, in order to determine whether T has a chance of winning the division, one needs to determine how many games each of its closest competitors wins against each other. One can determine this as a max flow problem. But the transformation into a max flow problem is pretty tricky. It's much easier to see that it is a special case of a min cost flow problem. We will show this transformation next.

First, assume Tampa wins all its games.

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

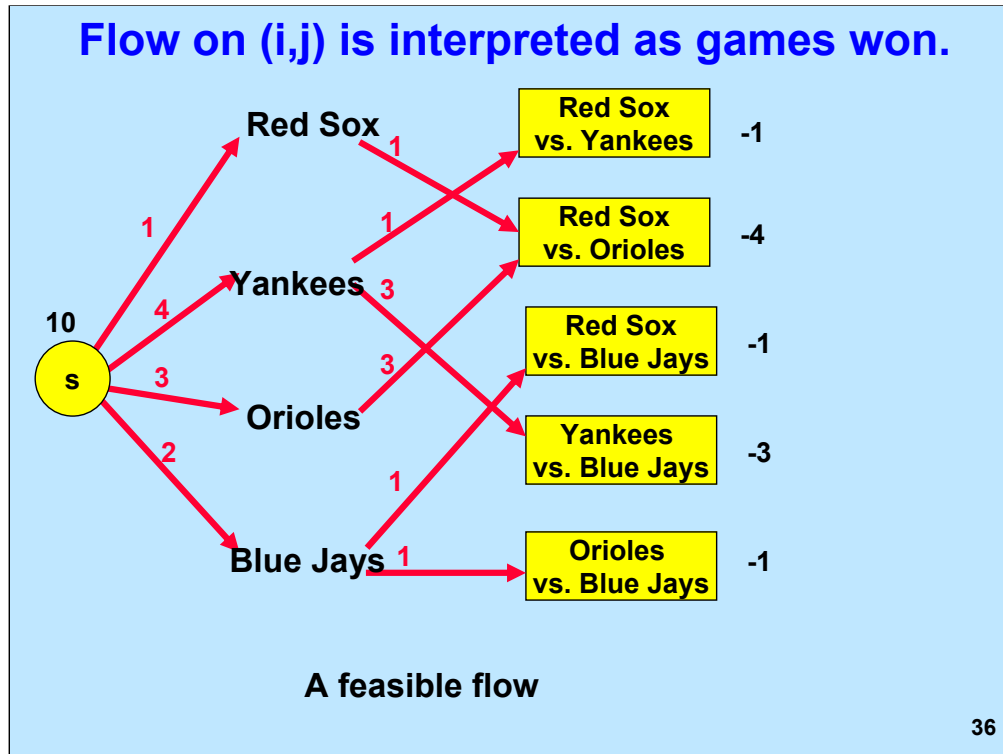
	Bos	NY	Balt	Tor	Tamp
Bos	--	1	4	1	0
NY	1	--	0	3	0
Balt	4	0	--	1	0
Tor	1	3	1	0	0
Tamp	0	0	0	0	0

Now check to see if the remaining games can be played so that no team wins more than 83 games.



The capacities on the arcs from the source represent the maximum number of games that each of the teams can win while still having Tampa as the division winner. The supplies of each of the nodes on the right represent the number of games that must be played between pairs of teams.

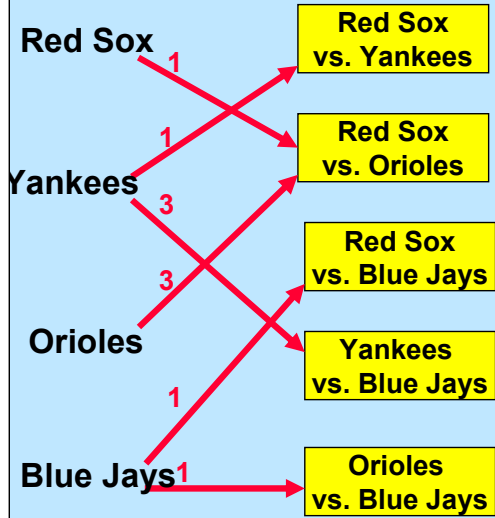
The flows from the team nodes to the yellow boxes on the left represent the number of wins for a team in matchups. For example, the flow from the Red Sox node to the “Red Sox vs Orioles” node is the number of games that the Red Sox beat the Orioles. This is a decision variable because we are trying to determine whether we can allocate victories in all remaining games so that Tampa wins the division.



The capacities on the arcs from the source represent the maximum number of games that each of the teams can win while still having Tampa as the division winner. The supplies of each of the nodes on the right represent the number of games that must be played between pairs of teams.

The flows from the team nodes to the yellow boxes on the left represent the number of wins for a team in matchups. For example, the flow from the Red Sox node to the “Red Sox vs Orioles” node is the number of games that the Red Sox beat the Orioles. This is a decision variable because we are trying to determine whether we can allocate victories in all remaining games so that Tampa wins the division.

Tampa Can Tie for the Leader

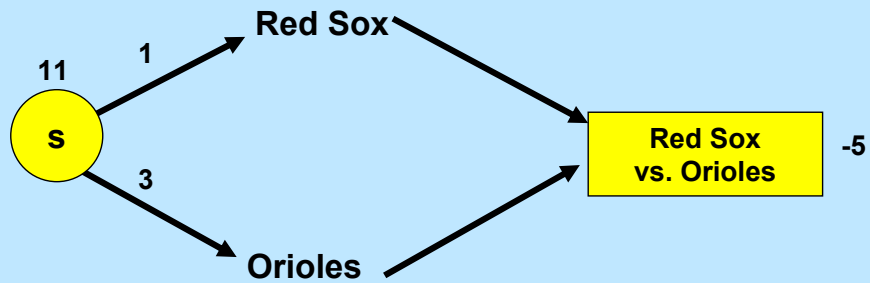


	Bos	NY	Balt	Tor	Tamp
Bos	--	0	1	0	0
NY	1	--	0	3	0
Balt	3	0	--	0	0
Tor	1	0	1	0	0
Tamp	0	0	0	0	0

	Games Won	Games Left
Bos	83	0
NY	81	0
Balt	83	0
Tor	81	0
Tamp	83	0

On the baseball elimination problem

- Note: if Baltimore and Boston has had five games left instead of four, then Tampa could not have won. The flow problem would have been infeasible.



38

Boston is permitted to win at most one more game. Otherwise, they would have more wins than Tampa. The Orioles are permitted to win at most three games. This would not be possible if the Red Sox and the Orioles had five more games to play.

And now, it's time for

Summary slide

- **Min Cost flow Problem**
- **Integrality Property**
- **Assignment Problem**
- **Matching Problem and Line Cover Problem**
- **Baseball Elimination Problem**
- **Next Lecture: Network Simplex**