# Network Flows

RAMESH K. JALLU

Lec-06

DT. 03/02//22

# Motivation

- We often use graphs to model *transportation* (or *flow*) *networks*—networks whose edges carry some sort of traffic (flow) and whose nodes act as “switches” passing traffic between different edges
- We can interpret a directed graph as a *flow network* and use it to answer questions about material flows
- A computer network in which the edges are links that can carry packets and the nodes are switches
- A fluid/gas/electrical network in which edges are pipes/wires that carry liquid/gas/electricity, and the nodes are junctures where pipes/wires are plugged/joined together

# Basic terminology

- Network models of this type have several ingredients:
  1. ***Capacities*** on the edges, indicating how much they can carry
  2. ***Traffic/flow***, an abstract entity which is transmitted across the edges
  3. ***Source*** nodes in the graph, which generate traffic/flow
  4. ***Sink*** (or destination) nodes in the graph, which can "absorb" traffic as it arrives
- The source produces the material at some steady rate, and the sink consumes the material at the same rate
- The "flow" of the material at any point in the system is intuitively the rate at which the material moves

# Flow network

- A *flow network* is a directed graph $G = (V, E)$ such that each edge $(u, v)$ is associated with a *capacity* $c(u, v) \geq 0$ (i.e., a capacity function $c$ is given) and has two distinguished nodes source $s$ and sink $t$
  - No edge enters the source $s$ and no edge leaves the sink $t$
  - Nodes other than $s$ and $t$ will be called *internal* nodes
- If $(u, v) \notin E$, then $c(u, v) = 0$, and we disallow self-loops
- We further require that if $E$ contains an edge $(u, v)$, then $(v, u) \notin E$
- The flow is generated at the source node, transmitted across edges, and absorbed at the sink node
- WLG, we assume that each vertex lies on some path from the source to the sink
- The graph is therefore connected and, since each vertex other than $s$ has at least one entering edge, $|E| \geq |V| - 1$

# Flow

- Let $G = (V, E)$ be a flow network with a capacity function $c$
- Let $s$ be the source of the network, and let $t$ be the sink
- A ***flow*** in $G$ is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties:
  - **Capacity constraint:** For each edge $(u, v) \in E$, we require $0 \leq f(u, v) \leq c(u, v)$
  - **Flow conservation:** For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

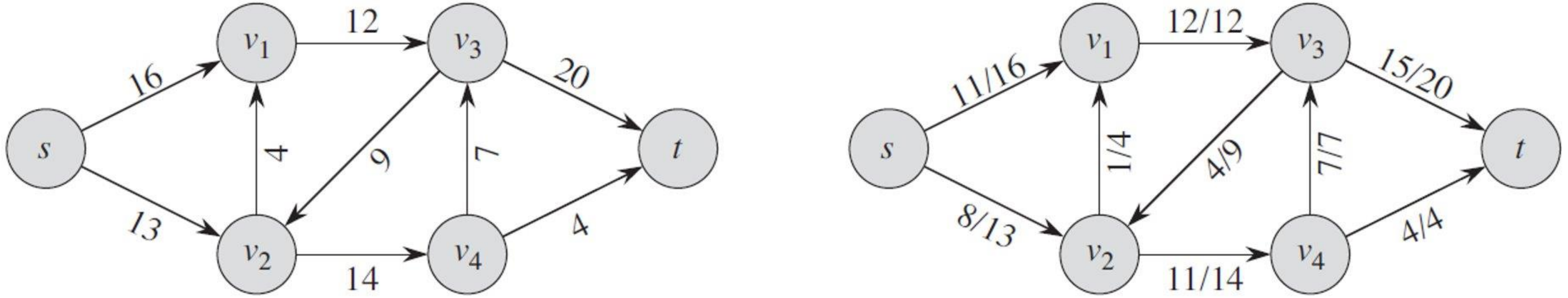
  - When $(u, v) \notin E$, there can be no flow from $u$ to $v$, and $f(u, v) = 0$

# Flow value

- We call the non-negative quantity $f(u, v)$ the flow from $u$ to $v$
- The ***value*** $|f|$ of a flow $f$ is defined as $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$
  - The total flow out of the source minus the flow into the source
- A flow network will not have any edges into the source, and the flow into the source is zero, but it is allowed to have flow going out
  - Therefore, the *value* of a flow $f$ is defined to be the amount of flow generated at the source
- Symmetrically, the sink is allowed to have flow coming in, even though it has no edges leaving it

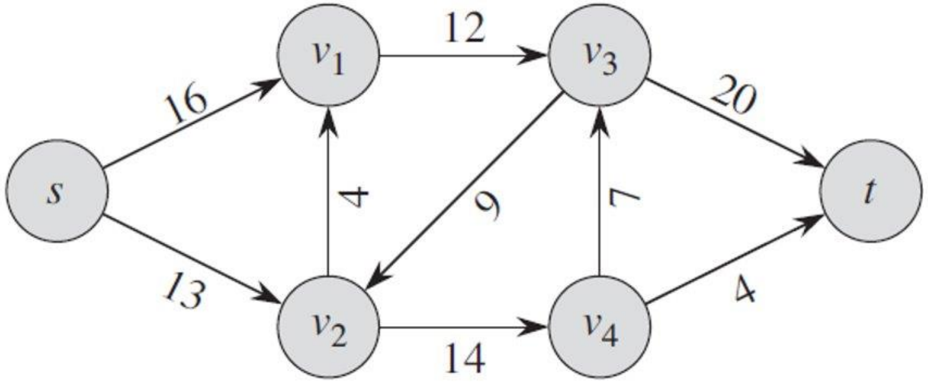
# The maximum-flow problem

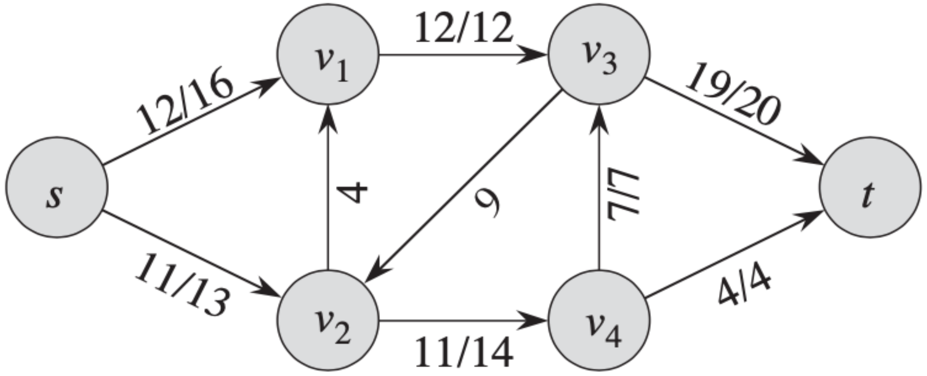
- **Given**: A flow network $G$ with source $s$ and sink $t$
- **Objective**: Arrange the flow so as to make as efficient use as possible of the available capacity
  - In other words, the goal is to find a flow of maximum possible value



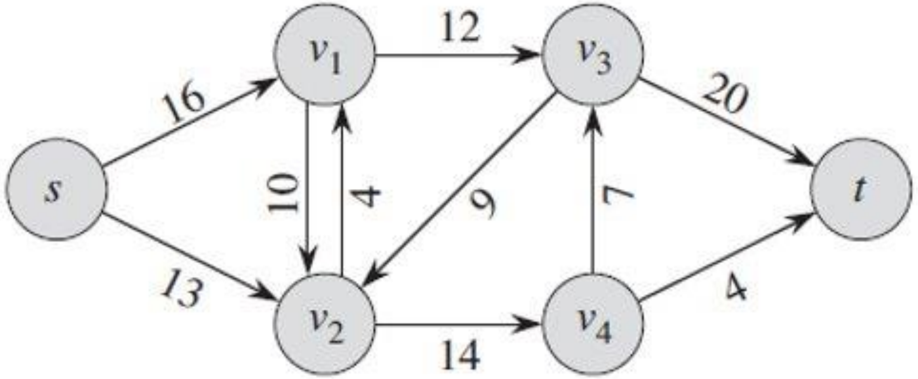
# The maximum-flow problem

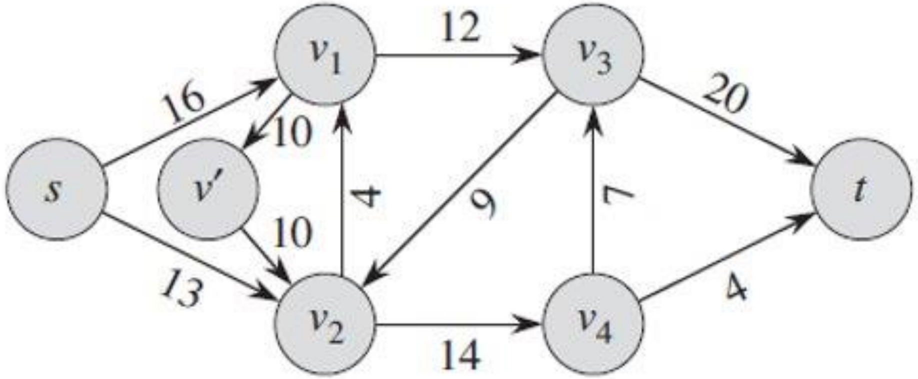
- **Given**: A flow network $G$ with source $s$ and sink $t$
- **Objective**: Arrange the flow so as to make as efficient use as possible of the available capacity
  - In other words, the goal is to find a flow of maximum possible value



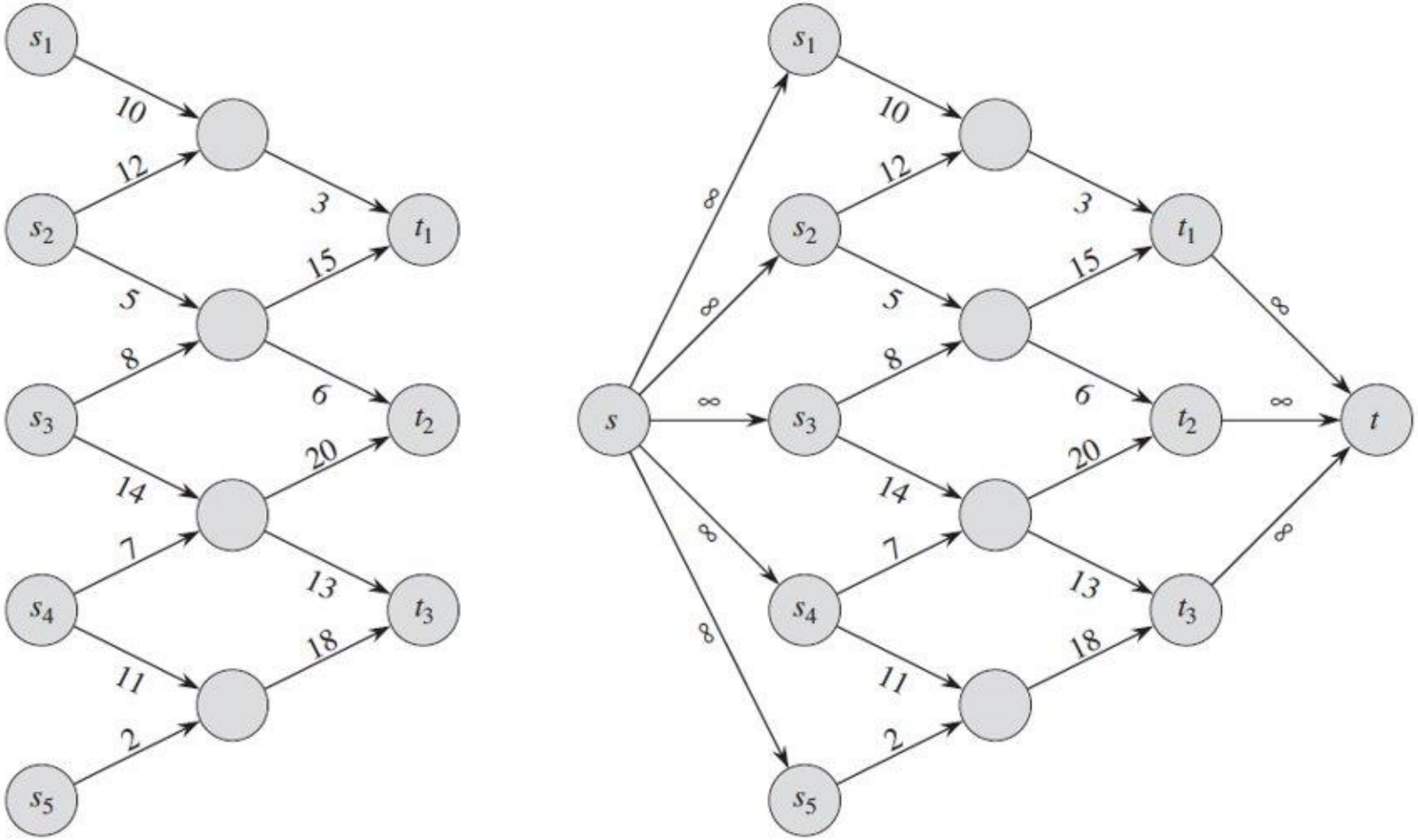
# Modeling problems with antiparallel edges




- **Claim:** Splitting an edge in a flow network yields an equivalent network
- More formally, suppose that flow network $G$ contains edge $(u, v)$ and we create a new flow network $G'$ by creating a new vertex $x$ and replacing $(u, v)$ by new edges $(u, x)$ and $(x, v)$ with $c(u, x) = c(x, v) = c(u, v)$. The maximum flow in $G'$ has the same value as a maximum flow in $G$

# Networks with multiple sources and sinks



# The Ford-Fulkerson method

- This method was developed by Lester Randolph Ford Jr. and Delbert Ray Fulkerson in 1956
- It is called as a method rather than an algorithm
- The method depends on 3 ideas: residual networks, augmenting paths, and cuts
- The idea behind the algorithm is as follows
  - As long as there is a path from the source to the sink, with available capacity on all edges in the path, we send flow along one of the paths
  - Then we find another path, and so on
  - A path with available capacity is called an ***augmenting path***

| | |
|---|---|
| Ford–Fulkerson algorithm | $O(E\|f_{max}\|)$ |
| Edmonds–Karp algorithm | $O(VE^2)$ |
| Dinic's algorithm | $O(V^2E)$ |
| MKM (Malhotra, Kumar, Maheshwari) algorithm[10] | $O(V^3)$ |
| Dinic's algorithm with dynamic trees | $O(VE\log V)$ |
| General push–relabel algorithm[11] | $O(V^2E)$ |
| Push–relabel algorithm with *FIFO* vertex selection rule[11] | $O(V^3)$ |
| Push–relabel algorithm with *maximum distance* vertex selection rule[12] | $O(V^2\sqrt{E})$ |
| Push-relabel algorithm with dynamic trees[11] | $O\left(VE\log\frac{V^2}{E}\right)$ |
| KRT (King, Rao, Tarjan)'s algorithm[13] | $O\left(VE\log_{\frac{E}{V\log V}}V\right)$ |
| Binary blocking flow algorithm[14] | $O\left(E\cdot\min\{V^{2/3},E^{1/2}\}\cdot\log\frac{V^2}{E}\cdot\log U\right)$ |
| James B Orlin's + KRT (King, Rao, Tarjan)'s algorithm[9] | $O(VE)$ |

# Cont...

- Although each iteration of the Ford-Fulkerson method increases the value of the flow, we shall see that the flow on any particular edge of $G$ may increase or decrease
- Decreasing the flow on some edges may be necessary in order to enable an algorithm to send more flow from the source to the sink
- We repeatedly augment the flow until the residual network has no more augmenting paths

FORD-FULKERSON-METHOD$(G, s, t)$

```
initialize flow f to 0
while there exists an augmenting path p in the residual network G_f
    augment flow f along p
return f
```

# Residual networks

- Given a flow network $G$ and a flow $f$, the residual network $G_f$ consists of edges with capacities that represent how we can change the flow on edges of $G$
- An edge of the flow network can admit an amount of additional flow equal to the edge's capacity minus the flow on that edge
- If that value is positive, we place that edge into $G_f$ with a "residual capacity" of $c_f(u, v) = c(u, v) - f(u, v)$
- The only edges of $G$ that are in $G_f$ are those that can admit more flow
- Those edges $(u, v)$ whose flow equals their capacity have $c_f(u, v) = 0$, and they are not in $G_f$

# Cont…

- The residual network $G_f$ may also contain edges that are not in $G$
- As an algorithm manipulates the flow, with the goal of increasing the total flow, it might need to decrease the flow on a particular edge
- In order to represent a possible decrease of a positive flow $f(u, v)$ on an edge in G, we place an edge $(u, v)$ into $G_f$ with residual capacity $c_f(v, u) = f(u, v)$
  - That is, an edge that can admit flow in the opposite direction to $(u, v)$ at most canceling out the flow on $(u, v)$
- These reverse edges in the residual network allow an algorithm to send back flow it has already sent along an edge
- Sending flow back along an edge is equivalent to decreasing the flow on the edge

- Thank you!