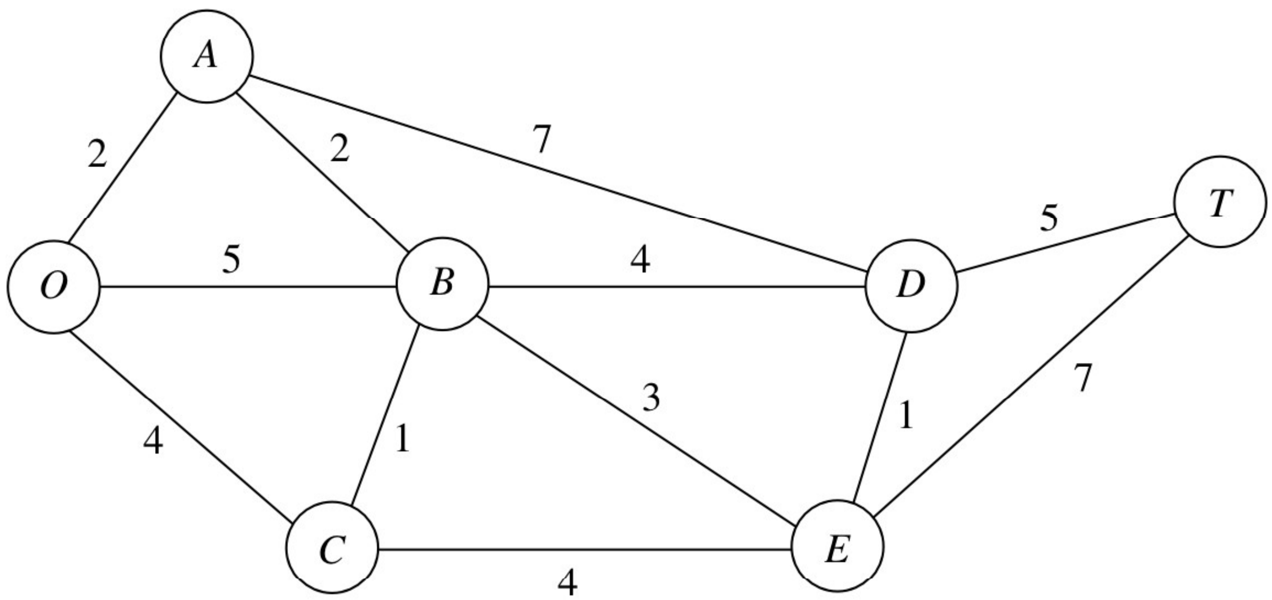
NETWORK OPTIMIZATION MODELS

Network models

- Network representation is widely used in:
  - Production, distribution, project planning, facilities location, resource management, financial planning, etc.
- Transportation, electrical and communication networks pervade our daily lives.
- Algorithms and software are being used to solve huge network problems on a routine basis.
- Many network problems are special cases of linear programming problems.

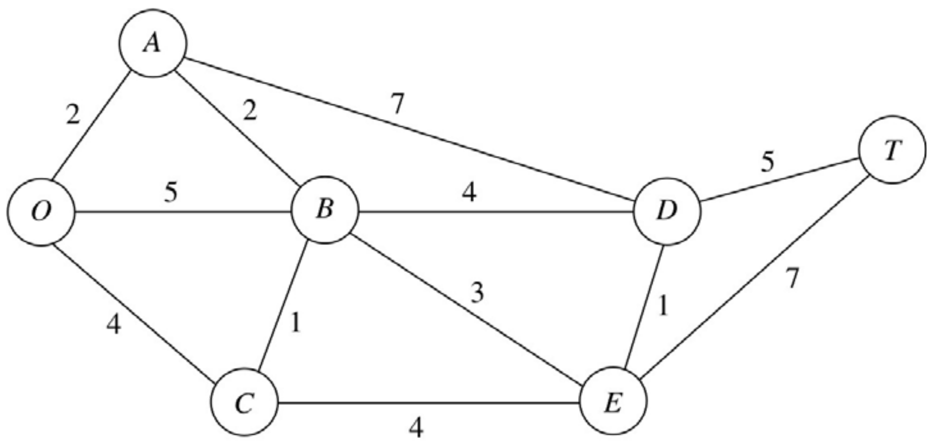
Prototype example

- Seervada Park has a limited amount of sightseeing and backpack hiking.
  - *O*: entrance of the park.
  - *T*: station with scenic wonder.



Park problems

- Determine route from park entrance to station *T* with *smallest total distance* for the operation of trams.
- Telephone lines must be installed under the roads to establish communication among all the stations. This should be accomplished with a *minimum* total number of miles of lines.
- Route the various trips of trams to *maximize* the number of trips per day without violating the limits of any road.



Typical networks

Nodes	**Arcs**	**Flow**
Intersections	Roads	Vehicles
Airports	Air lanes	Aircraft
Switching points	Wires, channels	Messages
Pumping stations	Pipes	Fluids
Work centers	Material-handling sources	Jobs

Terminology of networks

- Network is a set of *points* (**nodes** or vertices) and a set of *lines* (**arcs** or links or edges or branches) connecting certain pairs of the nodes.
  - Example: road system of Seervada Park has 7 nodes and 12 arcs.
- Flow in only one direction is a **directed arc**.
- Flow allowed in either direction: undirected arc or **link**.
- Network with only directed arcs is a **directed network.**

Terminology of networks

- Network with only undirected arcs is an **undirected network**.
- A **path** between two nodes is a sequence of distinct arcs connecting these nodes.
- A **directed path** from node *i* to node *j* is a sequence of connecting arcs whose direction is *toward* node *j*.
- An **undirected path** from node *i* to node *j* is a sequence of connecting arcs whose direction (if any) can be *either* toward or away from node *j*.

Terminology of networks

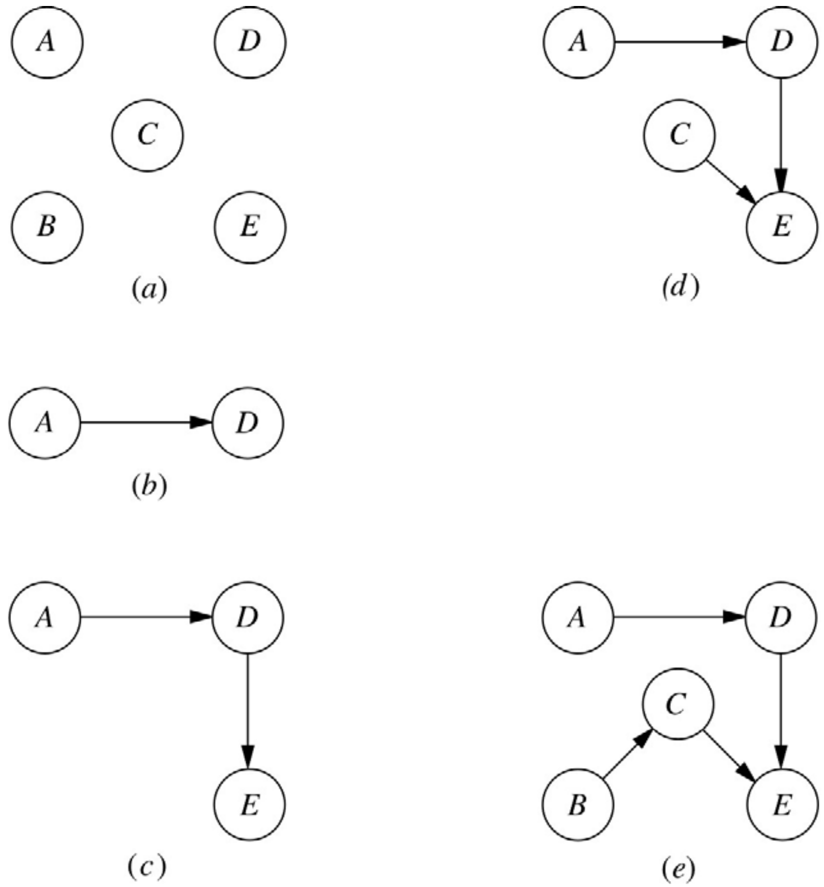
- A path that begins and ends at the same node is a **cycle**.
- Two nodes are **connected** if the network contains at least one undirected path between them.
- A **connected network** is a network where every pair of nodes is connected.
- A **tree** is a *connected network* (for some subset of the n nodes of the original network) that contains *no undirected cycles*.

Terminology of networks

- A **spanning tree** is a *connected network* for all *n* nodes of the original network that contains *no undirected cycles*. A spanning tree has exactly $n - 1$ arcs.
- The maximum amount of flow that can be carried on a directed arc is the **arc capacity**.
- **Supply node**: the flow *out* of the node exceeds the flow *into* the node. The reverse in a **demand node**.
- **Transshipment node**: node that satisfies *conservation of flow*, i.e., flow in equals flow out.

Growing a tree one arc at a time

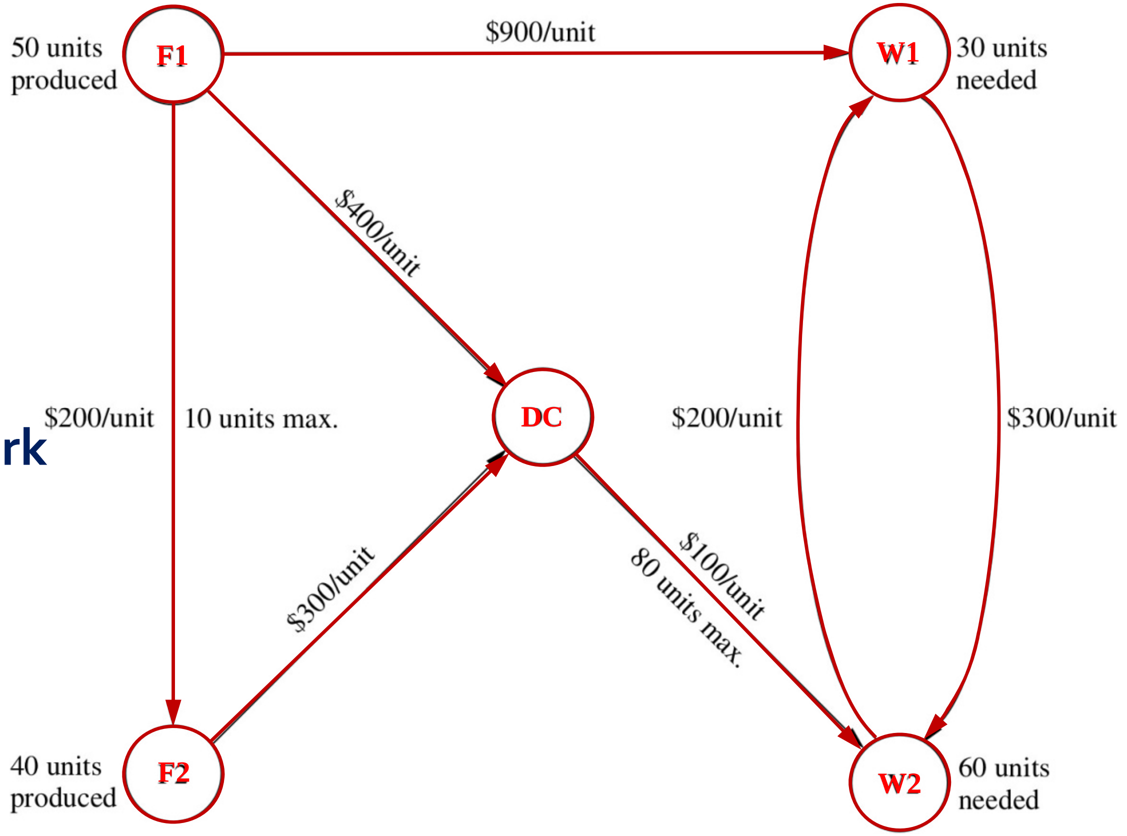
a) Nodes without arcs
b) Tree with one arc
c) Tree with two arcs
d) Tree with three arcs
e) A spanning tree



Example

- Distribution Unlimited Co. produces the same new product at two different factories. Products must be shipped to two warehouses (a distribution center is available).

Example of a directed network

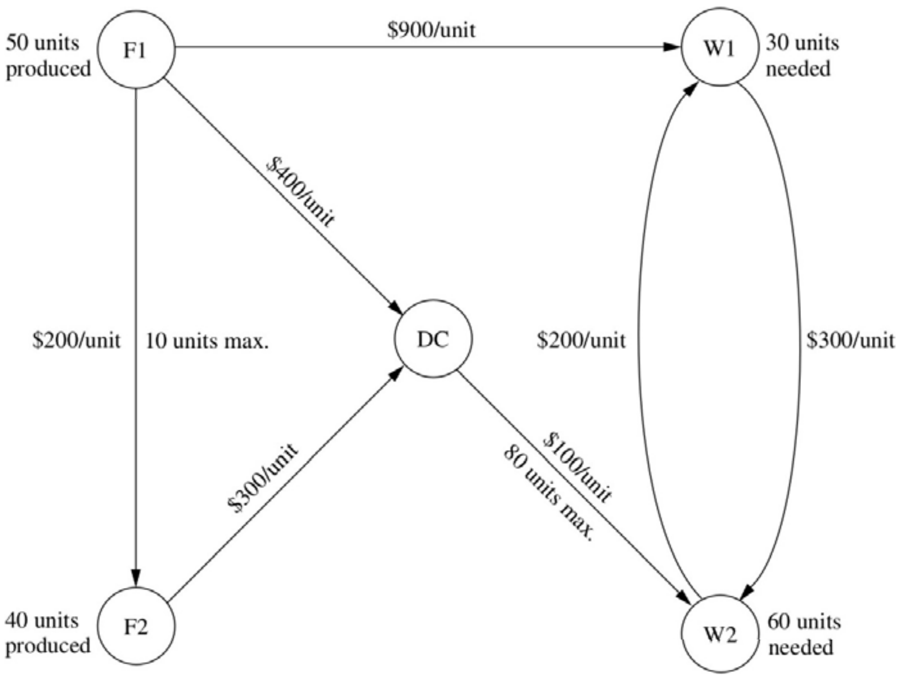


Linear Programming model

$$\text{Minimize } Z = 2x_{F1-F2} + 4x_{F1-DC} + 9x_{F1-W1} + 3x_{F2-DC} + x_{DC-W2} + 3x_{W1-W2} + 2x_{W2-W1}$$

subject to:

$$\begin{aligned}
x_{F1-F2} + x_{F1-DC} + x_{F1-W1} &= 50 \\
-x_{F1-F2} + x_{F2-DC} &= 40 \\
-x_{F1-DC} - x_{F2-DC} + x_{DC-W2} &= 0 \\
-x_{F1-W1} + x_{W1-W2} - x_{W2-W1} &= -30 \\
-x_{DC-W2} - x_{W1-W2} + x_{W2-W1} &= -60 \\
x_{F1-F2} &\leq 10 \\
x_{DC-W2} &\leq 80 \\
x_i &\geq 0, \forall i
\end{aligned}$$



Shortest-path problem

- Consider an *undirected* and *connected* network with the special nodes called *origin* and *destination*.
- Each *link* (undirected arc) has an associated *distance*.
- **Objective**: *find the shortest path from the origin to the destination*.
- **Algorithm**: starting from the origin, successively identify the shortest path to each of the nodes in the ascending order of their distances from the origin.
- The problem is solved when the destination is reached.

Algorithm for shortest-path problem

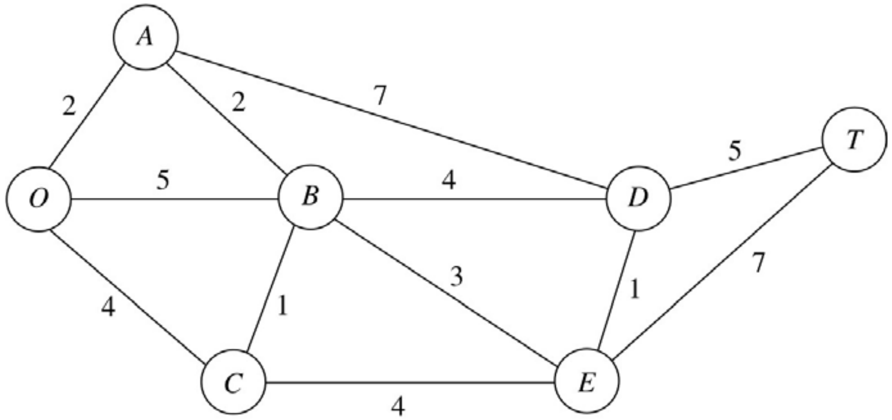
- *Objective of* n*th iteration*: find the *n*th nearest node to the origin (n = 1, 2, ...) until the *n*th nearest node is reached.
- *Input for the* n*th iteration*: $n - 1$ nearest nodes to the origin, including their shortest path and distance to the origin (these are the *solved nodes*).
- *Candidates for the* n*th nearest node*: each solved that is directly connected to unsolved nodes provides *one* candidate – the unsolved node with the *shortest* connecting link.

Algorithm for shortest-path problem

- *Calculation of the* n*th nearest node*: for each solved node and its candidate, add the distance between them to the distance of the shortest path from the origin to this solved node.

  The candidate with the smallest total distance is the *n*th nearest node, and its shortest path is the one generating this distance.

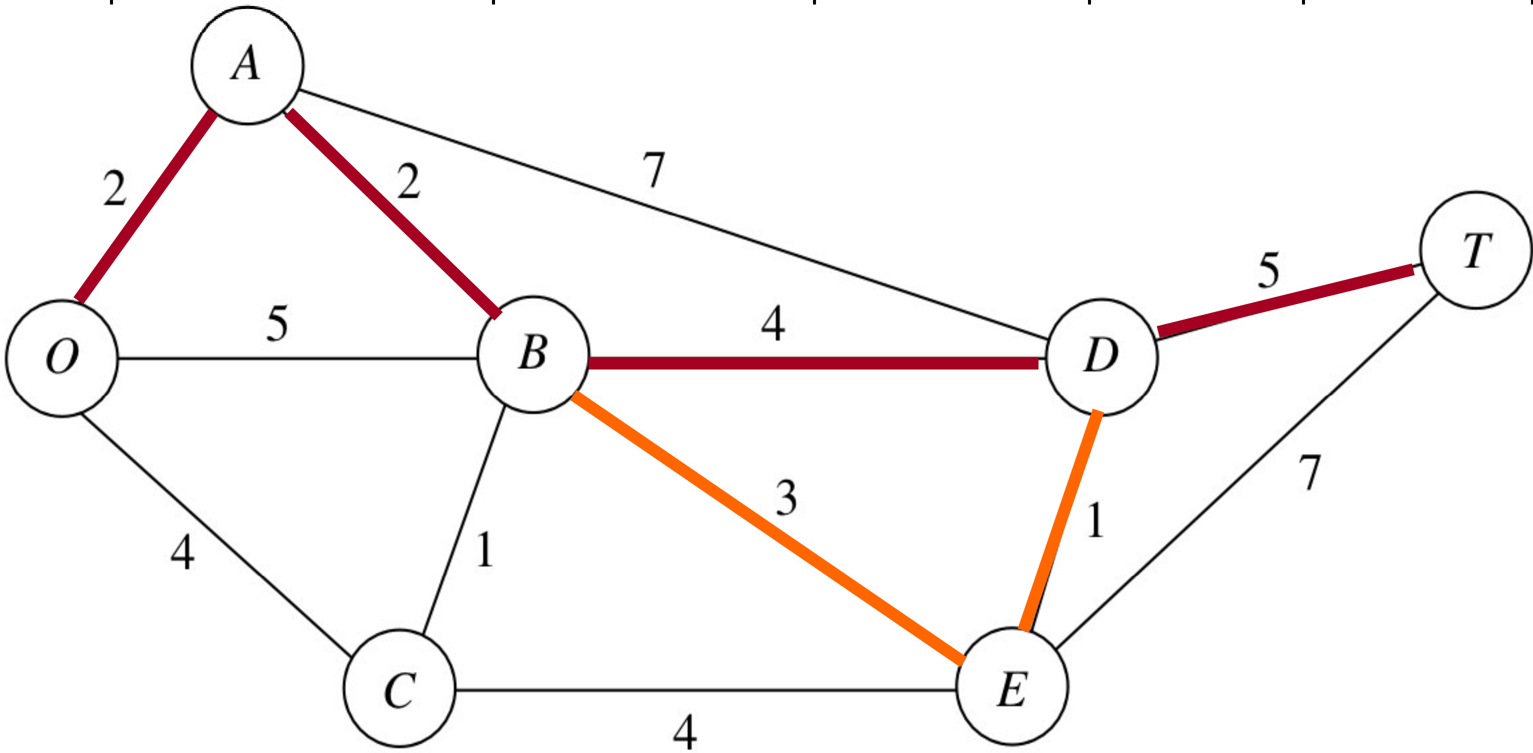
Example: Seervada park



n	Solved nodes directly connected to unsolved nodes	Closest connected unsolved node	Total distance involved	nth nearest node	Minimum distance	Last connection
1	O	A	2	A	2	OA
2,3	O	C	4	C	4	OC
	A	B	$2 + 2 = 4$	B	4	AB
4	A	D	$2 + 7 = 9$	E	7	BE
	B	E	$4 + 3 = 7$			
	C	E	$4 + 4 = 8$			

Example: Seervada park

n	Solved nodes directly connected to unsolved nodes	Closest connected unsolved node	Total distance involved	*n*th nearest node	Minimum distance	Last connection
						OA
						OC *AB*
						BE
						BD *ED*
						DT



Using simplex to solve the problem

Seervada Park Shortest-Path Problem

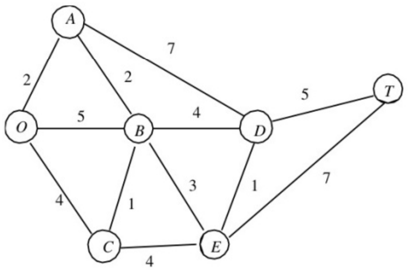
	From	To	On Route	Distance
4	O	A	1	2
5	O	B	0	5
6	O	C	0	4
7	A	B	1	2
8	A	D	0	7
9	B	C	0	1
10	B	D	1	4
11	B	E	0	3
12	C	B	0	1
13	C	E	0	4
14	D	E	0	1
15	D	T	1	5
16	E	D	0	1
17	E	T	0	7
19		Total Distance	13	

Nodes	Net Flow		Supply/Demand
O	1	=	1
A	0	=	0
B	0	=	0
C	0	=	0
D	0	=	0
E	0	=	0
T	-1	=	-1

Range Name	Cells
Distance	F4:F17
From	B4:B17
NetFlow	I4:I10
Nodes	H4:H10
OnRoute	D4:D17
SupplyDemand	K4:K10
To	C4:C17
TotalDistance	D19

Using simplex to solve the problem

	A	B	C	D	E	F	G	H	I	J
1		Seervada Park Shortest-Path Problem								
2										
3		From	To	On Route	Distance		Nodes	Net Flow		Supply/ Demand
4		O	A	1	2		O	1	=	1
5		O	B	0	5		A	0	=	0
6		O	C	0	4		B	0	=	0
7		A	B	1	2		C	0	=	0
8		A	D	0	7		D	0	=	0
9		B	C	0	1		E	0	=	0
10		B	D	0	4		T	-1	=	-1
11		B	E	1	3					
12		C	B	0	1					
13		C	E	0	4					
14		D	E	0	1					
15		D	T	1	5					
16		E	D	1	1					
17		E	T	0	7					
18										
19			Total Distance	13						



Solver Parameters

Set Target Cell: TotalDistan

Equal To: ○ Max ◉ Min ○

By Changing Cells:

OnRoute

Subject to the Constraints:

NetFlow = SupplyDemand

	H
3	Net Flow
4	=SUMIF(From,G4,OnRoute)-SUMIF(To,G4,OnRoute)
5	=SUMIF(From,G5,OnRoute)-SUMIF(To,G5,OnRoute)
6	=SUMIF(From,G6,OnRoute)-SUMIF(To,G6,OnRoute)
7	=SUMIF(From,G7,OnRoute)-SUMIF(To,G7,OnRoute)
8	=SUMIF(From,G8,OnRoute)-SUMIF(To,G8,OnRoute)
9	=SUMIF(From,G9,OnRoute)-SUMIF(To,G9,OnRoute)
10	=SUMIF(From,G10,OnRoute)-SUMIF(To,G10,OnRoute)

Solver Options

☑ Assume Linear Model

☑ Assume Non-Negative

Range Name	Cells
Distance	E4:E17
From	B4:B17
NetFlow	H4:H10
Nodes	G4:G10
OnRoute	D4:D17
SupplyDemand	J4:J10
To	C4:C17
TotalDistance	D19

	C	D
19	Total Distance	=SUMPRODUCT(D4:D17,E4:E17)

Applications of shortest-path

Main applications:

1. Minimize the total *distance* traveled.
2. Minimize the total *cost* of a sequence of activities.
3. Minimize the total *time* of a sequence of activities.
4. Combination of the previous three.

- What happens if *the network is directed*?
- How to optimize from the source to *all* other nodes?
- How to find the shortest path from *every* node to every other node?

Minimum spanning tree problem

- Also for *undirected* and *connected* networks.
- A positive *length* (distance, cost, time, etc.) is associated with each link.
- Both the shortest-path problem and the minimum spanning tree problem choose a set of links that satisfy a certain property.
- **Objective**: *find the shortest total length that provide a path between* each *pair of nodes*.

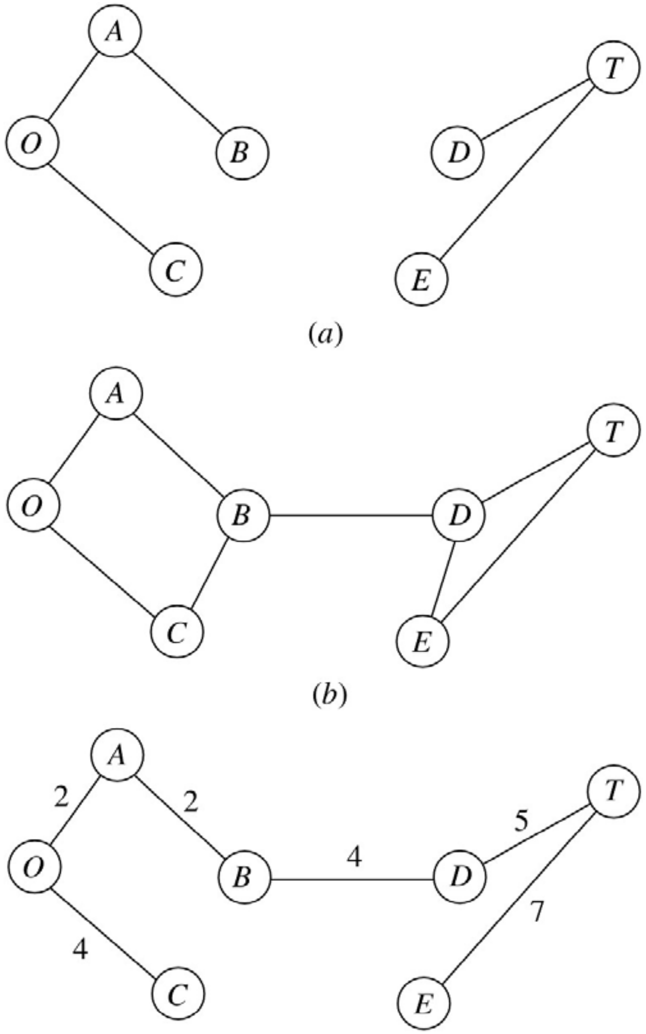
Minimum spanning tree problem

Definition:

1. *Nodes* of the network are given, as well as *potential links* and positive *length* for each if it is inserted in the network.
2. Design network inserting links in order to have a path between *each* pair of nodes.
3. These links must minimize the total length of the links inserted into the network.

Properties

- A network with *n* nodes requires *n* – 1 links to provide a path between each pair of nodes.
- The *n* – 1 links form a *spanning tree*.
- Which are spanning trees?



(*a*)

(*b*)

(*c*)

Applications

- Design of telecommunication networks: fiber-optic, computer, leased-line telephone cable television, etc.
- Design of a lightly used transportation network to minimize the total cost of providing the links.
- Design of a network of high-voltage electrical power transmission lines.
- Design of a network of wiring on electrical equipment to minimize the total length of wire.
- Design of a network of pipelines to connect locations.

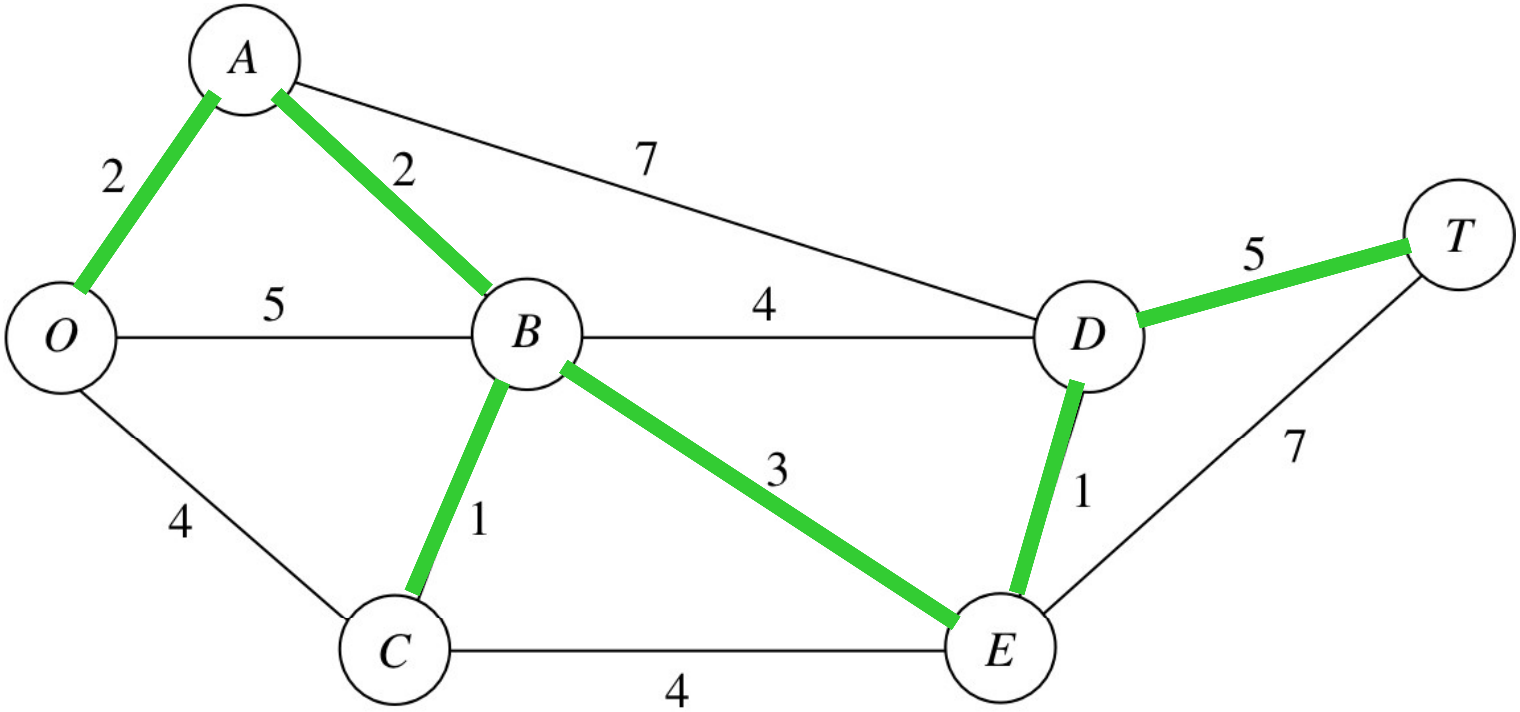
Algorithm for minimum spanning tree

- Can be solved in a straightforward way using a *greedy* algorithm.

1. Select any node arbitrarily, and connect it to the nearest distinct node.
2. Identify the unconnected node that is closest to a connected node, and connect the two nodes. Repeat this step until all nodes have been connected.

- *Tie breaking*: can be done arbitrarily. It can indicate that more than one optimal solution exist.

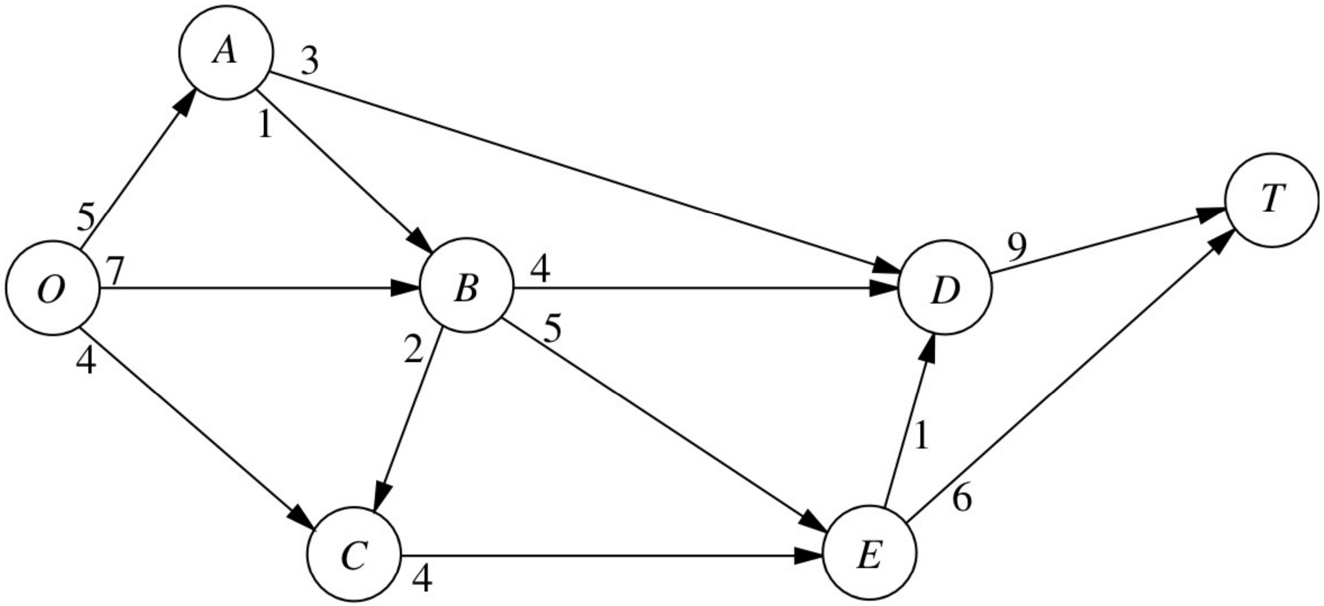
Application to Seervada park



- Total length of the links: 14 Km.
- Verify that the choice of the initial node does not affect the final solution!

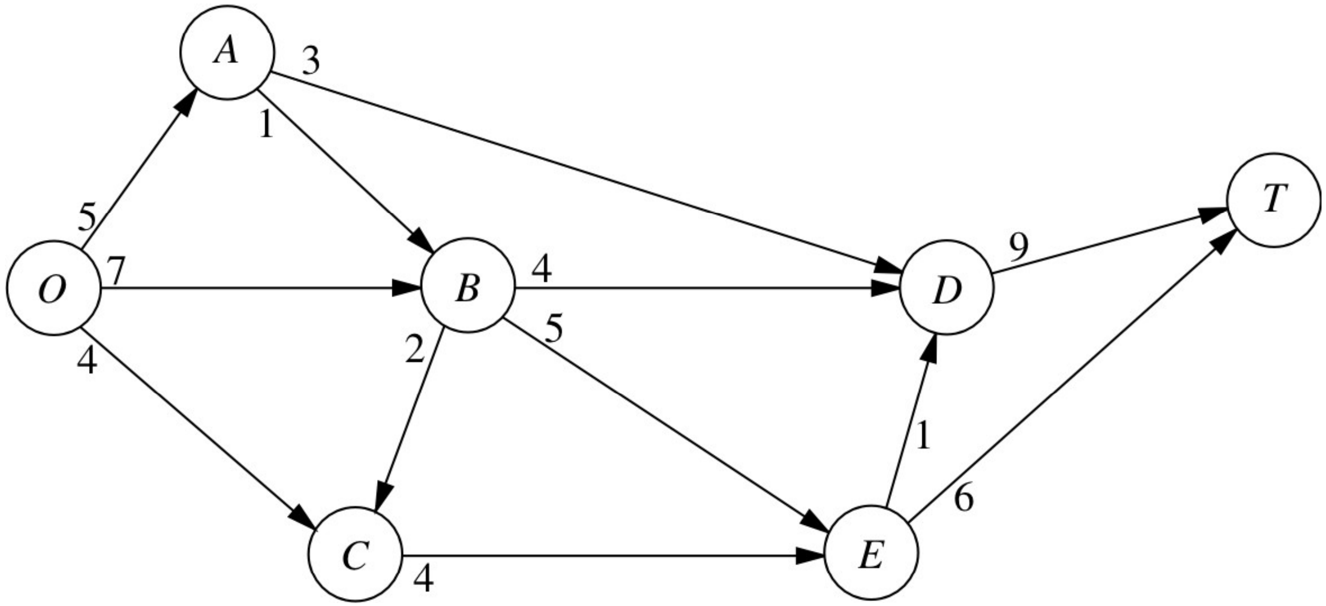
Maximum flow problems

- Third problem in Seervada Park: route the tram trips from the park entrance O to the scenic wonder T *maximizing* the number of trips per day.
- Outgoing trips allowed per day:



Feasible solution

- One solution (not optimal):
  - 5 trams using the route $O \longrightarrow B \longrightarrow E \longrightarrow T$
  - 1 tram using $O \longrightarrow B \longrightarrow C \longrightarrow E \longrightarrow T$
  - 1 tram using $O \longrightarrow B \longrightarrow C \longrightarrow E \longrightarrow D \longrightarrow T$



Definition of maximum flow problem

- All flow through a directed and connected network from the **source** to the **sink**.
- All remaining nodes are *transshipment nodes*.
- Flow through an arc is only allowed in the direction indicated by the arrowhead. Maximum amount of flow is given by the *capacity* of that node.
- ***Objective***: *maximize the total amount of flow from the source to the sink.*

Applications

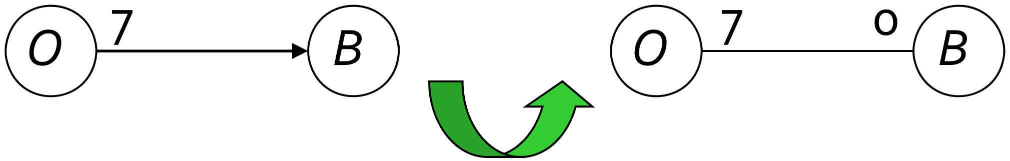
- Maximize the flow through a company's distribution network from its factories to its costumers.
- Maximize the flow through a company's supply network from its vendors (suppliers) to its factories.
- Maximize the flow of oil through a system of pipelines.
- Maximize the flow of water through a system aqueducts.
- Maximize the flow of vehicles through a transportation network.

Some applications

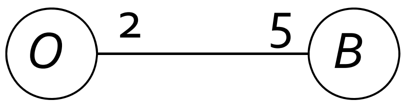
- For some applications, the flow may be originated at more than one node, and may also terminate at more than one node.
  - More than one source: include a *dummy source* with capacity equal to the maximum flow.
  - More than one sink: include a *dummy sink* with capacity equal to the maximum flow.
- Maximum flow problem is a *linear programming problem*; it can be solved by the simplex method.
- However, *augmented path algorithm* is more efficient.

Augmented path algorithm

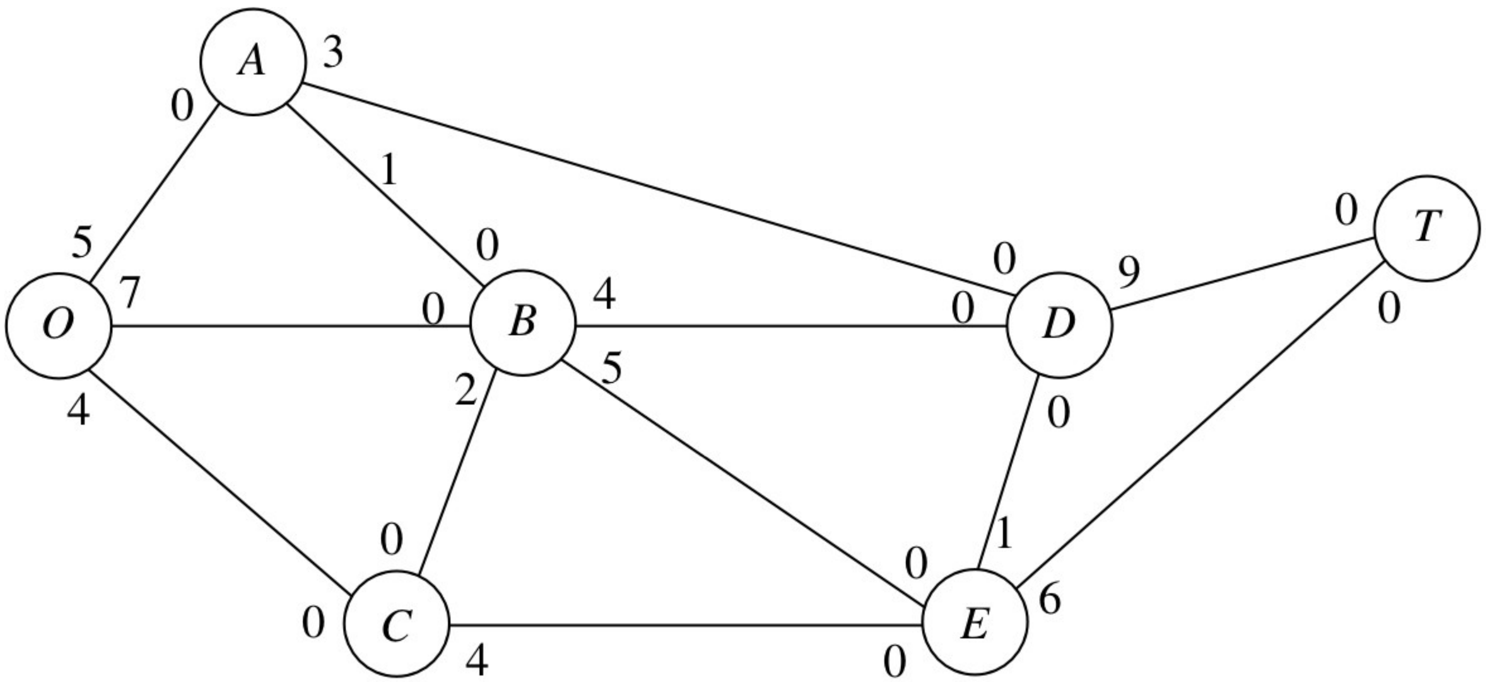
- Every arc is changed from a directed to an undirected arc:



- **Residual network** shows the **residual capacities** for assigning additional flows:



Servada Park residual network



- **Augmenting path** is a directed path from the source to the sink in the residual network, such that *every* arc on this path has *strictly positive* residual capacity.

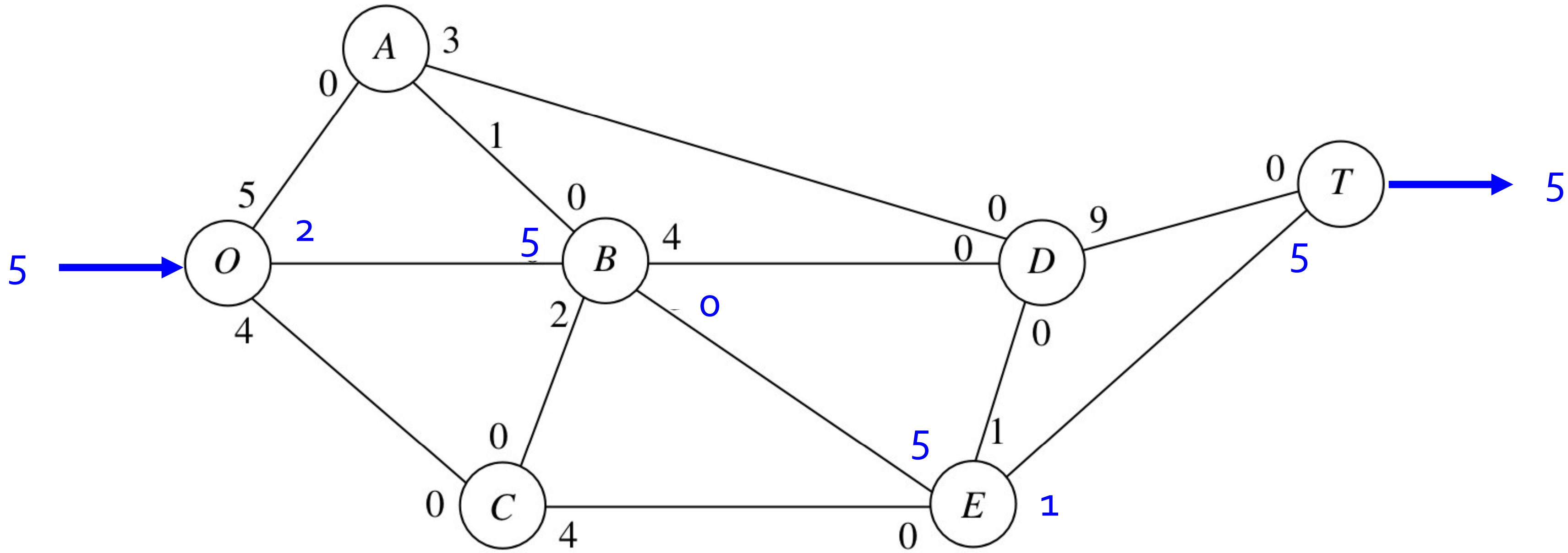
Iteration of augmented path algorithm

1. Identify an augmenting path (directed path from source to sink with positive residual capacity).
2. Identify residual capacity c^* (*minimum* of residual capacities of the arcs on path). *Increase* flow by c^*.
3. *Decrease* by c^* the residual capacity of each arc on this augmented path. *Increase* by c^* the residual capacity of each arc in the opposite direction. Return to Step 1.

- Several augmenting paths can be chosen. Its choice is important for the efficiency of large-scale networks.

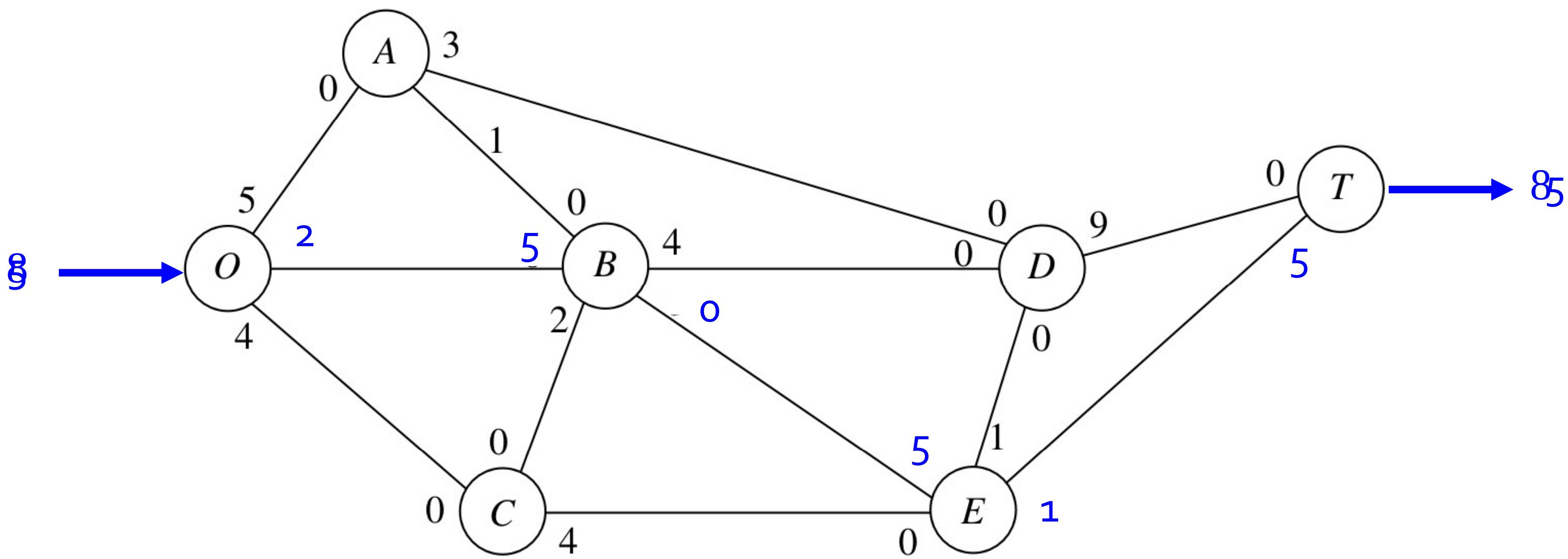
Application to Seervada Park

- *Iteration 1*: one of several augmenting paths is $O \rightarrow B \rightarrow E \rightarrow T$ residual capacity is min{7, 5, 6} = 5.



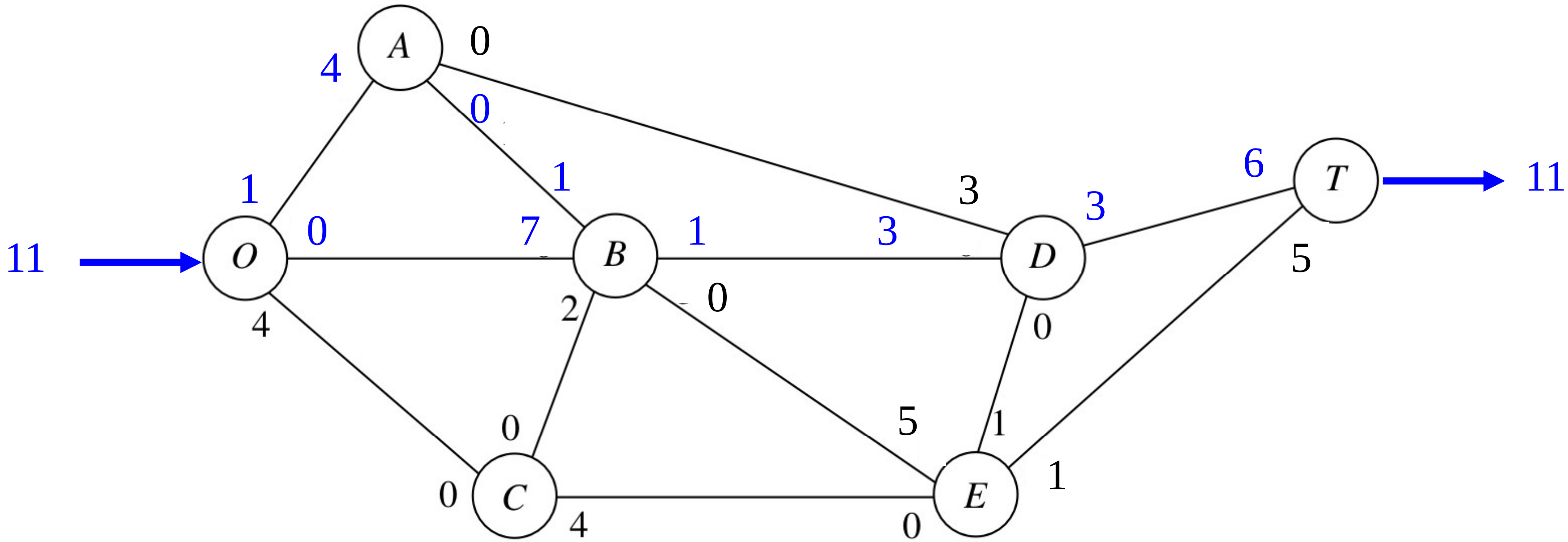
Iteration 2

- *Iteration 2*: assign a flow of 3 to the augmenting path $O \rightarrow A \rightarrow D \rightarrow T$



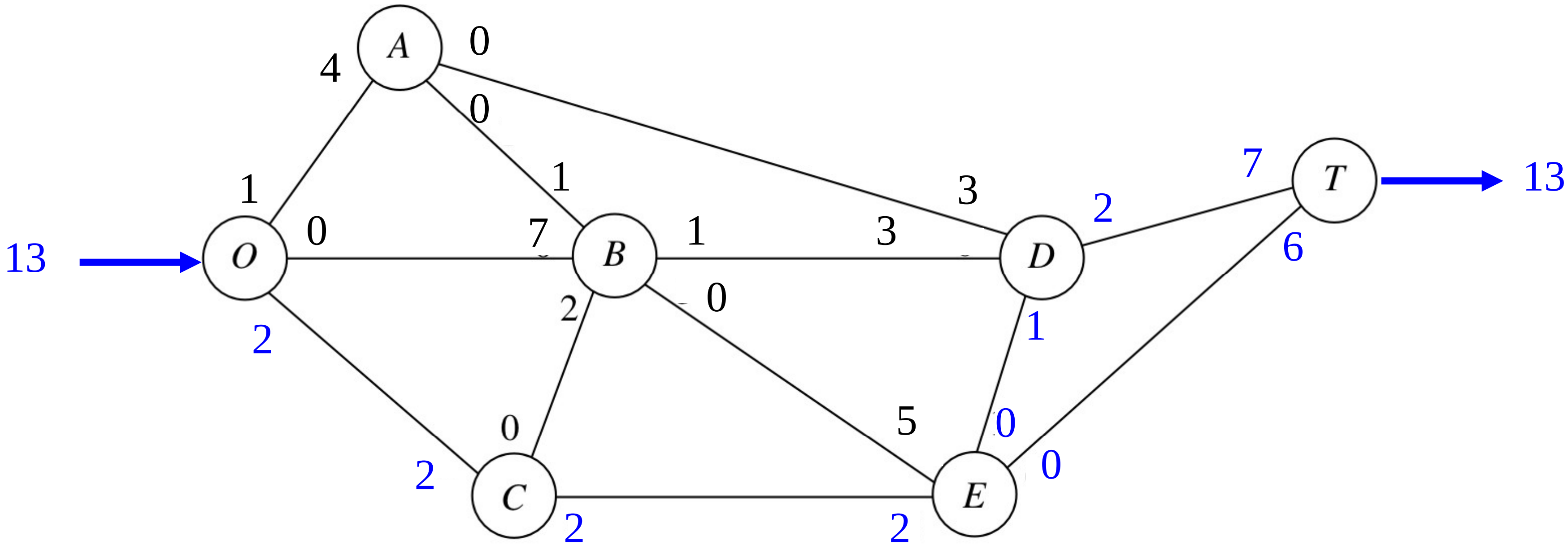
Iterations 3 and 4

- *Iteration 3*: assign flow of 1 to augmenting path $O \rightarrow A \rightarrow B \rightarrow D \rightarrow T$
- *Iteration 4*: assign flow of 2 to augmenting path $O \rightarrow B \rightarrow D \rightarrow T$



Iterations 5 and 6

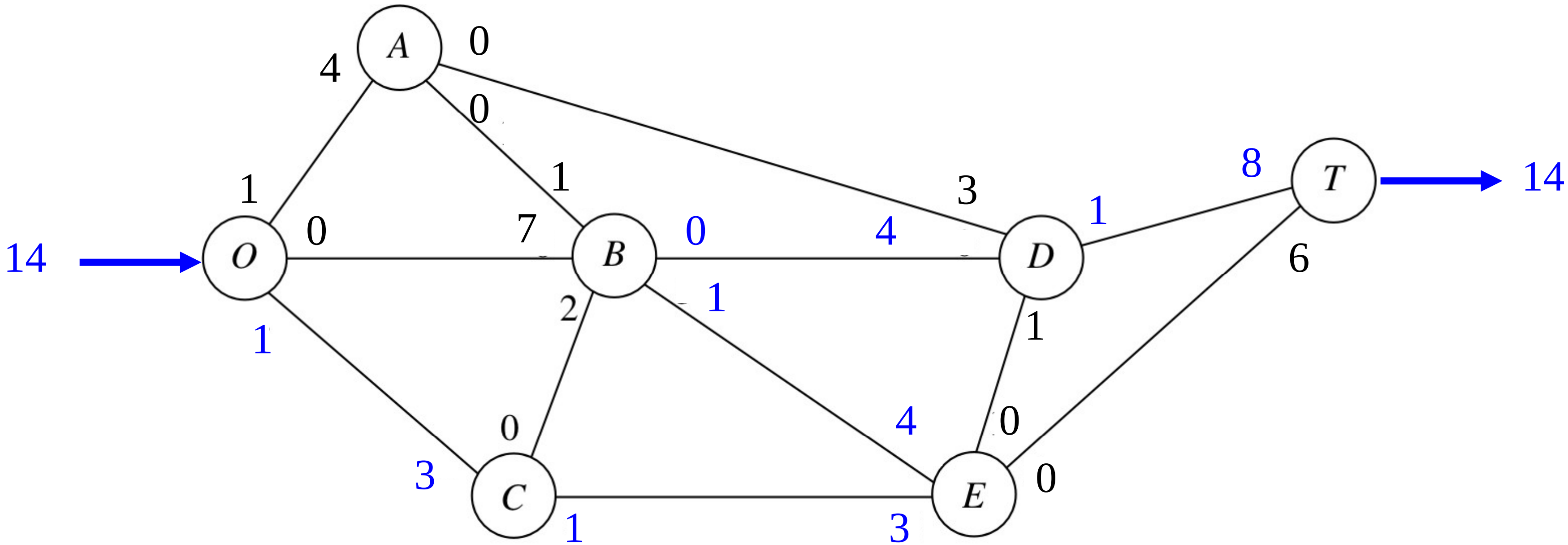
- *Iteration 5*: assign flow of 1 to augmenting path $O \rightarrow C \rightarrow E \rightarrow D \rightarrow T$
- *Iteration 6*: assign flow of 1 to augmenting path $O \rightarrow C \rightarrow E \rightarrow T$



Iteration 7

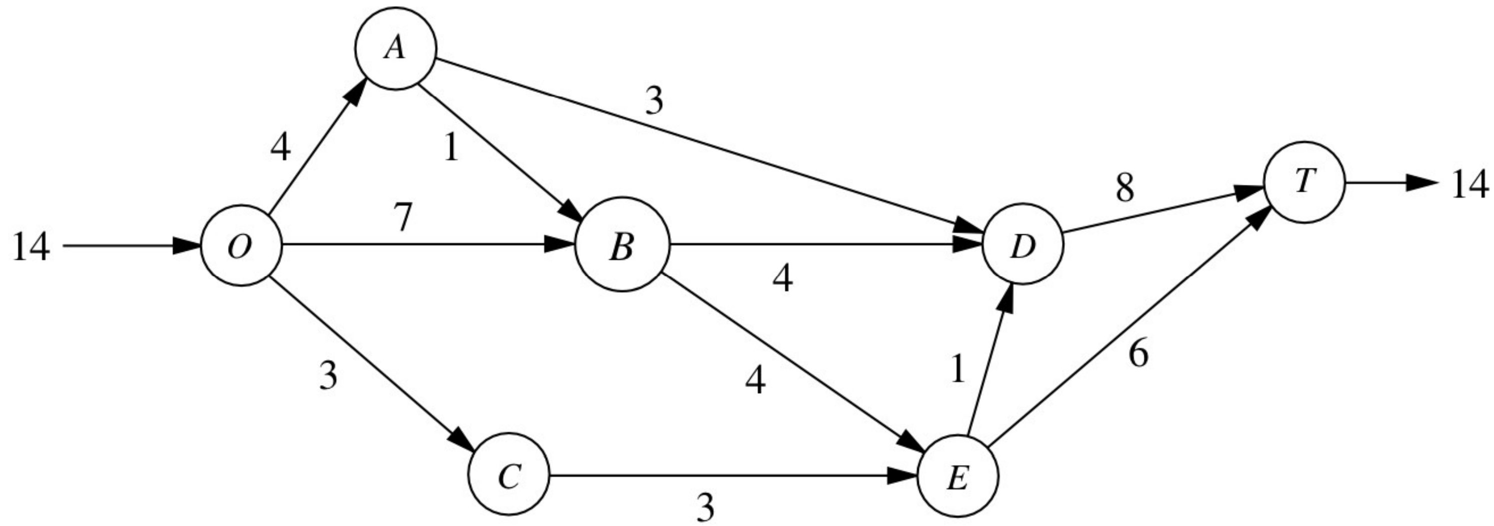
- *Iteration 7*: assign flow of 1 to augmenting path

  $O \to C \to E \to B \to D \to T$



Optimal solution

- After iteration 7 there are no more augmenting paths. Optimal flow pattern is:



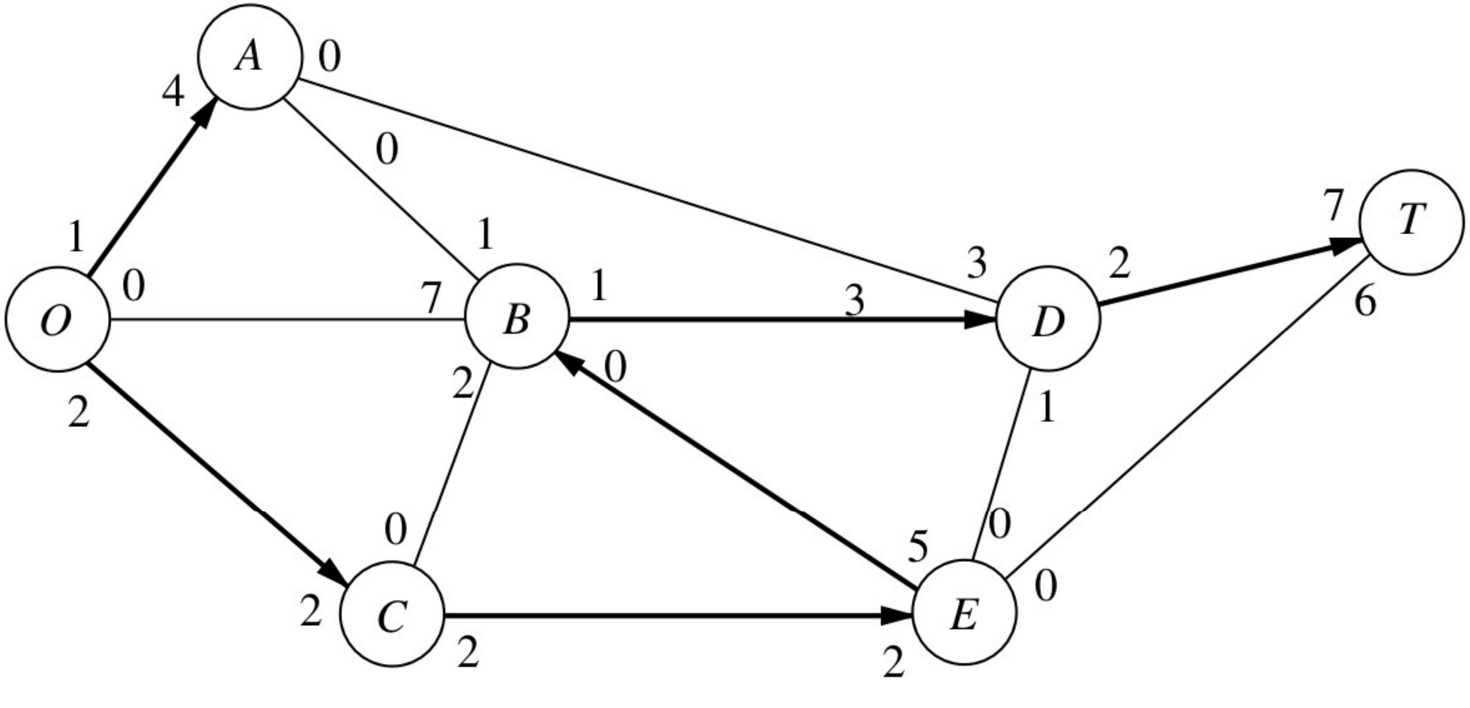
- The flow assignment of 1 for $O \rightarrow C \rightarrow E \rightarrow B \rightarrow D \rightarrow T$ cancels 1 unit of flow assigned at iteration 1 ($O \rightarrow B \rightarrow E \rightarrow$ T) and replaces it by assignments of 1 unit to both $O \rightarrow B \rightarrow D \rightarrow$ T and $O \rightarrow C \rightarrow E \rightarrow T$.

Finding an augmenting path

- This can be difficult, especially for *large* networks.
- **Procedure**:
  - Find all nodes that can be reached from the source along a single arc with strictly positive residual capacity.
  - For each reached node, find all *new* nodes from this node that can be reached along an arc with strictly positive residual capacity.
  - Repeat this successively with the new nodes as they are reached.
  - Identification of a tree of all nodes reached from the source along a path with strictly positive residual flow capacity.

Example in Seervada Park

❖ Residual network after Iteration 6 is given, as well as the possible augmenting path.

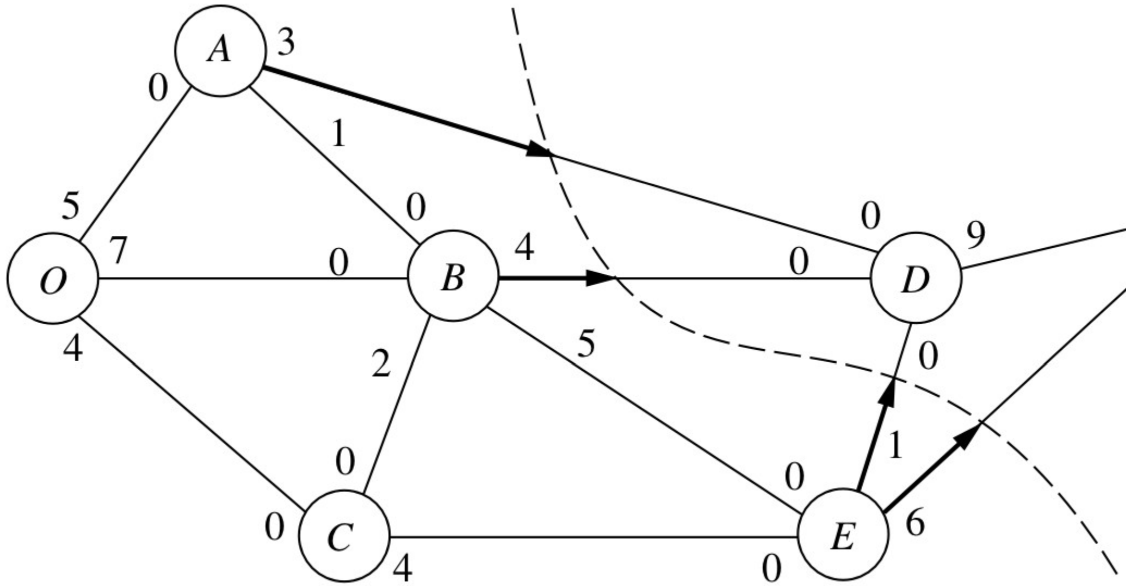


How to recognize the optimal?

- Using the *maximum-flow min-cut theorem*.
- **Cut:** any set of directed arcs containing at least one arc from every directed path from the source to the sink.
- **Cut value:** sum of the arc capacities of the arcs (in the specified direction) of the cut.
- **Maximum-flow min-cut theorem:** for any network with a single source and sink, the *maximum feasible flow* from the source to the sink *equals* the *minimum cut value* for all cuts of the network.

Maximum-flow min-cut theorem

- *F* is the amount of flow from the source to the sink for any feasible flow pattern.
- **Example**: value of cut is 3 + 4 + 1 + 6 = 14. This is the maximum value of *F*, so this is the minimum cut.



Using simplex to solve the problem

Seervada Park Maximum Flow Problem

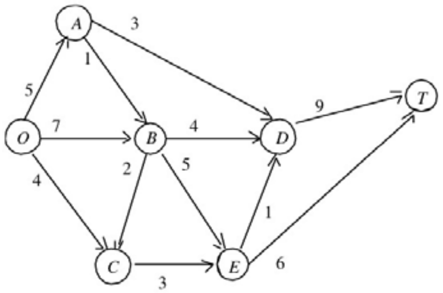
From	To	Flow		Capacity
O	A	3	<=	5
O	B	7	<=	7
O	C	4	<=	4
A	B	0	<=	1
A	D	3	<=	3
B	C	0	<=	2
B	D	4	<=	4
B	E	3	<=	5
C	E	4	<=	4
D	T	8	<=	9
E	D	1	<=	1
E	T	6	<=	6

Maximum Flow: 14

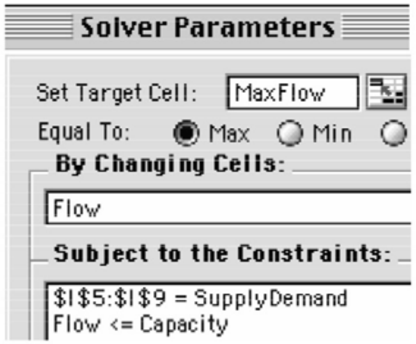
Nodes	Net Flow		Supply/Demand
O	14		
A	0	=	0
B	0	=	0
C	0	=	0
D	0	=	0
E	0	=	0
T	-14		

Range Name	Cells
Capacity	F4:F15
Flow	D4:D15
From	B4:B15
MaximumFlow	D17
NetFlow	I4:I10
Nodes	H4:H10
SupplyDemand	K5:K9
To	C4:C15

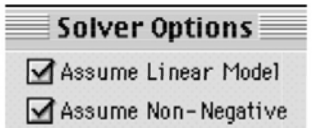
Using simplex to solve the problem



	A	B	C	D	E	F	G	H	I	J	K
1		Seervada Park Maximum Flow Problem									
2											
3		From	To	Flow		Capacity		Nodes	Net Flow		Supply/ Demand
4		O	A	4	≤	5		O	14		
5		O	B	7	≤	7		A	0	=	0
6		O	C	3	≤	4		B	0	=	0
7		A	B	1	≤	1		C	0	=	0
8		A	D	3	≤	3		D	0	=	0
9		B	C	0	≤	2		E	0	=	0
10		B	D	4	≤	4		T	-14		
11		B	E	4	≤	5					
12		C	E	3	≤	4					
13		D	T	8	≤	9					
14		E	D	1	≤	1					
15		E	T	6	≤	6					
16											
17			Maximum Flow	14							



	I
3	Net Flow
4	=SUMIF(From,H4,Flow)-SUMIF (To,H4,Flow)
5	=SUMIF(From,H5,Flow)-SUMIF(To,H5,Flow)
6	=SUMIF(From,H6,Flow)-SUMIF(To,H6,Flow)
7	=SUMIF(From,H7,Flow)-SUMIF(To,H7,Flow)
8	=SUMIF(From ,H8,Flow)-SUMIF(To,H8,Flow)
9	=SUMIF(From,H9,Flow)-SUMIF(To,H9,Flow)
10	=SUMIF(From,H10,Flow)-SUMIF(To,H10,Flow)



	C	D
17	Maximum Flow	=I4

Range Name	Cells
Capacity	F4:F15
Flow	D4:D15
From	B4:B15
MaxFlow	D17
NetFlow	I4:I10
Nodes	H4:H10
SupplyDemand	K5:K9
To	C4:C15

Minimum cost flow problem

- It contains a large number of applications and it can be solved extremely efficiently.
  - Like the *maximum flow problem*, it considers flow through a network with limited arc capacities.
  - Like the *shortest-path problem*, it considers a cost (or distance) for flow through an arc.
  - Like the *transportation problem* or *assignment problem*, it can consider multiple sources (supply nodes) and multiple destinations (demand nodes) for the flow, again with associated costs.

Minimum cost flow problem

- The four previous problems are all special cases of the **minimum cost flow problem**.
- This problem can be formulated as a linear programming problem, solved using a streamlined version of the simplex method:

The **network simplex method**.

Definition of minimum cost flow problem

1. The network is a *directed* and *connected* network.
2. *At least one* of the nodes is a *supply node*.
3. *At least one* of the other nodes is a *demand node*.
4. All the remaining nodes are *transshipment nodes*.
5. Flow through an arc is in the direction of the arrow. Maximum amount of flow given by the *capacity* of the arc.
6. Network has enough arcs with sufficient capacity such that all flow generated at supply nodes reaches the demand nodes.
7. Cost of flow through each arc is proportional to the amount of that flow.

- ***Objective***: *minimize (maximize) the total cost (profit) of sending the available supply through the network to satisfy the demand.*

Applications

Kind of application	Supply nodes	Transshipment nodes	Demand nodes
Operation of a distributed network	Sources of goods	Intermediate storage facilities	Customers
Solid waste management	Sources of solid waste	Processing facilities	Landfill locations
Operation of a supply network	Vendors	Intermediate warehouses	Processing facilities
Coordinating product mixes at plants	Plants	Production of a specific product	Market for a specific product
Cash flow management	Sources of cash at a specific time	Short-term investment options	Needs for cash at a specific time

Formulation of the model

- Consider directed network where n nodes include at least one supply node and one demand node.
- Decision variables:
  - x_{ij} = flow through arc $i \rightarrow j$
- Given information:
  - c_{ij} = cost per unit flow through arc $i \rightarrow j$
  - u_{ij} = arc capacity for arc $i \rightarrow j$
  - b_i = net flow generated at node i
- Value of b_i depends on nature of node i:
  - $b_i > 0$ if node i is a supply node
  - $b_i < 0$ if node i is a demand node
  - $b_i = 0$ if node i is a transshipment node

Formulation of the model

Minimize $\quad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$

subject to

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i, \quad \text{for each node } i,$$

Node constraints

and $\quad 0 \leq x_{ij} \leq u_{ij}, \quad$ for each arc $i \rightarrow j$

Minimum cost flow problem

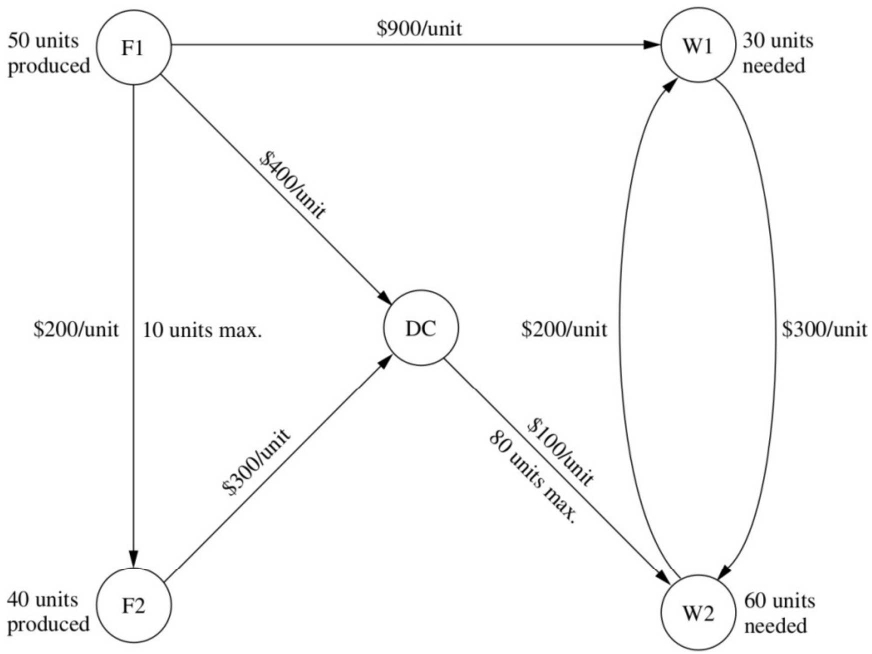
- **Feasible solutions property:** necessary condition for a minimum cost flow problem to have any feasible solutions:

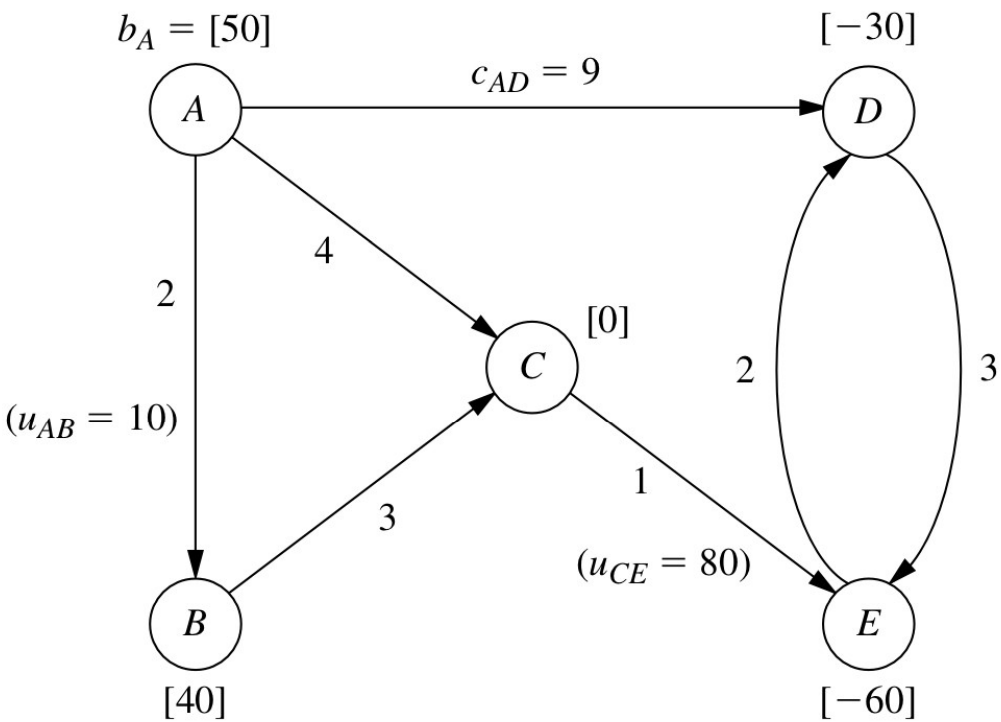
$$\sum_{i=1}^{n} b_i = 0$$

- If this condition does not hold, a dummy supply node or a dummy demand node is needed (as in the transportation problem).
- **Integer solutions property:** when every b_i and u_{ij} have integer values, all basic variables in *every* basic feasible (BF) solution also have integer values.

Example

- *Distribution network* for the Distribution Unlimited Co.





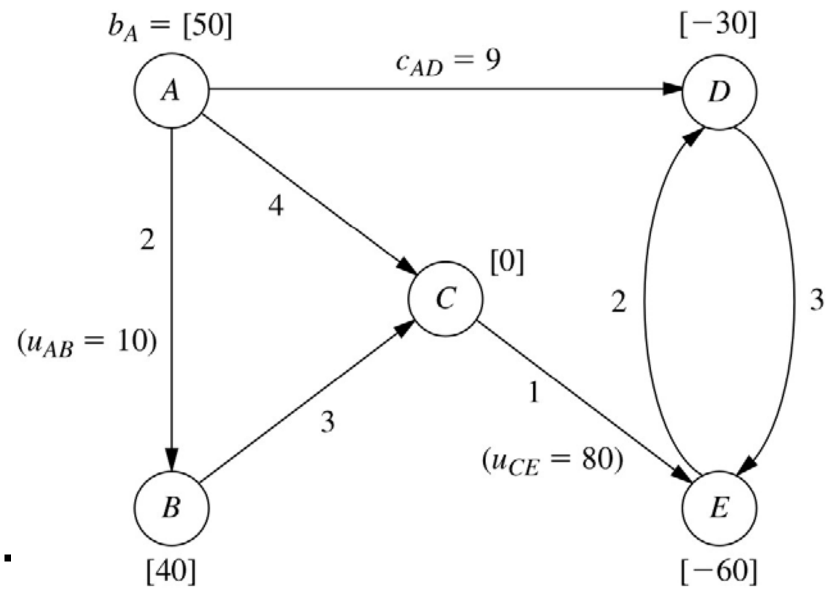
Example

- Linear programming problem:

$$\text{Minimize} \quad Z = 2x_{AB} + 4x_{AC} + 9x_{AD} + 3x_{BC} + x_{CE} + 3x_{DE} + 2x_{ED},$$

subject to

$$\begin{aligned} x_{AB} + x_{AC} + x_{AD} &= 50 \\ -x_{AB} + x_{BC} &= 40 \\ -x_{AC} - x_{BC} + x_{CE} &= 0 \\ -x_{AD} + x_{DE} - x_{ED} &= -30 \\ -x_{CE} - x_{DE} + x_{ED} &= -60 \end{aligned}$$



$$\text{and} \quad x_{AB} \le 10, \quad x_{CE} \le 80, \quad \text{all } x_{ij} \ge 0.$$

Using simplex to solve the problem

Distribution Unlimited Co. Minimum Cost Flow Problem

From	To	Ship		Capacity	Unit Cost	Nodes	Net Flow		Supply/Demand
A	B	0	<=	10	2	A	50	=	50
A	C	40			4	B	40	=	40
A	D	10			9	C	0	=	0
B	C	40			3	D	-30	=	-30
C	E	80	<=	80	1	E	-60	=	-60
D	E	0			3				
E	D	20			2				
	Total Cost	490							

Range Name	Cells
From	B4:B10
NetFlow	J4:J8
Nodes	I4:I8
Ship	D4:D10
SupplyDemand	L4:L8
To	C4:C10
TotalCost	D12
UnitCost	G4:G10

Using simplex to solve the problem

	A	B	C	D	E	F	G	H	I	J	K	L
1	Distribution Unlimited Co. Minimum Cost Flow Problem											
2												
3		From	To	Ship		Capacity	Unit Cost		Nodes	Net Flow		Supply/ Demand
4		A	B	0	≤	10	2		A	50	=	50
5		A	C	40			4		B	40	=	40
6		A	D	10			9		C	0	=	0
7		B	C	40			3		D	-30	=	-30
8		C	E	80	≤	80	1		E	-60	=	-60
9		D	E	0			3					
10		E	D	20			2					
11												
12		Total Cost		490								

Solver Parameters

Set Target Cell: TotalCost

Equal To: Max Min

By Changing Cells:

Ship

Subject to the Constraints:

D4 <= F4
D8 <= F8
NetFlow = SupplyDemand

Range Name	Cells
Capacity	F4:F10
From	B4:B10
NetFlow	J4:J8
Nodes	I4:I8
Ship	D4:D10
SupplyDemand	L4:L8
To	C4:C10
TotalCost	D12
UnitCost	G4:G10

	J
3	Net Flow
4	=SUMIF(From,I4,Ship)-SUMIF(To,I4,Ship)
5	=SUMIF(From,I5,Ship)-SUMIF(To,I5,Ship)
6	=SUMIF(From,I6,Ship)-SUMIF(To,I6,Ship)
7	=SUMIF(From,I7,Ship)-SUMIF(To,I7,Ship)
8	=SUMIF(From,I8,Ship)-SUMIF(To,I8,Ship)

Solver Options

☑ Assume Linear Model
☑ Assume Non-Negative

	C	D
12	Total Cost	=SUMPRODUCT(D4:D10,G4:G10)

Special cases

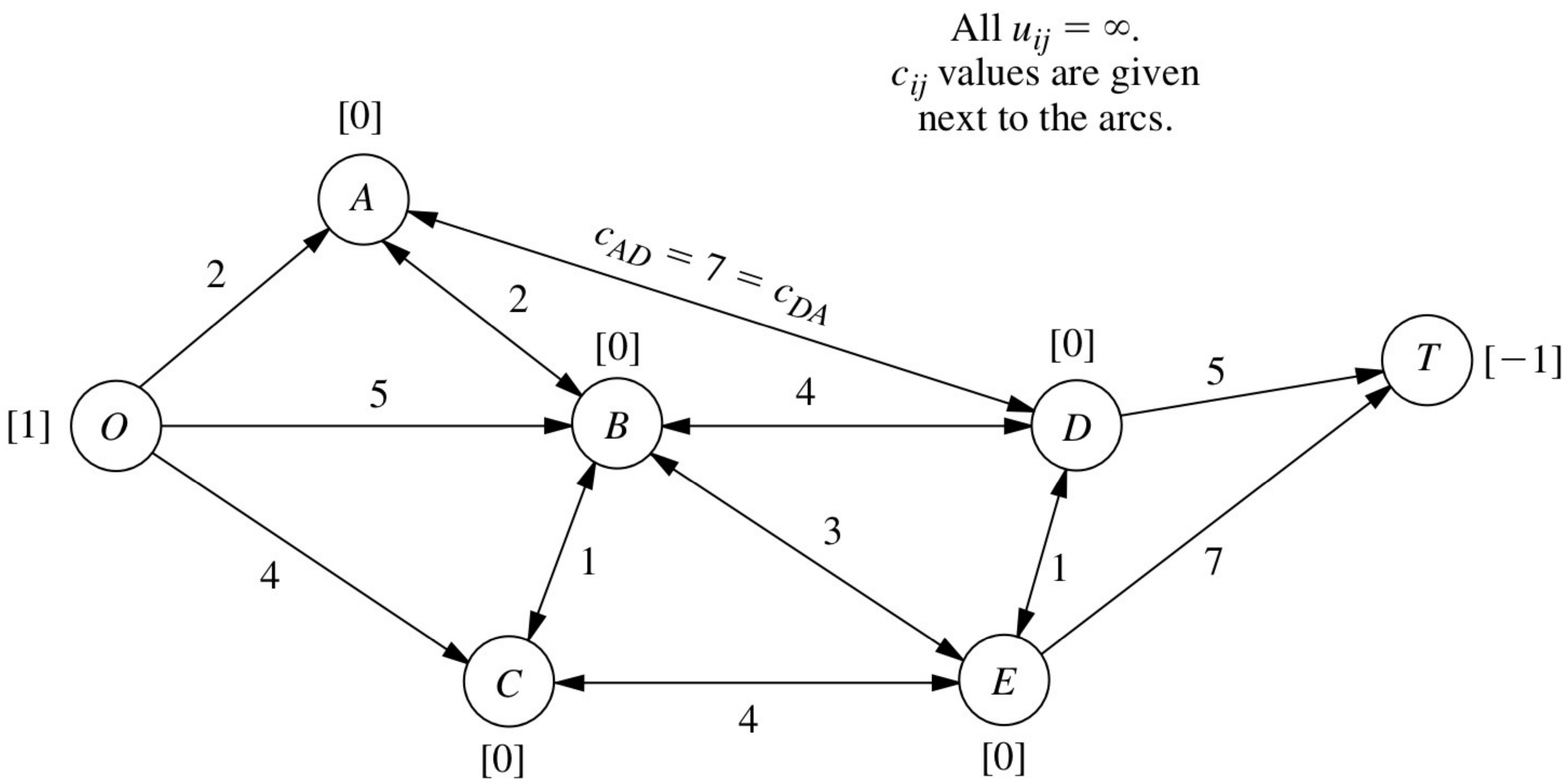
- **Transportation problem.** A supply node is provided for each *source* and a demand node for each *destination*. No transshipment nodes are included. Also, $u_{ij} = \infty$.
- **Assignment problem.** As in the transportation problem and
  - Number of supply nodes equal to number of demand nodes.
  - $b_i = 1$ for supply nodes and $b_i = -1$ for demand nodes.
- **Transshipment problem.** A minimum cost flow problem with unlimited arc capacities: $u_{ij} = \infty$ (like previous example if u_{AB} and u_{CE} were ∞).

Special cases

Shortest-path problem

- A supply node with the supply of 1 is provided for the origin.
- A demand node with the demand of 1 is provided for the destination.
- Rest of nodes are transshipment nodes.
- Each undirected link is replaced by a pair of directed arcs in opposite directions ($c_{ij} = c_{ji}$), except arcs *into* supply node or *out of* demand node.
- No arc capacities are imposed: $u_{ij} = \infty$.

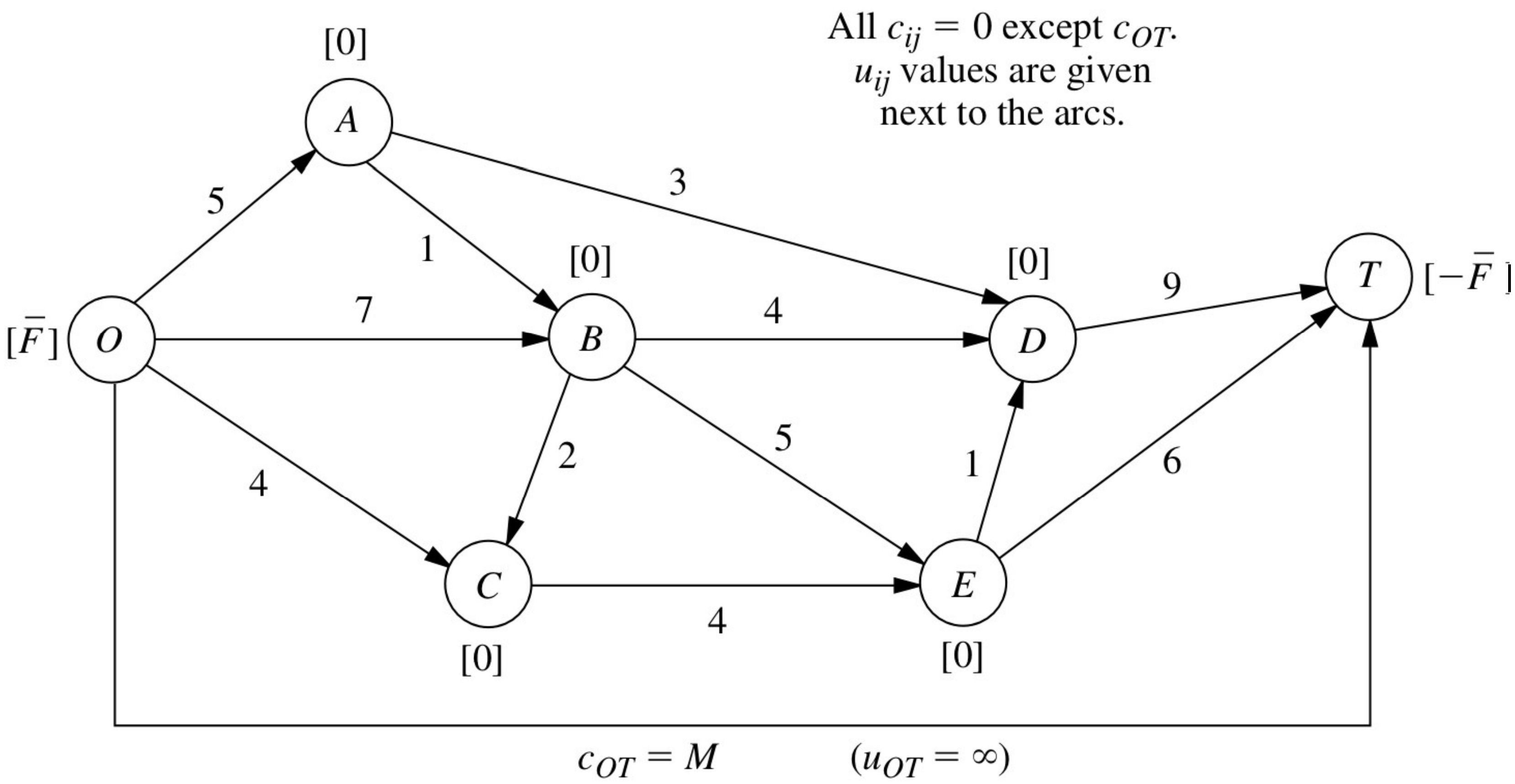
Example of shortest-path problem



Special cases

- **Maximum flow problem.** Network already provided with one supply node (source) and one demand node (sink). Adjustments needed:
  - Set c_{ij} = 0 for all existing arcs (absence of costs).
  - Select $\overline{F}$, a safe upper bound on the maximum feasible flow through the network, and assign it as supply and demand.
  - Add an arc from the supply node to the demand node and assign it an arbitrarily large unit cost, $c_{ij} = M$, as well as unlimited capacity, $u_{ij} = \infty$.

Example of maximum flow problem



Final comments

- Except for transshipment problem, each of these special cases was seen in this or the previous lesson (Chap.8 and 9).
- For these, we already saw special-purpose algorithms.
- When a computer code is not readily available for the special-purpose algorithm, it is reasonable to use the ***network simplex method***, a highly streamlined version of the simplex method for solving minimum cost flow problems.