NETWORK OPTIMIZATION MODELS

Network models

- Transportation, electrical and communication networks pervade our daily lives.
- Network representation are widely used in:
  - Production, distribution, project planning, facilities location, resource management, financial planning, etc.
- 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.

![Graph of Seervada Park's layout]

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 distance 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

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Arcs</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersections</td>
<td>Roads</td>
<td>Vehicles</td>
</tr>
<tr>
<td>Airports</td>
<td>Air lanes</td>
<td>Aircraft</td>
</tr>
<tr>
<td>Switching points</td>
<td>Wires, channels</td>
<td>Messages</td>
</tr>
<tr>
<td>Pumping stations</td>
<td>Pipes</td>
<td>Fluids</td>
</tr>
<tr>
<td>Work centers</td>
<td>Material-handling sources</td>
<td>Jobs</td>
</tr>
</tbody>
</table>

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 directions: 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 $j$.

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).
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**.

---

**Example of a directed network**

![Directed Network Diagram](image)

**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. 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 where the amount of flow out equals the amount of flow in.

Growing tree one arc a time

- Nodes without arcs
- Tree with one arc
- Tree with two arcs
- Tree with three arcs
- A spanning tree
**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 identifies 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 nth iteration:** find the nth nearest node to the origin \((n = 1, 2, \ldots)\) until the nth nearest node is reached.
- **Input for the nth 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 nth 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 nth nearest node**: for each solved 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 nth nearest node, and its shortest path is the one generating this distance.

Example: Seervada park

<table>
<thead>
<tr>
<th>n</th>
<th>Solved nodes directly connected to unsolved nodes</th>
<th>Closest connected unsolved node</th>
<th>Total distance involved</th>
<th>nth nearest node</th>
<th>Minimum distance</th>
<th>Last connection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Example: Seervada park**

<table>
<thead>
<tr>
<th>n</th>
<th>Solved nodes directly connected to unsolved nodes</th>
<th>Closest connected unsolved node</th>
<th>Total distance involved</th>
<th>nth nearest node</th>
<th>Minimum distance</th>
<th>Last connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>A</td>
<td>2</td>
<td>A</td>
<td>2</td>
<td>OA</td>
</tr>
<tr>
<td>2,3</td>
<td>O</td>
<td>C</td>
<td>4</td>
<td>C</td>
<td>4</td>
<td>OC</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>2 + 2 = 4</td>
<td>B</td>
<td>4</td>
<td>AB</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>D</td>
<td>2 + 7 = 9</td>
<td>E</td>
<td>7</td>
<td>BE</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>E</td>
<td>4 + 3 = 7</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>E</td>
<td>4 + 4 = 8</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>D</td>
<td>2 + 7 = 9</td>
<td>D</td>
<td>8</td>
<td>BD</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>D</td>
<td>4 + 4 = 8</td>
<td>D</td>
<td>8</td>
<td>ED</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>D</td>
<td>7 + 1 = 8</td>
<td>D</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td>T</td>
<td>8 + 5 = 13</td>
<td>T</td>
<td>13</td>
<td>DT</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>T</td>
<td>7 + 7 = 14</td>
<td>T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Solution**

![Diagram of Seervada park network with distances and connections labeled.]
Using simplex to solve the problem

Seervada Park Shortest-Path Problem

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>On Route</th>
<th>Distance</th>
<th>Nodes</th>
<th>Net Flow</th>
<th>Supply/Demand</th>
<th>Range Name</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>O</td>
<td>1</td>
<td>1</td>
<td>Distance</td>
<td>F4:F17</td>
</tr>
<tr>
<td>O</td>
<td>B</td>
<td>0</td>
<td>5</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>From</td>
<td>B4:B17</td>
</tr>
<tr>
<td>O</td>
<td>C</td>
<td>4</td>
<td>0</td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>NetFlow</td>
<td>H4:H10</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>1</td>
<td>2</td>
<td>C</td>
<td>0</td>
<td>0</td>
<td>Nodes</td>
<td>H4:H10</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>0</td>
<td>7</td>
<td>D</td>
<td>0</td>
<td>0</td>
<td>OnRoute</td>
<td>D4:D17</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>E</td>
<td>0</td>
<td>0</td>
<td>Supply/Demand</td>
<td>K4:K10</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>1</td>
<td>4</td>
<td>T</td>
<td>-1</td>
<td>-1</td>
<td>To</td>
<td>C4:C17</td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>TotalDistance</td>
<td>D19</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>T</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>T</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total Distance: 13

Using simplex to solve the problem

Seervada Park Shortest-Path Problem

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>On Route</th>
<th>Distance</th>
<th>Nodes</th>
<th>Net Flow</th>
<th>Supply/Demand</th>
<th>Range Name</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>O</td>
<td>1</td>
<td>1</td>
<td>Distance</td>
<td>F4:F17</td>
</tr>
<tr>
<td>O</td>
<td>B</td>
<td>0</td>
<td>5</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>From</td>
<td>B4:B17</td>
</tr>
<tr>
<td>O</td>
<td>C</td>
<td>4</td>
<td>0</td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>NetFlow</td>
<td>H4:H10</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>1</td>
<td>2</td>
<td>C</td>
<td>0</td>
<td>0</td>
<td>Nodes</td>
<td>H4:H10</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>0</td>
<td>7</td>
<td>D</td>
<td>0</td>
<td>0</td>
<td>OnRoute</td>
<td>D4:D17</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>E</td>
<td>0</td>
<td>0</td>
<td>Supply/Demand</td>
<td>K4:K10</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>1</td>
<td>4</td>
<td>T</td>
<td>-1</td>
<td>-1</td>
<td>To</td>
<td>C4:C17</td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>TotalDistance</td>
<td>D19</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>T</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>T</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total Distance: 13

Solver Parameters
- Set Target Cell: Total Distance
- By Changing Cells: Distance
- Subject to the Constraints: Net Flow = Supply/Demand

Solver Options
- Assume Linear Model
- Assume Non-negative

Solver Options
- Range Name Cell: D19
- Total Distance: D19

Total Distance: 13
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 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

Definitions:
- **Nodes** of the network are given, as well as potential **links** and positive **length** for each if it is inserted in the network.
- The network should insert links in order to have a path between each pair of nodes.
- 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 is a spanning tree: (a), (b) or (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, 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.
- 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 to the scenic wonder to maximize the number of trips per day.
- Outgoing trips allowed per day:
Feasible solution

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

Definition of maximum flow problem

- All flows through a directed and connected network from the source to the sink.
- All remaining nodes are transshipment nodes.
- Flow through an arc in 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 customers.
- 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 of 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 the 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; can be solved by the simplex method.
- However, the augmented path algorithm is more efficient.
Augmented path algorithm

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

  ![Directed and undirected arcs]

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

  ![Residual network]

Seervada Park residual network

- Augmenting path is a directed path from the source to the sink, such that every path has strictly positive residual capacity.

  ![Seervada Park residual network diagram]
Iteration of augmented path algorithm

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

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

Application to Seervada Park

- Initial residual network:
Iteration 1

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

Iteration 2

- **Iteration 2**: assign a flow of 3 to the augmenting path $O \to A \to D \to T$.
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
  \[ O \rightarrow C \rightarrow E \rightarrow B \rightarrow D \rightarrow T \]

**Optimal solution**

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

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

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Flow</th>
<th>Capacity</th>
<th>Nodes</th>
<th>Net Flow</th>
<th>Supply/Demand</th>
<th>Range Name Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>A</td>
<td>3</td>
<td>&lt;= 5</td>
<td>O</td>
<td>14</td>
<td>Capacity</td>
<td>F4:F15</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
<td>7</td>
<td>&lt;= 7</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>D4:D15</td>
</tr>
<tr>
<td>O</td>
<td>C</td>
<td>4</td>
<td>&lt;= 4</td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>B4:B15</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>0</td>
<td>&lt;= 1</td>
<td>C</td>
<td>0</td>
<td>0</td>
<td>A4:A15</td>
</tr>
<tr>
<td>A</td>
<td>U</td>
<td>3</td>
<td>&lt;= 3</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>U4:U15</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>0</td>
<td>&lt;= 2</td>
<td>E</td>
<td>0</td>
<td>0</td>
<td>E4:E15</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>4</td>
<td>&lt;= 4</td>
<td>Y</td>
<td>.14</td>
<td>-14</td>
<td>Y4:Y15</td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td>3</td>
<td>&lt;= 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>3</td>
<td>&lt;= 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>1</td>
<td>&lt;= 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>T</td>
<td>1</td>
<td>&lt;= 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Y</td>
<td>6</td>
<td>&lt;= 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Range Name Cells:
- Capacity: F4:F15
- Flow: D4:D15
- MaximumFlow: D17
- NetFlow: H4:H10
- Nodes: K5:K9
- Supply/Demand: C4:C15

Maximum Flow: 14

Using simplex to solve the problem

Solver Parameters
- Set Target Cell: MaxFlow
- By Changing Cells: Flow

Subject to the Constraints:
- <= Supply/Demand (Flow <= Capacity)

Solver Options
- Assume Linear Model
- Assume Non-Negative

Solver Results
- Range Name Cells:
  - Supply/Demand: K5:K9
  - NetFlow: H4:H10
  - Nodes: C4:C15

Maximum Flow: 14
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:

  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 is the capacity of the arc.
6. Network has enough arcs with sufficient capacity such that all flow generated at supply nodes reach the demand nodes.
7. Cost of flow through each arc is proportional to the amount of that flow.
8. Objective: minimize the total cost (profit) of sending the available supply through the network to satisfy the demand.

Applications

<table>
<thead>
<tr>
<th>Kind of application</th>
<th>Supply nodes</th>
<th>Transshipment nodes</th>
<th>Demand nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation of a distributed network</td>
<td>Sources of goods</td>
<td>Intermediate storage facilities</td>
<td>Customers</td>
</tr>
<tr>
<td>Solid waste management</td>
<td>Sources of solid waste</td>
<td>Processing facilities</td>
<td>Landfill locations</td>
</tr>
<tr>
<td>Operation of a supply network</td>
<td>Vendors</td>
<td>Intermediate warehouses</td>
<td>Processing facilities</td>
</tr>
<tr>
<td>Coordinating product mixes at plants</td>
<td>Plants</td>
<td>Production of a specific product</td>
<td>Market for a specific product</td>
</tr>
<tr>
<td>Cash flow management</td>
<td>Sources of cash at a specific time</td>
<td>Short-term investment options</td>
<td>Needs for cash at a specific time</td>
</tr>
</tbody>
</table>
Formulation of the model

- Decision variables:
  - \( x_{ij} \) = flow through arc \( i \to j \)
- Given information:
  - \( c_{ij} \) = cost per unit flow through arc \( i \to j \)
  - \( u_{ij} \) = arc capacity for arc \( i \to j \)
  - \( b_i \) = 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

\[
\text{minimize}\quad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},
\]

subject to

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

and \( 0 \leq x_{ij} \leq u_{ij}, \quad \text{for each arc } i \to j \)
**Minimum cost flow problem**

- **Feasible solutions property:** necessary condition for a minimum flow problem to have feasible solutions:
  \[ \sum_{j=1}^{n} b_j = 0 \]

- If this condition does not hold, a dummy source or a dummy destination is needed.

- **Integer solutions property:** when every \( b_i \) and \( u_{ij} \) have integer values, all the basic variables in every basic feasible (BF) solution also have integer values.

---

**Example**

- **Distribution network** for the Distribution Unlimited Co.

  ![Distribution network diagram]

  - \( b_A = 50 \)
  - \( c_{AD} = 9 \)
  - \( [40] \)
  - \( b_B = 40 \)
  - \( u_{AB} = 10 \)
  - \( c_{BC} = 3 \)
  - \( u_{CE} = 80 \)
  - \( [0] \)
  - \( [2] \)
  - \( [3] \)
  - \( [1] \)
  - \( [2] \)
  - \( [3] \)
  - \( [4] \)
  - \( [5] \)
  - \( [6] \)
  - \( [7] \)
  - \( [8] \)
  - \( [9] \)
  - \( [10] \)
Example

- Linear programming problem:

  minimize \[ Z = 2x_{AB} + 4x_{AC} + 9x_{AD} + 3x_{BC} + x_{CE} + 3x_{DE} + 2x_{ED}, \]

  subject to

  \[
  \begin{align*}
  x_{AB} + x_{AC} + x_{AD} &= 50 \\
  -x_{AB} + x_{BC} &= 40 \\
  -x_{AC} - x_{BC} + x_{CE} &= 0 \\
  -x_{AD} + x_{DE} - x_{ED} &= -30 \\
  -x_{CE} - x_{DE} + x_{ED} &= -60 \\
  \end{align*}
  \]

  and \( x_{AB} \leq 10, \quad x_{CE} \leq 80, \quad \text{all } x_{ij} \geq 0. \)

Using simplex to solve the problem

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Ship</th>
<th>Capacity</th>
<th>Unit Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>40</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>10</td>
<td>9</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>40</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>80</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>20</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Cost</td>
<td>490</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using simplex to solve the problem

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Capacity</th>
<th>Unit Cost</th>
<th>Nodes</th>
<th>Net Flow</th>
<th>Supply/Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>10</td>
<td>2</td>
<td>A</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>9</td>
<td>1</td>
<td>C</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>3</td>
<td>1</td>
<td>D</td>
<td>-30</td>
<td>-30</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
<td>2</td>
<td>2</td>
<td>E</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total Cost: 80

Special cases

- **Transportation problem.** A supply node is provided for each *source* and a demand node for each *destination*. No transshipment nodes are included.

- **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 $b_j = -1$ for demand nodes.

- **Transshipment problem.** A minimum cost flow problem with unlimited arc capacities: $u_{ij} = \infty$. 
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 ($c_{ij} = c_{ji}$).
  - No arc capacities are imposed: $u_{ij} = +\infty$.

Example of shortest-path problem

All $a_{ij} = \infty$.
$c_{ij}$ values are given next to the arcs.
**Special cases**

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

---

**Example of maximum flow problem**

![Diagram of maximum flow problem]
Network simplex method

- Highly streamlined version of the simplex method for solving minimum cost flow problems.
- Goes from one BF solution to another one using the network itself (without using the simplex tableau).
- There is a correspondence between BF solutions and feasible spanning trees.
- See example in Hillier's book, OR and IOR examples, and computational implementations.