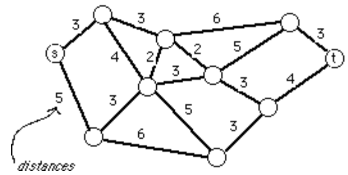
THE SHORTEST PATH PROBLEM

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu

author

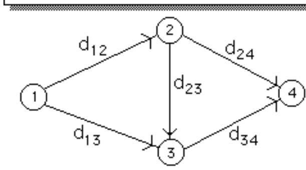
Shortest Path Problem



Find the shortest path in the network from node s to node t

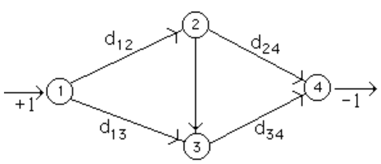
Contents

Formulation of the Shortest-Path Problem as a linear programming problem:



Example: Find the shortest path from node **1** to node **4**, where distances are as given.

Minimum-Cost Network Flow Model:
Let the unit "shipping cost" = d_{ij} for arc (i,j)
source node #1 supplies 1 unit
sink node #4 requires 1 unit



Minimize $d_{12}X_{12}+d_{13}X_{13}+d_{23}X_{23}+d_{24}X_{24}+d_{34}X_{34}$

subject to

LP model

$$\begin{cases} -X_{12} - X_{13} = -1 \\ +X_{12} - X_{23} - X_{24} = 0 \\ +X_{13} + X_{23} - X_{34} = 0 \\ +X_{24} + X_{34} = +1 \end{cases}$$

$X_{ij} \geq 0$ for each arc (i,j)

Minimize $d_{12}X_{12}+d_{13}X_{13}+d_{23}X_{23}+d_{24}X_{24}+d_{34}X_{34}$

subject to

LP model

$$\begin{cases} -X_{12} - X_{13} = -1 \\ +X_{12} - X_{23} - X_{24} = 0 \\ +X_{13} + X_{23} - X_{34} = 0 \\ +X_{24} + X_{34} = +1 \end{cases}$$

$X_{ij} \geq 0$ for each arc (i,j)

Dual LP

Maximize $-y_1 + y_4$

subject to

$$\begin{aligned} -y_1 + y_2 &\leq d_{12} \\ -y_1 + y_3 &\leq d_{13} \\ -y_2 + y_3 &\leq d_{23} \\ -y_2 + y_4 &\leq d_{24} \\ -y_3 + y_4 &\leq d_{34} \end{aligned}$$

(y_i unconstrained in sign)

Dual LP

Maximize $y_4 - y_1$

subject to

$$\begin{aligned} y_2 &\leq y_1 + d_{12} \\ y_3 &\leq y_1 + d_{13} \\ y_3 &\leq y_2 + d_{23} \\ y_4 &\leq y_2 + d_{24} \\ y_4 &\leq y_3 + d_{34} \end{aligned}$$

(y_i unconstrained in sign)

The algorithm which will be presented will find primal variables (X_{ij}) and dual variables (y_i) which are:
- each feasible
- satisfy complementary slackness conditions

(This will guarantee optimality!

Dijkstra's Shortest-Path Algorithm

Assume no negative-length cycles exist in the network.

To find: the shortest path from node **s** to **each** of the other nodes

Label each node j with **two** labels:

d(j) = length of shortest path from node **s** to node j passing through permanently-labelled nodes **only**

p(j) = immediate predecessor to node j in the path from node **s**.

At any stage of the algorithm, the label of each node is either **temporary** or **permanent**

Dijkstra's Shortest-Path Algorithm

Step 0: Initially, give node s permanent labels
$d(s) = 0$ and $p(s) = \varnothing$
and give all other nodes temporary labels
$d(j) = +\infty$ and $p(j) = \varnothing$

Step 1: Let k = node whose labels were most recently made permanent.
For every node j linked to node k and having temporary labels, update the labels:

$d(j) = \text{minimum} \{ d(j), d(k) + d_{kj} \}$
and, if $d(j) = d(k) + d_{kj}$, then $p(j) = k$

Step 2: Make permanent the label of the node having smallest d(j)

If some labels are temporary still, return to step 1; otherwise, stop.

Justification for step 2

Suppose node x has the smallest temporary label.

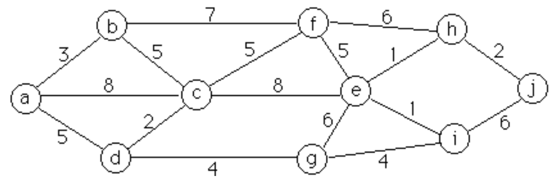
d(x) = shortest length of any path from node s to node x, using only intermediate nodes with permanent labels.

The shortest path from node s to node x which includes some node y with temporary label must be $\geq d(y) + d_{yx} \geq d(x)$

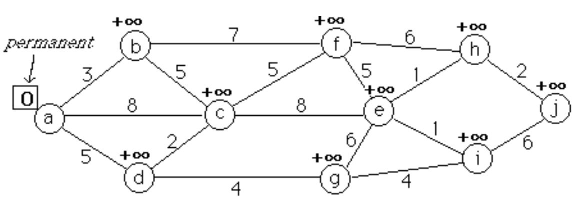
Therefore, we can make the label of node x permanent.

Example

Find the shortest paths from node **a** to all other nodes.

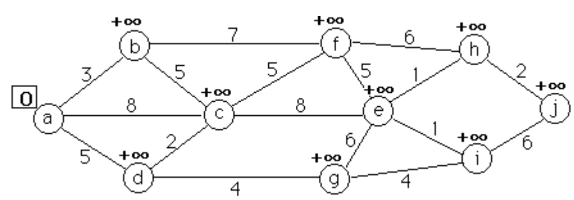


Details Solution



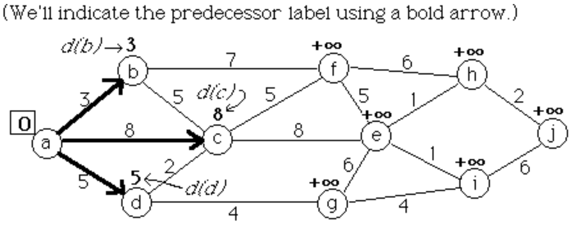
We start by assigning the initial labels (0 for node a, +∞ otherwise)

(A box around p(j) will indicate that the label is permanent.)

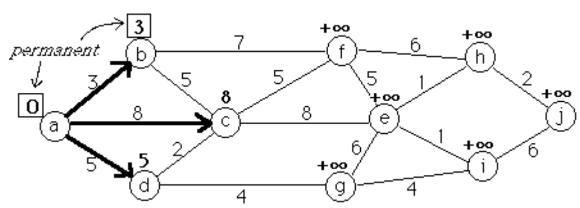


Next, we update the labels on nodes b, c, and d:
$d(b) = \text{minimum} \{ \infty, 0+3 \} = 3$
$d(c) = \text{minimum} \{ \infty, 0+8 \} = 8$
$d(d) = \text{minimum} \{ \infty, 0+5 \} = 5$
In each case, the predecessor label will indicate node a.

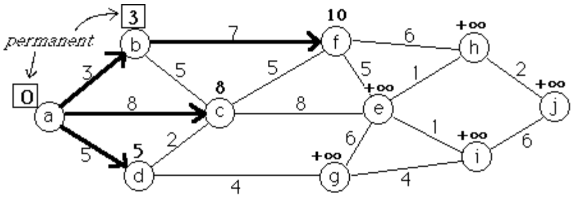
(We'll indicate the predecessor label using a bold arrow.)



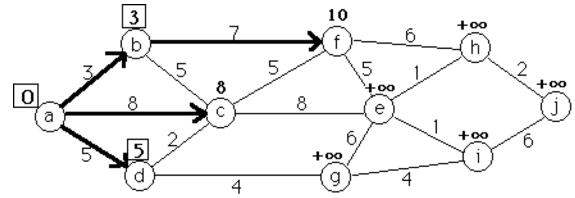
Next we select a temporary label to be made permanent.
This will be the label of node b, since it has the smallest temporary label.



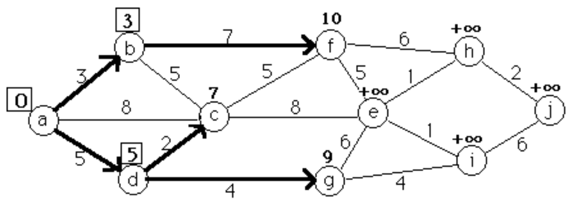
Next, we update the labels of nodes c and f:
$d(c) = \text{minimum} \{8, 3+5\} = 8$ & $d(f) = \text{minimum} \{ \infty, 3+7\} = 10$



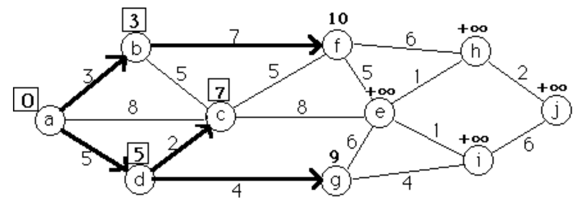
Next step is to choose the temporary label to be made permanent. This will be the label of node d, which is the smallest temporary label.



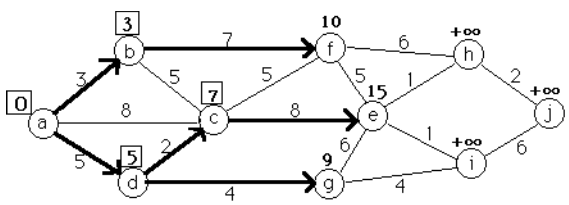
Next we update the temporary labels of the neighbors of node d:
d(c) = minimum {8, 5+2}= 7 & d(g) = minimum { ∞ , 5+4} = 9



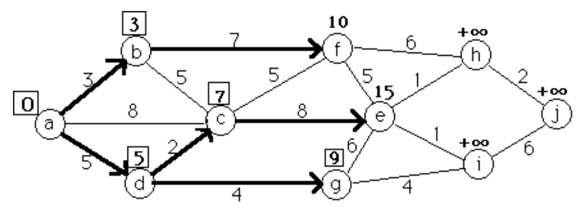
The next temporary label to be made permanent is that of node c.



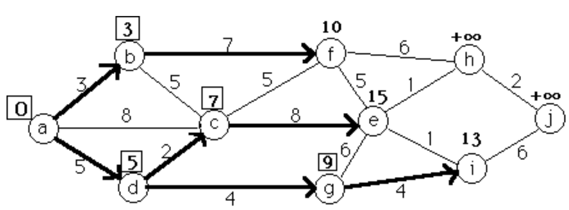
Update the label of nodes e and f:
d(e) = minimum {∞ , 7+8}=15 and d(f) = minimum{10, 7+5}=10



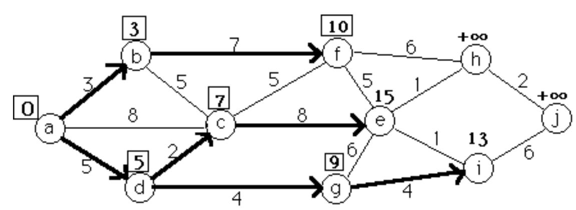
Next, we select the smallest temporary label (that of node g) and make it permanent.



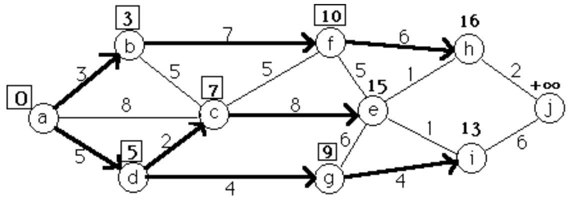
We next update the temporary labels of nodes e and i:
d(e) = minimum {15, 9+6}=15 and d(i) = minimum { ∞, 9+4}=13



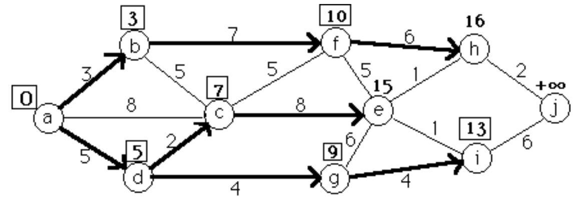
We next make the label of node f permanent.



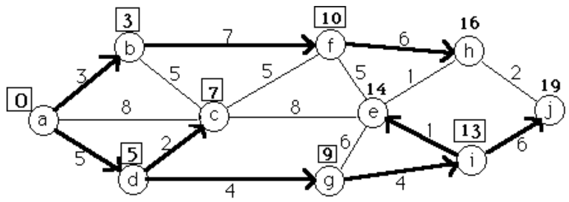
Update the temporary labels of nodes e and h:
d(e) = minimum {15, 10+5}=15 and d(h)=min{ ∞ , 10+6}=16



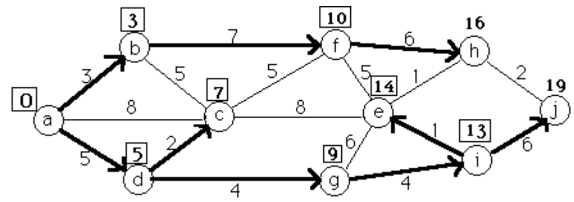
Choose the next temporary label to be made permanent. This will be that of node i.



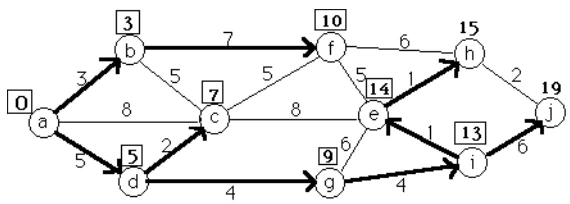
Update the temporary labels of nodes e and j:
d(e) = min{15, 13+1}=14 & d(j)= min{ ∞ , 13+6}=19



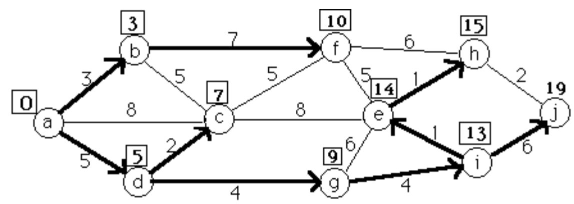
Next, choose the temporary label of node e to be made permanent.



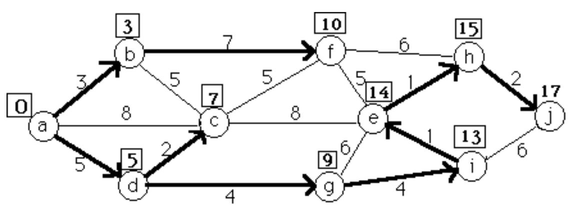
Update the temporary labels of node h:
d(h) = minimum{16, 14+1} = 15



Choose the temporary label of node h to be made permanent.

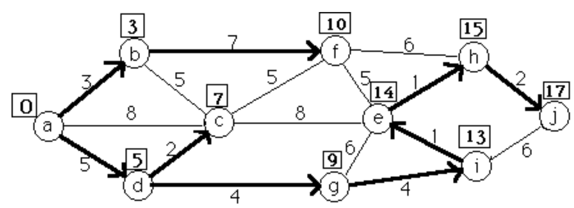


Update the temporary label of node j:
d(j) = minimum{19, 15+2}=17



Finally, we select the label of node j to be made permanent.

Since no temporary labels remain, the algorithm terminates.



The predecessor labels (indicated by the bold arrows above) allow us to "trace back" the shortest path.
For example, the shortest path to node j is: a → d → g → i → e → h → j

The distance labels which we have computed,
i.e., $y_j = d(j)$
are feasible in the dual LP

Dual LP

Maximize $y_4 - y_1$
subject to

$$
\begin{aligned}
y_2 &\le y_1 + d_{12}\\
y_3 &\le y_1 + d_{13}\\
y_3 &\le y_2 + d_{23}\\
y_4 &\le y_2 + d_{24}\\
y_4 &\le y_3 + d_{34}
\end{aligned}
$$

(y_i unconstrained in sign)

If we define $X_{ij} = \begin{cases} 1 & \text{if edge (i,j) is on the shortest path to t} \\ 0 & \text{otherwise} \end{cases}$

then X is feasible in the primal LP

Minimize $d_{12}X_{12}+d_{13}X_{13}+d_{23}X_{23}+d_{24}X_{24}+d_{34}X_{34}$

subject to

$$
\begin{cases}
-X_{12} - X_{13} = -1\\
+X_{12} - X_{23} - X_{24} = 0\\
+X_{13} + X_{23} - X_{34} = 0\\
+X_{24} + X_{34} = +1
\end{cases}
$$

$X_{ij} \ge 0$ for each arc (i,j)

Complementary slackness conditions are satisfied, i.e.,

If $X_{ij} > 0$, then $y_j = y_i + d_{ij}$

Complementary Slackness Theorem then guarantees that

X is optimal in the primal LP
Y is optimal in the dual LP

```
    ∇L←V DIJKSTRA D;P;T;N;TL;PRED;iter
     ⍝
     ⍝  Dijkstra's algorithm for finding the shortest path
     ⍝  from V to all other nodes
     ⍝  (or from V[1] to V[2] if 2=⍴V)
     ⍝  Returns result:
     ⍝         L[1;] = Shortest path lengths
     ⍝         L[2;] = Predecessors on shortest path
     ⍝
     →OK IF ∧/∧/D≥0
     'Error: Distance matrix is not non-negative'
     L←⍳0 ⋄ →0

     OK:T←⍳N←1↑⍴D   ⋄ V←V,0   ⋄ PRED←N⍴0   ⋄ TL←L←N⍴BIG
        L[P←V[1]]←0    ⋄ iter←0

     NEXT:→FINIS IF 0=⍴T←(T≠P)/T
        TL[T]←L[T]⌊,L[P]+D[P;T]
        PRED[(L[T]≠TL[T])/T]←P
        P←T[1↑⍋L[T]←TL[T]]
        →NEXT IF P≠V[2]
     FINIS:L←L,[0.5] PRED
    ∇
```

APL Code for Dijkstra's Algorithm

Identifying the shortest path, given the predecessor labels

```
    ∇PATH←V DIJKSTRA∆PATH PRED;I
     ⍝
     ⍝   Find shortest path to node V,
     ⍝   given predecessor list computed by
     ⍝   the function DIJKSTRA
     ⍝
     PATH←V

     NEXT:V←PRED[1↑V],V
     →NEXT IF 0≠1↑V

     PATH←1↓V
    ∇
```

Floyd's Algorithm for Shortest Paths

Assume no negative-length cycles exist in the network.

To find: the shortest path between **each** pair of nodes

Triangle operation

Given an nxn distance matrix $D = \{d_{ij}\}$, a triangle operation for a fixed node k is:

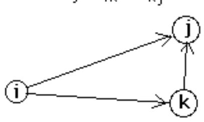
$$d_{ij} \leftarrow \min\{d_{ij}, d_{ik} + d_{kj}\} \text{ for all } i,j=1,\dots n \text{ but } i,j \ne k$$

Theorem If we perform a triangle operation on the distance matrix for successive values of k=1,2, ...n, each entry d_{ij} becomes equal to the length of the shortest path from i to j.

Triangle operation

Given an nxn distance matrix $D = \{d_{ij}\}$, a triangle operation for a fixed node k is:

$$d_{ij} \leftarrow \min\{d_{ij}, d_{ik} + d_{kj}\} \text{ for all } i,j=1,\dots n \text{ but } i,j \ne k$$



Theorem If we perform a triangle operation on the distance matrix for successive values of k=1,2, ...n, each entry d_{ij} becomes equal to the length of the shortest path from i to j.

Justification for Floyd's Algorithm

After the triangle operation for step k_0 is performed,

d_{ij} = length of shortest path from i to j with only intermediate nodes $k \le k_0$

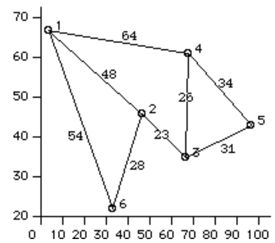
Proof (by induction): Assume true for $k_0 - 1$
Consider triangle operation for k_0:

$$d_{ij} = \text{minimum}\{ d_{ij}, d_{ik_0} + d_{k_0 j} \}$$

If the shortest path from i to j using only intermediate nodes 1, 2, ... k_0 does not pass through k_0, then d_{ij} is unchanged by this operation, and d_{ij} will still satisfy the above property for k_0. Otherwise, $d_{ij} = d_{ik_0} + d_{k_0 j}$ and since d_{ik_0} & $d_{k_0 j}$ each satisfy the property, $d_{ik_0} + d_{k_0 j}$ will satisfy the property.

APL Code for Floyd's Algorithm

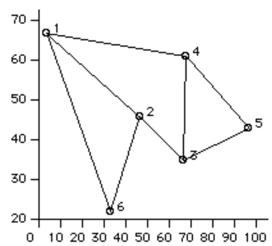
```
     ∇DΔ←FLOYD D;K;EQ;N;P;R
      ⍝
      ⍝        FLOYD'S SHORTEST PATH ALGORITHM
      ⍝              D is distance matrix
      ⍝              P is predecessor matrix
      ⍝              DΔ is matrix of shortest path lengths
      ⍝                   between every pair of nodes
      ⍝
      P←⍉(⍴D)⍴⍳N←1↑⍴D
      K←1

      NEXT:DΔ←D⌊D[;K]∘.+D[K;]
         P←(((⍴D)⍴P[K;])×~EQ)+P×EQ←D=DΔ
         D←DΔ
         P[K;K]←0
      END_LOOP:→NEXT IF N≥K←K+1

      DΔ←DΔ,[0.5]P
     ∇
```



DISTANCE MATRIX

from \ to	1	2	3	4	5	6
1	0	48	999	64	999	54
2	48	0	23	999	999	28
3	999	23	0	26	31	999
4	64	999	26	0	34	999
5	999	999	31	34	0	999
6	54	28	999	999	999	0

FLOYD'S ALGORITHM



Path Lengths

from \ to	1	2	3	4	5	6
1	0	48	71	64	98	54
2	48	0	23	49	54	28
3	71	23	0	26	31	51
4	64	49	26	0	34	77
5	98	54	31	34	0	82
6	54	28	51	77	82	0

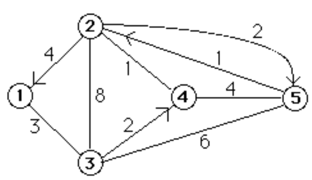
Predecessors

from \ to	1	2	3	4	5	6
1	0	1	2	1	4	1
2	2	0	2	3	3	2
3	2	3	0	3	3	2
4	4	3	4	0	4	2
5	4	3	5	5	0	2
6	6	6	2	3	3	0

Applications

- ➪ Traffic Assignment Problem
- ➪ Equipment Replacement Problem
- ➪ Excavation Planning Problem

TRAFFIC ASSIGNMENT PROBLEM



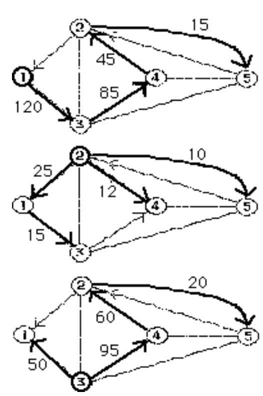
Trip Generation Matrix

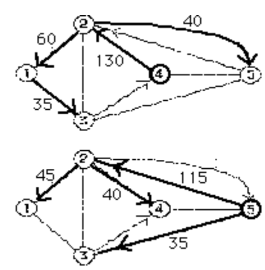
from \ to	1	2	3	4	5
1	--	30	35	40	15
2	10	--	15	12	10
3	50	40	--	35	20
4	25	30	35	--	40
5	45	30	35	40	--

trips/hour (x10)

Assume autos follow the shortest path from origin to destination

Find the flow in the network.

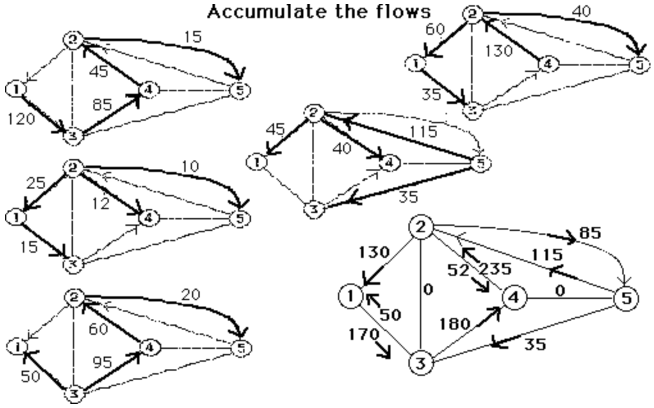




Apply Floyd's algorithm to compute the shortest paths & SD-trees

Assign the flow to each shortest path

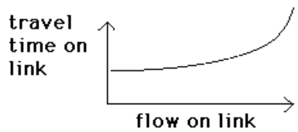
Accumulate the flows



Limitations of the model & algorithm

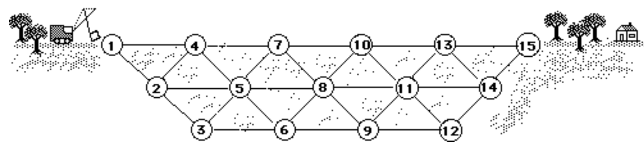
- In reality, shortest distance (or shortest travel time) is not the sole criterion for route selection
- The capacity of any link in a transportation network is finite

- Travel speeds are a function of the amount of congestion on a link, so that, if the criterion is shortest travel time, finding the shortest paths cannot be done independently of computing the traffic flow.

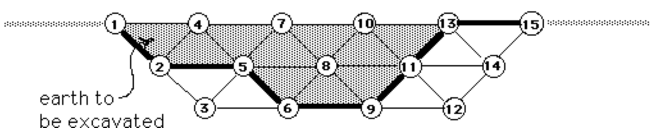


Excavation Planning

An excavation plan for a new open-pit mine is characterized by a continuous path, starting at node #1 and ending at node #15.

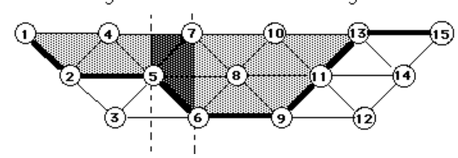


An example of such an excavation plan:



The net profit of such a plan depends upon the depth of the excavation as well as the expected amount of recoverable ore.

For example, if the edge (5,6) is part of the plan, then the associated net profit can be calculated by estimating the recoverable ore in the earth above and subtracting the cost of removing this earth.



Let p_{ij} be the net profit associated with edge (i,j).

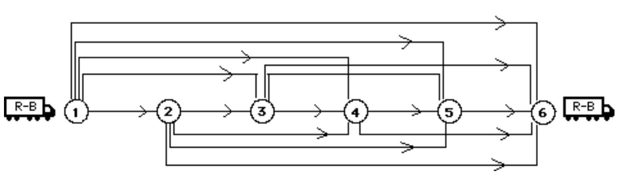
(Not all p_{ij} are positive!)

How can this be modeled as a shortest path problem?

Equipment Replacement

The Rhode-Bloch Trucking Co. is preparing a leasing plan for transportation equipment extending over the next five years. The company can meet its requirement for a truck by leasing a new truck at the beginning of Year **1** and keeping it until the beginning of Year **j** (≤6). If **j**<6, then the company replaces the truck at the beginning of year **j** and keeps it until the beginning of Year **k** (≤6), etc.

The cost of leasing a truck at the beginning of Year **i** until the beginning of Year **j** (denoted c_{ij}) embodies the rental fee plus the expected cost of operating and maintaining the truck.



The shortest path from node #1 to node #6 corresponds to the least-cost schedule for replacement of the truck.