ECE 307 – Techniques for Engineering Decisions

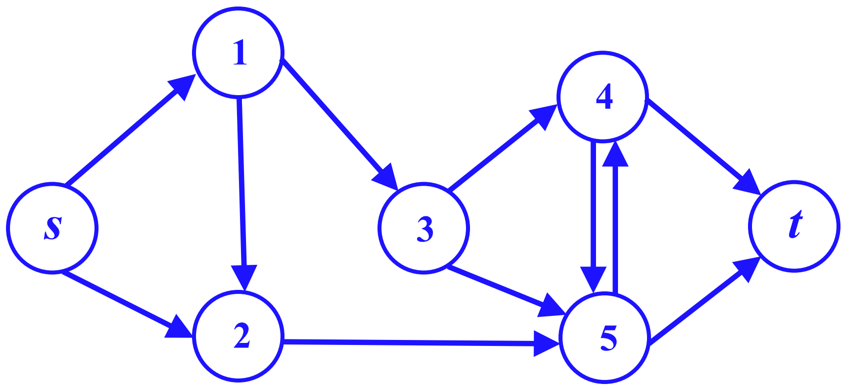
Lecture 6. Transshipment and Shortest Path Problems

George Gross
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

TRANSSHIPMENT PROBLEMS

- We consider the shipment of a *homogeneous* commodity or product from a specified point or *source* to a particular destination or *sink*: the homogeneity characteristic ensures that each unit shipped is identical and is independent of point of origin
- Typically, the *source* and the *sink* are not directly connected; rather, the flows pass through the *transshipment points*, i.e., the intermediate nodes
- The objective is to determine the *maximal flow* from the *source* to the *sink*

DIRECTED FLOW NETWORK EXAMPLE



TRANSSHIPMENT PROBLEMS

- nodes 1, 2, 3, 4 and 5 are the *transshipment points*
- *directed arcs* of the network are (s, 1), (s, 2), (1, 2), (1, 3), (2, 5), (3, 4), (3, 5), (4, 5), (5, 4), (4, t), (5, t) ; the existence of an arc from 4 to 5 and from 5 to 4 allows bi-directional flows between the two nodes
- each arc may be constrained in terms of a *limit on the flow through the arc*

MAX FLOW PROBLEM

- We denote by f_{ij} the flow from i to j, which equals the amount of the commodity shipped from i to j on the arc (i, j) that directly connects the node i to the node j
- The problem is to determine the maximal flow f from s to t taking into account the *flow limits* k_{ij} of each arc (i, j)
- The mathematical statement of the problem is

MAX FLOW PROBLEM

$$\max \quad Z = f$$

s.t.

$$0 \leq f_{ij} \leq k_{ij} \qquad \forall \text{ arc } (i, j) \text{ that connects nodes } i \text{ and } j$$

$$\left.\begin{array}{lll} f = \sum_i f_{si} & \text{at source } s \\ \sum_i f_{it} = f & \text{at sink } t \end{array}\right\} \text{conservation of flow relations}$$

$$\sum_i f_{ij} = \sum_k f_{jk} \quad \Big\} \text{ at each transshipment node } j$$

MAX FLOW PROBLEM

- While we may use the simplex approach to solve the *max flow* problem, we construct a numerically, highly efficient *network* method to determine f
- We develop such a scheme by making detailed use of graph theoretic notions
- We start out by introducing some definitions

DEFINITIONS OF NETWORK TERMS

- Each *arc* is directed and so for an arc (i, j),

$$f_{ij} \geq 0$$

- A *forward* arc at a node i is one that leaves the node i to some node j and is denoted by (i, j)
- A *backward* arc at node i is one that enters node i from some node j and is denoted by (j, i)

DEFINITIONS OF NETWORK TERMS

- A *path* connecting node i to node j is a *sequence* of arcs that starts at node i and terminates at node j
  - we denote a path by

$$\mathscr{P} = \{ (i, k), (k, l), \; . \; . \; . \; , (m, j) \}$$

  - in the example network
    - $\{ (1,2),(2,5),(5,4) \}$ is a path from 1 to 4
    - $\{ (1,3),\ (3,4) \}$ is another path from 1 to 4

DEFINITIONS OF NETWORK TERMS

- A *cycle* is a path with the condition $j = i$, i.e.,

$$\mathscr{P} = \{ (i, k), (k, l), \; . \; . \; . \; , (m, i) \}$$

- We denote the set of nodes of the network by $\mathscr{N}$
  - the definition is

$$\mathscr{N} = \{ i : i \text{ is a node of the network } \}$$

  - In the example network

$$\mathscr{N} = \{ s , 1, 2, 3, 4, 5, t \}$$

NETWORK CUT CAPACITY

- A *cut* is a partitioning of nodes into two distinct subsets $\mathcal{S}$ and $\mathcal{T}$ with the properties

$$\mathcal{N} = \mathcal{S} \cup \mathcal{T} \text{ and } \mathcal{S} \cap \mathcal{T} = \varnothing$$

- We are interested in cuts with the property that

$$s \in \mathcal{S} \text{ and } t \in \mathcal{T}$$

- We say that the sets $\mathcal{S}$ and $\mathcal{T}$ provide an $s-t$ cut; in the example network,

$$\mathcal{S} = \{s, 1, 2\} \quad \text{and} \quad \mathcal{T} = \{3, 4, 5, t\}$$

provide an $s-t$ cut

NETWORK CUT

- The capacity of a cut is

$$K(\mathcal{S},\mathcal{T}) = \sum_{\substack{s \in \mathcal{S} \\ t \in \mathcal{T}}} k_{st}$$

- In the example network with

$$\mathcal{S} = \{s, 1, 2\} \quad \text{and} \quad \mathcal{T} = \{3, 4, 5, t\}$$

we have

$$K(\mathcal{S},\mathcal{T}) = k_{13} + k_{25}$$

but for the cut with

$$\mathcal{S} = \{s, 1, 2, 3, 4\} \quad \text{and} \quad \mathcal{T} = \{5, t\}$$

$$K(\mathcal{S},\mathcal{T}) = k_{4,t} + k_{4,5} + k_{3,5} + k_{2,5}$$

NETWORK CUT

- Note: arc $(5, 4)$ is directed from a node in $\mathcal{T}$ to a node in $\mathcal{S}$ and is not included in the summation
- A *salient characteristic* of the $s - t$ cuts of interest is that when all the arcs in the cut are removed, then ***no*** path exists from s to t ; consequently, no flow is possible since any flow from s to t must go through the arcs in a cut
- The flow is *limited* by the capacity of the cut

NETWORK CUT LEMMA

- For any directed network, the flow f from s to t is constrained by an $s - t$ cut so that

$$f \leq K(\mathcal{S}, \mathcal{T}) \;\; \textit{for every } s-t \textit{ cut set } \mathcal{S}, \mathcal{T}$$

- Corollaries of this lemma are

$$(i) \quad \textit{max flow} \leq K(\mathcal{S}, \mathcal{T}) \;\; \forall \;\; \mathcal{S}, \mathcal{T}$$

and

$$(ii) \quad \textit{max flow} \leq \min_{\mathcal{S}, \mathcal{T}} K(\mathcal{S}, \mathcal{T})$$

MAX – FLOW – MIN – CUT THEOREM

- For any network, the value of the maximal flow from s to t is equal to the minimal cut, i.e., the cut $\mathcal{S}, \mathcal{T}$ with the smallest capacity
- The *max-flow min-cut* theorem allows us, in principle, to find the maximal flow in a network, we find the capacity of each of the cuts and determine the cut with the smallest capacity

MAX FLOW

- The *maximal flow* algorithm is based on the identification of a *path* through which a positive flow from s to t can be sent – the so-called *flow augmenting path*
- The procedure is continued until no such *flow augmenting path* can be found and therefore we have the *maximal flow*
- The maximal flow algorithm is based on the repeated application of the *labeling procedure*

LABELING PROCEDURE

- The *labeling procedure* is the basic scheme to determine the maximum flow in a network
- The *labeling procedure* is used to find a *flow augmenting path* from s to t
- We say that a node j can be *labeled* if and only if flow can be sent from s to t and node j is on a path to make such flow possible

 17

LABELING PROCEDURE

Step 0 : start with node s

Step 1 : given that node i is already labeled, label node j only if

(i) either there exists an arc (i, j) and

$$f_{ij} < k_{ij}$$

(ii) or, there exists an arc (j, i) and

$$f_{ji} > 0$$

Step 2 : if $j = t$, stop; else, return to Step 1

 18

THE MAX FLOW ALGORITHM

Step 0 : start with a feasible flow

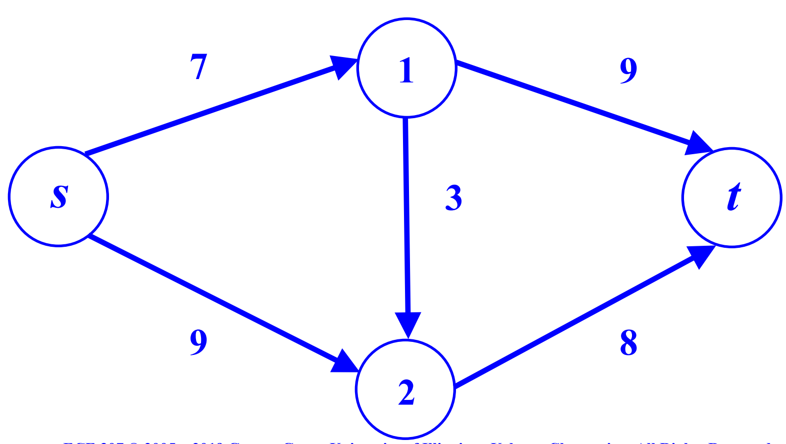
Step 1 : use the *labeling procedure* to find a flow augmenting path

Step 2 : determine the maximum value δ for the largest increase (decrease) of flow on all forward (backward) arcs

Step 3 : use the *labeling procedure* to find a flow augmenting path: if no such path exists, stop; else, go to Step 2

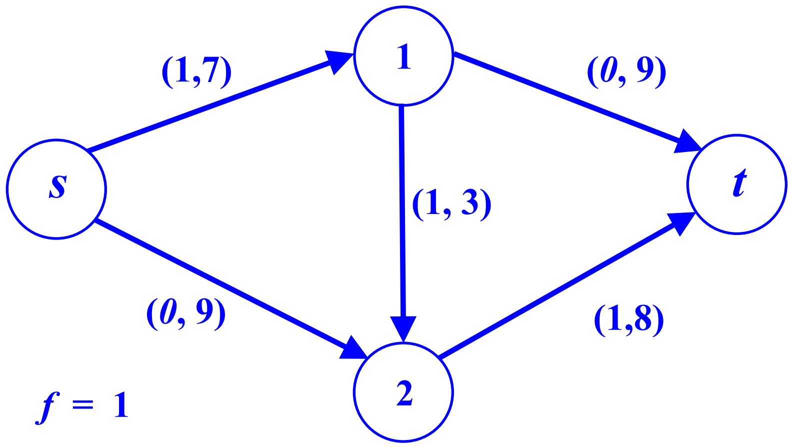
ILLUSTRATIVE EXAMPLE

- Consider the simple network with the flow capacities on each arc indicated



ILLUSTRATIVE EXAMPLE

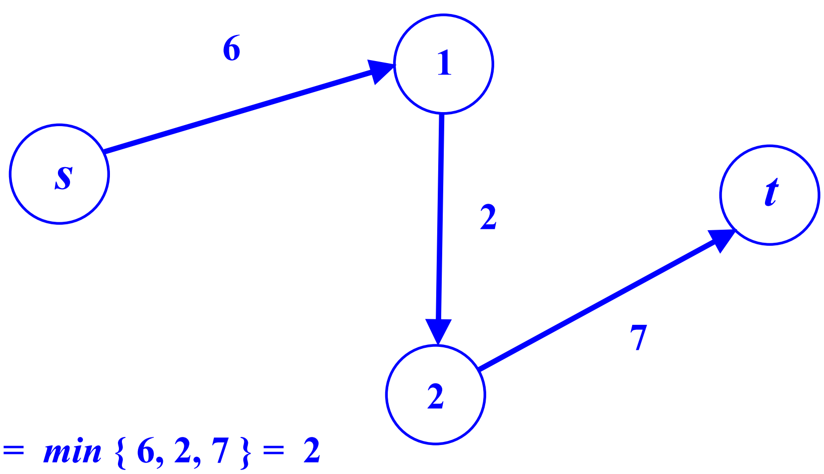
- We initialize the network with a flow 1



$f = 1$

ILLUSTRATIVE EXAMPLE

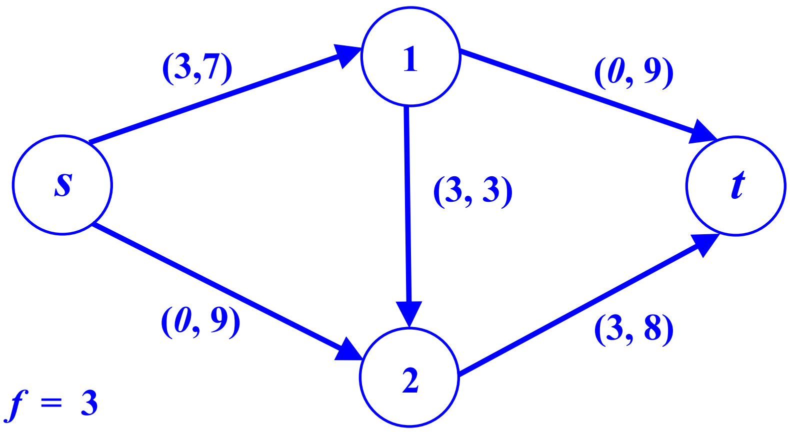
- We apply the labeling procedure



$f = min\ \{ 6, 2, 7 \} = 2$

ILLUSTRATIVE EXAMPLE

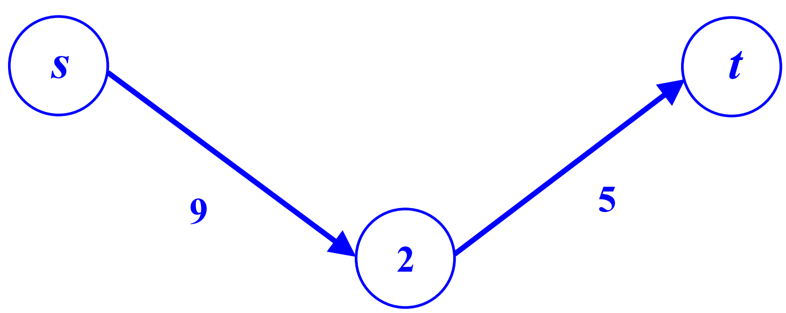
- Consider the simple network with the flow and the capacity on each arc (i, j) indicated by (f_{ij}, k_{ij})



$f = 3$

ILLUSTRATIVE EXAMPLE

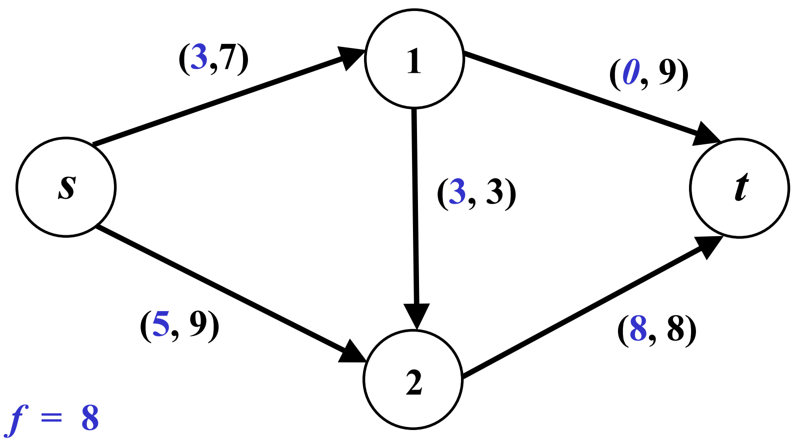
- We repeat application of the labeling procedure



$f = min\ \{ 5, 9 \} = 5$

ILLUSTRATIVE EXAMPLE

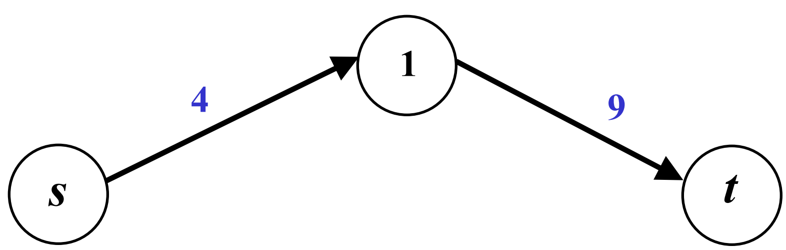
- We increase the flow by 5



$f = 8$

ILLUSTRATIVE EXAMPLE

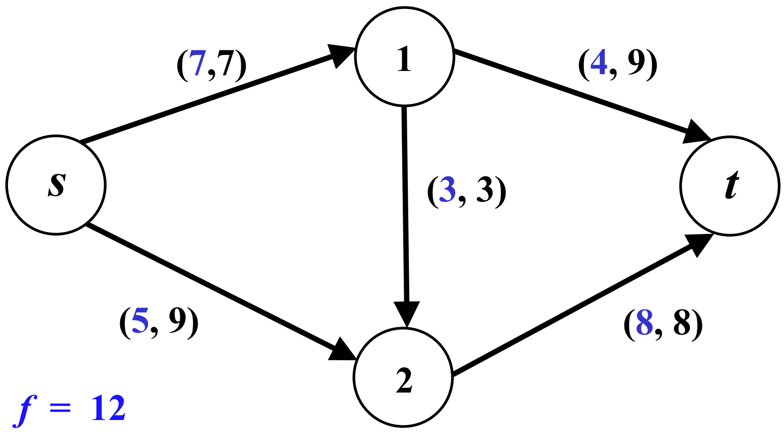
- We repeat application of the labeling procedure



$f = min\ \{ 4, 9 \} = 4$

ILLUSTRATIVE EXAMPLE

❑ We increase the flow by 4 to obtain

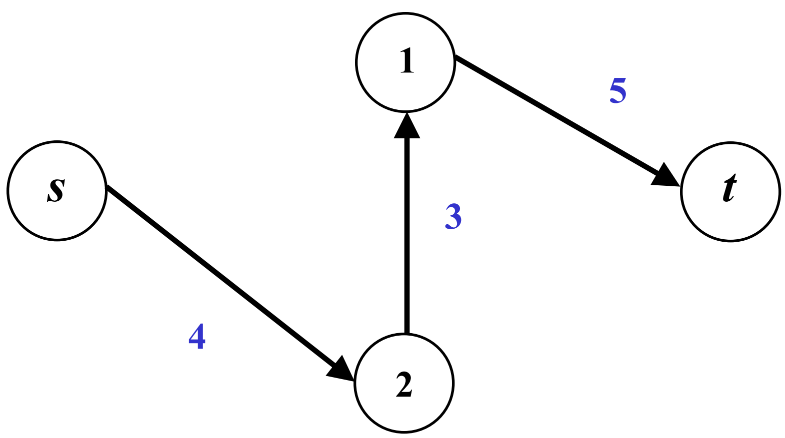


$f = 12$

 27

ILLUSTRATIVE EXAMPLE

❑ We repeat application of the *labeling procedure*

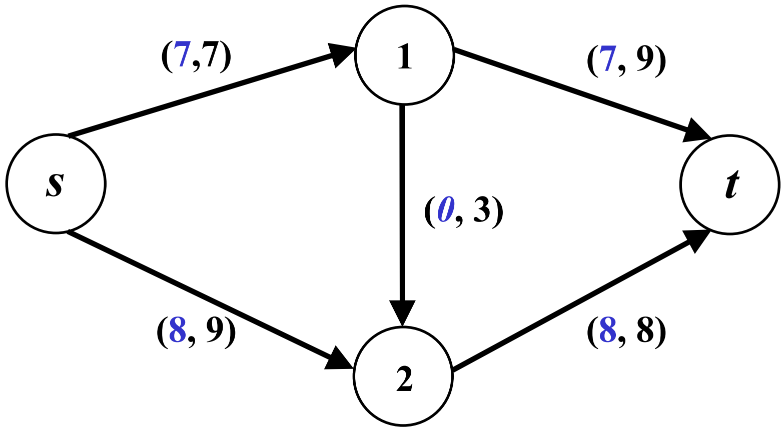


$f = min\ \{ 4, 3, 5 \} = 3$

 28

ILLUSTRATIVE EXAMPLE

- We increase the flow by 3



$f = 15$ with no *flow augmenting path*

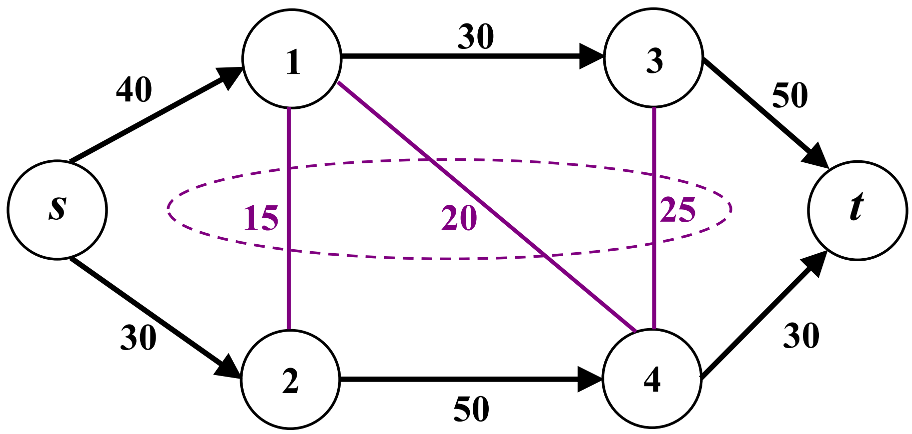
UNDIRECTED NETWORKS

- A network with undirected arcs is called an *undirected network*: the flows on each arc (i, j) with the limit k_{ij} cannot violate the capacity constraints in either direction
- Mathematically, we require

$$\left.\begin{aligned} f_{ij} &\leq k_{ij} \\ f_{ji} &\leq k_{ji} \\ f_{ij} f_{ji} &= 0 \end{aligned}\right\}$$

interpretation of unidirectional flow below capacity limit

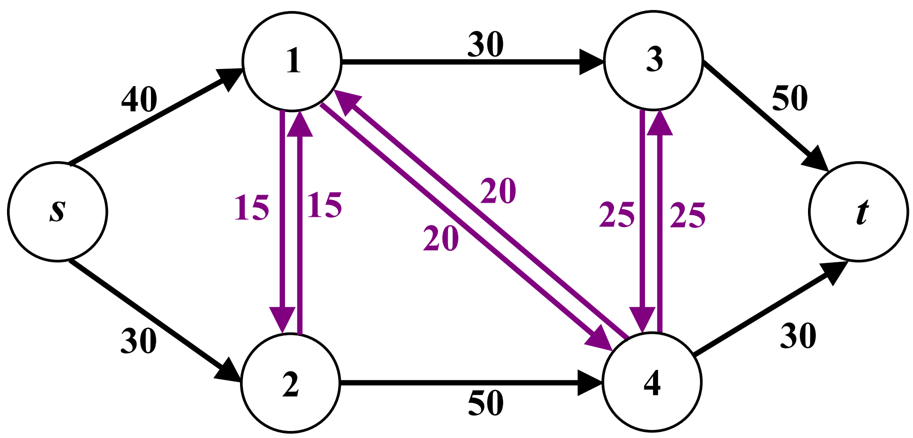
EXAMPLE OF A NETWORK WITH 3 UNDIRECTED ARCS



EXAMPLE OF A NETWORK WITH UNDIRECTED ARCS

- To make the problem realistic, we may view the capacities as representing traffic flow limits: the directed arcs correspond to *unidirectional* streets and the problem is to place *one-way signs* on each undirected street (i, j) so as to *maximize* the traffic flow from s to t
- The procedure is to replace each *undirected arc* by two *directed* arcs (i, j) and (j, i) to determine the maximal $s - t$ flow

EXAMPLE OF A NETWORK WITH 3 UNDIRECTED ARCS



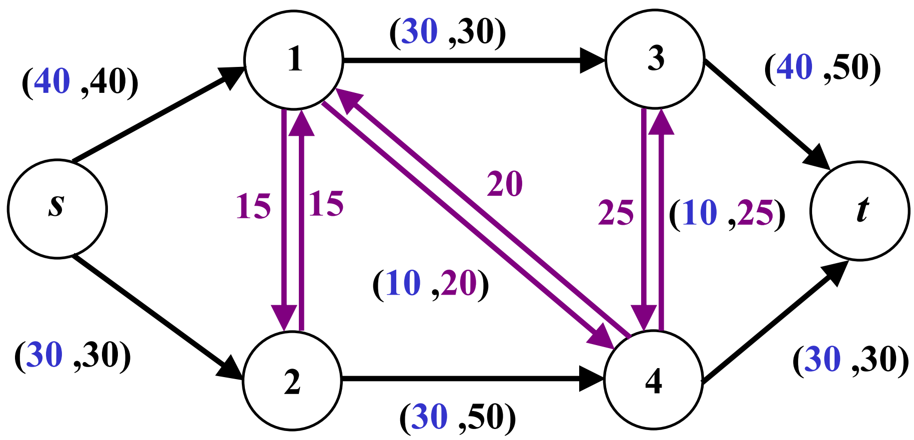
EXAMPLE OF A NETWORK WITH 3 UNDIRECTED ARCS

- We apply the *max flow* scheme to the directed network and give the following interpretations to the flows on the *max flow* bidirectional arcs that are the initially undirected arcs (i, j) : if

$$f_{ij} > 0 \ , \ f_{ji} > 0 \ \text{and} \ f_{ij} > f_{ji} \ ,$$

  set up the flow from i to j with value $f_{ij} - f_{ji}$ and remove the arc (j, i)
- The determination of the max flow f for this example is easily performed

EXAMPLE OF A NETWORK WITH 3 UNDIRECTED ARCS



EXAMPLE OF A NETWORK WITH 3 UNDIRECTED ARCS : RESULT

flow: $s \rightarrow 1 \rightarrow 3 \rightarrow t$ = 30

flow: $s \rightarrow 2 \rightarrow 4 \rightarrow t$ = 30

flow: $s \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow t$ = 10

and so the maximum flow is $30 + 30 + 10 = 70$

one way signs must be put from $1 \rightarrow 4$ and $4 \rightarrow 3$;

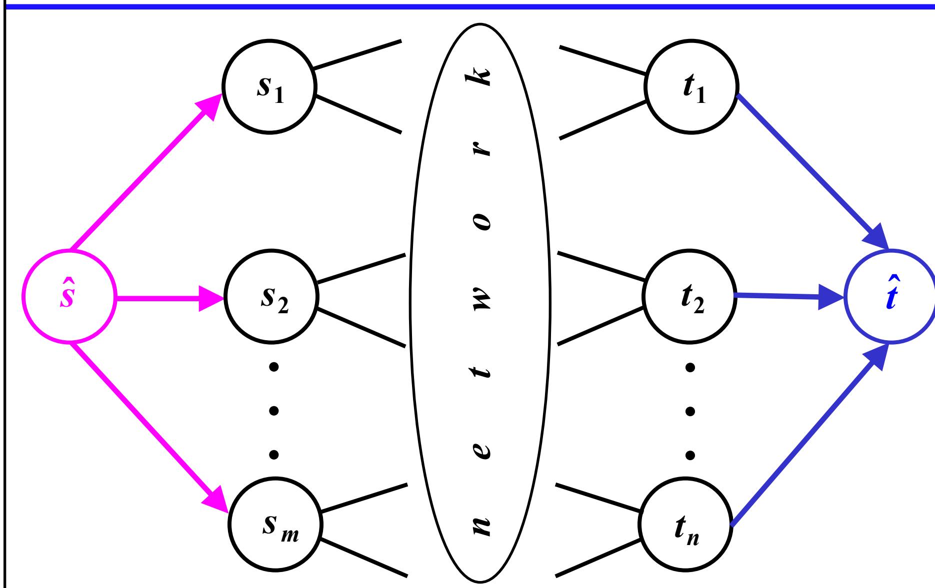
an alternative path of a flow of 10 is the path:

$s \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow t$, which requires one-way signs from $1 \rightarrow 2$ and $4 \rightarrow 3$

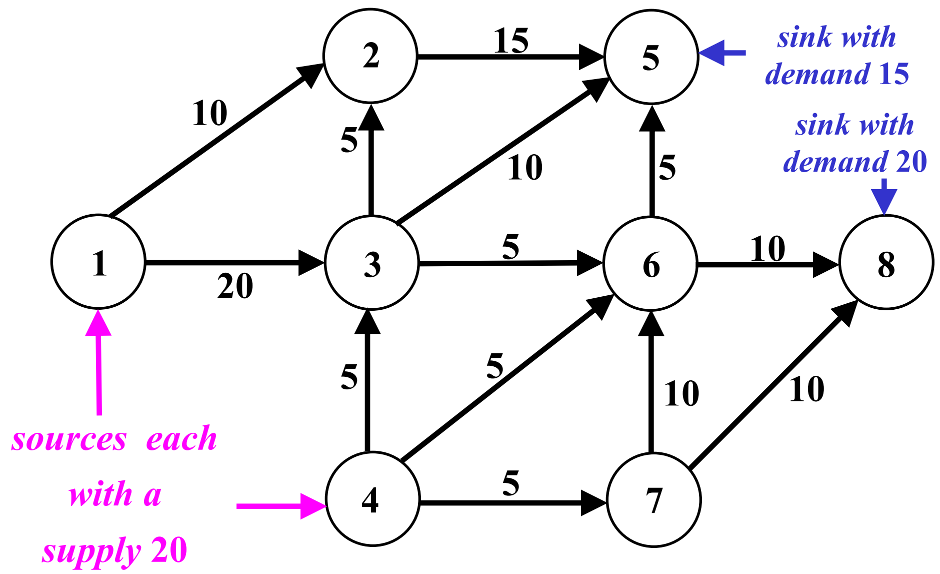
NETWORKS WITH MULTIPLE SOURCES AND MULTIPLE SINKS

- We next consider a network with several supply and several demand points
- We introduce a super *source* $\hat{s}$ linking to all the *sources* and a super *sink* $\hat{t}$ linking all the *sinks*
- We can consequently apply the *max flow* algorithm to the modified network

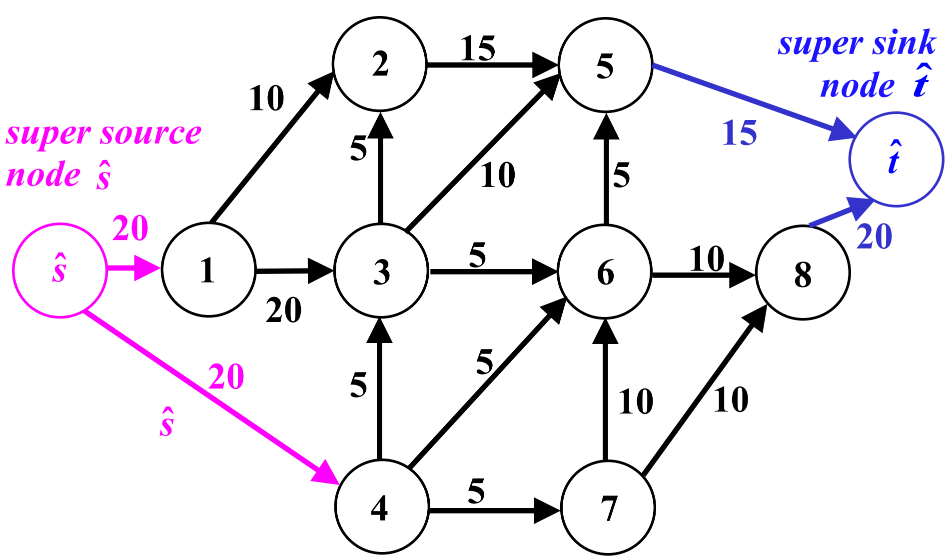
NETWORKS WITH MULTIPLE SOURCES AND MULTIPLE SINKS



MULTIPLE – SOURCE / MULTIPLE – SINK NETWORK EXAMPLE



MULTIPLE – SOURCE / MULTIPLE – SINK NETWORK EXAMPLE



MULTIPLE – SOURCE / MULTIPLE – SINK NETWORK EXAMPLE

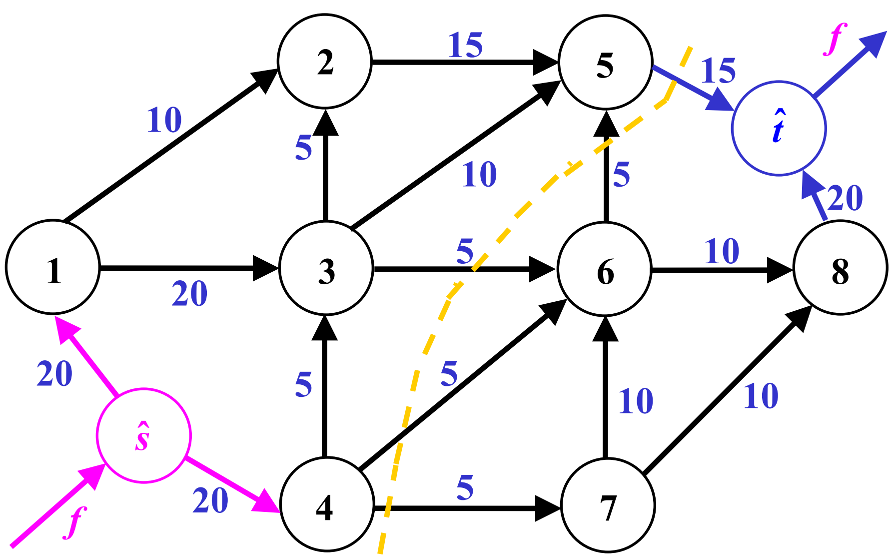
- The transshipment problem is feasible if and only if the maximal $\hat{s}-\hat{t}$ flow f satisfies

$$f = \sum_{sinks} demands$$

- We need to show that
  - the transshipment problem is infeasible since the network cannot accommodate the total demand of 35
  - the smallest shortage for this problem is 5

 41

MULTIPLE – SOURCE / MULTIPLE – SINK NETWORK EXAMPLE



 42

MULTIPLE – SOURCE / MULTIPLE – SINK NETWORK EXAMPLE

- The minimum cut is shown and has capacity

$$15 + 5 + 5 + 5 = 30 ;$$

the maximum flow is, therefore, 30

- Since the maximum flow fails to meet the total demand of 35 units by the super sink, the problem is infeasible; the minimum shortage is 5

APPLICATION TO MANPOWER SCHEDULING

- Consider the case of a company that must complete its 4 engineering projects within 6 months

project	*earliest start month*	*latest finish month*	*manpower requirements (man month)*
A	1	4	6
B	1	6	8
C	2	5	3
D	1	6	4

APPLICATION TO MANPOWER SCHEDULING

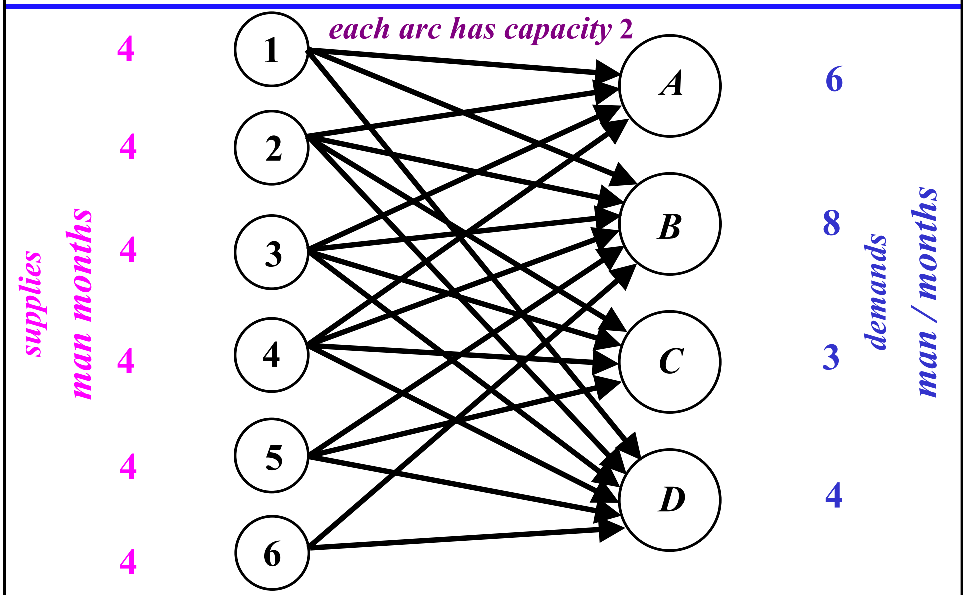
- There are the following additional constraints:
  - the company has only 4 engineers
  - at most 2 engineers may be assigned to any one project in a given month
  - no engineer may be assigned to more than one project at any time
- The question is whether there is a *feasible assignment* and, if so, determine the *optimal assignment*

APPLICATION TO MANPOWER SCHEDULING

- The solution approach is to set up the problem as a transshipment network
  - the *sources* are the 6 months of engineer labor
  - the *sinks* are the 4 projects that must be done
  - an arc (i, j) is introduced whenever a feasible assignment of the engineers who work in month i can be made to project j with

    $$k_{ij} = 2 \qquad i = 1, 2, \dots, 6 \ , \quad j = A, B, C, D$$
  - there is no arc $(1, C)$ since project C cannot start before month 2

APPLICATION TO MANPOWER SCHEDULING



 47

APPLICATION TO MANPOWER SCHEDULING

- The transshipment problem is feasible if the total demand

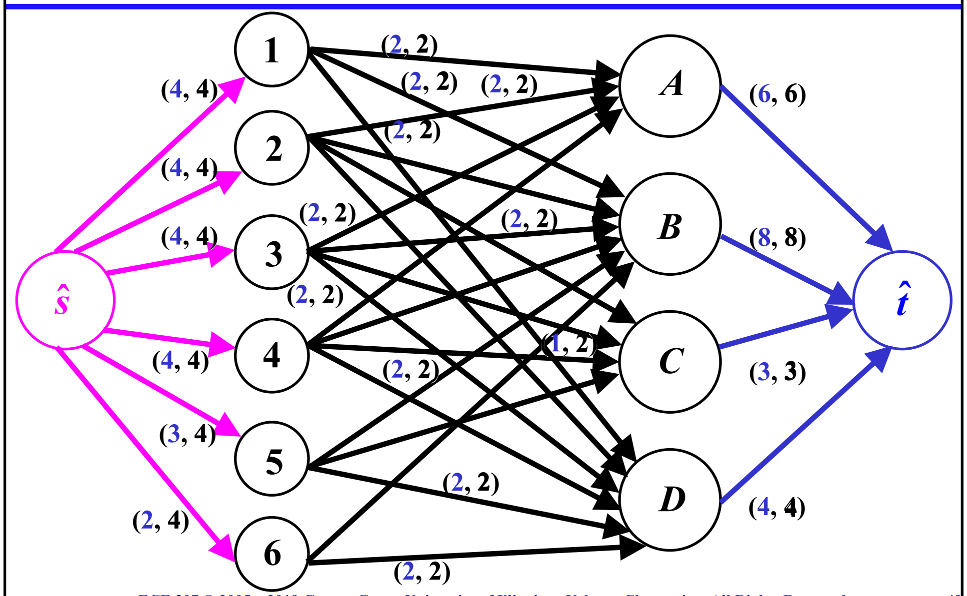
$$6 + 8 + 3 + 4 = 21$$

can be met

- We determine whether a feasible schedule exists and if so, we find it

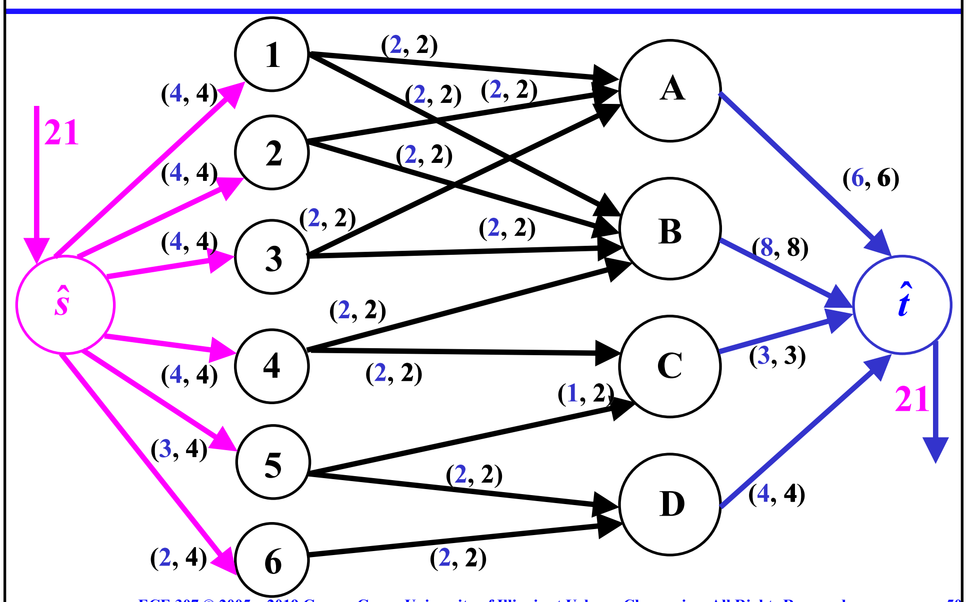
 48

APPLICATION TO MANPOWER SCHEDULING



 49

APPLICATION TO MANPOWER SCHEDULING



 50

SHORTEST ROUTE PROBLEM

- The problem is to determine the *shortest path* from $s = 1$ to $t = n$ in a network with the set of nodes

$$\mathcal{N} = \{1, 2, \ldots, n\}$$

and the set of arcs $\{(i, j)\}$, where for each arc (i, j)

$$d_{ij} = \textit{distance or transit time}$$

- The determination of the shortest path from 1 to n requires the specification of the path

$$\{ (1, i_1), (i_1, i_2), \ldots, (i_q, n) \}$$

SHORTEST ROUTE PROBLEM

- We can write an *LP* formulation of this problem in the form of a *transshipment problem*:

  ship 1 unit from node 1 to node n by minimizing the shipping costs using the costs

$$d_{ij} = \begin{cases} \textit{shipping costs for } 1 \textit{ unit from } i \textit{ to } j \\ \infty \textit{ whenever } i \textit{ and } j \textit{ are not directly connected} \end{cases}$$

- But, in practice, we use the *Dijkstra scheme solution*

THE DIJKSTRA ALGORITHM

- The solution is very efficiently performed using *the Dijkstra algorithm*
- The assumptions are
  - d_{ij} is given for each pair of connected nodes
  - $d_{ij} \geq 0$
- The scheme is, basically, a label assignment procedure, which assigns nodes with either a *permanent* or a *temporary* label

THE DIJKSTRA ALGORITHM

- The *temporary* label of a node i is an upper bound on the shortest distance from node 1 to node i
- The *permanent* label is the actual shortest distance from node 1 to node i
- A temporary label becomes permanent when we find the tightest upper bound, i.e., the shortest distance

THE DIJKSTRA ALGORITHM

Step 0 : assign the *permanent* label 0 to node 1

Step 1 : assign *temporary* labels to all the other nodes

- d_{1j} if node j is directly connected to node 1
- ∞ if node j is not directly connected to node 1

and select the minimum of the *temporary* labels and declare it *permanent* ; in case of ties, the choice is arbitrary (but a rule is required to break the tie in a systematic way)

THE DIJKSTRA ALGORITHM

Step 2 : let ℓ be the node most recently assigned a *permanent* label and consider each node j with a *temporary* label; recompute each label

$$min \left\{ \begin{matrix} \textit{temporary label} \\ \textit{of node } j \end{matrix} \; , \; \begin{matrix} \textit{permanent label} \\ \textit{of node } \ell \end{matrix} \; + \; d_{\ell j} \right\}$$

Step 3 : select the smallest valued *temporary* label & declare its node *permanent* ; in case of ties, the choice is arbitrary – *but, we need a rule*

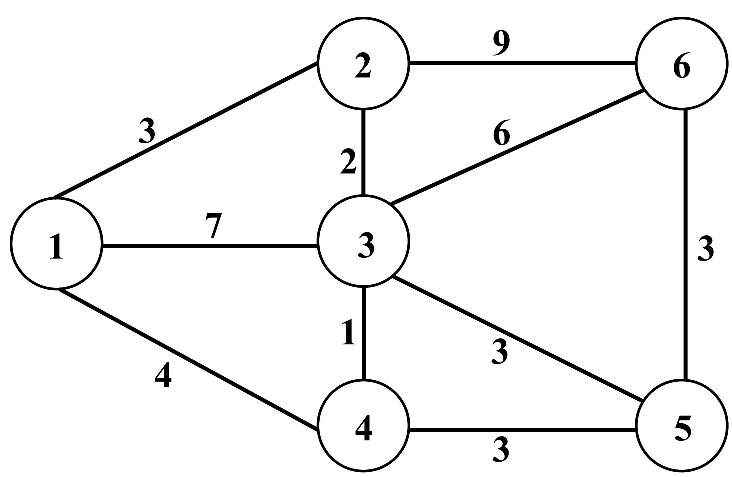
Step 4 : if the selected node is n , stop; else, return to Step 2

THE DIJKSTRA ALGORITHM

- We obtain the **shortest path** by **retracing** the sequence of nodes with permanent labels starting at node n and returning back to node 1
- The path is then given in the **forward direction** starting from node 1 and ending at node n

EXAMPLE : SHORTEST PATH

- Consider the undirected network



EXAMPLE : SHORTEST PATH

- The problem is to
  - find the shortest path from 1 to 6
  - compute the length of the shortest path
- We apply the Dijkstra algorithm and assign iteratively a *permanent* label to each node

EXAMPLE : SHORTEST PATH

$\boxed{0}$

Steps 0 and 1 : $\mathscr{L}(0) = [0,3,7,4,\infty,\infty]$

initial label

$\boxed{1}$

Step 2 : $\mathscr{L}(1) = [0,3,5,4,\infty,12]$

label in iteration 1

$\boxed{2}$

Steps $2,3$ and 4 : $\mathscr{L}(2) = [0,3,5,4,7,12]$

label in iteration 2

$\boxed{3}$

EXAMPLE : SHORTEST PATH

Steps 2,3 and 4 : $\mathscr{L}(3) = [0,3,5,4,7,11]$

$\boxed{4}$

label in iteration 3

Steps 2,3 and 4 : $\mathscr{L}(4) = [0,3,5,4,7,10]$

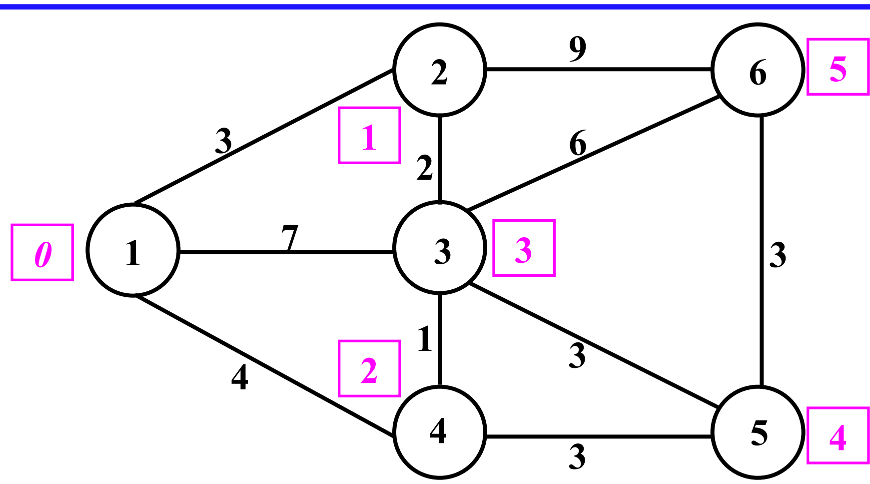
$\boxed{5}$

label in iteration 4

$\mathscr{L}(4) = [0,3,5,4,7,10]$

$\boxed{6}$

EXAMPLE : SHORTEST PATH



- The shortest distance is 10 obtained with the path

$\{(1, 4), (4, 5), (5, 6)\}$

PATH RETRACING

- We retrace the path from n back to 1 using the scheme:

  pick node j preceding node n as the node with the property

$$\underset{\text{node } j}{\textit{permanent label of}} + d_{jn} = \underset{\textit{distance}}{\textit{shortest}}$$

- In the retracing scheme, certain nodes may be skipped

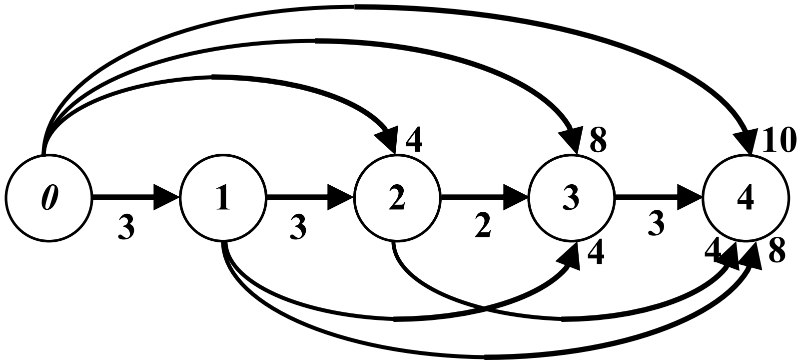
 63

SHORTEST PATH BETWEEN ANY TWO NODES

- The Dijkstra algorithm may be applied to compute the *shortest distance* between *any* pair of nodes i, j by taking i as the *source* node and j as the *sink* node
- We give as an example the following five – node network

 64

EXAMPLE : FIVE – NODE NETWORK



EXAMPLE : FIVE – NODE NETWORK

$$\mathscr{L}(0) = [0,3,4,8,10]$$

$$\boxed{0}$$

$$\mathscr{L}(1) = [0,3,4,7,10]$$

$$\boxed{1}$$

$$\mathscr{L}(2) = [0,3,4,6,8]$$

$$\boxed{2}$$

EXAMPLE : FIVE – NODE NETWORK

$$\mathscr{L}(3) \;=\; \left[\,0,3,4,6,8\,\right]$$

$$\boxed{3}$$

determines the shortest distance from *0* to every other node

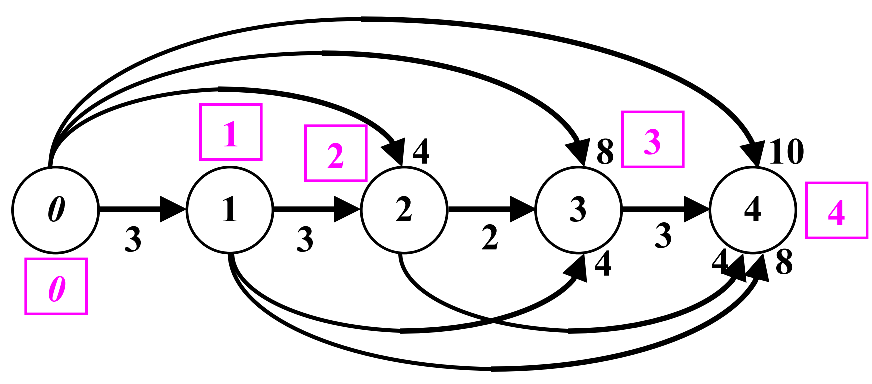
We retrace the path to get

$$8 = \underbrace{4}_{node\,2} + \underbrace{d_{24}}_{4}$$

and so the path is

$$0 \rightarrow 2 \rightarrow 4$$

EXAMPLE : FIVE – NODE NETWORK



APPPLICATION : EQUIPMENT REPLACEMENT PROBLEM

- We consider the problem of old equipment replacement or its continued maintenance
- As equipment ages, the level of maintenance required increases and, typically, results in increased operating costs
- *O&M* costs may be reduced by replacing aging equipment; however, replacement requires additional capital investment and so higher fixed costs

APPPLICATION : EQUIPMENT REPLACEMENT PROBLEM

- The problem is how often to replace equipment so as to minimize the total costs given by

$$\underset{costs}{total} = \underbrace{\underset{costs}{capital}}_{fixed} + \underbrace{\underset{costs}{O\&M}}_{variable}$$

EXAMPLE: EQUIPMENT REPLACEMENT

- Equipment replacement is planned during the next 5 years
- The cost elements are

p_j = *purchase costs in year j*

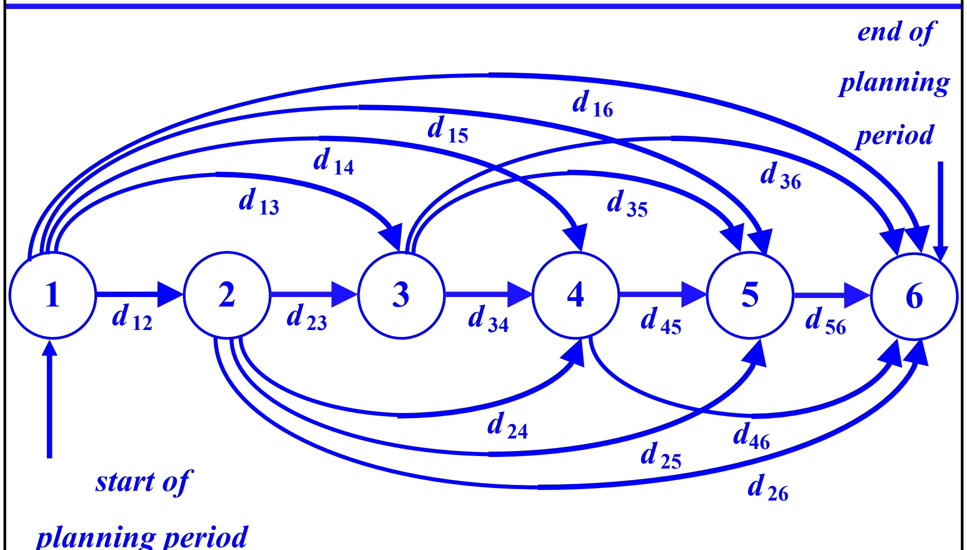
s_j = *salvage value of original equipment after j years of use*

c_j = *O&M costs in year j of operation of equipment with the property that*

$$\ldots\ c_j < c_{j+1} < c_{j+2} < \ \ldots$$

- We formulate this problem as a *shortest route* problem on a directed network

EQUIPMENT REPLACEMENT PROBLEM



APPPLICATION : EQUIPMENT REPLACEMENT PROBLEM

The "distances" d_{ij} are defined to be *finite* if $i < j$, i.e., year i precedes the year j , with

$$d_{ij} = \underset{\uparrow}{p_i} - \underset{\uparrow}{s_{j-i}} + \underbrace{\sum_{\tau=1}^{j-i} c_\tau}_{} \qquad j > i$$

purchase price in year i

salvage value after j – i years of use

O&M costs for j – i years of operation

APPPLICATION : EQUIPMENT REPLACEMENT PROBLEM

- For example, if the purchase is made in year 1

$$d_{16} = p_1 - s_5 + \sum_{\tau=1}^{5} c_\tau$$

- The solution is the shortest distance path from year 1 to year 6; if for example the path is

$$\{ (1, 2) , (2, 3) , (3, 4) , (4, 5) , (5, 6) \}$$

then the solution is interpreted as the replacement of the equipment each year with

$$\textit{total costs} = \sum_{\tau=1}^{5} p_\tau - 5s_1 + 5c_1$$

COMPACT BOOK STORAGE IN A LIBRARY

- This problem concerns the storage of books in a limited size library
- Books are stored according to their size, in terms of height and thickness, with books placed in groups of same or higher height; the set of book heights $\{ H_i \}$ is arranged in ascending order with

$$H_1 < H_2 < \ldots < H_n$$

 75

COMPACT BOOK STORAGE IN A LIBRARY

- Any book of height H_i may be shelved on a shelf of height at least H_i , i.e., $H_i, H_{i+1}, H_{i+2}, \ldots$
- The length L_i of shelving required for height H_i is computed given the thickness of each book; the total shelf area required is $\sum_i H_i L_i$
  - if only 1 height class [corresponding to the tallest book] exists, total shelf area required is the total length of the thickness of all books times the height of the tallest book

 76

COMPACT BOOK STORAGE IN A LIBRARY

- if 2 or more height classes are considered, the total area required is less than the total area required for a single class

- The costs of construction of shelf areas for each height class H_i have the components

  s_i fixed costs [independent of shelf area]

  c_i variable costs / unit area

 77

COMPACT BOOK STORAGE IN A LIBRARY

- For example, if we consider the problem with 2 height classes H_m and H_n with $H_m < H_n$
  - all books of height $\leq H_m$ are shelved in shelf with the height H_m
  - all the other books are shelved on the shelf with height H_n
- The corresponding total costs are

$$\left[s_m + c_m H_m \sum_{j=1}^{m} L_j \right] + \left[s_n + c_n H_n \sum_{j=m+1}^{n} L_j \right]$$

 78

COMPACT BOOK STORAGE IN A LIBRARY

- The problem is to find the set of shelf heights and lengths to *minimize* the *total shelving costs*
- The solution approach is to use a network flow model for a network with
  - the set of $(n+1)$ nodes

    $$\mathcal{N} = \{\, 0, 1, 2, \ldots, n \,\}$$

    corresponding to the n book heights with

    $$1 \leftrightarrow H_1 < H_2 < \ldots < H_n \leftrightarrow n$$

    and the starting node with height 0

COMPACT BOOK STORAGE IN A LIBRARY

- directed arcs (i, j) only if $j > i$ resulting in a total of $\dfrac{n(n+1)}{2}$ arcs
- "distance" d_{ij} on each arc given by

$$d_{ij} = \begin{cases} s_j + c_j H_j \displaystyle\sum_{k=i+1}^{j} L_k & \text{if } \; j > i \\ \infty & \text{otherwise} \end{cases}$$

COMPACT BOOK STORAGE IN A LIBRARY

- For this network, we solve the shortest route problem for the specified "distances" d_{ij}
- Suppose that for a problem with $n = 17$, we determine the optimal trajectory to be

$$\{ (0,7), (7,9), (9,15), (15,17) \}$$

the interpretation of this solution is :

COMPACT BOOK STORAGE IN A LIBRARY

- store all the books of height $\leq H_7$ on the shelf of height H_7
- store all the books of height $\leq H_9$ but $> H_7$ on the shelf of height H_9
- store all the books of height $\leq H_{15}$ but $> H_9$ on the shelf of height H_{15}
- store all the books of height $\leq H_{17}$ but $> H_{15}$ on the shelf of height H_{17}

