COMS21103: Maximum Flow

Dima Damen
Dima.Damen@bristol.ac.uk

Bristol University, Department of Computer Science
Bristol BS8 1UB, UK

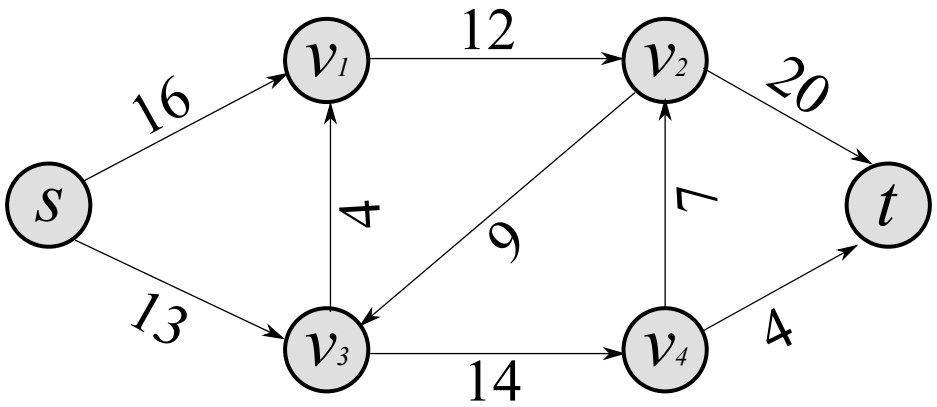
November 26, 2015

Flow Networks

- Directed graphs can also be interpreted as **flow networks**
- In a flow network, exactly one node is referred to as a **source**, and one node as a **sink**.
- The aim is to push the maximum amount of *material* from the source to the sink.

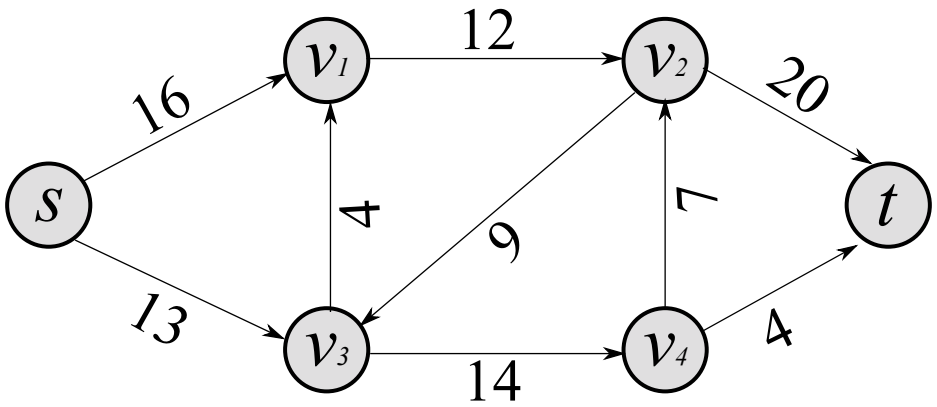
Flow Network - Definitions

A *flow network* $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative *capacity* $c(u, v) \geq 0$. We distinguish two vertices in a flow network: a ***source*** s and a ***sink*** t.



Flow Network - Definitions

A *flow network* $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative *capacity* $c(u, v) \geq 0$. We distinguish two vertices in a flow network: a ***source*** s and a ***sink*** t.



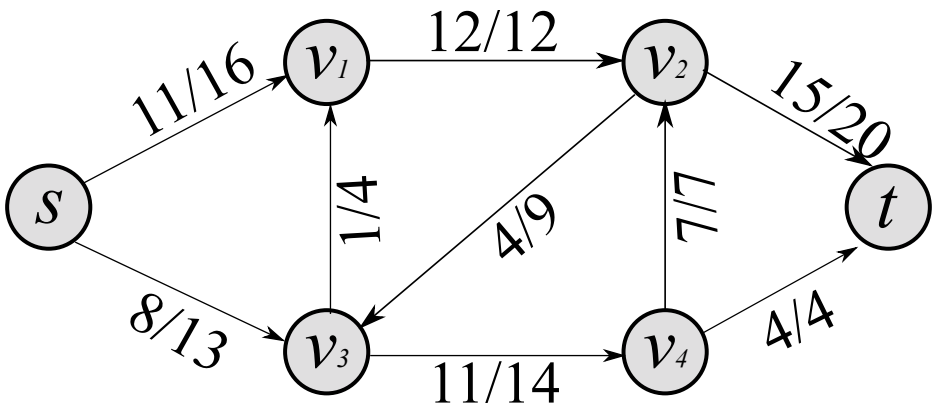
The capacity could be represented by a real-valued function $c : V \times V \rightarrow \mathbb{R}^+$

** edges are removed from visualisation when $c(u, v) = 0$

Flow Network - Definitions

A *flow f* in the *network G* is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies two conditions

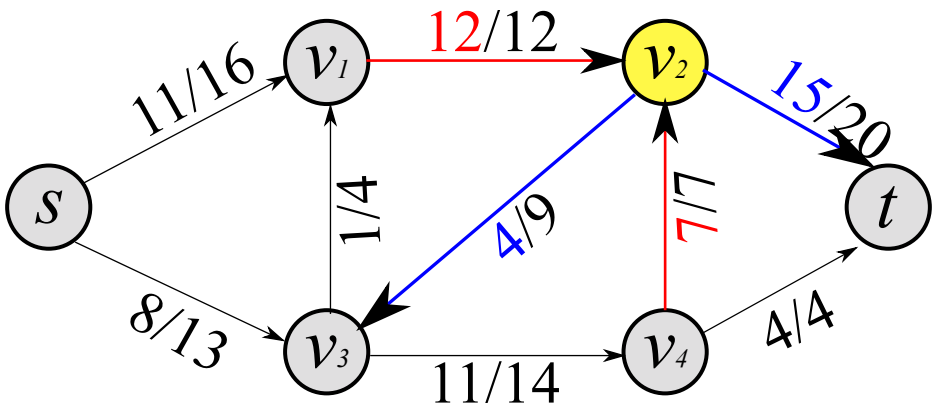
- $\forall u, v \in V \quad \Rightarrow \quad 0 \leq f(u, v) \leq c(u, v)$
- $\forall u \in V - \{s, t\} \quad \Rightarrow \quad \sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$



Flow Network - Definitions

A *flow f* in the *network G* is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies two conditions

- $\forall u, v \in V \quad \Rightarrow \quad 0 \leq f(u, v) \leq c(u, v)$
- $\forall u \in V - \{s, t\} \quad \Rightarrow \quad \sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$

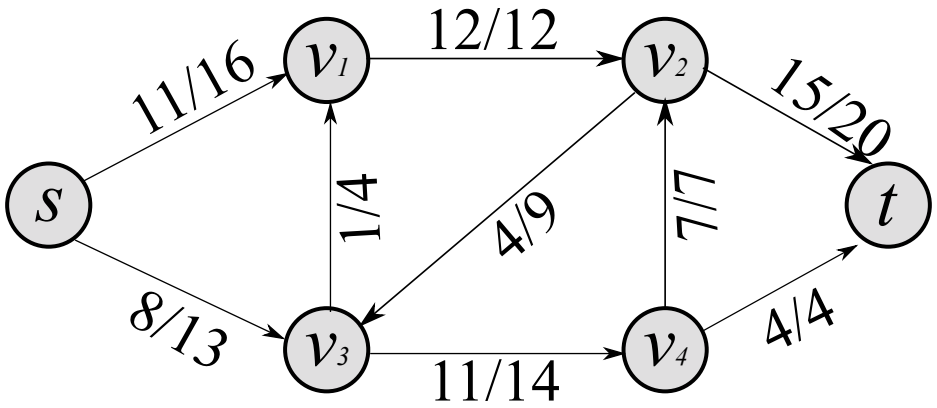


Flow Network - Definitions

The **value** $|f|$ of the flow function f is defined as

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

is the total flow out of the source minus the flow into the source



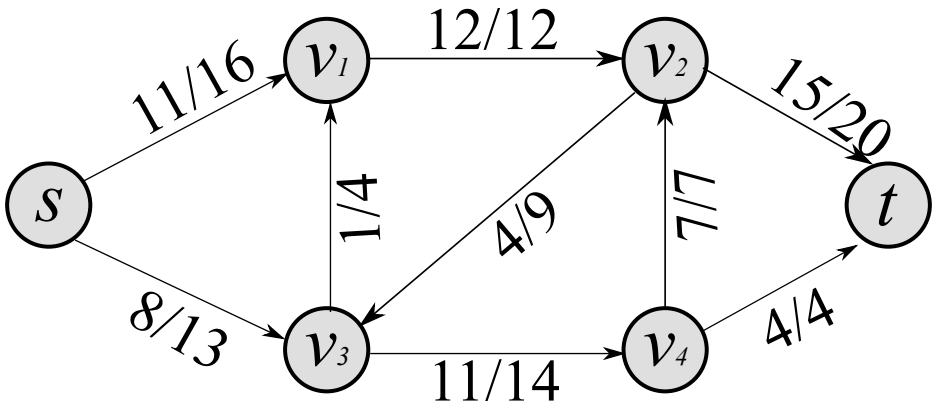
Flow Network - Definitions

The **value** $|f|$ of the flow function f is defined as

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

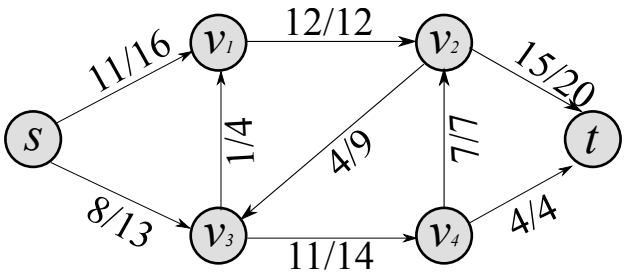
is the total flow out of the source minus the flow into the source

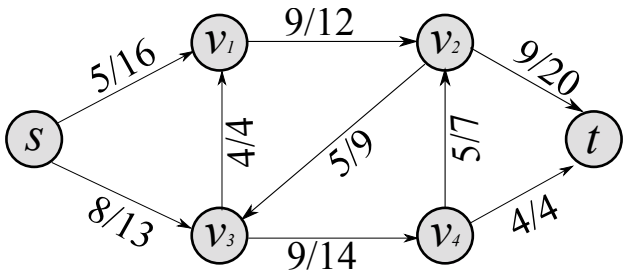
For the flow f below, $|f| = f(s, v_1) + f(s, v_3) = 19$



Flow Network - Definitions

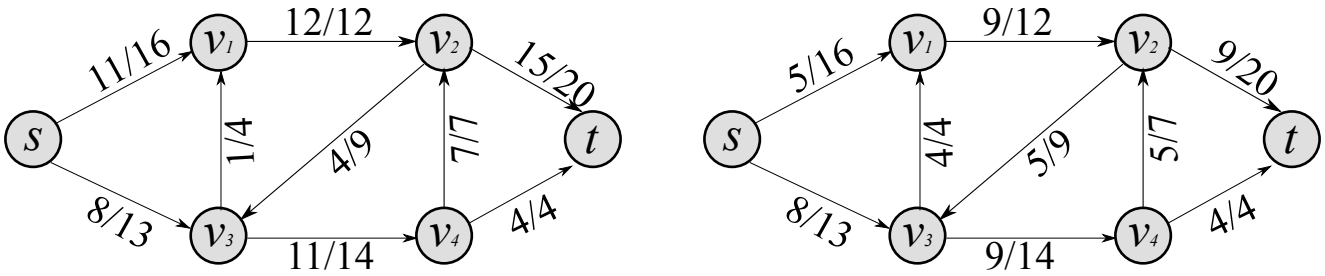
Multiple flow functions f_1, f_2 could be defined for the same flow network





Flow Network - Definitions

Multiple flow functions f_1, f_2 could be defined for the same flow network



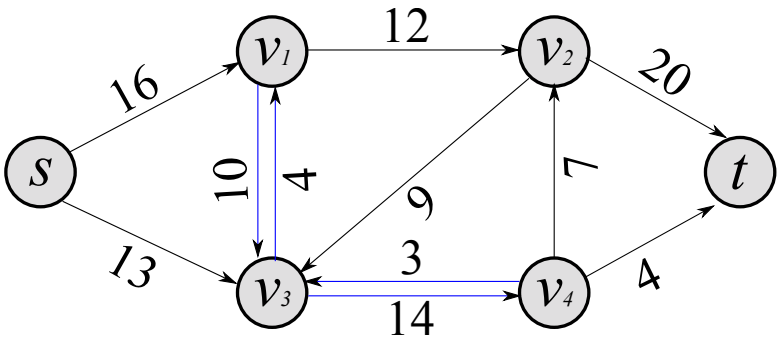
Finding the function with the **maximum flow** is making the most of a flow network with certain capacities.

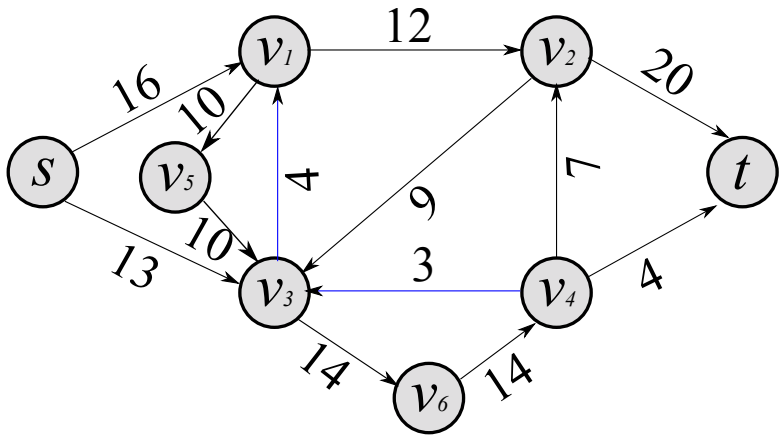
Max-Flow Problem

In the ***maximum-flow problem***, we are given a flow network G with source s and sink t, and we wish to find a flow of maximum value.

Max-Flow Problem

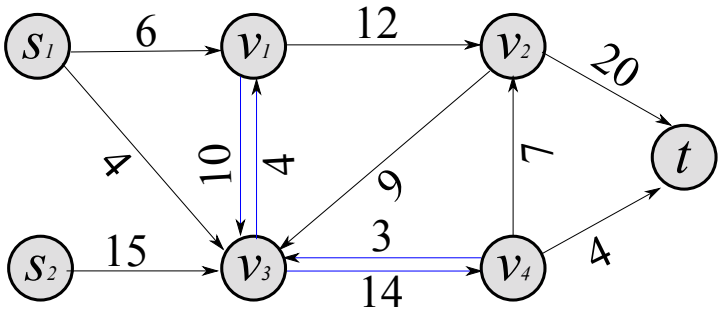
Even though the number of problems you can think of, with one source, one sink and non-antiparallel edges are limited. It is easy to re-formulate these cases as a flow network.

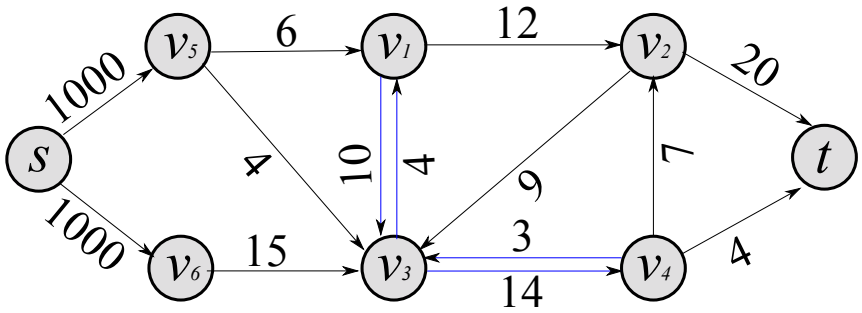




Max-Flow Problem

Even though the number of problems you can think of, with one source, one sink and non-antiparallel edges are limited. It is easy to re-formulate these cases as a flow network.





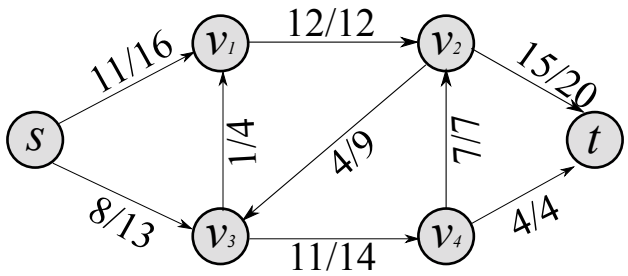
Ford-Fulkerson Method - Definitions

Given a flow network $G = (V, E)$ and a flow f, the ***residual network*** of G induced by f is $G_f = (V, E_f)$ where

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

where

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



f

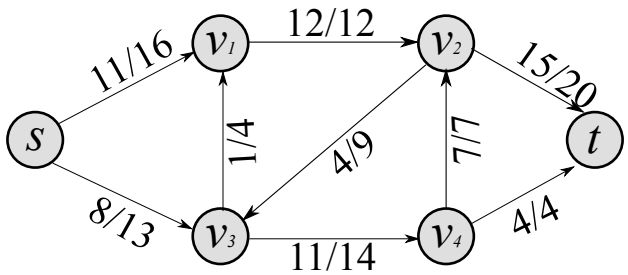
Ford-Fulkerson Method - Definitions

Given a flow network $G = (V, E)$ and a flow f, the ***residual network*** of G induced by f is $G_f = (V, E_f)$ where

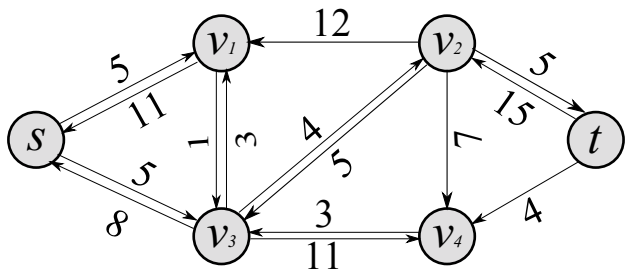
$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

where

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



f



G_f

Ford-Fulkerson Method - Definitions

An ***augmenting path*** p is a simple path from s to t in the residual network G_f.

The ***residual capacity*** of the augmenting path p is given by

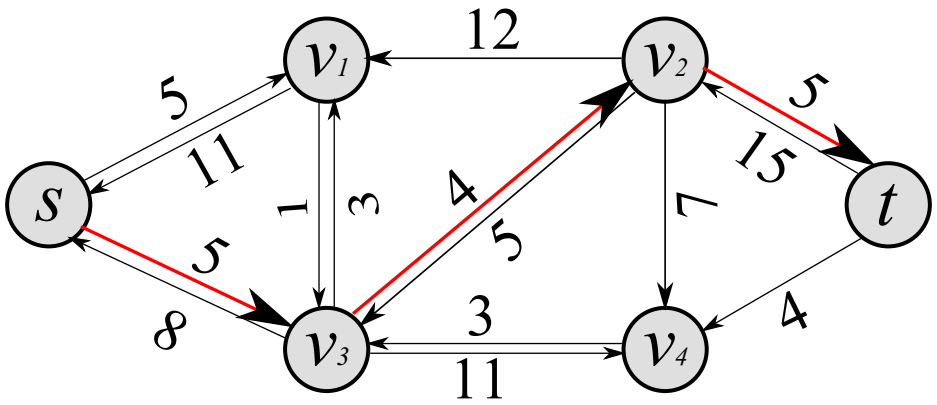
$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on p}\}$$

Ford-Fulkerson Method - Definitions

An ***augmenting path*** p is a simple path from s to t in the residual network G_f.

The ***residual capacity*** of the augmenting path p is given by

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

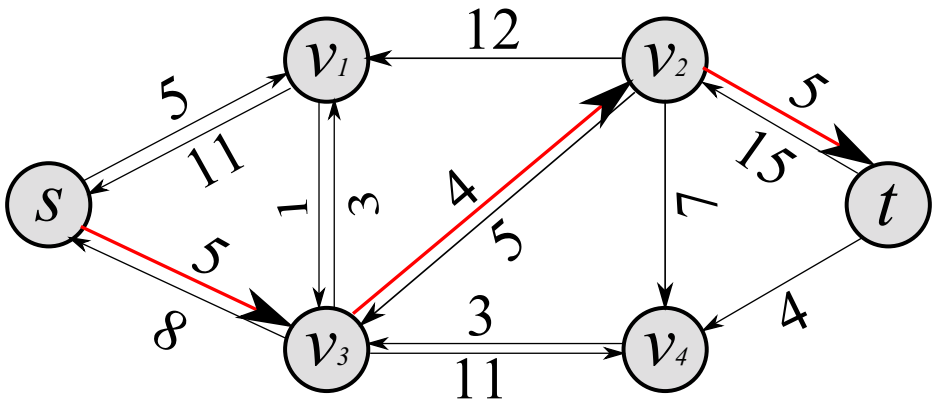


Ford-Fulkerson Method - Definitions

An ***augmenting path*** p is a simple path from s to t in the residual network G_f.

The ***residual capacity*** of the augmenting path p is given by

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



In the example above, $c_f(p) = \min\{5, 4, 5\} = 4$

Ford-Fulkerson Method - Definitions

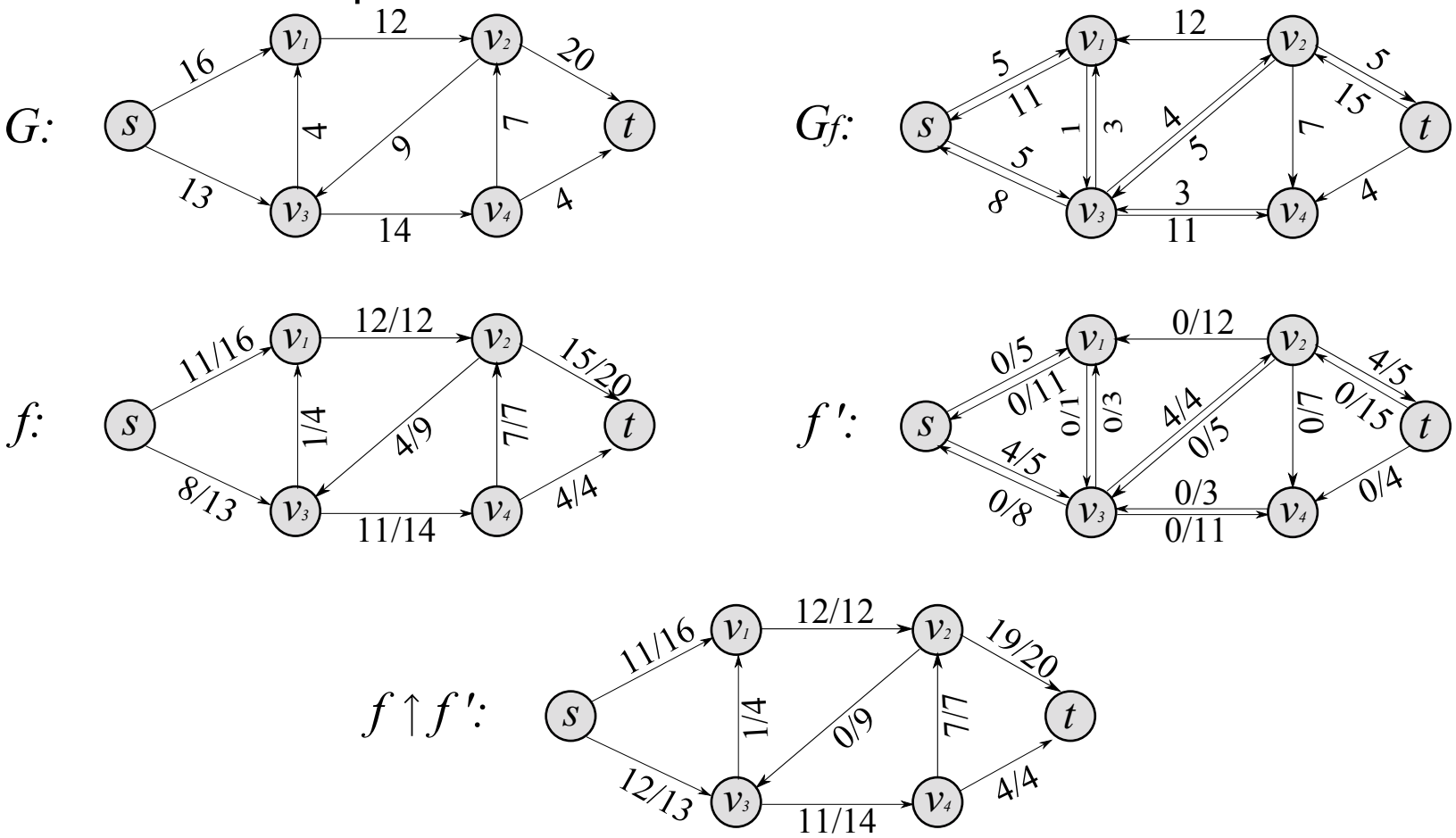
If f is a flow in G and f' is a flow in the corresponding residual network G_f, then $f \uparrow f'$ is the ***augmentation*** of flow f by f' defined as:

$f \uparrow f' : V \times V \Rightarrow \mathbb{R}$ such that

$$f \uparrow f' = f(u, v) + f'(u, v) - f'(v, u)$$

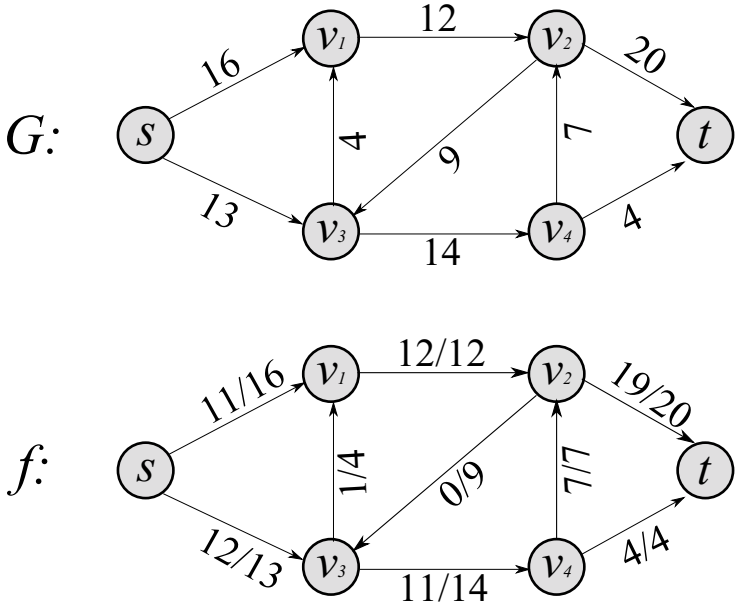
Ford-Fulkerson Method - Definitions

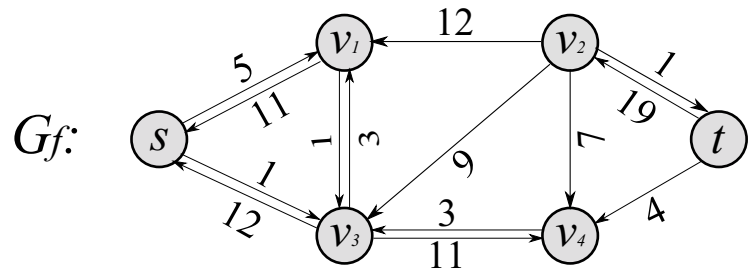
Augmentation example:



Ford-Fulkerson Method - Definitions

Augmentation example - continued:



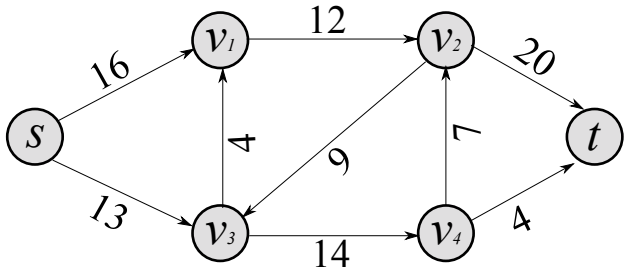


Ford-Fulkerson Method - Basic Algorithm

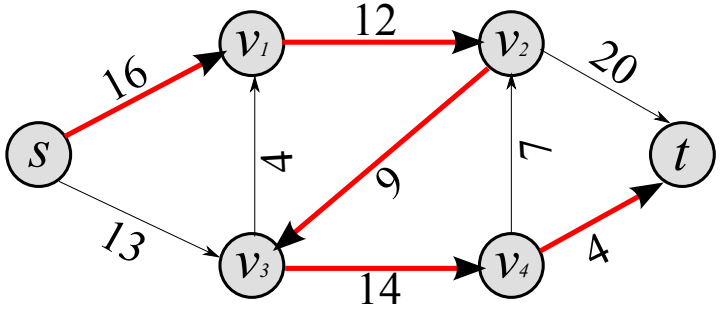
FORD-FULKERSON-METHOD(G, s, t)
begin
    initialise flow f to 0
    while *there exists an augmentation path p in the residual network G_f* **do**
        augment flow f along p
    end
end

Ford-Fulkerson Method - Example

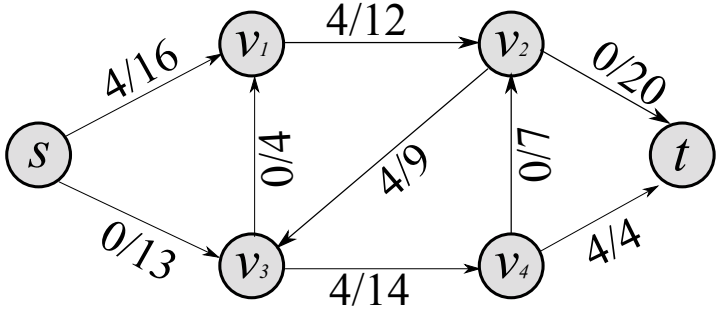
G:



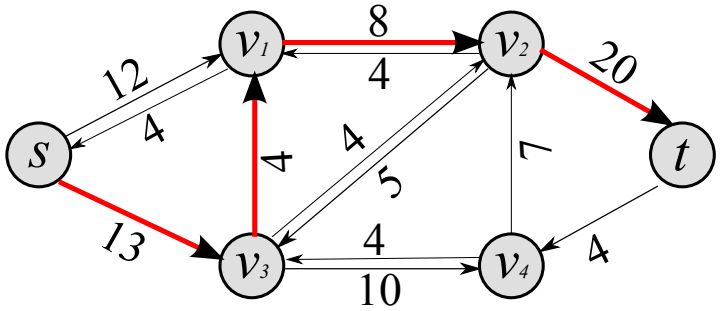
(1) G_f:



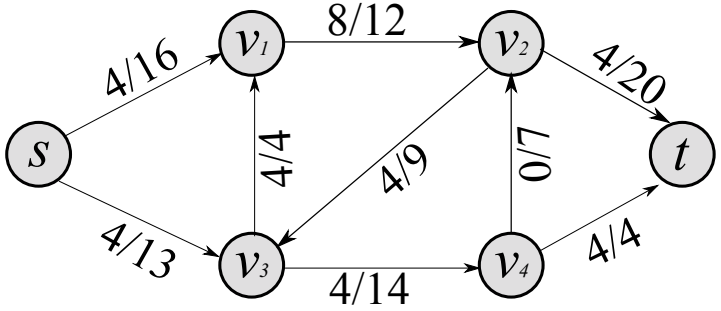
f:



(2) G_f:

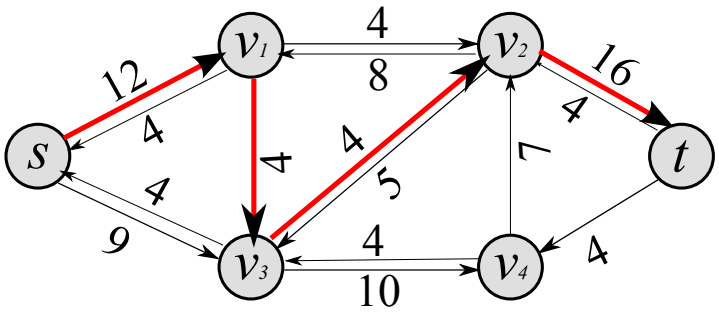


f:

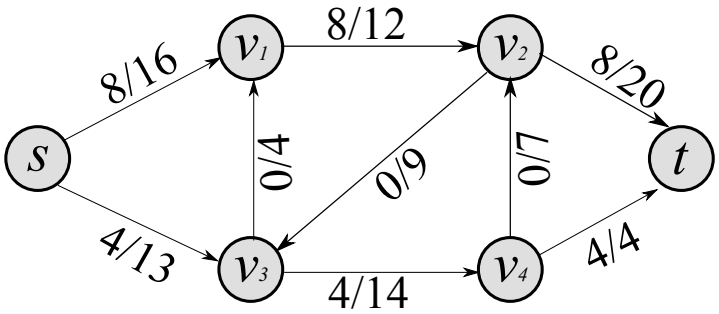


Ford-Fulkerson Method - Example

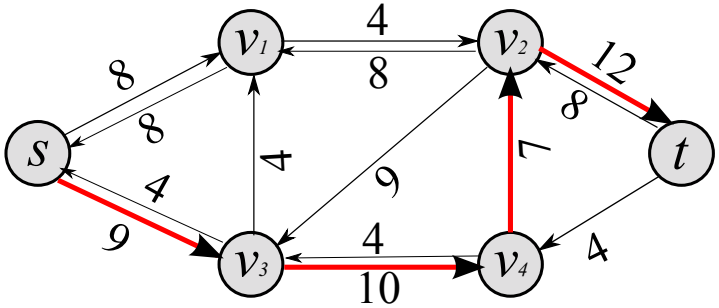
(3) G_f:



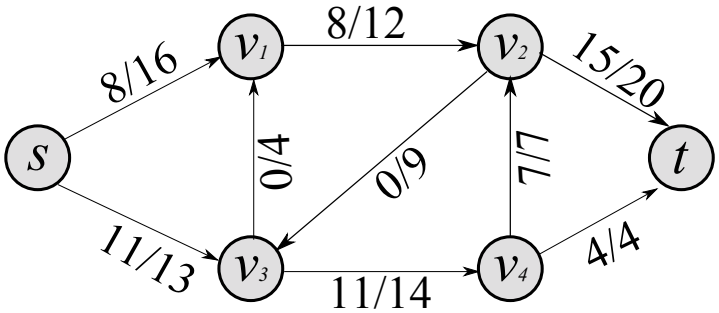
f:



(4) G_f:

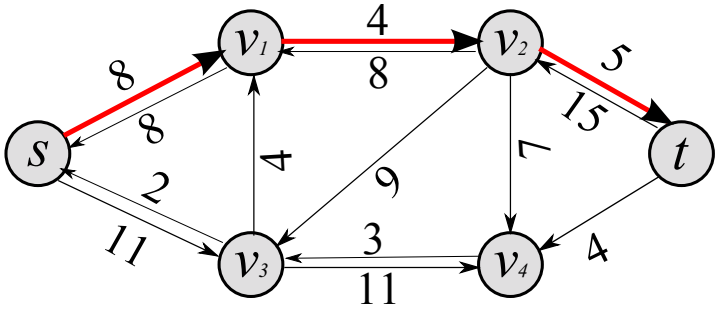


f:

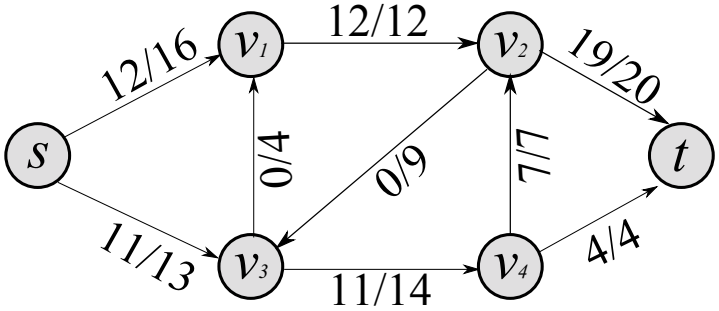


Ford-Fulkerson Method - Example

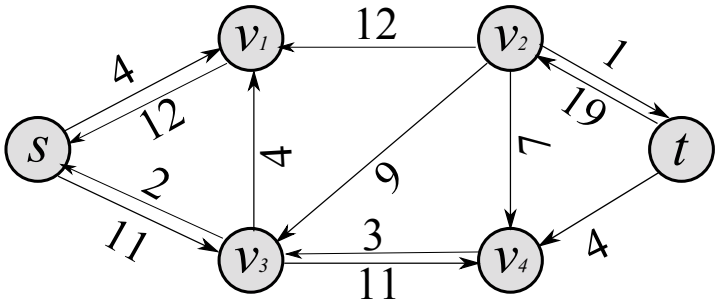
(5) G_f:



f:



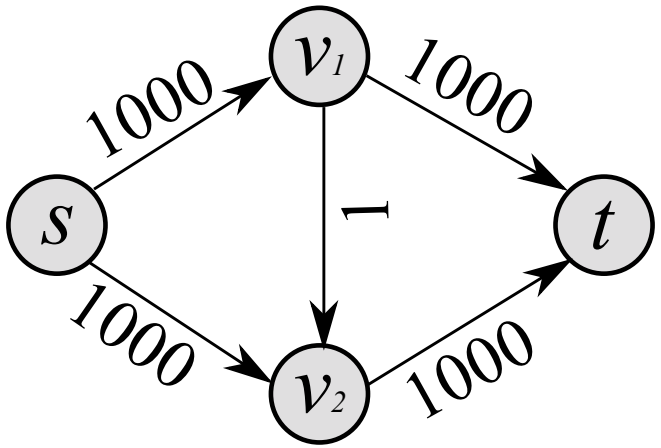
(6) G_f:



Basic Algorithm - Time Analysis

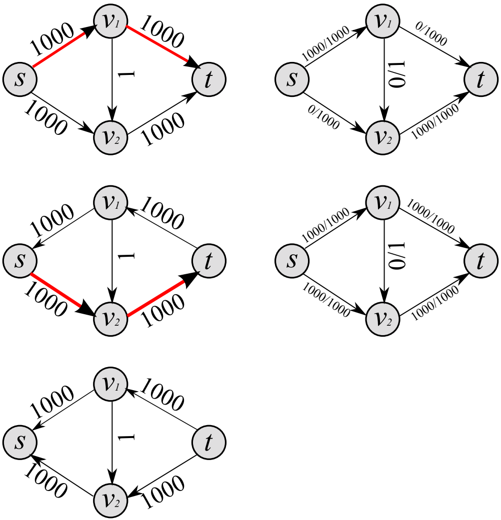
- running time depends on the order of selecting augmenting paths p.
- if we choose poorly ...

Ex.



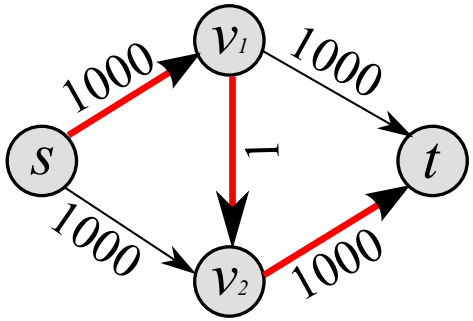
Basic Algorithm - Time Analysis

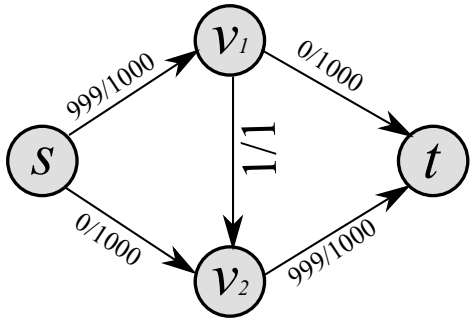
Good Choice:

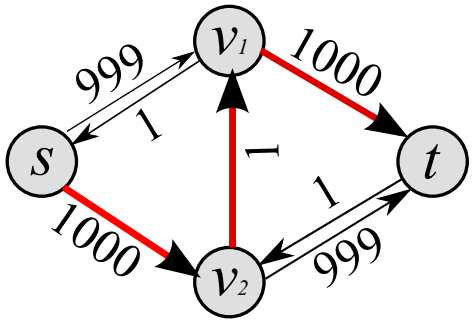


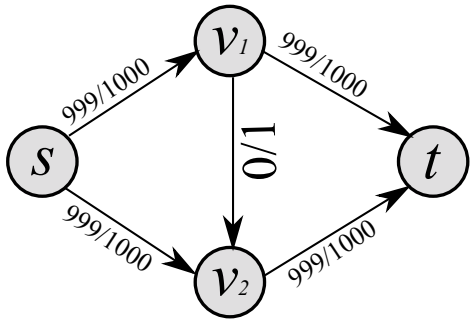
Basic Algorithm - Time Analysis

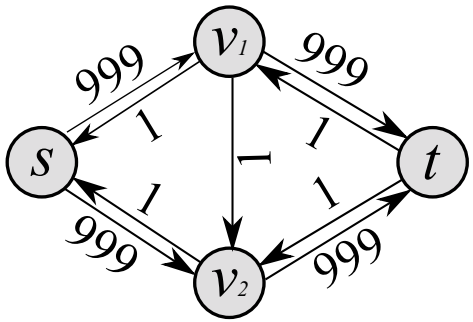
Bad Choice:











Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.
  - For the max flow f^*, the maximum value $|f^*|$ denotes a bound in the number of iterations

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.
  - For the max flow f^*, the maximum value $|f^*|$ denotes a bound in the number of iterations
- The total running time is thus

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.
  - For the max flow f^*, the maximum value $|f^*|$ denotes a bound in the number of iterations
- The total running time is thus $O(|f^*|E)$

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.
  - For the max flow f^*, the maximum value $|f^*|$ denotes a bound in the number of iterations
- The total running time is thus $O(|f^*|E)$
- What would you do if the minimum capacity in your network is 1000 units?

Basic Algorithm - Time Analysis

- running time depends on the order of selecting augmenting paths p.
- if we choose poorly...
- To prove the time bound:
  - each iteration takes E time, where E is the number of edges in the network... why?
  - the flow network increases by at least one unit in each iteration.
  - For the max flow f^*, the maximum value $|f^*|$ denotes a bound in the number of iterations
- The total running time is thus $O(|f^*|E)$
- What would you do if the minimum capacity in your network is 1000 units?
- We still prefer an algorithm that is independent of the value of the maximum flow (which we don't know in advance!)

Edmonds-Karp Algorithm

- Choose the augmenting path as the *shortest path* from s to t at each step
- Let $\delta_f(u, v)$ be the shortest-path distance from u to v in G_f, where each edge has a unit distance.

Lemma

For a network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest path $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Lemma

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, the total number of augmentations performed by the algorithm is $O(VE)$.

Edmonds-Karp Algorithm

- Choose the augmenting path as the *shortest path* from s to t at each step
- Let $\delta_f(u, v)$ be the shortest-path distance from u to v in G_f, where each edge has a unit distance.
- Note: the shortest path is the one with minimum number of edges, not the one with minimum capacity

Lemma

For a network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest path $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Lemma

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, the total number of augmentations performed by the algorithm is $O(VE)$.

Edmonds-Karp Algorithm

Proof.

- Each augmentation path has at least one critical edge that disappears from the residual network.
- If the edge (u, v) disappears, it cannot reappear until the flow from u to v is decreased.

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

□

Edmonds-Karp Algorithm

Proof.

continued..

- From the time (u, v) becomes critical, to the time it next becomes critical, the distance from s to u increases by at least 2
- Until u becomes unreachable from the source, if ever, its distance would be at most $|V| - 2$
- After the first time, it becomes critical at most $(|V| - 2)/2$ times.
- There are at most $O(E)$ pairs of vertices in the residual network.
- Total number of critical edges is $O(VE)$

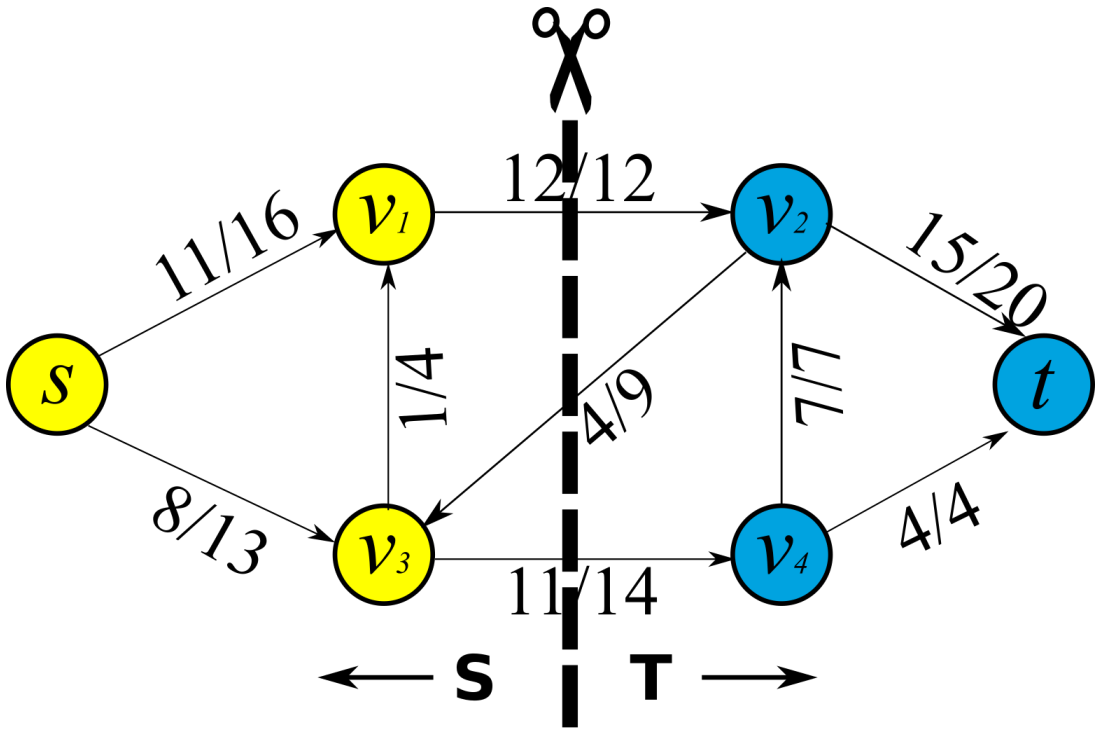
□

Edmonds-Karp Algorithm

- Total number of critical edges, and thus number of iterations, is $O(VE)$
- Each iteration runs in $O(E)$ times
- Total running time of the Edmonds-Karp algorithm is $O(VE^2)$

Cuts of Flow Networks - Definition

A ***cut*** (S, T) of flow network $G = (V, E)$ is a partition of vertices V into S and $T = V - S$ such that $s \in S$ and $t \in T$.



Cuts of Flow Networks - Definition

If f is a flow, then the ***net flow*** $f(S, T)$ across the cut is defined to be

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

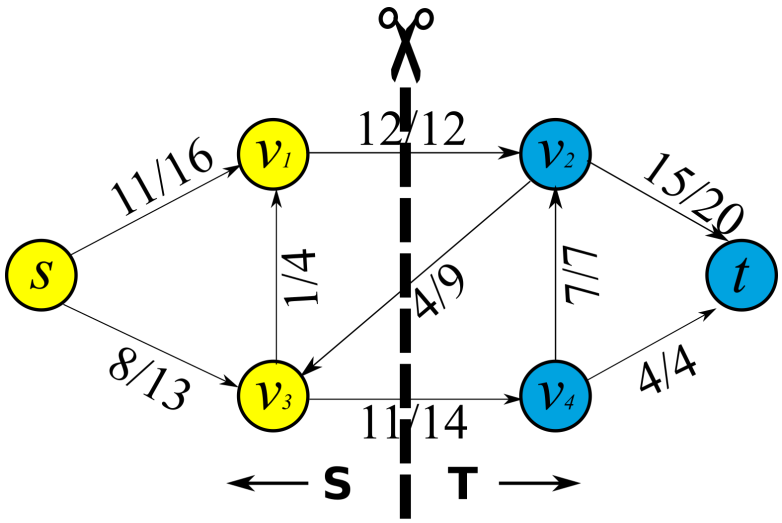
The ***capacity*** of the cut is

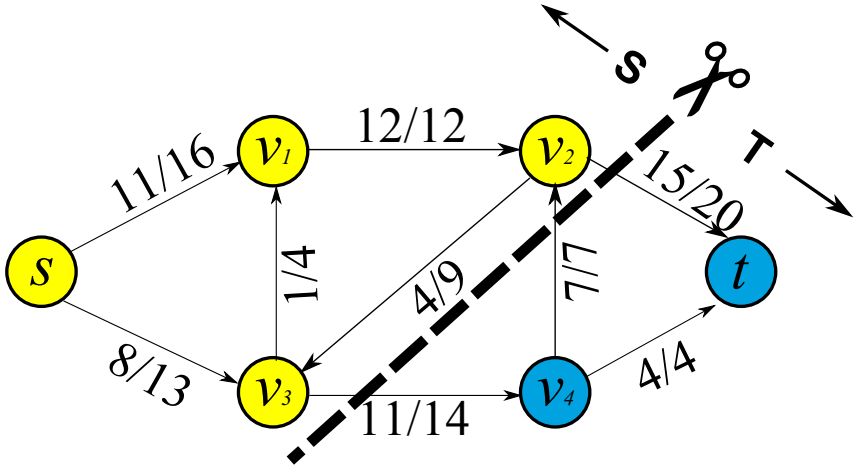
$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

The ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network.

Cuts of Flow Networks - Definition

Calculate the *net flow* and the *capacity* of these two cuts...





What do you conclude?

Cuts of Flow Networks - Definition

Lemma

Let f be a flow in a flow network G with source s and sink t, and let (S, T) be any cut of G, then $f(S, T) = |f|$

Cuts of Flow Networks - Definition

Proof.

$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) \quad (1)$$

(2)

(3)

(4)

(5)

(6)

$$= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) \quad (7)$$

$$= f(S,T) \quad (8)$$

Cuts of Flow Networks - Definition

Proof.

$$
\begin{aligned}
|f| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) && (1)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S - \{s\}} \left(\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) \right) && (2)\\
& && (3)\\
& && (4)\\
& && (5)\\
& && (6)\\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) && (7)\\
&= f(S,T) && (8)
\end{aligned}
$$

Cuts of Flow Networks - Definition

Proof.

$$
\begin{aligned}
|f| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) && (1)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \Big(\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) \Big) && (2)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \sum_{v \in V} f(u,v) - \sum_{u \in S-\{s\}} \sum_{v \in V} f(v,u) && (3)\\
& && (4)\\
& && (5)\\
& && (6)\\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) && (7)\\
&= f(S,T) && (8)
\end{aligned}
$$

Cuts of Flow Networks - Definition

Proof.

$$
\begin{aligned}
|f| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) && (1)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \left(\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) \right) && (2)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \sum_{v \in V} f(u,v) - \sum_{u \in S-\{s\}} \sum_{v \in V} f(v,u) && (3)\\
&= \sum_{v \in V} \left(f(s,v) + \sum_{u \in S-\{s\}} f(u,v) \right) - \sum_{v \in V} \left(f(v,s) + \sum_{u \in S-\{s\}} f(v,u) \right) && (4)\\
& && (5)\\
& && (6)\\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) && (7)\\
&= f(S,T) && (8)
\end{aligned}
$$

Cuts of Flow Networks - Definition

Proof.

$$
\begin{aligned}
|f| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) && (1)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \left(\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) \right) && (2)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \sum_{v \in V} f(u,v) - \sum_{u \in S-\{s\}} \sum_{v \in V} f(v,u) && (3)\\
&= \sum_{v \in V} \left(f(s,v) + \sum_{u \in S-\{s\}} f(u,v) \right) - \sum_{v \in V} \left(f(v,s) + \sum_{u \in S-\{s\}} f(v,u) \right) && (4)\\
&= \sum_{v \in V} \sum_{u \in S} f(u,v) - \sum_{v \in V} \sum_{u \in S} f(v,u) && (5)\\
& && (6)\\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) && (7)\\
&= f(S,T) && (8)
\end{aligned}
$$

Cuts of Flow Networks - Definition

Proof.

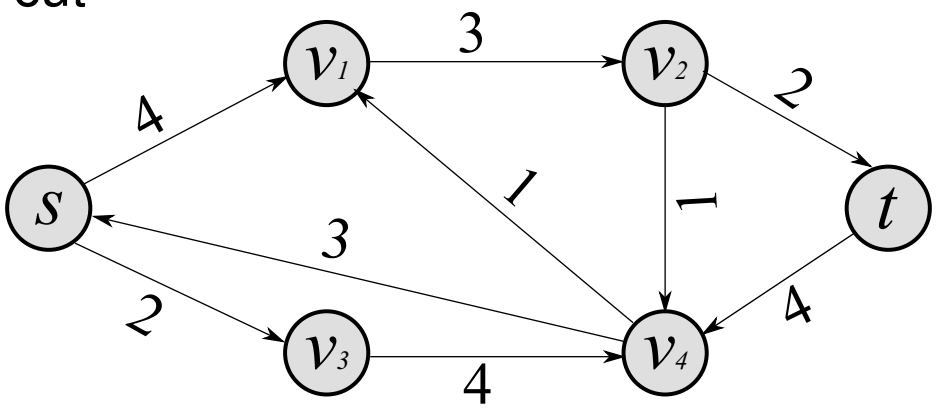
$$
\begin{aligned}
|f| &= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) && (1)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \left(\sum_{v \in V} f(u,v) - \sum_{v \in V} f(v,u) \right) && (2)\\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{u \in S-\{s\}} \sum_{v \in V} f(u,v) - \sum_{u \in S-\{s\}} \sum_{v \in V} f(v,u) && (3)\\
&= \sum_{v \in V} \left(f(s,v) + \sum_{u \in S-\{s\}} f(u,v) \right) - \sum_{v \in V} \left(f(v,s) + \sum_{u \in S-\{s\}} f(v,u) \right) && (4)\\
&= \sum_{v \in V} \sum_{u \in S} f(u,v) - \sum_{v \in V} \sum_{u \in S} f(v,u) && (5)\\
&= \sum_{v \in S} \sum_{u \in S} f(u,v) + \sum_{v \in T} \sum_{u \in S} f(u,v) - \sum_{v \in S} \sum_{u \in S} f(v,u) - \sum_{v \in T} \sum_{u \in S} f(v,u) && (6)\\
&= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) && (7)\\
&= f(S,T) && (8)
\end{aligned}
$$

Max Flow - Min Cut

Theorem

The value of the maximum flow is equal to the capacity of the minimum cut

Find the minimum cut



Application - Image Segmentation

Max-Flow Min-Cut has been used for image segmentation.

Application - Image Segmentation

How can you automatically segment the flowers from the image below?

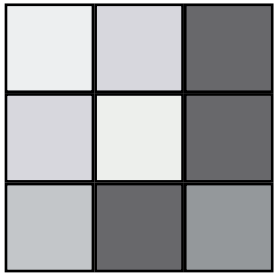
Application - Image Segmentation

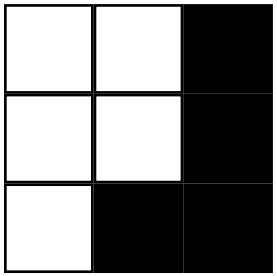
How can you automatically segment the flowers from the image below?

Colour-based selection in photoshop would not be able to perform this segmentation

Application - Image Segmentation

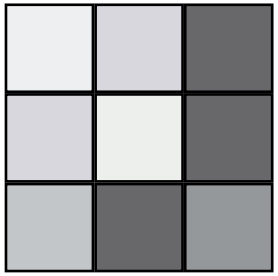
For an image of 3 $\times$ 3 pixels, you aim to label each pixel as *foreground* or *background*

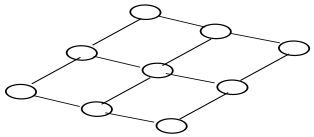


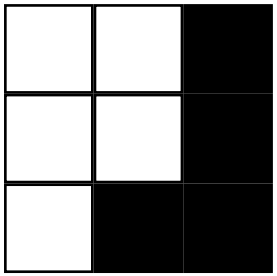


Application - Image Segmentation

Each pixel is represented by a single node, neighbouring pixels are connected by an edge

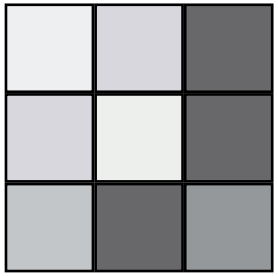


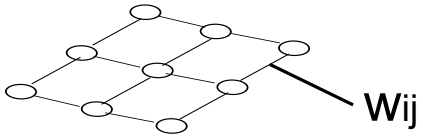


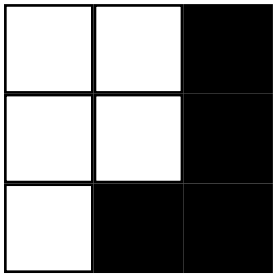


Application - Image Segmentation

The edge weight is usually represented by the difference in colour (/intensity) between neighbouring pixels

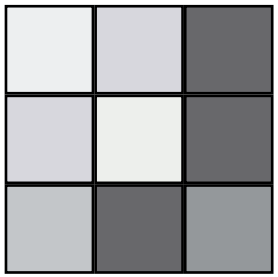


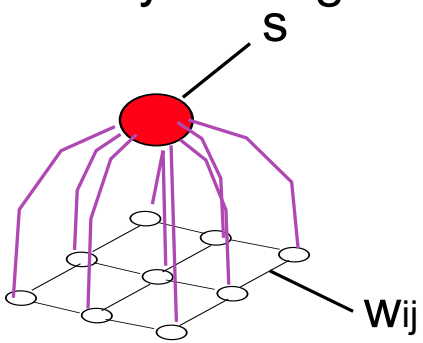


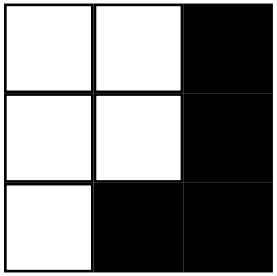


Application - Image Segmentation

This is transformed to a network flow by adding a *source*.

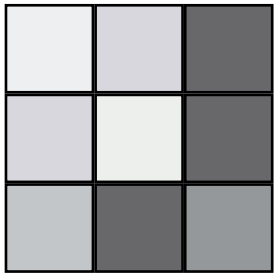


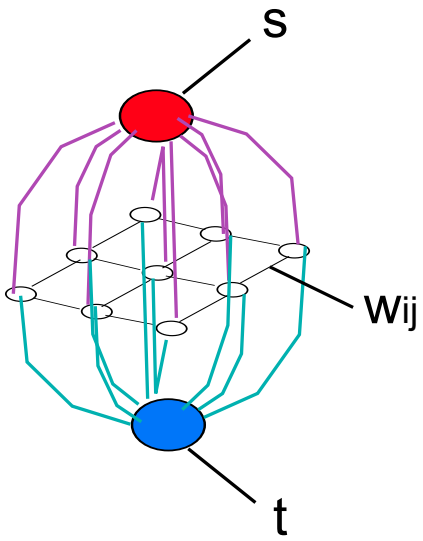


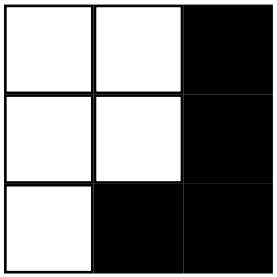


Application - Image Segmentation

... and a *sink*.

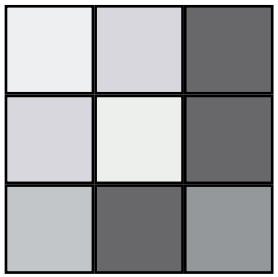


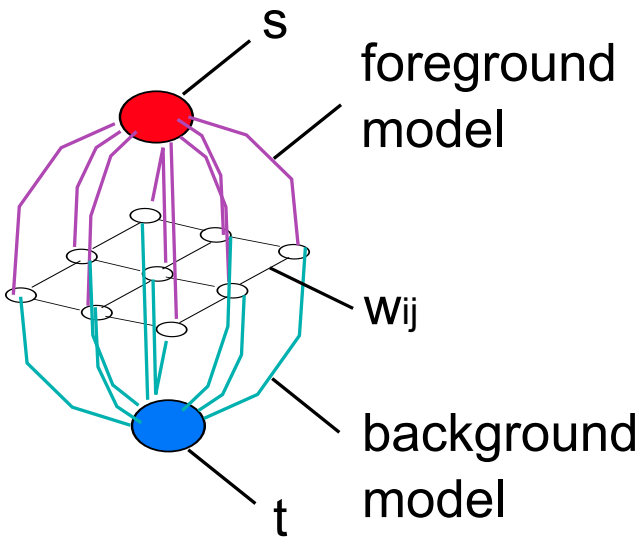


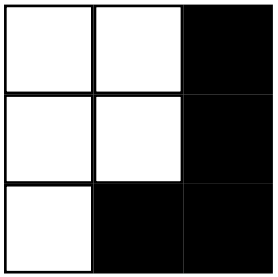


Application - Image Segmentation

The flow from the source is represented by the foreground model and to the sink is represented by the background model

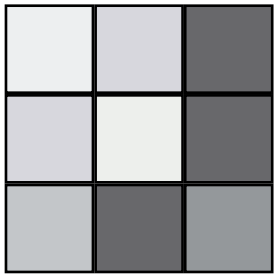


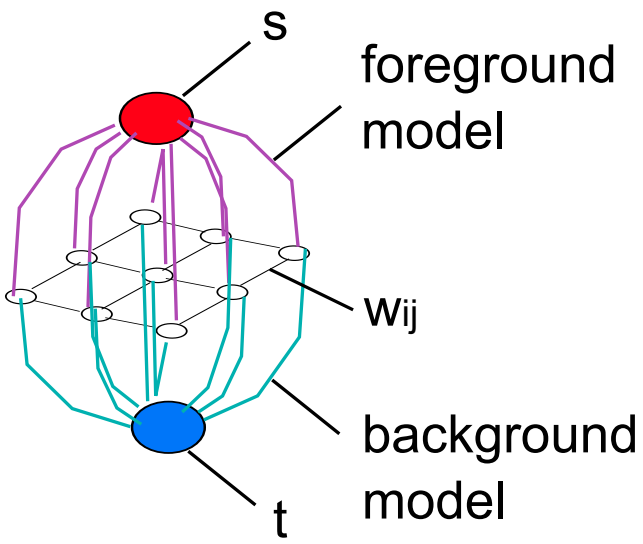


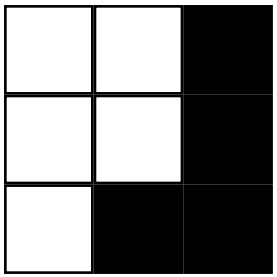


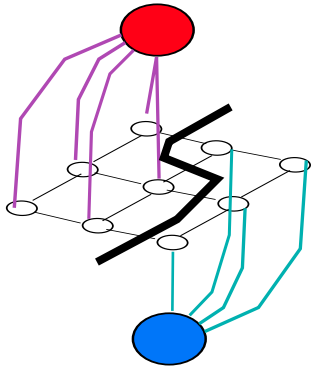
Application - Image Segmentation

The max-flow min-cut would result in an image segmentation



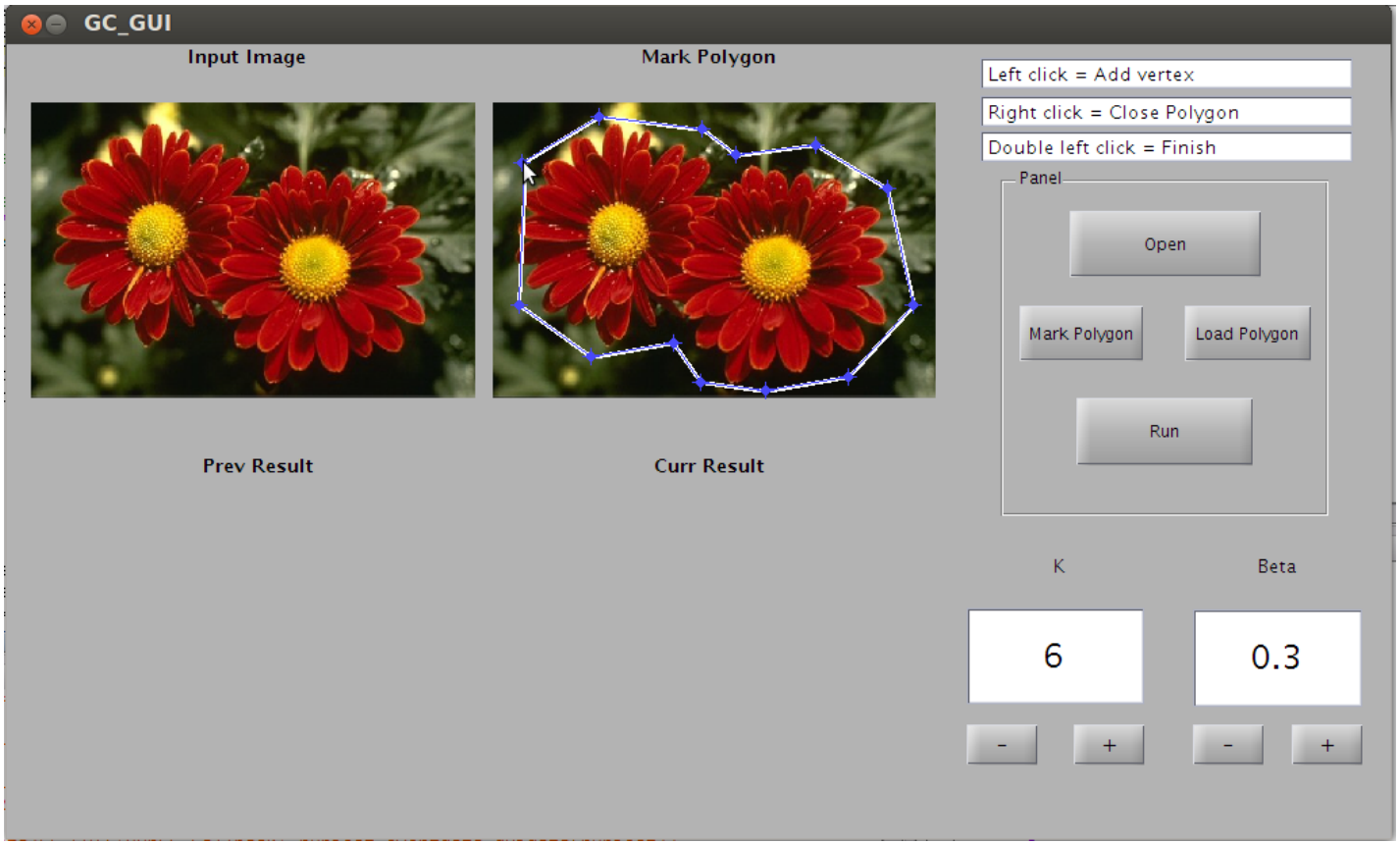






Application - Image Segmentation*

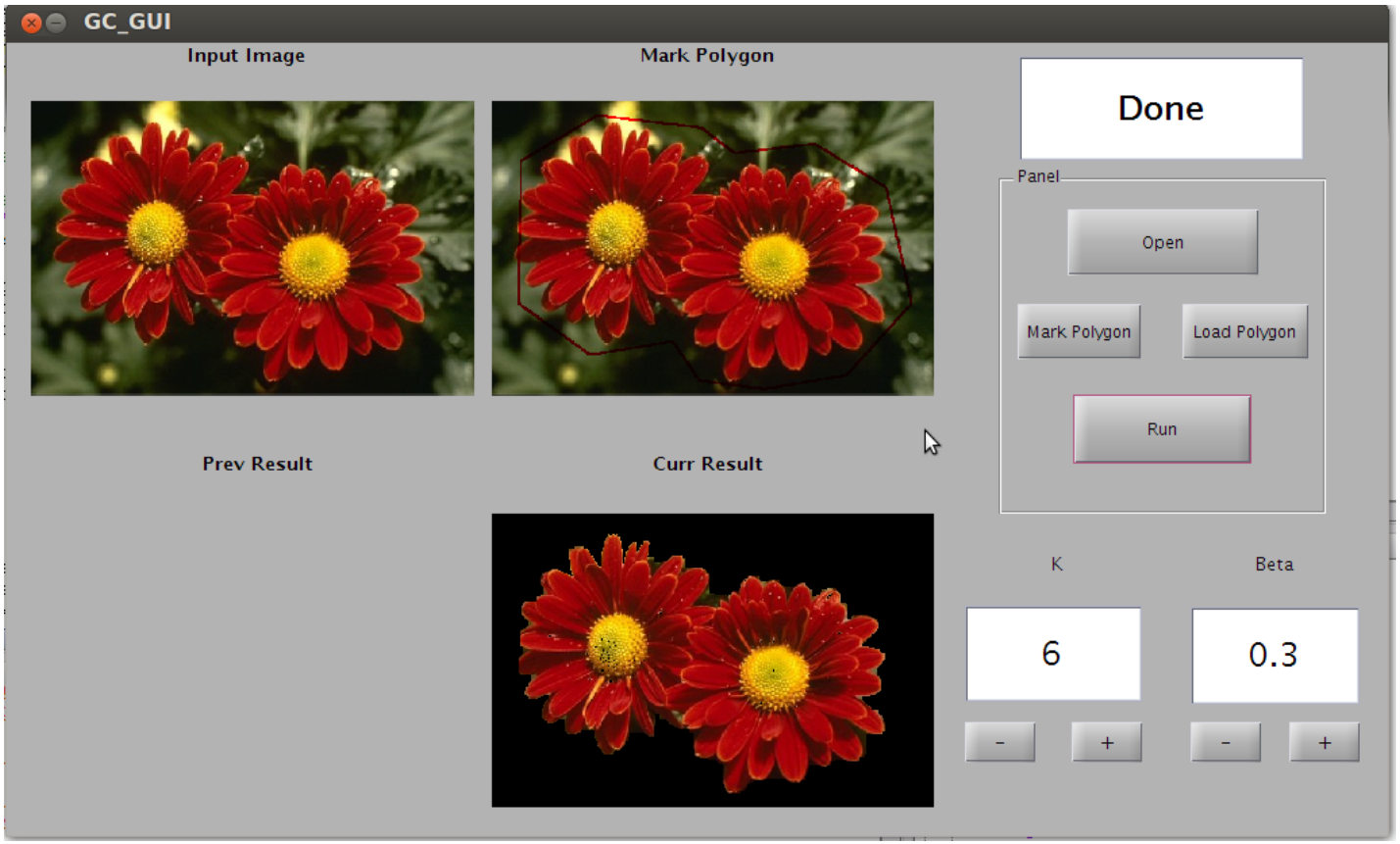
Using an interactive approach the user can specify foreground/background prior regions



*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

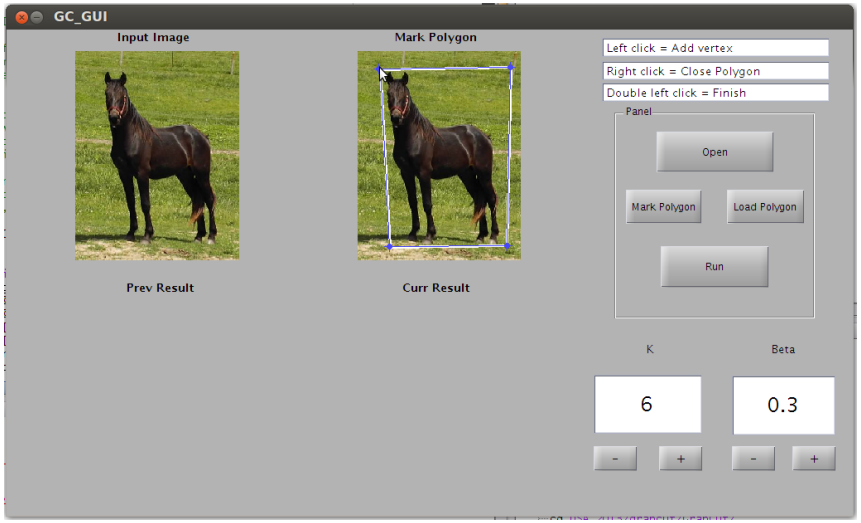
Foreground and background models are created, and min-cut solution is found



*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

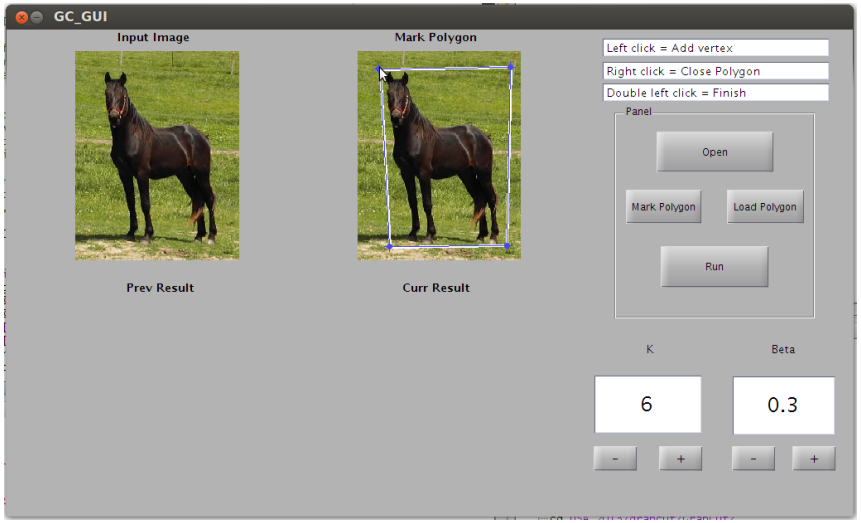
Models need not be tight

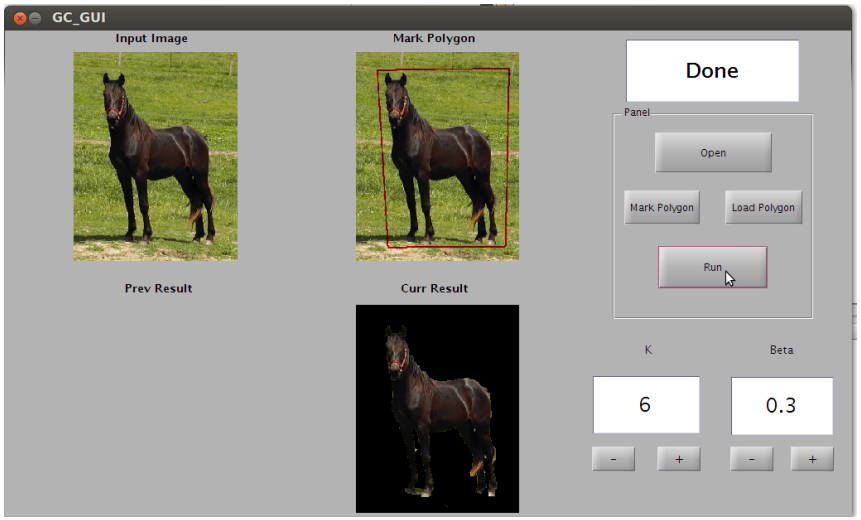


*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

Models need not be tight





*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

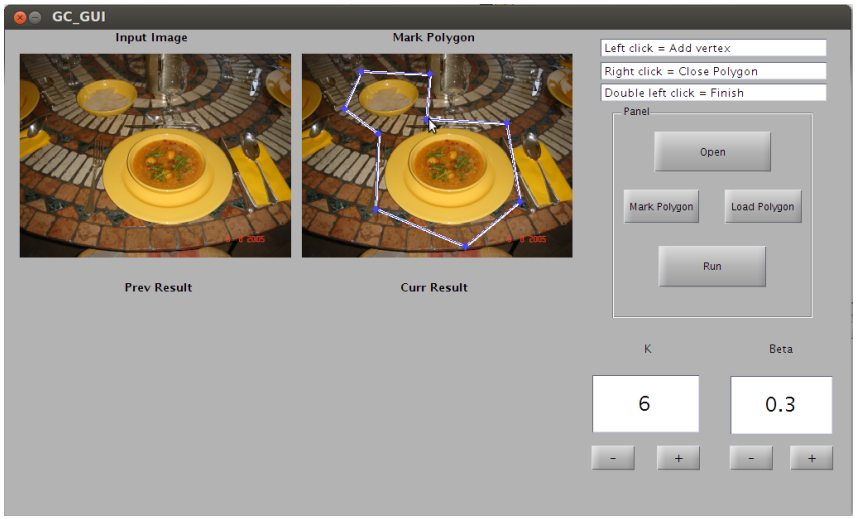
Application - Image Segmentation*

The cut need not generate one connected region

*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

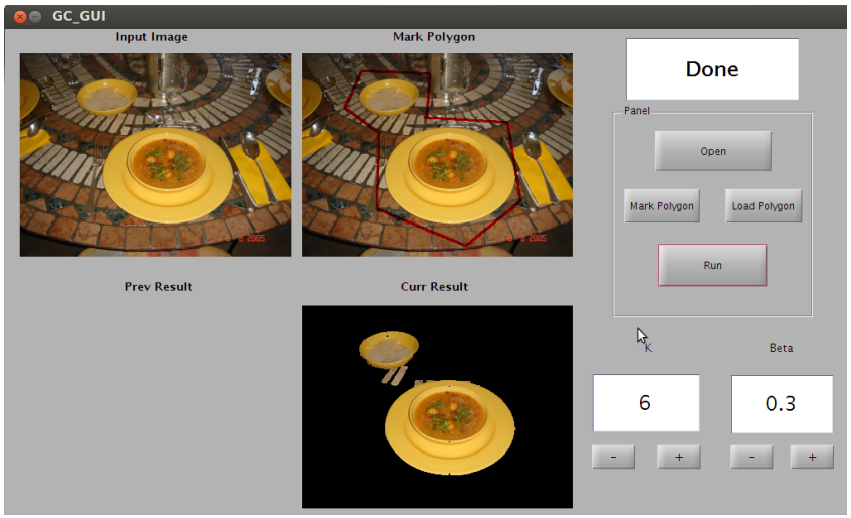
The cut need not generate one connected region



*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

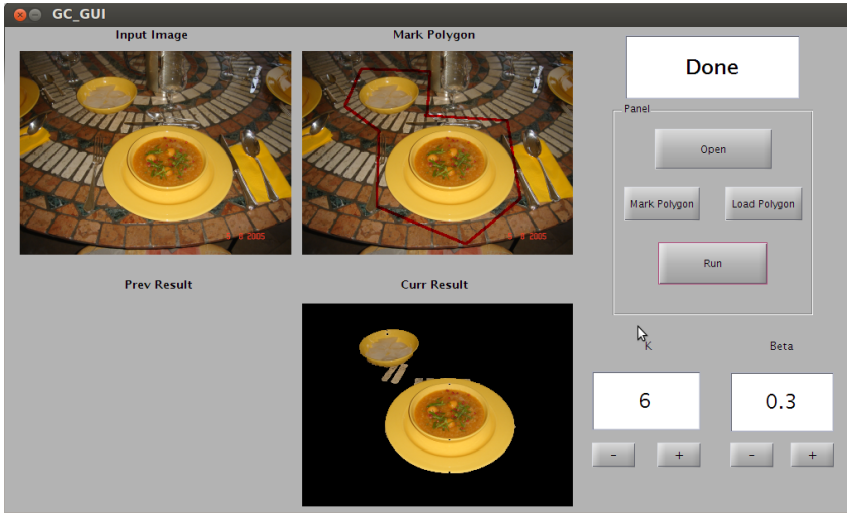
The cut need not generate one connected region

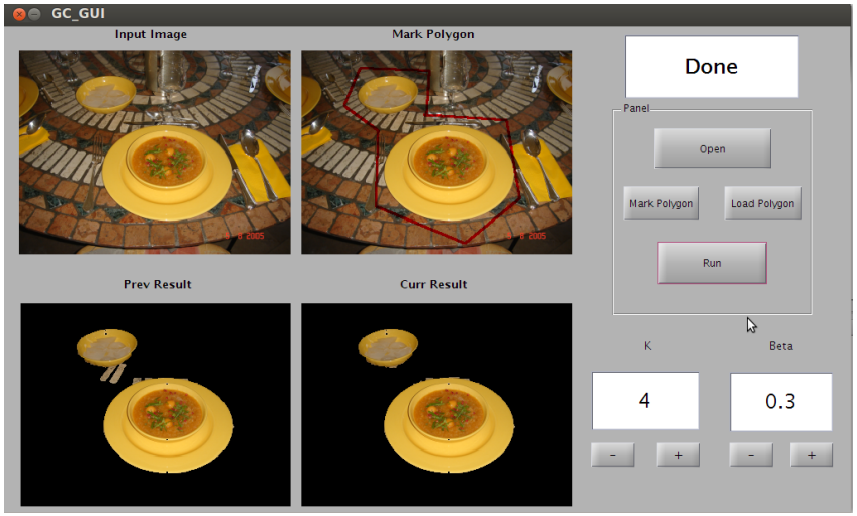


*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

The cut need not generate one connected region





*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

Think about this method's limitation.....

*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

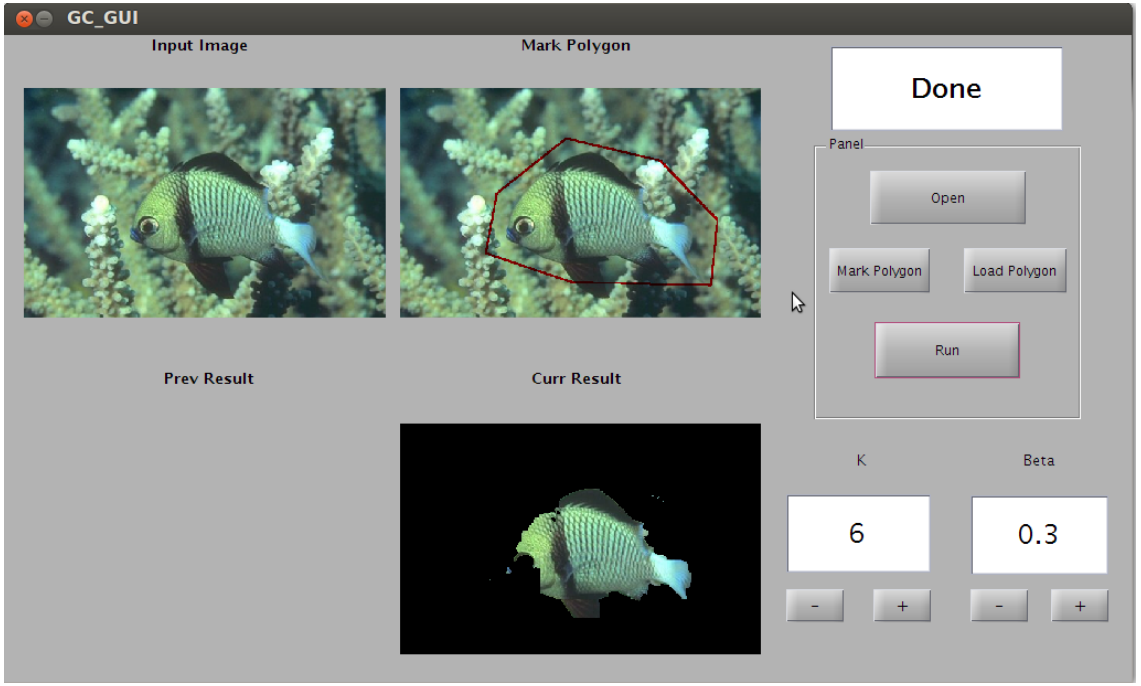
Application - Image Segmentation*

Think about this method's limitation.....

*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Application - Image Segmentation*

Think about this method's limitation.....



*Rother et al (2004). GrabCut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH. Online code and GUI.

Further Reading

- Introduction to Algorithms
  T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein.
  MIT Press/McGraw-Hill, ISBN: 0-262-03293-7.
  - Chapter 26 – Maximum Flow
- GrabCut: Interactive foreground extraction using iterated graph cuts
  C. Rother, V. Kolmogorov, and A. Blake.
  ACM Trans. Graph., 2004

Further Reading

- Youtube Lessons - Class of 2013/2014 - Ford Fulkerson
  - James Burnside `http://www.youtube.com/watch?v=v1VgJmkEJW0` [16,600 views - Nov 2015]
  - George Field `http://www.youtube.com/watch?v=LfbKwot9sZA` [20,140 views - Nov 2015]
  - Andrew Stuart `http://www.youtube.com/watch?v=XZHOI5xXLxs` [27,600 views - Nov 2015]
- Youtube Lessons - Class of 2014/2015 - Edmonds Karp
  - Ben Owain `https://www.youtube.com/watch?v=MczX0SM3I84` [4,250 views - Nov 2015]
  - Akmal Ahmad `https://www.youtube.com/watch?v=L9B0oBQlXQ0` [3,700 views - Nov 2015]
  - Laser Oz `https://www.youtube.com/watch?v=QJtYqHfczTo` [518 views - Nov 2015]