COMP9334: Capacity Planning of Computer Systems and Networks

Optimisation – Part 2

A/Prof. Chun Tung Chou

CSE, UNSW

Last week

- Linear programming (LP)
  - Real values for decision variables, linear in objective function, linear in constraints
  - Large LP problems can be solved routinely

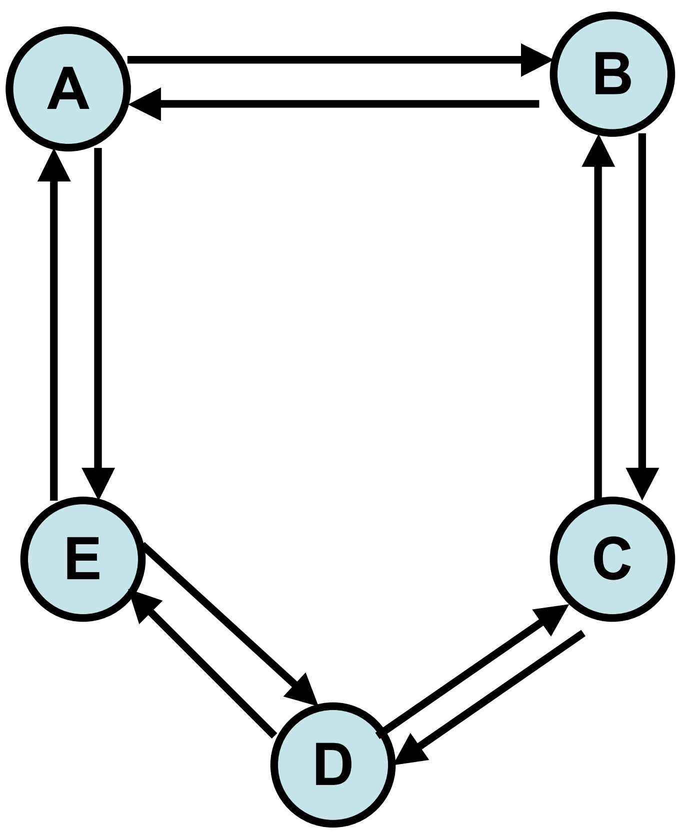
- Integer programming (IP)
  - Some decision variables can only take integer values
  - Some decision variables can only take binary values, e.g. for making yes-or-no decisions
  - IP problems can be solved using branch and bound in principle
  - Computation complexity is generally exponential except for unimodular problems

This week

- Applications of integer programming for network flow problems
  - Traffic Engineering
  - Dimensioning problem
  - Topology design
- Power of binary variables
  - We have seen using binary variables (either 0 or 1) to capture yes-or-no type of logical decisions
  - Binary variables can be used to capture many other requirements

Traffic Engineering Example (1)

An ISP owns the following network which connects
5 cities A, B, C, D and E.
Capacity of each link is 1000 Mbps



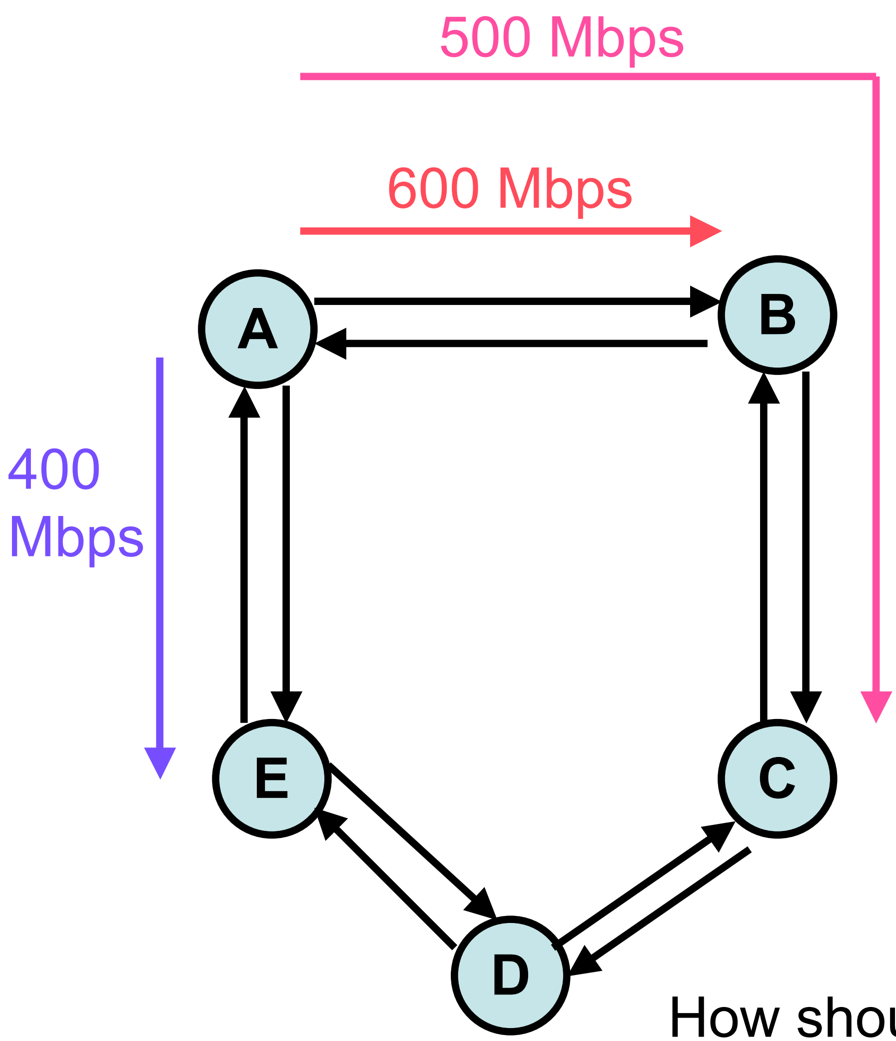
The traffic demands between cities are:

A to B: 600 Mbps
A to E: 400 Mbps
A to C: 500 Mbps

Question: How should we route the traffic so that the links are at most 80% utilised and we use the minimum amount of resources?

Traffic Engineering Example (2)

Capacity of each link is 1000 Mbps



The traffic demands between cities are:
A to B: 600 Mbps
A to E: 400 Mbps
A to C: 500 Mbps

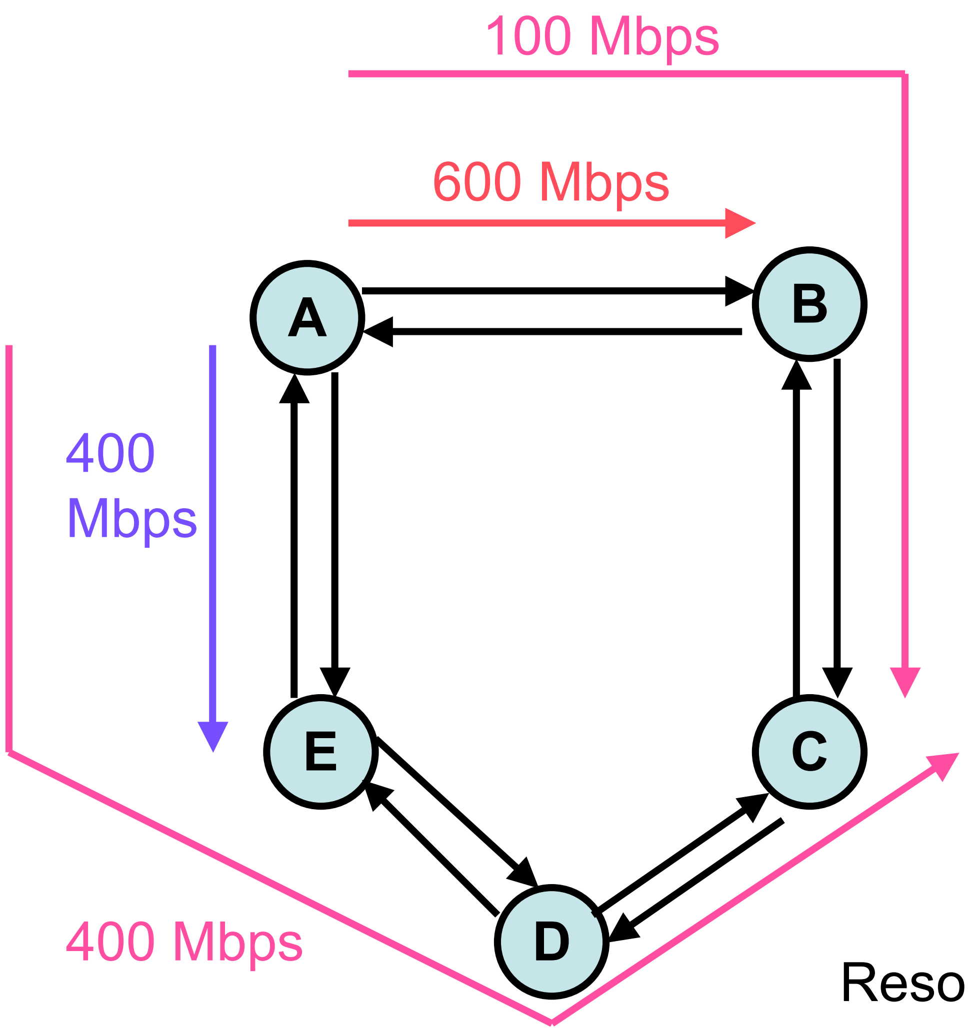
Let us assume that the traffic demand will take the shortest path between its end points

But 1100 Mbps in link A-B! The link is over-utilised!

How should you route the traffic?

Traffic Engineering Example (3)

Capacity of each link is 1000 Mbps



- The traffic demands between cities are:

A to B: 600 Mbps
A to E: 400 Mbps
A to C: 500 Mbps

- Traffic in links

A-B = 700 Mbps
B-C = 100 Mbps
A-E = 800 Mbps
E-D = 400 Mbps
D-C = 400 Mbps

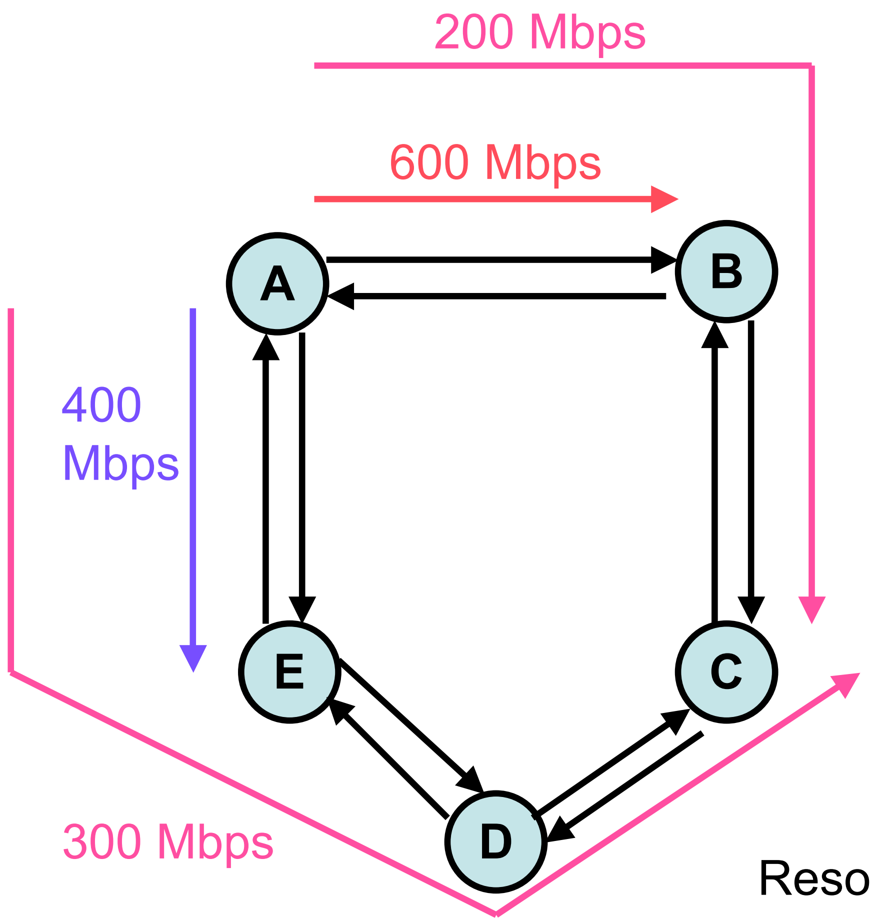
+

- Link A-E is 80% utilised. Others are less utilised.

Resources used = 2400Mbps

Traffic Engineering Example (4)

Capacity of each link is 1000 Mbps



- The traffic demands between cities are:
  A to B: 600 Mbps
  A to E: 400 Mbps
  A to C: 500 Mbps

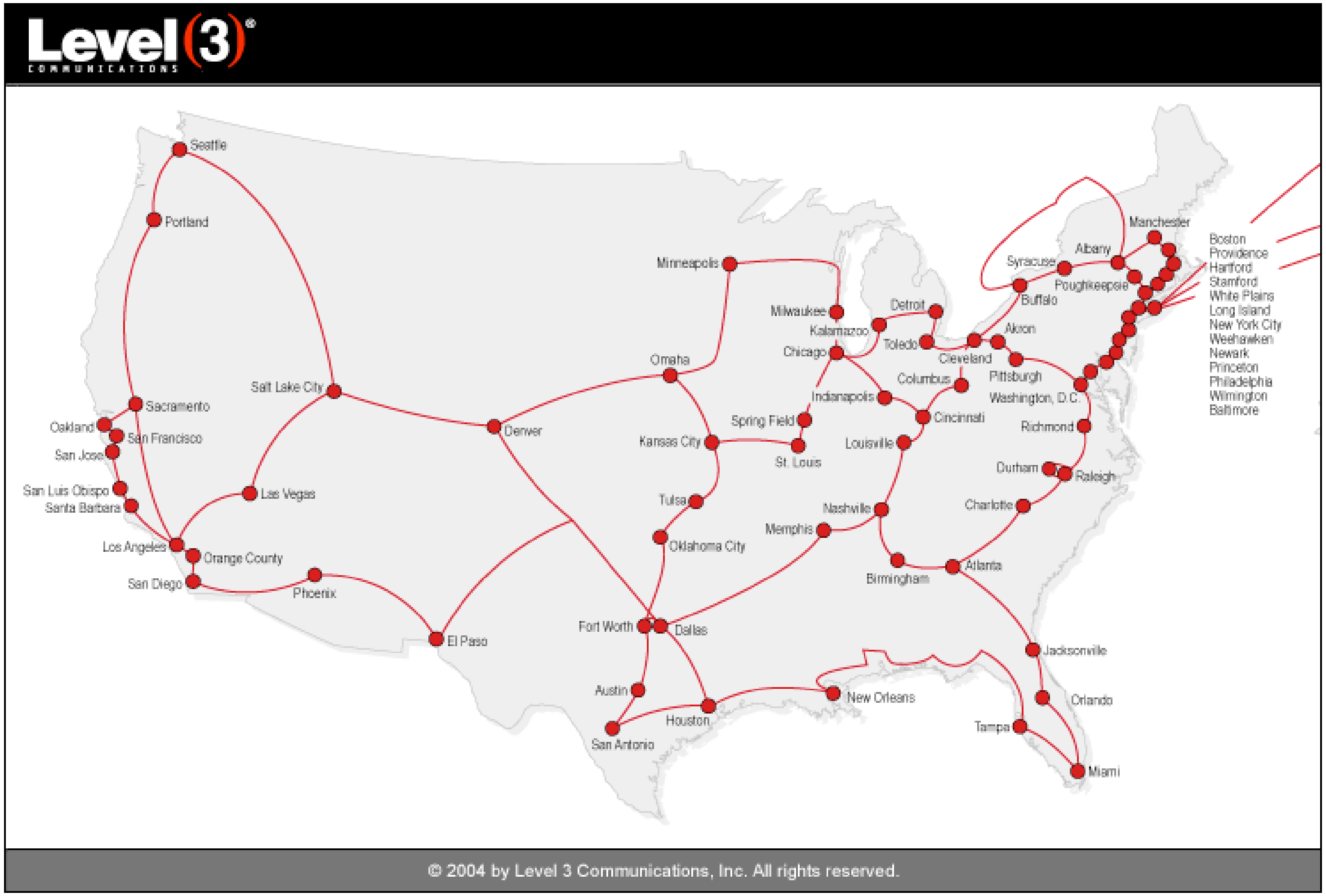
- Traffic in links
  A-B = 800 Mbps
  B-C = 200 Mbps
  A-E = 700 Mbps
  E-D = 300 Mbps
  D-C = 300 Mbps
  +

- Link A-B is 80% utilised. Others are less utilised.

Resources used = 2300Mbps

Traffic Engineering

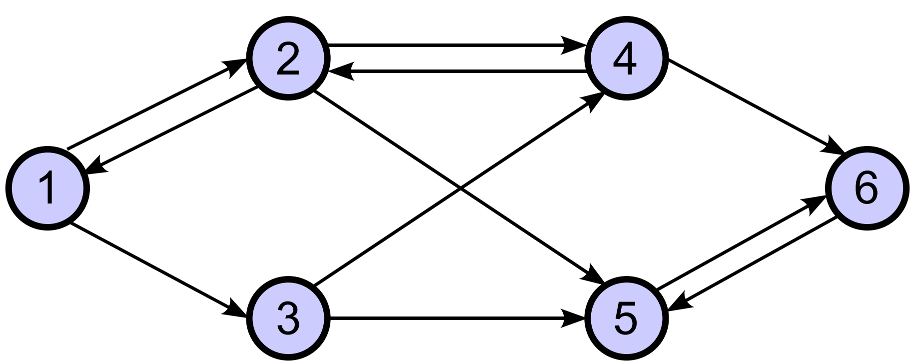
- General traffic engineering problem:
  - Given:
    - A network (i.e. nodes, links and their capacites)
    - The traffic demand between each pair of nodes.
  - Find: how to route the traffic to best utilise the resource
- The traffic engineering example earlier was simple, but for a commercial carrier (Next slide shows the network map of a commercial carrier.), it's no longer so.
  - Traffic engineering problems can be solved systematically using integer programming
  - These problems are generally known as network flow problems. *Note: flow is synonymous with traffic demand between a pair of nodes.*
  - We will start with the simplest network flow problem, finding the shortest path for one flow.


Level(3) COMMUNICATIONS
Seattle
Portland
Sacramento
Oakland
San Francisco
San Jose
San Luis Obispo
Santa Barbara
Los Angeles
Orange County
San Diego
Phoenix
Las Vegas
Salt Lake City
Denver
El Paso
Minneapolis
Omaha
Kansas City
Tulsa
Oklahoma City
Fort Worth
Dallas
Austin
San Antonio
Houston
New Orleans
Milwaukee
Kalamazoo
Chicago
Spring Field
St. Louis
Indianapolis
Memphis
Nashville
Louisville
Detroit
Toledo
Cleveland
Akron
Columbus
Cincinnati
Pittsburgh
Washington, D.C.
Richmond
Durham
Raleigh
Charlotte
Birmingham
Atlanta
Jacksonville
Orlando
Tampa
Miami
Syracuse
Buffalo
Albany
Poughkeepsie
Manchester
Boston
Providence
Hartford
Stamford
White Plains
Long Island
New York City
Weehawken
Newark
Princeton
Philadelphia
Wilmington
Baltimore
© 2004 by Level 3 Communications, Inc. All rights reserved.

Network flow problems

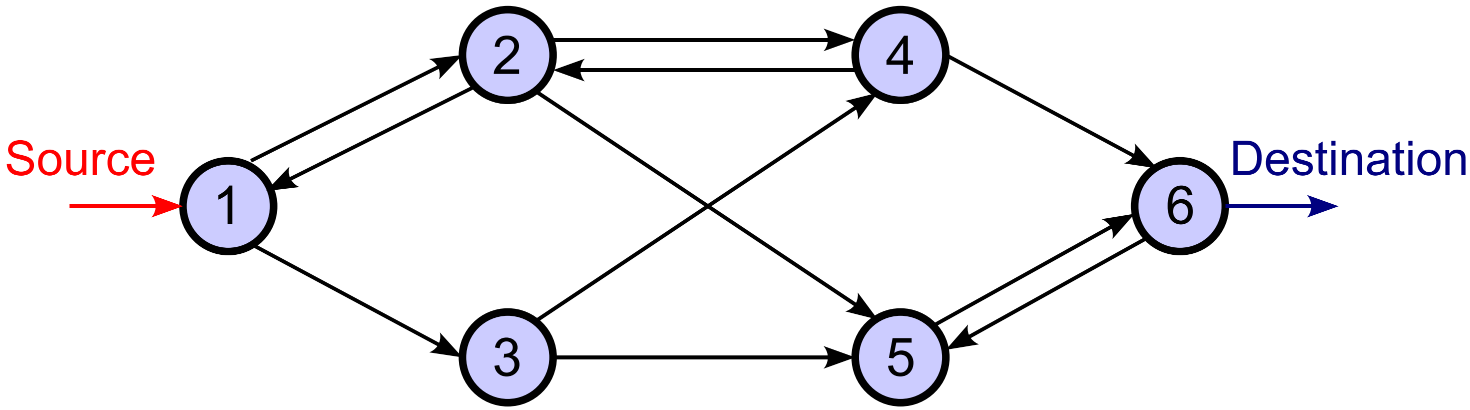
- Network flow problems are important applications of integer programming
  - Move some entity from one point to another in the network
  - Given alternative ways, find the most efficient one, e.g. minimum cost, maximum profit, etc.

- Network is represented as a directed graph $G = (N, E)$
  - N = the set of nodes, e.g. $N = \{1, 2, 3, 4, 5, 6\}$
  - E = the set of directed edges, e.g. $E = \{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 2, 1\rangle, \ldots\}$



Finding the shortest path

- Aim: Find the shortest path from the source node to the destination node of a flow



- Cost is generally assumed to be additive, i.e. cost of a path = sum of the cost of using each edge in the path
  - E.g. cost of using edges $\langle 1, 2\rangle$, $\langle 2, 4\rangle$ and $\langle 4, 6\rangle$ = cost of edge $\langle 1, 2\rangle$ + cost of edge $\langle 2, 4\rangle$ + cost of edge $\langle 4, 6\rangle$

Shortest path problem (SPP)

- Given
  - A directed graph $G = (N, E)$
  - A flow of size 1 enters at node s (source) and leaves at node d (destination)
  - It costs $c_{i,j}$ for using directed edge $\langle i, j \rangle$
- Find which directed edges the flow should use in order that
  - The total cost is minimized
  - The entire flow must use only one path
- Logical decision: Should I use a directed edge or not?

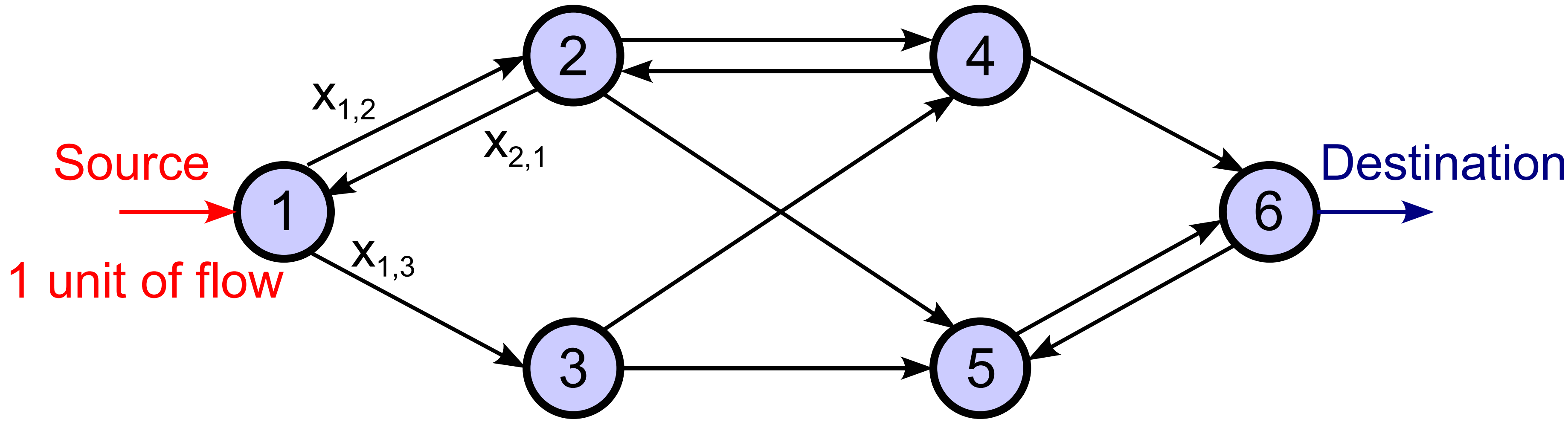
Formulating SPP

- Decision variables

$$x_{i,j} = \begin{cases} 1 & \text{if directed edge } \langle i,j \rangle \text{ in } E \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

- We assume 1 unit of flow from the source to the destination
- An important part of the formulation is to make sure the directed edges selected actually form a connected path from the source to the destination
  - This is by adding the conservation of flow constraints

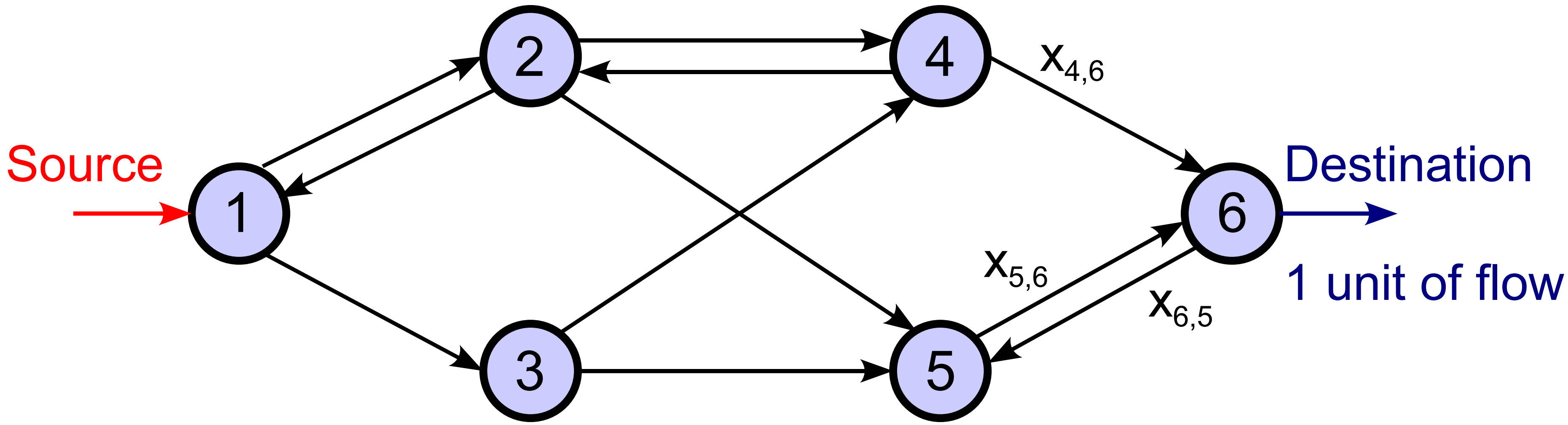
Conservation of flow: Source node



- What goes in = What goes out
  - Flow going into node 1 from external = 1
  - Flow going into node 1 from neighboring nodes = $x_{2,1}$
    - Second index is “1”
  - Flow going from node 1 to neighboring nodes = $x_{1,2} + x_{1,3}$
    - First index is “1”
  - Therefore, we have

$$1 + x_{2,1} = x_{1,2} + x_{1,3}$$

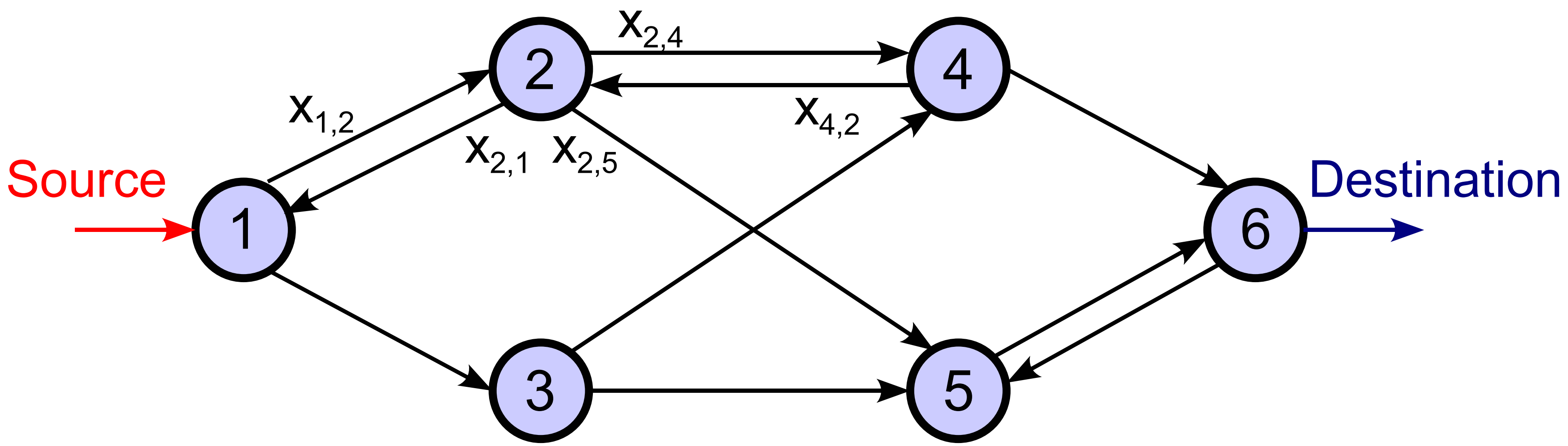
Conservation of flow: Destination node



- What goes in = What goes out
  - Flow going into node 6 from neighboring nodes = $x_{4,6} + x_{5,6}$
    - Second index is "6"
  - Flow going from node 6 to neighboring nodes = $x_{6,5}$
    - First index is "6"
  - Flow going from node 6 to external = 1
  - Therefore, we have

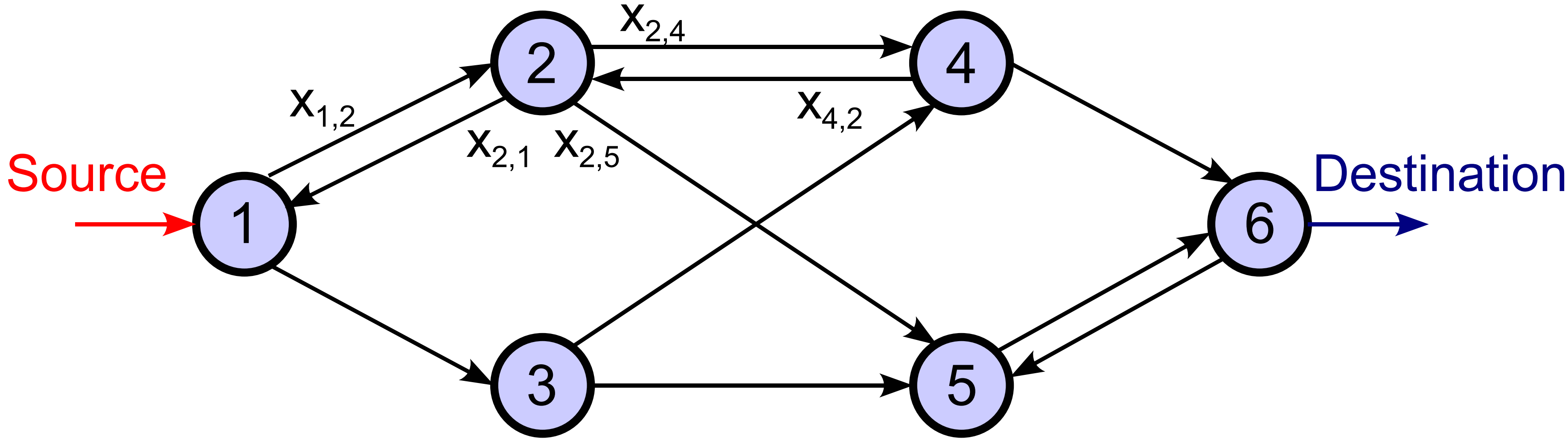
$$x_{4,6} + x_{5,6} = x_{6,5} + 1$$

Conservation of flow: Other nodes



- Exercise: Work out the conservation of flow for Node 2

Conservation of flow: Other nodes



- E.g. flow conservation at node 2: What goes in = What goes out
  - Flow going into node 2 from neighboring nodes = $x_{1,2} + x_{4,2}$
    - Second index is “2”
  - Flow going from node 2 to neighboring nodes = $x_{2,1} + x_{2,4} + x_{2,5}$
    - First index is “2”
  - Therefore, we have

$$x_{1,2} + x_{4,2} = x_{2,1} + x_{2,4} + x_{2,5}$$

Conservation of flow constraints

- In our example network, the source is node 1, so the constraint is

$$1 + x_{2,1} = x_{1,2} + x_{1,3}$$

  - This can be rewritten as

$$\sum_{j:\langle 1,j\rangle \in E} x_{1,j} - \sum_{j:\langle j,1\rangle \in E} x_{j,1} = 1$$

- The destination is node 6, so the constraint is

$$x_{4,6} + x_{5,6} = x_{6,5} + 1$$

  - This can be rewritten as

$$\sum_{j:\langle 6,j\rangle \in E} x_{6,j} - \sum_{j:\langle j,6\rangle \in E} x_{j,6} = -1$$

Conservation of flow constraints (cont.)

- For all other nodes (neither a source or a destination), e.g. node 2, the constraint is

$$x_{1,2} + x_{4,2} = x_{2,1} + x_{2,4} + x_{2,5}$$

  - This can be rewritten as

$$\sum_{j:\langle 2,j\rangle \in E} x_{2,j} - \sum_{j:\langle j,2\rangle \in E} x_{j,2} = 0$$

- The flow conservation constraints can be written in a compact form

$$\sum_{j:\langle i,j\rangle \in E} x_{i,j} - \sum_{j:\langle j,i\rangle \in E} x_{j,i} = 0, \;\; i \in N - \{s, d\}$$

IP formulation for SPP

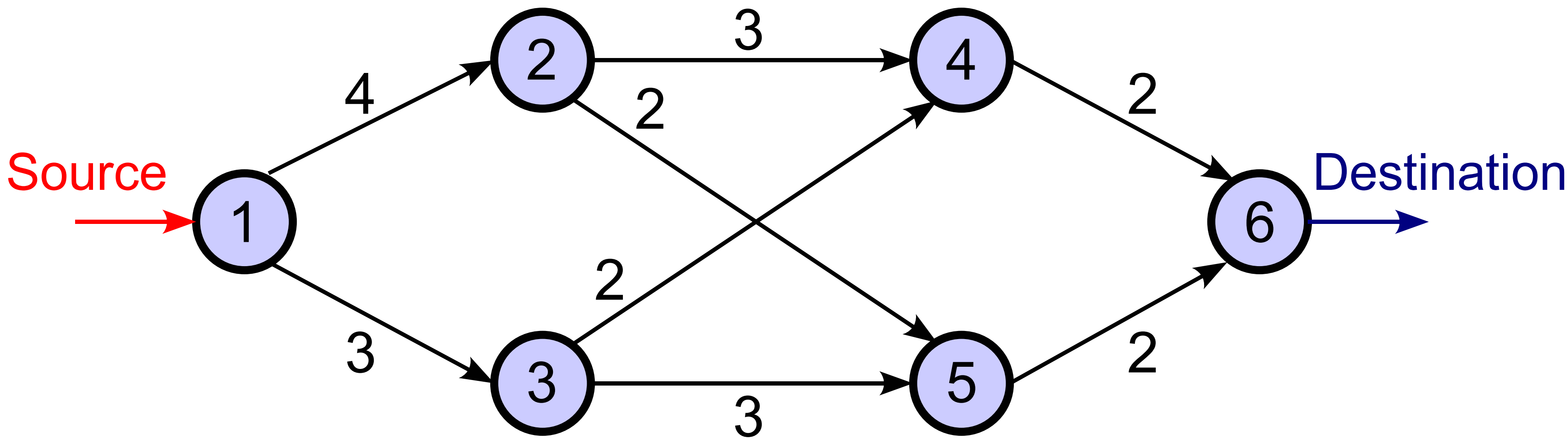
- SPP can be formulated as

$$\min \sum_{\langle i,j\rangle \in E} c_{i,j} x_{i,j}$$

subject to

$$\begin{aligned}
\sum_{j:\langle i,j\rangle \in E} x_{i,j} - \sum_{j:\langle j,i\rangle \in E} x_{j,i} &= \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in N - \{s,d\} \\ -1 & \text{if } i = d \end{cases} \\
x_{i,j} &\in \{0,1\} \text{ for all } \langle i,j\rangle \in E
\end{aligned}$$

SPP example



- We will use AMPL/CPLEX for solving SPP in this example network
- Note:
  - It is far more efficient to use Dijkstra's algorithm for solving SPP
  - The reason of using integer programming here is for illustration only
- The files are `shortest.dat`, `shortest.mod` and `shortest_batch`

Introducing non-unit flow and link capacity (1)

(Note: A dot point preceded by ★ indicates that it is different from the setting of the shortest path problem.)

- Given
  - A directed graph (N, E)
  - ★ A flow of size f with source node s and destination node d
  - It costs c_{ij} (per unit flow) for the flow to use directed edge (i, j)
  - ★ The capacity of the directed edge (i, j) is b_{ij}
- Find which directed edges the flow should use in order that
  - The total cost is minimised
  - The entire flow must use only one path
  - ★ The flow on any directed edge does not exceed its capacity

Introducing non-unit flow and link capacity (2)

- Decision variables are the same as before

$$x_{ij} = \begin{cases} 1 & \text{if directed edge } (i,j) \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

- The amount of flow on directed edge (i,j) will be fx_{ij}

Introducing non-unit flow and link capacity (3)

The problem formulation is

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

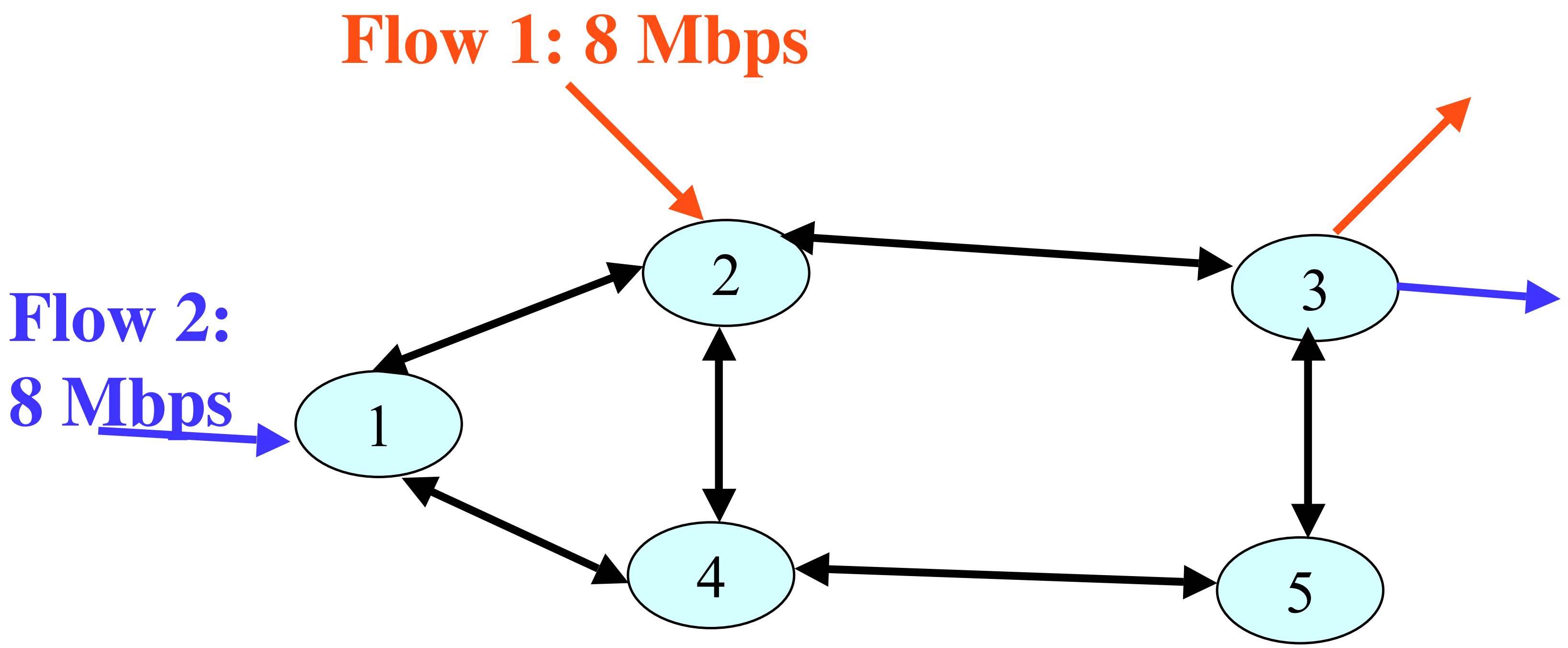
subject to

$$\begin{aligned}
\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} &= \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in N - \{s,d\} \\ -1 & \text{if } i = d \end{cases} \\
f x_{ij} &\leq b_{ij} \text{ for all } (i,j) \in E \qquad (***) \\
x_{ij} &\in \{0,1\} \text{ for all } (i,j) \in E
\end{aligned}$$

Note: $(***)$ — this constraint ensures that only links with sufficient capacity may be chosen to carry the flow.
Solution: Eliminate edges with insufficient capacity, then Dijkstra.

Multiple flows (1)

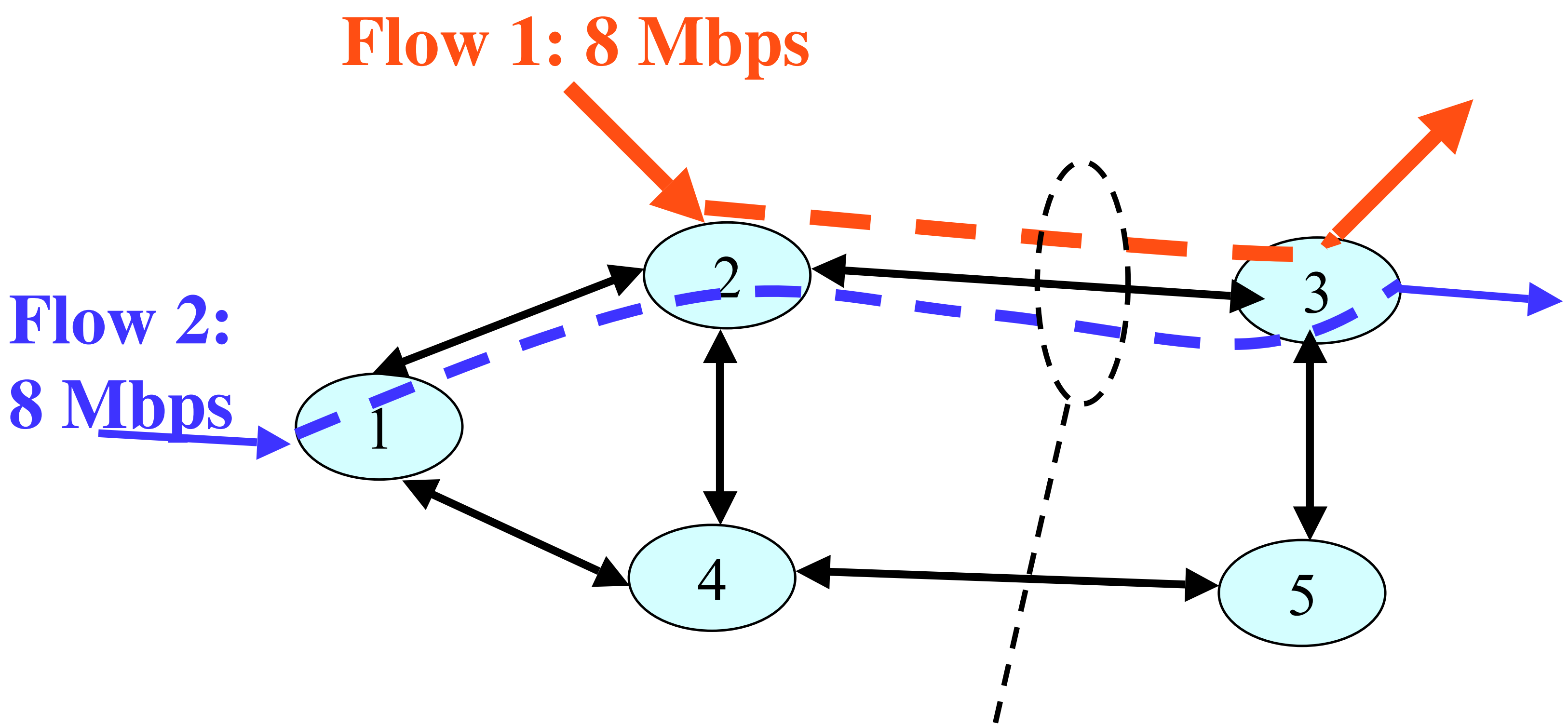
Given: Each link has capacity of 10 Mbps



Assuming cost for each link is 1.
What if both flows use the shortest path?

Multiple flows (2)

Given: Each link has capacity of 10 Mbps



16 Mbps of flows on 10Mbps
⇒ Utilisation > 1, High packet delay and loss

Traffic engineering problem

- Given
  - A directed graph (N, E)
  - ★ m flows (indexed by $k = 1, 2, ...m$)
  - ★ Flow k has size f_k, source node s_k, destination d_k
  - ★ It costs c_{ij} for a unit of flow to use directed edge (i, j)
  - The capacity of directed edge is b_{ij}
- Find the directed edges that **each flow** should use in order that
  - The total cost is minimised
  - The entire flow must use only one path
  - ★ The *total flow* on a directed edge does not exceed its capacity

Digression: Integral versus continuous traffic engineering

- There are two versions of traffic engineering problem
- The *integral* version where each flow must use only one path, i.e. all packets in a flow must use the same path
  - In order to ensure that the packets use a certain path, you can use source routing (available in IP version 6) or MPLS (multi-protocol label switching - covered in COMP9332)
- The *continuous* version where each flow may use multiple paths, e.g. the one described on pages 5 – 6 of this lecture.
  - In order to split the flow, a classifier will be required at the router to send packets on different paths
- We will see how we can formulate the integral traffic engineering problem

Traffic engineering IP formulation (1)

- Decision variables: m sets of decision variables, one for *each flow*

$$x_{ijk} = \begin{cases} 1 & \text{if flow } k \text{ uses directed edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

- The flow on directed edge (i,j) will be

$$\sum_{k=1}^{m} f_k x_{ijk}$$

- Ex: $m = 3$. Flows 1 and 3 use edge (1,2) but flow 2 doesn't.

  Total flow in edge (1,2) = $f_1 + f_3$

  $\sum_{k=1}^{m} f_k x_{12k} = f_1 \times \underbrace{x_{121}}_{=1} + f_2 \times \underbrace{x_{122}}_{=0} + f_3 \times \underbrace{x_{123}}_{=1} = f_1 + f_3$

Traffic engineering IP formulation (2)

The problem formulation is

$$\min \sum_{(i,j)\in E} \sum_{k=1}^{m} c_{ij} f_k x_{ijk}$$

subject to

$$\sum_{j:(i,j)\in E} x_{ijk} - \sum_{j:(j,i)\in E} x_{jik} = \begin{cases} 1 & \text{if } i = s_k \\ 0 & \text{if } i \in N - \{s_k, d_k\} \\ -1 & \text{if } i = d_k \end{cases} \quad k = 1, ..., m \ (*)$$

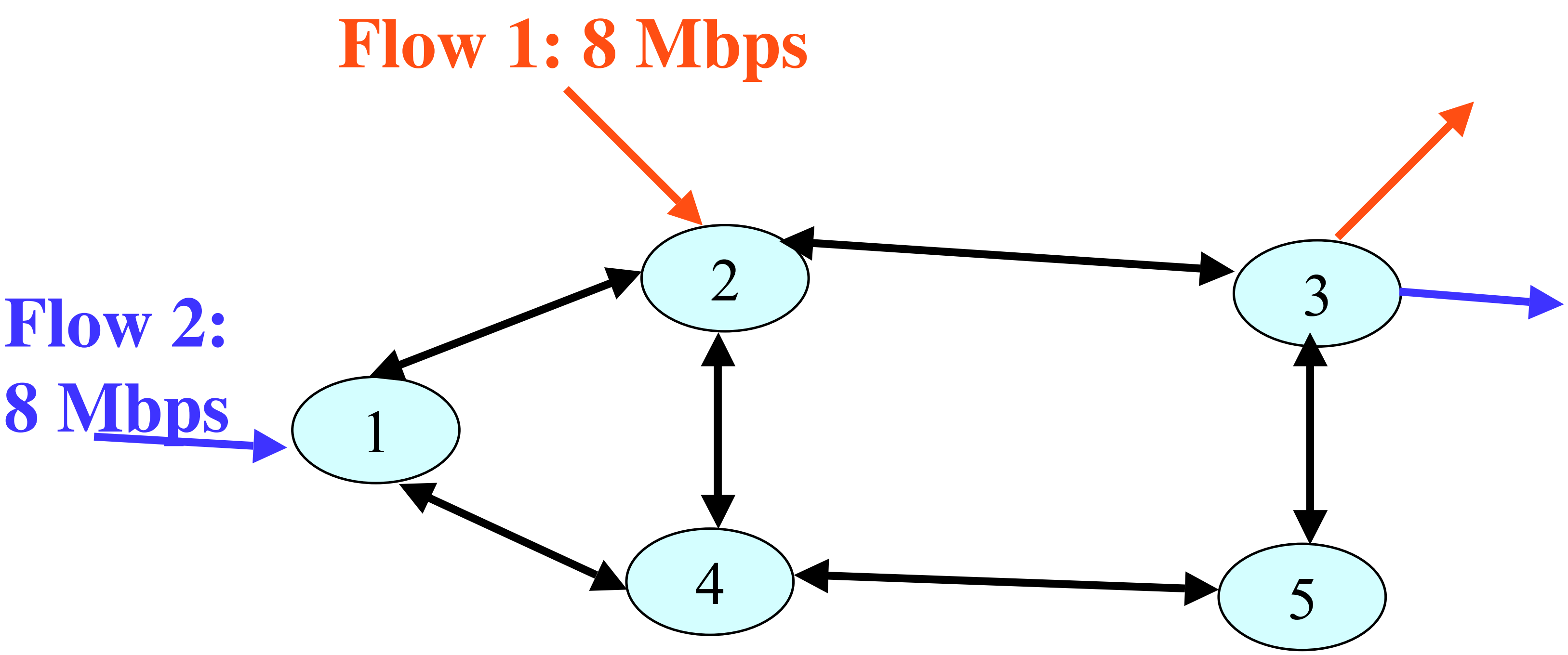
$$\sum_{k=1}^{m} f_k x_{ijk} \leq b_{ij} \quad \text{for all } (i,j) \in E \ \ (**)$$

$$x_{ijk} \in \{0,1\} \text{ for all } (i,j) \in E,\ k = 1, ..., m$$

$(*)$ – One set of flow balance constraint per flow. Enforces flow k is from s_k to d_k
$(**)$ – Total flow on a link does not exceed its capacity.

AMPL Example

Given: Each link has capacity of 10 Mbps



Assuming cost for each link is 1.
What if both flows use the shortest path?

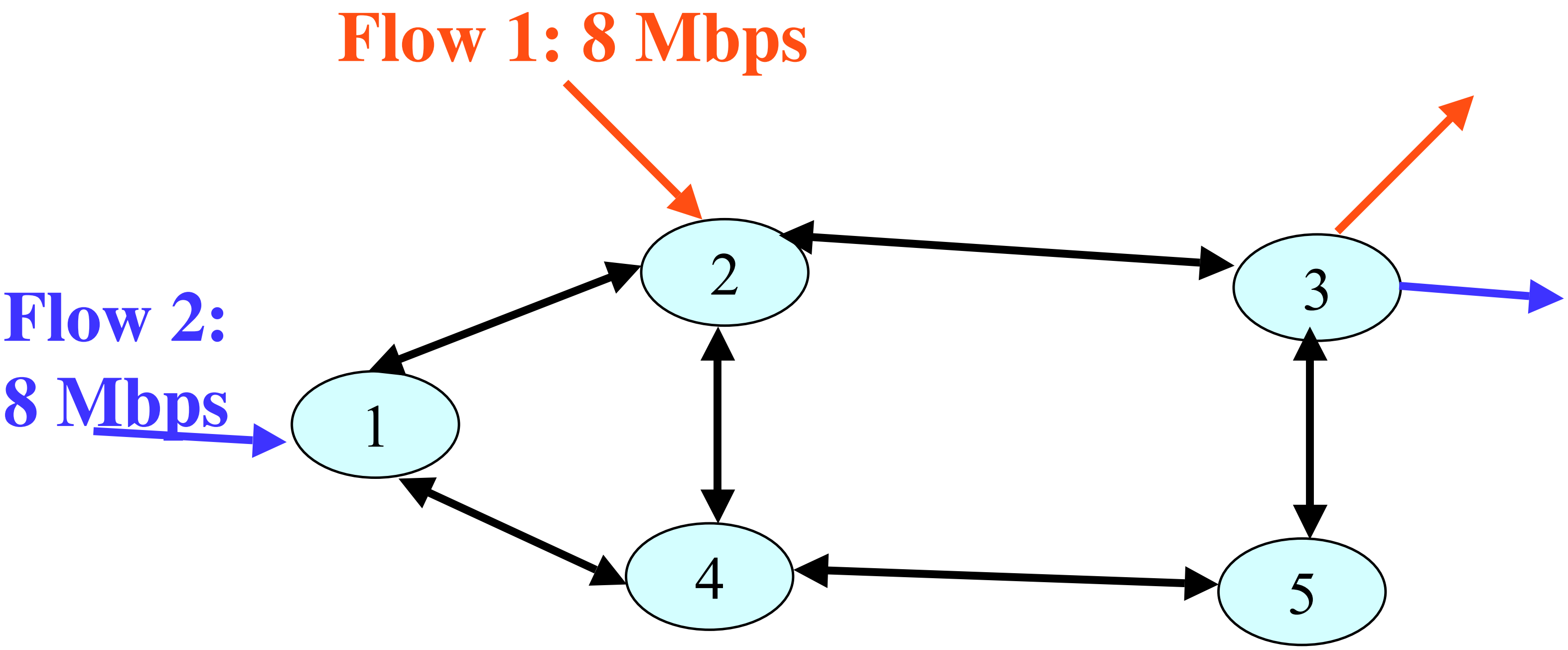
Files are `mcf1.dat`, `mcf1.mod` and `mcf1_batch`,

Traffic engineering problem

- Also known as
  - The multi-commodity flow problem in operations research
  - Flow assignment problem
- Essence: assign a flow to a path so that performance is met
  - i.e. routing problem
- Many variations possible
  - Constraint on the path delay / number of hops
  - Constraint on packet loss rate

Network design problem(1)

In flow assignment, we assume the network topology and link capacities are given.



Why should we choose capacity 10Mbps? Why not 100Mbps? Why should we choose to have a link between (2,3) but not (2,5)?

Network design problem(2)

- Given
  - A set of nodes N
  - m flows of size f_k, source s_k, destination d_k
  - Maximum network building cost
- Design options in network design problems
  - Topology: Which directed links to include
  - Capacity of the link
  - How the flows are routed?
- There are a few different network design problems

Different network design problems

- Flow Assignment Problem
  - Given: flows, topology, capacity
  - Find: paths for the flows
- Capacity and Flow Assignment Problem
  - Given: flows, topology, network cost
  - Find: paths for the flows, capacity
- Topology, Capacity and Flow Assignment Problem
  - Given: flows, network cost
  - Find: paths for the flows, capacity, topology

Power of binary variables

- Not only for making yes-or-no type of decisions, binary variables can be used to capture many other requirements
  - Restricted range of values
  - Either-or constraints
  - If-then constraints
  - Piecewise linear functions

Restricted range of values

- Some variables can only take certain values
  - E.g. network links can only be of capacity 155 Mbps, 466 Mbps, 622 Mbps, etc
- If decision variable x can only take values from $\{a_1, a_2, ..., a_m\}$, this can be modeled by using an additional set of binary decision variables

$$y_i = \begin{cases} 1 & \text{if } a_i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

Restricted range of values

- Then, the above requirement can be captured by

$$\begin{aligned} x &= \sum_{i=1}^{m} a_i y_i \\ \sum_{i=1}^{m} y_i &= 1 \\ y_i &\in \{0, 1\} \end{aligned}$$

- E.g. if $a_1 = 155, a_2 = 466, a_3 = 622$, we have
  - $y_1 = 1 \;\Rightarrow\; y_2 = y_3 = 0 \;\Rightarrow\; x = 155$
  - $y_2 = 1 \;\Rightarrow\; y_1 = y_3 = 0 \;\Rightarrow\; x = 466$
  - $y_3 = 1 \;\Rightarrow\; y_1 = y_2 = 0 \;\Rightarrow\; x = 622$

Either-or constraints

- A Cloud computing service provider offers 3 different packages with different speed and cost for each package. You can buy any cycles from any package but the deal requires that
  - # cycles from Package 1 $+$ # cycles from Package 2 $\geq$ 10000, or,
  - # cycles from Package 2 $+$ # cycles from Package 3 $\geq$ 50000
    - At least one of these two inequalities must hold, but not necessarily both
- Let $w_i =$ number of cycles to be bought from Package i

Either-or constraints (cont.)

- The above requirement can be captured by using an additional binary decision variable p

$$\begin{aligned} w_1 + w_2 &\geq 10000p \\ w_2 + w_3 &\geq 50000(1-p) \\ p &\in \{0, 1\} \\ w_i &\geq 0, \quad i = 1, 2, 3 \end{aligned}$$

Case 1: $p = 0$, we have

$$\begin{aligned} w_1 + w_2 &\geq 0 \leftarrow \text{Trivially satisfied} \\ w_2 + w_3 &\geq 50000 \\ w_i &\geq 0, \quad i = 1, 2, 3 \end{aligned}$$

Case 2: $p = 1$, we have

$$\begin{aligned} w_1 + w_2 &\geq 10000 \\ w_2 + w_3 &\geq 0 \leftarrow \text{Trivially satisfied} \\ w_i &\geq 0, \quad i = 1, 2, 3 \end{aligned}$$

Either-or constraints (cont.)

- In general, if one of the following two constraints must be satisfied

$$\begin{aligned}\sum_{i=1}^{n} a_{1,i} x_i &\geq b_1\\ \sum_{i=1}^{n} a_{2,i} x_i &\geq b_2\end{aligned}$$

where $a_{j,i}$ are given parameters, $x_i (\geq 0)$ are decision variables, b_j are scalar, then the either-or constraints can be modeled by

$$\begin{aligned}\sum_{i=1}^{n} a_{1,i} x_i &\geq b_1 p\\ \sum_{i=1}^{n} a_{2,i} x_i &\geq b_2(1-p)\\ p &\in \{0,1\}\end{aligned}$$

If-then constraints

- We may want to impose if-then constraints, e.g.

$$\text{if } x_1 + x_2 > 1, \text{ then } y \geq 4$$

where x_1, x_2 are binary variables, and $0 \leq y \leq 10$

- The above if-then constraint can be captured by using an additional binary decision variable p

$$\begin{array}{rcl} x_1 + x_2 - 1 & \leq & 1 - p \\ -y + 4 & \leq & 4p \\ p & \in & \{0, 1\} \end{array}$$

If-then constraints (cont.)

- To understand how this works, consider the two cases:
  - Case 1: If $x_1 + x_2 > 1$ holds
    - Since $x_1 + x_2 > 1$, $x_1 + x_2 - 1 > 0$
    - Since p can only be 1 or 0, the inequality constraint $x1 + x2 - 1 \leq 1 - p$ forces p to be 0
    - Since $p = 0$, from the inequality constraint $-y + 4 \leq 4p$, we have $y \geq 4$ which is the condition that we want to impose when $x_1 + x_2 > 1$ holds
  - Case 2: If $x_1 + x_2 > 1$ does not hold
    - In this case, since $x_1 + x_2 - 1 \leq 0$, p can be either 0 or 1
    - If $p = 0$, the inequality constraint $-y + 4 \leq 4p$ becomes $y \geq 4$
    - If $p = 1$, the inequality constraint $-y + 4 \leq 4p$ becomes $y \geq 0$
    - Thus, p can be chosen such that there is no restriction on the value of y

If-then constraints (cont.)

- In general, the if-then constraint

  $$\text{if } f(x_1, x_2, \ldots, x_n) > 0, \text{ then } g(x_1, x_2, \ldots, x_n) \geq 0$$

  can be modeled by

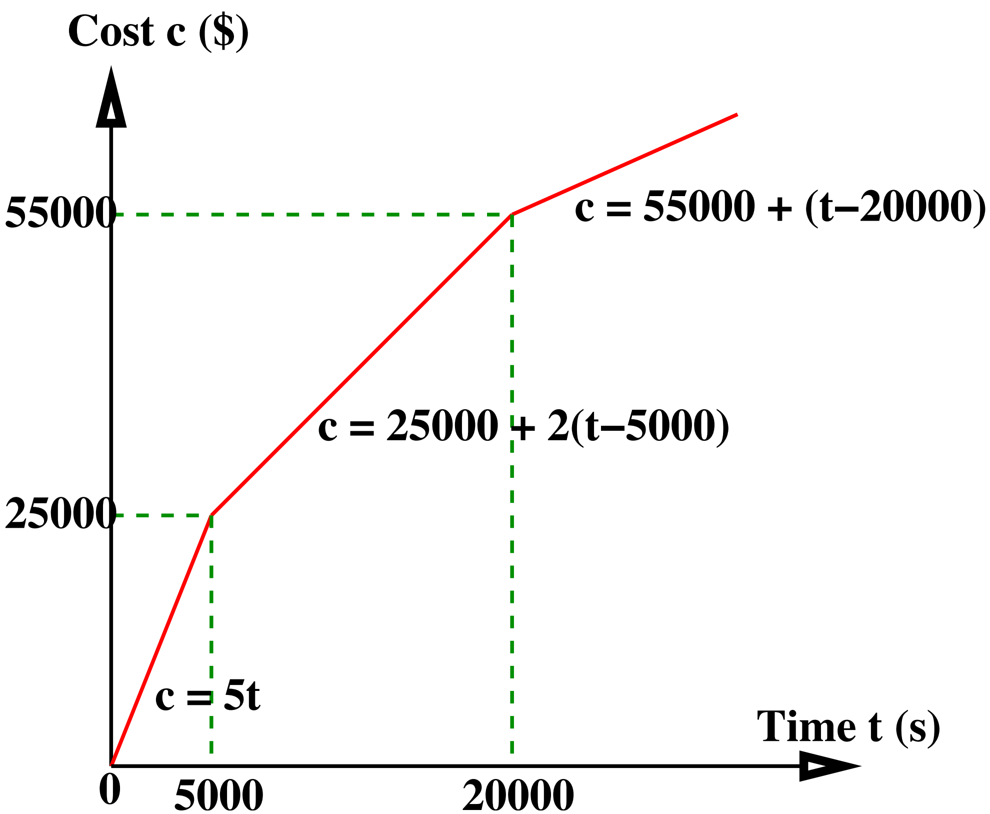
  $$\begin{aligned} f(x_1, x_2, \ldots, x_n) &\leq M_1(1 - p) \\ -g(x_1, x_2, \ldots, x_n) &\leq M_2\, p \end{aligned}$$

  where p is a binary variable, M_1 and M_2 are constants chosen large enough such that $f(x_1, x_2, \ldots, x_n) \leq M_1$ and $-g(x_1, x_2, \ldots, x_n) \leq M_2$ hold for all possible choices of $x_1, x_2, \ldots, x_n$

Piecewise linear functions

- We can use binary variables to model piecewise linear functions
- Example: A Cloud computing service provider may use a progressive charging scheme
  - 5 dollars/sec for the first 5,000 sec
  - 2 dollars/sec for the next 15,000 sec
  - 1 dollar/sec thereafter

Piecewise linear functions (cont.)



- Decision variables

$$y_i = \begin{cases} 1 & \text{if segment } i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

- Segment 1: $0 \leq t \leq 5000$, cost = $5t$
- Segment 2: $5000 \leq t \leq 20000$, cost = $2t$ + 15000
- Segment 3: $20000 \leq t$, cost = t + 35000

Piecewise linear functions (cont.)

- We have
  - $y_1 = 1 \Rightarrow 0 \leq t \leq 5000$ and cost = $5t$
  - $y_2 = 1 \Rightarrow 5000 \leq t \leq 20000$ and cost = $2t$ + 15000
  - $y_3 = 1 \Rightarrow 20000 \leq t$ and cost = t + 35000
  - $y_1 + y_2 + y_3 = 1$
- We can rewrite these as
  - $0 \leq ty_1 \leq 5000y_1$
  - $5000y_2 \leq ty_2 \leq 20000y_2$
  - $20000y_3 \leq ty_3$
  - cost = $y_1(5t) + y_2(2t + 15000) + y_3(t + 35000)$
  - $y_1 + y_2 + y_3 = 1$
- Problem: non-linear constraints

Piecewise linear functions (cont.)

- Define $t_i = ty_i$ for $i = 1, 2, 3$
  - cost = $5t_1 + 2t_2 + 15000y_2 + t_3 + 35000y_3$
  - $0 \leq t_1 \leq 5000y_1$
  - $5000y_2 \leq t_2 \leq 20000y_2$
  - $20000y_3 \leq t_3 \leq My_3$
  - $y_1 + y_2 + y_3 = 1$
  - $t = t_1 + t_2 + t_3$
- Note
  - t_i is non-zero if the corresponding $y_i = 1$
  - M is a sufficiently large number to enforce

$$y_3 = 0 \Rightarrow t_3 = 0 \quad \text{and} \quad t_3 \geq 20000 \Rightarrow y_3 = 1$$

  - This is a non-standard expression
  - An alternative expression can be found in Winston Chapter 9

References

- Advanced formulation of integer programming problems
  - Winston, “Operations Research”, Section 9.2
- Network flow problems
  - Ahuja et al, “Network Flows”, Sections 1.2