Correctness of the Ford–Fulkerson Algorithm

Claim: The Ford–Fulkerson Algorithm gives a maximum flow.
Proof: We must show that the algorithm always stops, and that when it stops, the output is indeed a maximum flow.

⋆ We will consider the case of integer capacities.

The algorithm terminates.

- Each iteration increases the throughput of the flow by an integer.
- The sum of the capacities on the edges out of s is finite.

The output is a maximum flow. Upon termination:

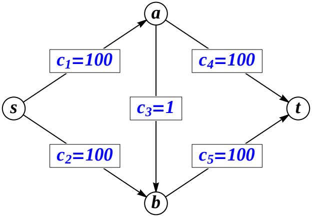
- There are no flow augmenting paths in the companion graph, so:
- Edges from X to X^c are full and edges from X^c to X are empty.
- The capacity of $[X, X^c]$ equals the throughput of the flow.

Conclusion: The flow is a max flow and the *st*-cut is a min cut.

Closing Remarks

- When using the algorithm, it is important to increase the flow by as much as possible at each step.

- When the capacities are integers, we always increase the throughput by integers. The algorithm does work when the capacities are not integers, but the proof is more involved.
- As presented here, this algorithm may be very slow.

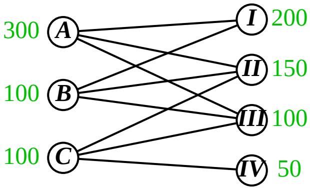


Transshipment

The Transshipment Problem: Given m suppliers, each with some amount of product, and n customers, each desiring some amount of product, where each supplier delivers to a subset of the customers.

Is it possible for the suppliers (customers) to have their orders filled?

Example: let A, B, and C be suppliers with 300, 100, and 100 units of product, respectively, and I, II, III, and IV be customers desiring 200, 150, 100, and 50 units of product, and where neither supplier A nor B delivers to customer IV and supplier C does not deliver to customer I. We can model this situation using a graph:



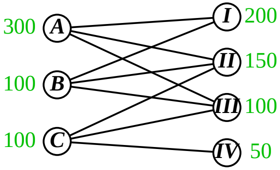
Is there a transshipment that satisfies all the suppliers?

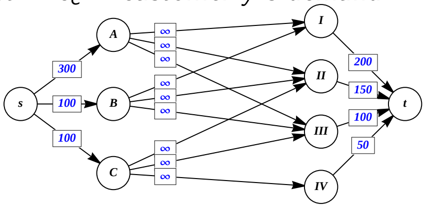
Transshipment

We can solve transshipment problems using the Ford–Fulkerson. We must convert the transshipment problem and corresponding graph G into a network flow problem.

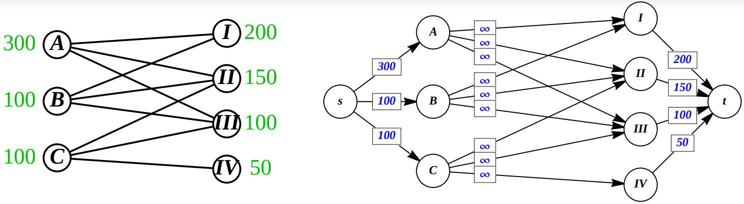
Augment the graph G to become a network $\widehat{G}$ by creating a "super-source" s that is adjacent to every supplier x and a "super-sink" t that is adjacent to every customer y.
Assign capacities to the edges as follows:

$$\begin{cases} \text{if } e : s \longrightarrow x, \text{ set} & c_e = \text{supplier } x\text{'s supply} \\ \text{if } e : x \longrightarrow y, \text{ set} & c_e = \infty \\ \text{if } e : y \longrightarrow t, \text{ set} & c_e = \text{customer } y\text{'s demand} \end{cases}$$





Transshipment



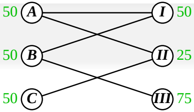
Important: Any valid transshipment in G corresponds to a valid flow in $\widehat{G}$, and vice versa.

Therefore, finding a maximum transshipment in G corresponds to finding a maximum flow in $\widehat{G}$, and vice versa.

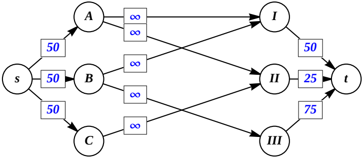
If all suppliers are satisfied, the min cut in the network will be ______ Otherwise, the min cut will tell you the problem: There will be some set of suppliers whose customers demand less than the suppliers supply.

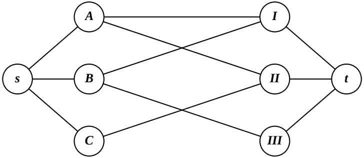
If you are customer-centric, instead orient the edges from right to left to find a set of customers who can not be satisfied by their suppliers.

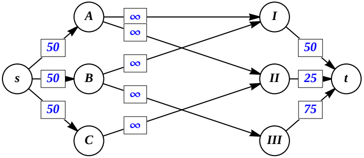
Transshipment Example

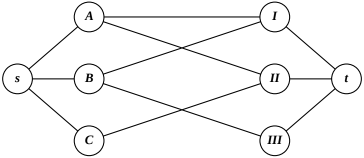


Supplier-centric:



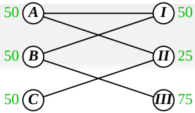




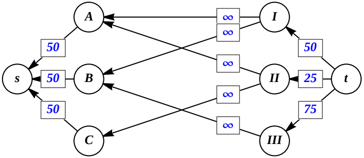


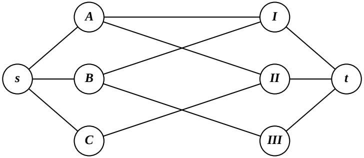
Problem:

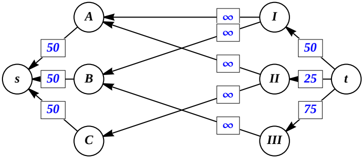
Transshipment Example

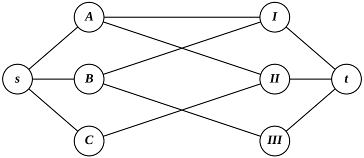


Customer-centric:









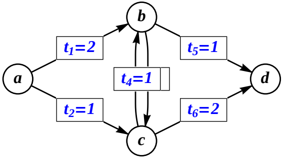
Problem:

Dynamic Networks

The networks we have been considering are all static and are good for modeling the maximum throughput of a system. We need to use dynamic networks to model the act of sending a shipment.

Definition: In a **dynamic network**, every edge e has a travel time t_e associated to it in addition to a capacity c_e.

Example: Consider four cities a, b, c, and d with warehouses such that one truck per day can leave along any route, and the travel time for each route is given by:



We wish to determine the maximum number of shipments which can make it from city a on day 0 and arrive at city d by day 5.