Maximum Flow
Ch 26 CLRS

Kimberly Nestor
Boston University
CS566 Project

Flow networks - overview

- Flow networks model the movement of a material from a source **s**, to a sink **t**
- Each edge in the network has a total **capacity** at which material can move
- As well as a current **flow** rate at which the material is moving
- Material is not allowed to accumulate at any of the vertices
    - **Flow conservation** - the rate at which the material enters the vertex should be the rate at which it leaves the vertex
- The goal of maximum flow problems is to determine the largest amount of material we can move through the network while abiding network capacity

CLRS

Flow networks - requirements

- For a flow network $G = (V, E)$
  - Directed graph
    - With capacity function c
  - Each edge $c(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$
  - No reverse edges
    - If E has an edge (u, v), there is no edge (v, u)
  - If $(u, v) \notin E$ then $c(u, v) = 0$
  - No self loops

CLRS

Flow networks - requirements

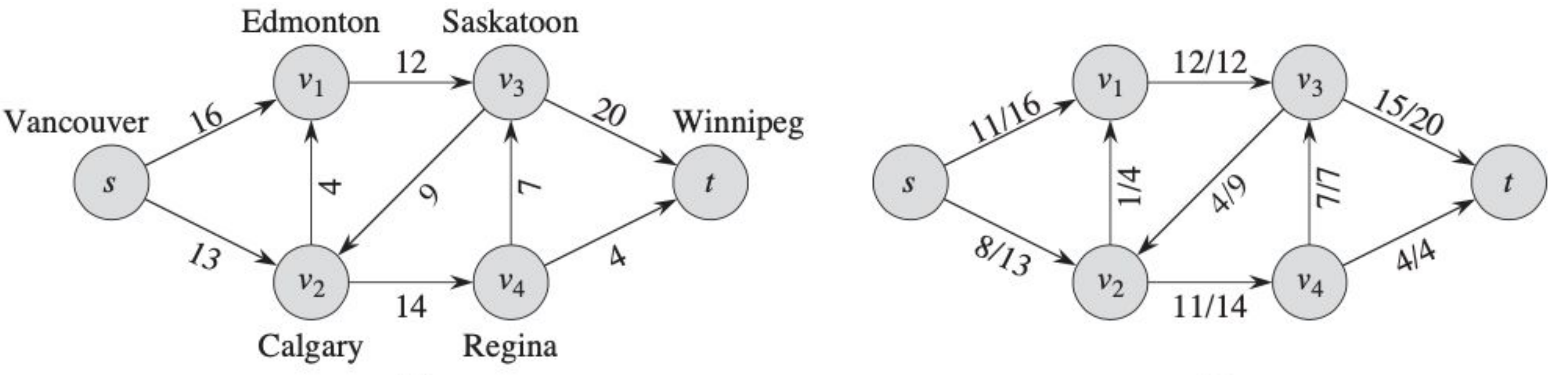
- For a flow network $G = (V, E)$
  - For each vertex $v \in V$ the follow network should have a path s $\rightsquigarrow$ v $\rightsquigarrow$ t
  - Each vertex, other than *s* has one entering edge
    - Therefore $|E| \geq |V| - 1$
- A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$
  - **Capacity constraint:** For all $u, v \in V$, 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 $f(u, v) = 0$.

CLRS

Flow networks



(a) (b)

- The nonnegative quantity of function *f(u, v)*, is a value | *f* | of a flow *f*

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) ,$$

  - In above e.g. flow *f* in *G* with value | *f* | = 19
  - Flow out of source should be 0, except when using residuals

CLRS

Residual networks

- For a flow network G and a flow f, the residual network $\boldsymbol{G_f}$ is a graph where edges are the remaining flow before exceeding capacity
- An edge has a residual capacity if
  - $c_f(u, v) = c(u, v) - f(u, v)$
- Only edges that can admit more flow are in G_f
  - Edges where the flow equals capacity are not in G_f and have $c_f(u, v) = 0$
- Residual G_f network may have edges that are not in G
  - To increase the total flow, may have to decrease flow on an edge
    - $c_f(v, u) = f(u, v)$

CLRS

Augmenting paths

- For a flow network *G* = (*V*, *E*) and a flow *f*
  - We can **increase** the **flow** on *(u, v)* by ***f'(u, v)*** but **decrease** it by ***f'(v, u)***
  - This is equivalent to decreasing flow, also called **cancellation**, by pushing flow in the reverse edge of the residual network

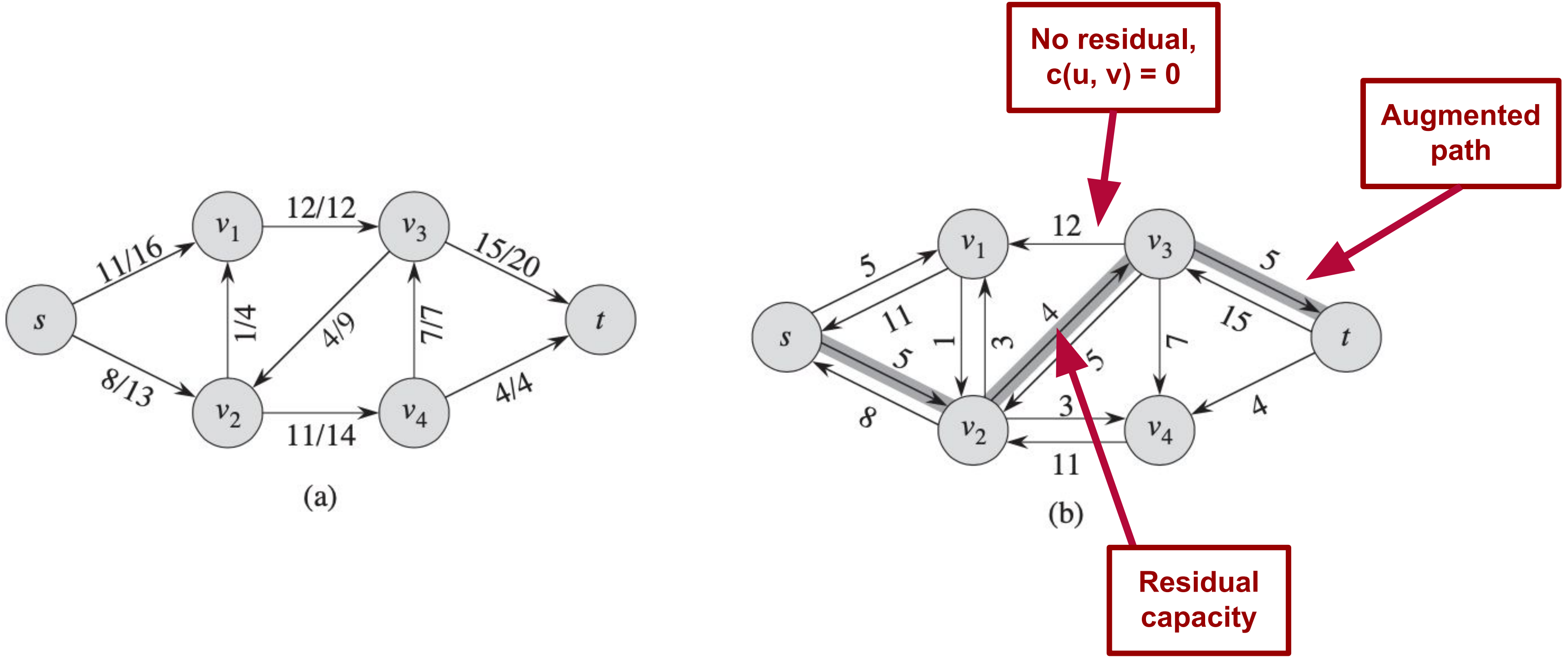
we define $f \uparrow f'$, the ***augmentation*** of flow f by f',

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{if } (u, v) \in E , \\ 0 & \text{otherwise} . \end{cases}$$

- The **residual capacity** of augmenting path *p* is the maximum amount by which we can increase the flow on **each edge**

CLRS

Residual networks with augmented paths



CLRS

Cuts

- As a part of the max-flow min-cut theorem, a flow is only at maximum when its residual network no longer has augmenting paths
  - s must be a part of set S, $s \in S$
  - t must be a part of set T, $t \in T$
- For flow *f*, the **net flow *f(S, T)*** across cut *(S, T)*
  - We consider the flow going from *S* to *T* minus the flow from *T* to *S*

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) .$$

CLRS

Cuts

- The **capacity** of the cut *(S, T)*
    - We consider only flow going from *S* to *T*, not from *T* to *S*

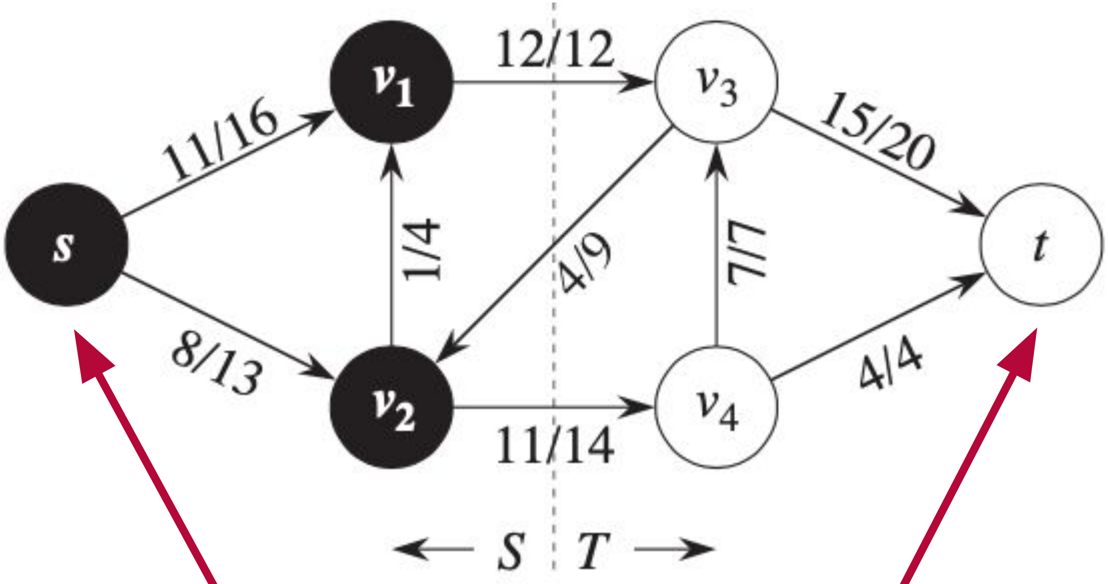
$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v) \ .$$

- A **minimum cut** of a network is a cut where the capacity is minimum over all the cut options in the network

CLRS

Minimum cut

Net flow *f(S, T)* = 19
Capacity c(S, T) = 26



s and t in separate partitions

CLRS

Min-cut max-flow theorem

If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Proof (1) ⇒ (2)

Proof (2) ⇒ (3)

Proof (3) ⇒ (1)

Cuts

- As a part of the max-flow min-cut theorem, a flow is only at maximum when its residual network no longer has augmenting paths
  - s must be a part of set S, s ∈ S
  - t must be a part of set T, t ∈ T
- For flow *f*, the **net flow *f(S, T)*** across cut *(S, T)*
  - We consider the flow going from *S* to *T* minus the flow from *T* to *S*

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) .$$

CLRS

Ford-Fulkerson algorithm

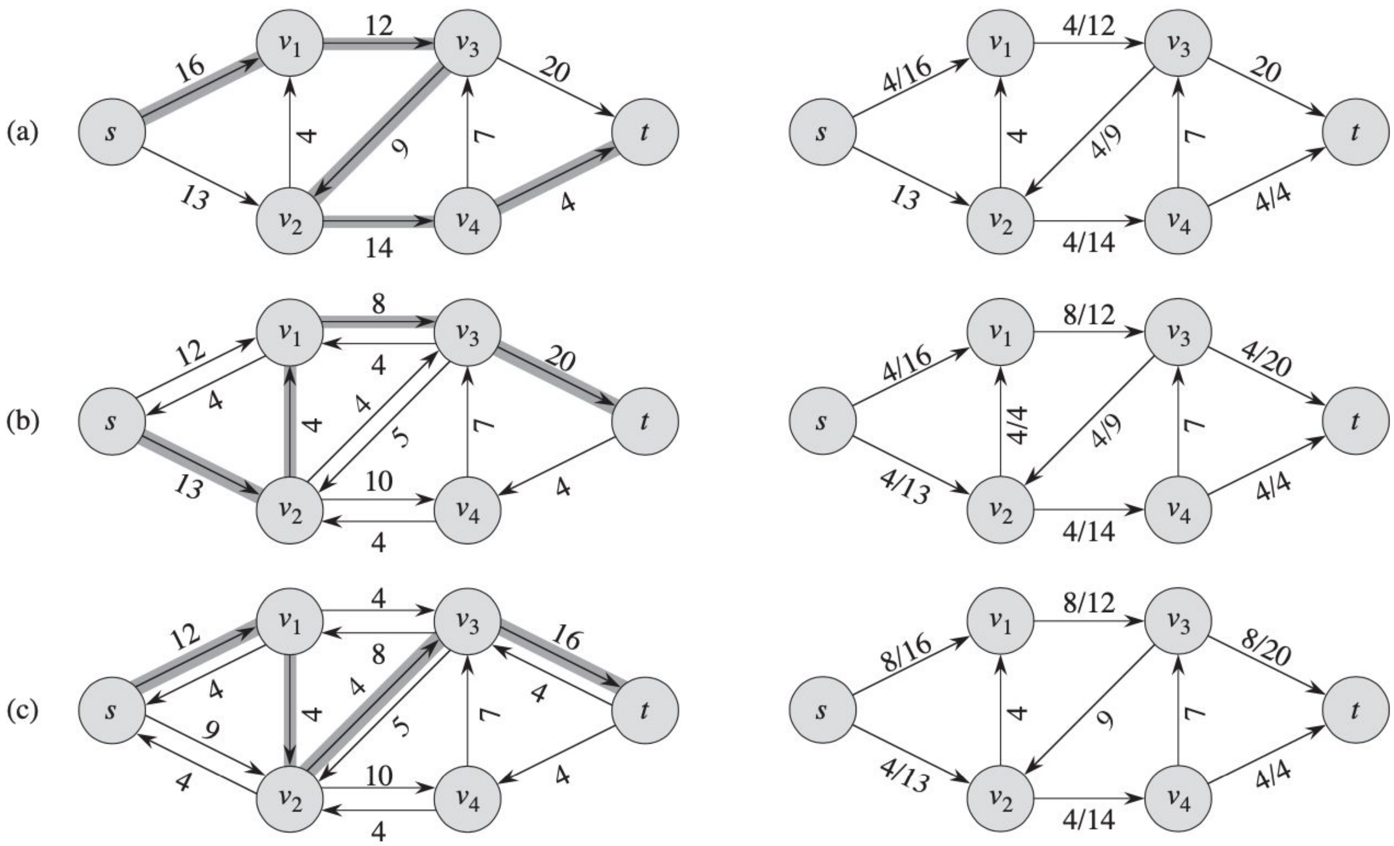
- We find some augmenting path p
  - Use p to modify the flow f
  - By replacing f by $f \uparrow f_p$, we get a new flow of $|f| + |f_p|$
- Running time is ***O(E)***

FORD-FULKERSON(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

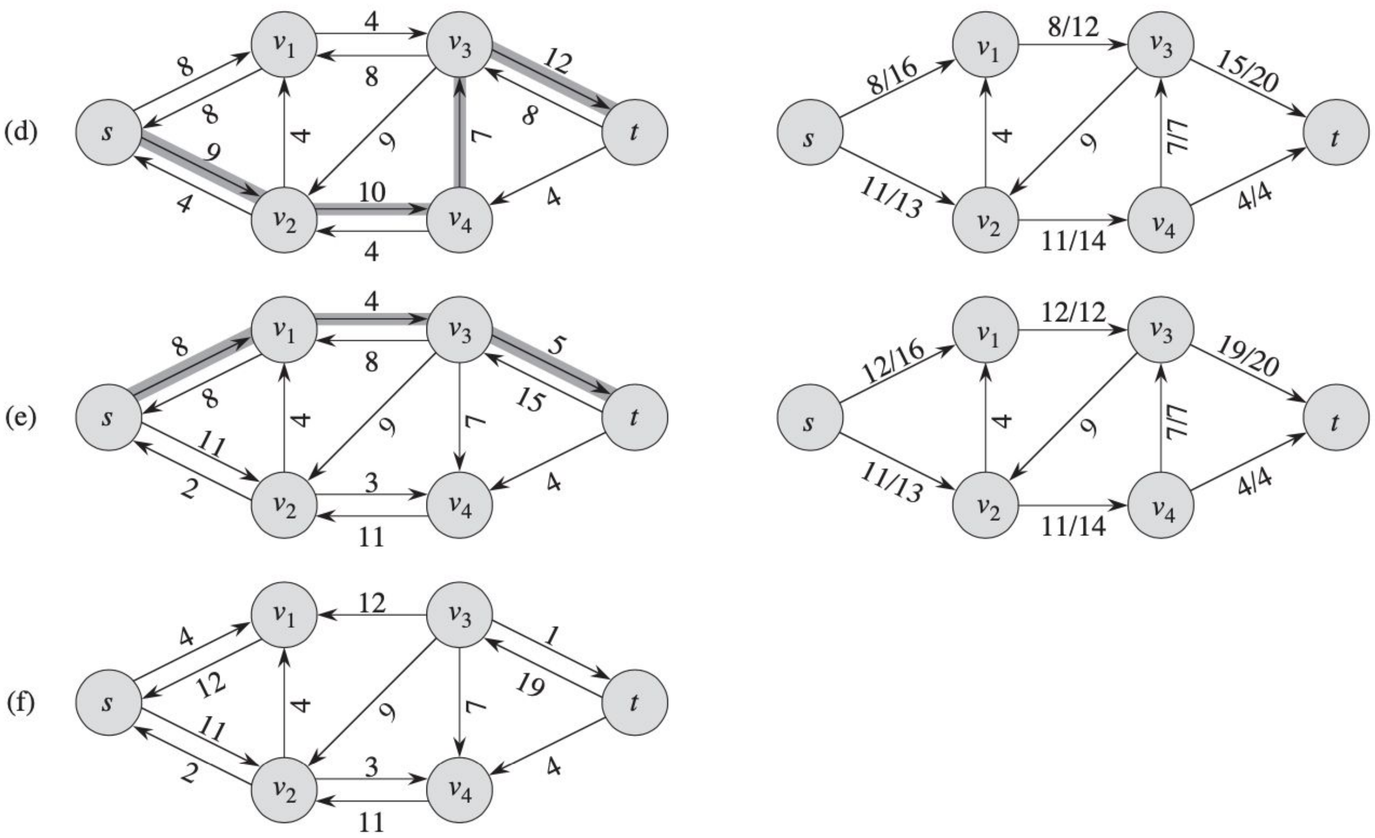
CLRS

Ford-Fulkerson algorithm



CLRS

Ford-Fulkerson algorithm



CLRS

Edmonds-Karp algorithm

- Developed due to high running time on Ford-Fulkerson with large edge weights
- Improves bound by finding augmenting paths using breadth-first search
  - Augmenting path is shortest path from *s* to *t* in residual network
- Running time is $O(VE^2)$

CLRS

References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd. ed.). The MIT Press.

Chapter 26. Maximum Flow pgs 708 - 730

Erik Demaine, Srini Devadas, and Nancy Lynch. 6.046J Design and Analysis of Algorithms. Spring 2015. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

Lecture 13. Incremental Improvement: Max Flow, Min Cut

Lecture 14. Incremental Improvement: Matching

Back To Back SWE. Network Flows: Max-Flow Min-Cut Theorem (& Ford-Fulkerson Algorithm). Oct 2019. Youtube.