

Computer Science & Engineering 423/823  
Design and Analysis of Algorithms  
Lecture 10 — Maximum Flow (Chapter 26)

Stephen Scott  
(Adapted from Vinodchandran N. Variyam)

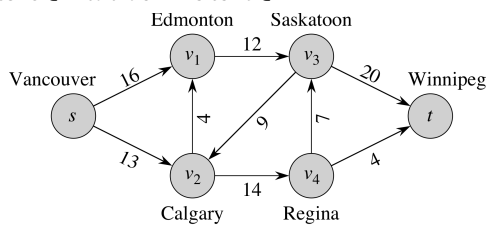
sscott@cse.unl.edu

## Introduction

- Can use a directed graph as a **flow network** to model:
  - Data through communication networks, water/oil/gas through pipes, assembly lines, etc.
- A **flow network** is a directed graph with two special vertices: **source**  $s$  that produces flow and **sink**  $t$  that takes in flow
- Each directed edge is a conduit with a certain capacity (e.g., 200 gallons/hour)
- Vertices are conduit junctions
- Except for  $s$  and  $t$ , flow must be conserved: The flow into a vertex must match the flow out
- Maximum flow problem:** Given a flow network, determine the maximum amount of flow that can get from  $s$  to  $t$
- Other application: Bipartite matching

## Flow Networks

- A **flow network**  $G = (V, E)$  is a directed graph in which each edge  $(u, v) \in E$  has a nonnegative **capacity**  $c(u, v) \geq 0$
- If  $(u, v) \in E$  then  $(v, u) \notin E$  (workaround: Fig 26.2)
- If  $(u, v) \notin E$  then  $c(u, v) = 0$
- No self-loops
- Assume that every vertex in  $V$  lies on some path from the **source** vertex  $s \in V$  to the **sink** vertex  $t \in V$



## Flows

- A **flow** in graph  $G$  is a function  $f : V \times V \rightarrow \mathbb{R}$  that satisfies:
  - Capacity constraint:** For all  $u, v \in V$ ,  $0 \leq f(u, v) \leq c(u, v)$  (flow nonnegative and does not exceed capacity)
  - Flow conservation:** For all  $u \in V \setminus \{s, t\}$ ,

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

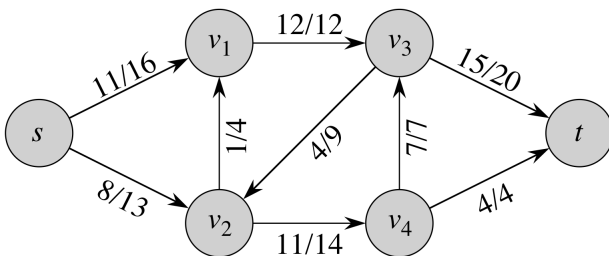
(flow entering a vertex = flow leaving)

- The **value** of a flow is the net flow out of  $s$  (= net flow into  $t$ ):

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

- Maximum flow problem:** given graph and capacities, find a flow of maximum value

## Flow Example

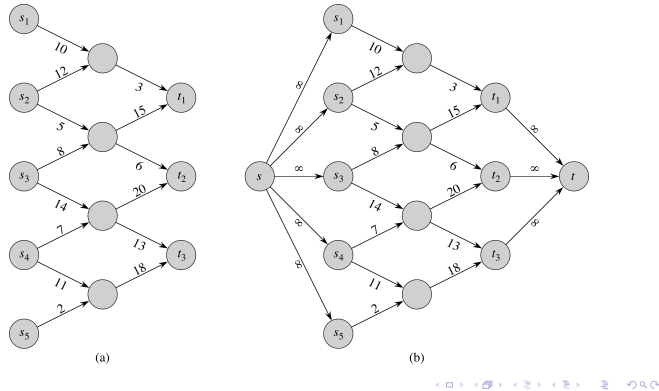


What is the value of this flow?

## Multiple Sources and Sinks

- Might have cases where there are multiple sources and/or sinks; e.g. if there are multiple factories producing products and/or multiple warehouses to ship to
- Can easily accommodate graphs with multiple sources  $s_1, \dots, s_k$  and multiple sinks  $t_1, \dots, t_\ell$
- Add to  $G$  a **supersource**  $s$  with an edge  $(s, s_i)$  for  $i \in \{1, \dots, k\}$  and a **supersink**  $t$  with an edge  $(t_j, t)$  for  $j \in \{1, \dots, \ell\}$
- Each new edge has a capacity of  $\infty$

## Multiple Sources and Sinks (2)



## Ford-Fulkerson Method

- ▶ A **method** (rather than specific algorithm) for solving max flow
- ▶ Multiple ways of implementing, with varying running times
- ▶ Core concepts:
  1. **Residual network:** A network  $G_f$ , which is  $G$  with capacities updated based on the amount of flow  $f$  already going through it
  2. **Augmenting path:** A simple path from  $s$  to  $t$  in residual network  $G_f$   
 $\Rightarrow$  If such a path exists, then can push more flow through network
  3. **Cut:** A partition of  $V$  into  $S$  and  $T$  where  $s \in S$  and  $t \in T$ ; can measure **net flow** and **capacity** crossing a cut
- ▶ Method repeatedly finds an augmenting path in residual network, adds in flow along the path, then updates residual network

## Ford-Fulkerson-Method( $G, s, t$ )

```

1 Initialize flow  $f$  to 0
2 while there exists augmenting path  $p$  in residual network  $G_f$ 
3   do
4     augment flow  $f$  along  $p$ 
5 end
6 return  $f$ 
    
```

## Residual Networks

- ▶ Given flow network  $G$  with capacities  $c$  and flow  $f$ , **residual network**  $G_f$  consists of edges with capacities showing how one can change flow in  $G$
- ▶ Define **residual capacity** of an edge as

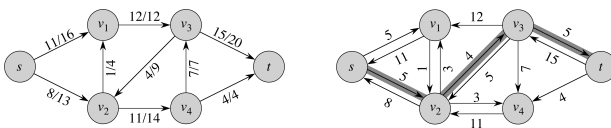
$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (v, u) \in E \\ 0 & \text{otherwise} \end{cases}$$

- ▶ E.g. if  $c(u, v) = 16$  and  $f(u, v) = 11$ , then  $c_f(u, v) = 5$  and  $c_f(v, u) = 11$
- ▶ Then can define  $G_f = (V, E_f)$  as

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

- ▶ So  $G_f$  will have some edges not in  $G$ , and vice-versa

## Residual Networks (2)



## Flow Augmentation

- ▶  $G_f$  is like a flow network (except that it can have an edge and its reversal); so we can find a flow within it
- ▶ If  $f$  is a flow in  $G$  and  $f'$  is a flow in  $G_f$ , can define the **augmentation** of  $f$  by  $f'$  as

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

- ▶ **Lemma:**  $f \uparrow f'$  is a flow in  $G$  with value  $|f \uparrow f'| = |f| + |f'|$
- ▶ **Proof:** Show that  $f \uparrow f'$  satisfies capacity constraint and flow conservation; then show that  $|f \uparrow f'| = |f| + |f'|$  (pp. 718–719) □
- ▶ **Result:** If we can find a flow  $f'$  in  $G_f$ , we can increase flow in  $G$

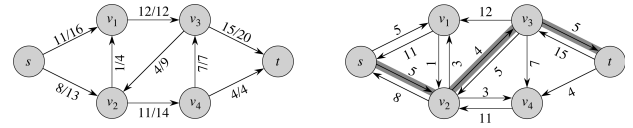
## Augmenting Path

- By definition of residual network, an edge  $(u, v) \in E_f$  with  $c_f(u, v) > 0$  can handle additional flow
- Since edges in  $E_f$  all have positive residual capacity, it follows that if there is a simple path  $p$  from  $s$  to  $t$  in  $G_f$ , then we can increase flow along each edge in  $p$ , thus increasing total flow
- We call  $p$  an **augmenting path**
- The amount of flow we can put on  $p$  is  $p$ 's residual capacity:

$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}$$

Navigation icons

## Augmenting Path (2)



$p$  is shaded; what is  $c_f(p)$ ?

Navigation icons

## Augmenting Path (3)

- Lemma:** Let  $G = (V, E)$  be a flow network,  $f$  be a flow in  $G$ , and  $p$  be an augmenting path in  $G_f$ . Define  $f_p : V \times V \rightarrow \mathbb{R}$  as

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \in p \\ 0 & \text{otherwise} \end{cases}$$

Then  $f_p$  is a flow in  $G_f$  with value  $|f_p| = c_f(p) > 0$

- Corollary:** Let  $G, f, p$ , and  $f_p$  be as above. Then  $f \uparrow f_p$  is a flow in  $G$  with value  $|f \uparrow f_p| = |f| + |f_p| > |f|$
- Thus, every augmenting path increases flow in  $G$
- When do we stop? Will we have a maximum flow if there is no augmenting path?

Navigation icons

## Max-Flow Min-Cut Theorem

- Used to prove that once we run out of augmenting paths, we have a maximum flow
- A **cut**  $(S, T)$  of a flow network  $G = (V, E)$  is a partition of  $V$  into  $S \subseteq V$  and  $T = V \setminus S$  such that  $s \in S$  and  $t \in T$
- Net flow** across the cut  $(S, T)$  is

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

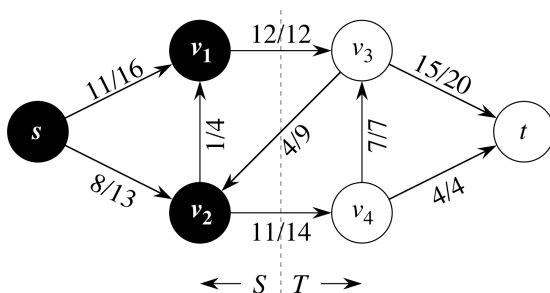
- Capacity** of cut  $(S, T)$  is

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

- A **minimum cut** is one whose capacity is smallest over all cuts

Navigation icons

## Max-Flow Min-Cut Theorem (2)



What are  $f(S, T)$  and  $c(S, T)$ ?

Navigation icons

## Max-Flow Min-Cut Theorem (3)

- Lemma:** For any flow  $f$ , the value of  $f$  is the same as the net flow across any cut; i.e.  $f(S, T) = |f|$  for all cuts  $(S, T)$
- Corollary:** The value of any flow  $f$  in  $G$  is upperbounded by the capacity of **any** cut of  $G$
- Proof:**

$$\begin{aligned} |f| &= f(S, T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \\ &\leq \sum_{u \in S} \sum_{v \in T} f(u, v) \\ &\leq \sum_{u \in S} \sum_{v \in T} c(u, v) \\ &= c(S, T) \end{aligned}$$

□

Navigation icons

## Max-Flow Min-Cut Theorem (4)

► **Max-Flow Min-Cut Theorem:** If  $f$  is a flow in flow network  $G$ , then these statements are equivalent:

1.  $f$  is a maximum flow in  $G$
2.  $G_f$  has no augmenting paths
3.  $|f| = c(S, T)$  for some (i.e. minimum) cut  $(S, T)$  of  $G$

► **Proof:** Show (1)  $\Rightarrow$  (2)  $\Rightarrow$  (3)  $\Rightarrow$  (1)

► (1)  $\Rightarrow$  (2): If  $G_f$  has augmenting path  $p$ , then  $f_p > 0$  and  $|f \uparrow f_p| = |f| + |f_p| > |f|$ , a contradiction

## Max-Flow Min-Cut Theorem (5)

► (2)  $\Rightarrow$  (3): Assume  $G_f$  has no path from  $s$  to  $t$  and define  $S = \{v \in V : s \rightsquigarrow v \text{ in } G_f\}$  and  $T = V \setminus S$

►  $(S, T)$  is a cut since it partitions  $V$ ,  $s \in S$  and  $t \in T$

► Consider  $u \in S$  and  $v \in T$ :

► If  $(u, v) \in E$ , then  $f(u, v) = c(u, v)$  since otherwise  $c_f(u, v) > 0 \Rightarrow (u, v) \in E_f \Rightarrow v \in S$

► If  $(v, u) \in E$ , then  $f(v, u) = 0$  since otherwise we'd have  $c_f(u, v) = f(v, u) > 0 \Rightarrow (u, v) \in E_f \Rightarrow v \in S$

► If  $(u, v) \notin E$  and  $(v, u) \notin E$ , then  $f(u, v) = f(v, u) = 0$

► Thus (by applying the Lemma as well)

$$\begin{aligned} |f| = f(S, T) &= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in T} \sum_{u \in S} f(v, u) \\ &= \sum_{u \in S} \sum_{v \in T} c(u, v) - \sum_{v \in T} \sum_{u \in S} 0 = c(S, T) \end{aligned}$$

## Max-Flow Min-Cut Theorem (6)

► (3)  $\Rightarrow$  (1):

► Corollary says that  $|f| \leq c(S', T')$  for all cuts  $(S', T')$

► We've established that  $|f| = c(S, T)$

$\Rightarrow |f|$  can't be any larger  
 $\Rightarrow f$  is a maximum flow

□

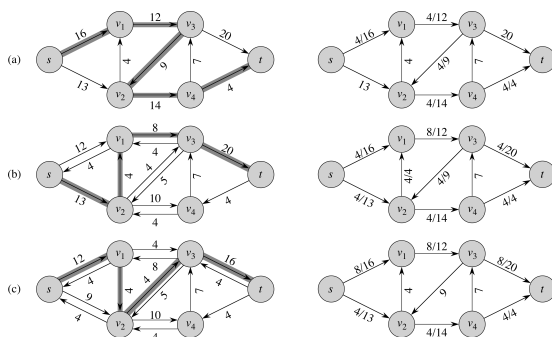
## Ford-Fulkerson( $G, s, t$ )

```

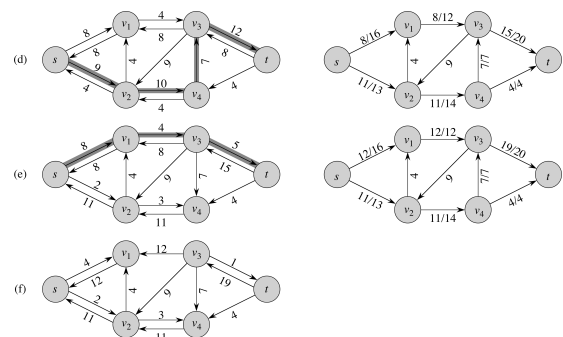
1 for each edge  $(u, v) \in E$  do
2    $f(u, v) = 0$ 
3 end
4 while there exists path  $p$  from  $s$  to  $t$  in  $G_f$  do
5    $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$ 
6   for each edge  $(u, v) \in p$  do
7     if  $(u, v) \in E$  then
8        $f(u, v) = f(u, v) + c_f(p)$ 
9     else
10       $f(v, u) = f(v, u) - c_f(p)$ 
11    end
12  end
13 end

```

## Ford-Fulkerson Example



## Ford-Fulkerson Example (2)



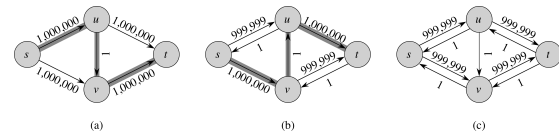
## Analysis of Ford-Fulkerson

- Assume all of  $G$ 's capacities are integers
  - If not, but values still rational, can scale them
  - If values irrational, might not converge ☹
- If we choose augmenting path arbitrarily, then  $|f|$  increases by at least one unit per iteration  $\Rightarrow$  number of iterations is  $\leq |f^*| = \text{value of max flow}$
- $|E_f| \leq 2|E|$
- Every vertex is on a path from  $s$  to  $t \Rightarrow |V| = O(|E|)$
- $\Rightarrow$  Finding augmenting path via BFS or DFS takes time  $O(|E|)$ , as do initialization and each augmentation step
- Total time complexity:  $O(|E||f^*|)$
- Not polynomial in size of input! (What is size of input?)

◀ ▶ ↺ ↻ 🔍

## Example of Large $|f^*|$

Arbitrary choice of augmenting path can result in small increase in  $|f|$  each step



Takes  $2 \times 10^6$  augmentations

◀ ▶ ↺ ↻ 🔍

## Edmonds-Karp Algorithm

- Uses Ford-Fulkerson Method
- Rather than arbitrary choice of augmenting path  $p$  from  $s$  to  $t$  in  $G_f$ , choose one that is shortest in terms of number of edges
  - How can we easily do this?
- Will show time complexity of  $O(|V||E|^2)$ , independent of  $|f^*|$
- Proof based on  $\delta_f(u, v)$ , which is length of shortest path from  $u$  to  $v$  in  $G_f$ , in terms of number of edges
- Lemma:** When running Edmonds-Karp on  $G$ , for all vertices  $v \in V \setminus \{s, t\}$ , shortest path distance  $\delta_f(u, v)$  in  $G_f$  increases monotonically with each flow augmentation

◀ ▶ ↺ ↻ 🔍

## Edmonds-Karp Algorithm (2)

- Theorem:** When running Edmonds-Karp on  $G$ , the total number of flow augmentations is  $O(|V||E|)$
- Proof:** Call an edge  $(u, v)$  **critical** on augmenting path  $p$  if  $c_f(p) = c_f(u, v)$
- When  $(u, v)$  is critical for the first time,  $\delta_f(s, v) = \delta_f(s, u) + 1$
- At the same time,  $(u, v)$  disappears from residual network and does not reappear until its flow decreases, which only happens when  $(v, u)$  appears on an augmenting path, at which time

$$\begin{aligned} \delta_{f'}(s, u) &= \delta_f(s, v) + 1 \\ &\geq \delta_f(s, v) + 1 \text{ (from Lemma)} \\ &= \delta_f(s, u) + 2 \end{aligned}$$

- Thus, from the time  $(u, v)$  becomes critical to the next time it does,  $u$ 's distance from  $s$  increases by at least 2

◀ ▶ ↺ ↻ 🔍

## Edmonds-Karp Algorithm (3)

- Since  $u$ 's distance from  $s$  is at most  $|V| - 2$  (because  $u \neq t$ ) and at least 0, edge  $(u, v)$  can be critical at most  $|V|/2$  times
- There are at most  $2|E|$  edges that can be critical in a residual network
- Every augmentation step has at least one critical edge
- $\Rightarrow$  Number of augmentation steps is  $O(|V||E|)$ , instead of  $O(|f^*|)$  in previous algorithm
- $\Rightarrow$  Edmonds-Karp time complexity is  $O(|V||E|^2)$  □

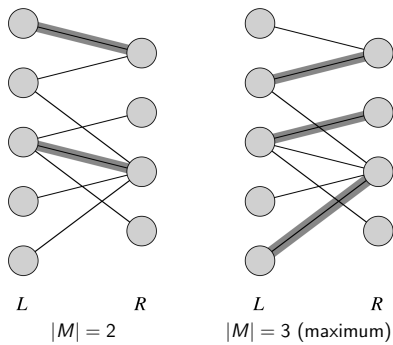
◀ ▶ ↺ ↻ 🔍

## Maximum Bipartite Matching

- In an undirected graph  $G = (V, E)$ , a **matching** is a subset of edges  $M \subseteq E$  such that for all  $v \in V$ , at most one edge from  $M$  is incident on  $v$
- If an edge from  $M$  is incident on  $v$ ,  $v$  is **matched**, otherwise **unmatched**
- Problem:** Find a matching of maximum cardinality
- Special case:**  $G$  is **bipartite**, meaning  $V$  partitioned into disjoint sets  $L$  and  $R$  and all edges of  $E$  go between  $L$  and  $R$
- Applications:** Matching machines to tasks, arranging marriages between interested parties, etc.

◀ ▶ ↺ ↻ 🔍

## Bipartite Matching Example



## Casting Bipartite Matching as Max Flow

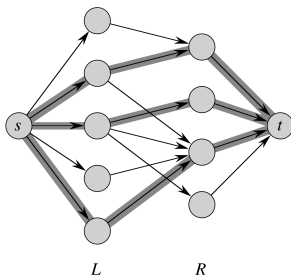
- ▶ Can cast bipartite matching problem as max flow
- ▶ Given bipartite graph  $G = (V, E)$ , define **corresponding flow network**  $G' = (V', E')$ :

$$V' = V \cup \{s, t\}$$

$$E' = \{(s, u) : u \in L\} \cup \{(u, v) : (u, v) \in E\} \cup \{(v, t) : v \in R\}$$

- ▶  $c(u, v) = 1$  for all  $(u, v) \in E'$

## Casting Bipartite Matching as Max Flow (2)



Value of flow across cut  $(L \cup \{s\}, R \cup \{t\})$  equals  $|M|$

## Casting Bipartite Matching as Max Flow (3)

- ▶ **Lemma:** Let  $G = (V, E)$  be a bipartite graph with  $V$  partitioned into  $L$  and  $R$  and let  $G' = (V', E')$  be its corresponding flow network. If  $M$  is a matching in  $G$ , then there is an **integer-valued flow**  $f$  in  $G'$  with value  $|f| = |M|$ . Conversely, if there is an integer-valued flow  $f$  in  $G'$ , then there is a matching  $M$  in  $G$  with cardinality  $|M| = |f|$ .

- ▶ **Proof:**  $\Rightarrow$  If  $(u, v) \in M$ , set  $f(s, u) = f(u, v) = f(v, t) = 1$ 
  - ▶ Set flow of all other edges to 0
  - ▶ Flow satisfies capacity constraint and flow conservation
  - ▶ Flow across cut  $(L \cup \{s\}, R \cup \{t\})$  is  $|M|$
- ▶  $\Leftarrow$  Let  $f$  be integer-valued flow in  $G'$ , and set

$$M = \{(u, v) : u \in L, v \in R, f(u, v) > 0\}$$

- ▶ Any flow into  $u$  must be exactly 1 in and exactly 1 out on one edge
- ▶ Similar argument for  $v \in R$ , so  $M$  is a matching with  $|M| = |f|$  □

## Casting Bipartite Matching as Max Flow (4)

- ▶ **Theorem:** If all edges in a flow network have integral capacities, then the Ford-Fulkerson method returns a flow with value that is an integer, and for all  $(u, v) \in V$ ,  $f(u, v)$  is an integer
- ▶ Since the corresponding flow network for bipartite matching uses all integer capacities, can use Ford-Fulkerson to solve matching problem
- ▶ Any matching has cardinality  $O(|V|)$ , so the corresponding flow network has a maximum flow with value  $|f^*| = O(|V|)$ , so time complexity of matching is  $O(|V||E|)$