Analysis of Algorithms

Maximum Flow

Andres Mendez-Vazquez

November 22, 2019

Outline

Cinvestav

Outline

Cinvestav

History of Max Flow

Long Ago in the Faraway Cold War

- It was first described by T. E. Harris (At RAND Corporation) as a simplified model of the Soviet traffic flow.

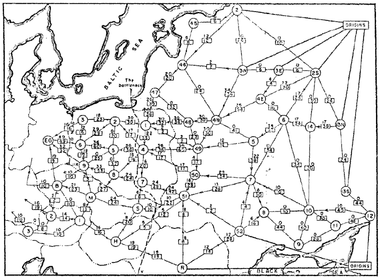


Figure: Railway network of the Western Soviet Union

Outline

Cinvestav

Flow Networks

Definition

- A **flow network** $G = (V, E)$ is a directed graph, where each edge $(u, v) \in E$ has a non-negative **capacity** $c(u, v) \geq 0$.
- In addition if E contains an edge (u, v), it does not contain the edge (v, u) (Reverse Direction).

- G has two vertices known as source s and sink t

Flow Networks

Definition

- A **flow network** $G = (V, E)$ is a directed graph, where each edge $(u, v) \in E$ has a non-negative **capacity** $c(u, v) \geq 0$.
- In addition if E contains an edge (u, v), it does not contain the edge (v, u) (Reverse Direction).

Constraints

- If $(u, v) \notin E$ we assume that $c(u, v) = 0$.
- G has two vertices known as source s and sink t.

Cinvestav

Flow Networks

Definition

- A **flow network** $G = (V, E)$ is a directed graph, where each edge $(u, v) \in E$ has a non-negative **capacity** $c(u, v) \geq 0$.
- In addition if E contains an edge (u, v), it does not contain the edge (v, u) (Reverse Direction).

Constraints

- If $(u, v) \notin E$ we assume that $c(u, v) = 0$.
- G has two vertices known as source s and sink t.

Flow Networks

Definition

- A **flow network** $G = (V, E)$ is a directed graph, where each edge $(u, v) \in E$ has a non-negative **capacity** $c(u, v) \geq 0$.
- In addition if E contains an edge (u, v), it does not contain the edge (v, u) (Reverse Direction).

Constraints

- If $(u, v) \notin E$ we assume that $c(u, v) = 0$.
- G has two vertices known as source s and sink t.

Cinvestav

Example that does not work

A Graph not Satisfying The Definition

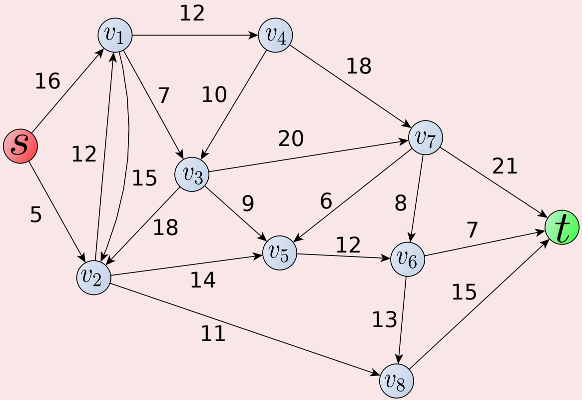


Figure: A simple example

Fixing the Example

Do not worry, we can transform it into...

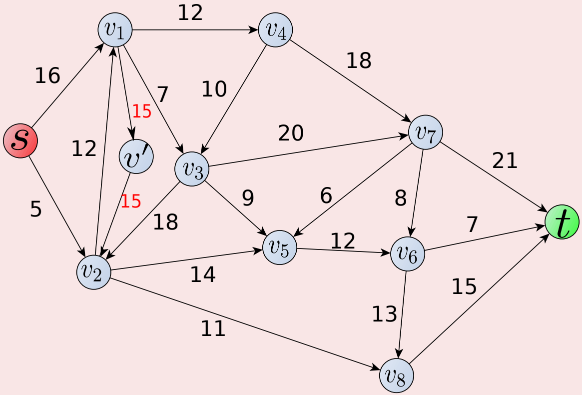


Figure: A simple example

Another Possible Problem

What if we have multiple sources and sinks?

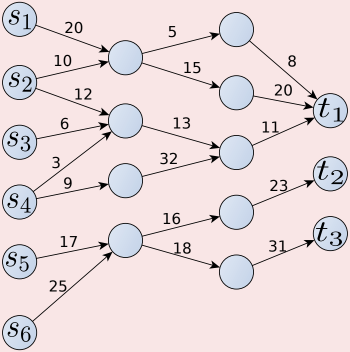


Figure: Ok no so simple!!!

Another Possible Problem

Use a Single Sink and Source

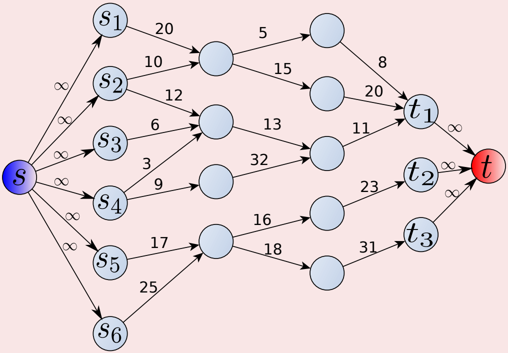


Figure: Ok!!! No so simple!!!

Outline

Cinvestav

Flow Properties

Definition

A **flow** in $G = (V, E)$ is a real valued function $f : V \times V \to \mathbb{R}$ satisfying the following properties:

$$0 \leq f(u,v) \leq c(u,v)$$

- **Flow conservation:** For all $u \in V - \{s,t\}$, we have that

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

Flow Properties

Definition

A **flow** in $G = (V, E)$ is a real valued function $f : V \times V \to \mathbb{R}$ satisfying the following properties:

Properties

- **Capacity Constraint**: For all $u, v \in V$

$$0 \leq f(u, v) \leq c(u, v).$$

- **Flow conservation**: For all $u \in V - \{s, t\}$, we have that

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

Cinvestav

Flow Properties

Definition

A **flow** in $G = (V, E)$ is a real valued function $f : V \times V \rightarrow \mathbb{R}$ satisfying the following properties:

Properties

- **Capacity Constraint**: For all $u, v \in V$

$$0 \leq f(u, v) \leq c(u, v).$$

- **Flow conservation**: For all $u \in V - \{s, t\}$, we have that

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

Flow Properties

Definition

A **flow** in $G = (V, E)$ is a real valued function $f : V \times V \rightarrow \mathbb{R}$ satisfying the following properties:

Properties

- **Capacity Constraint**: For all $u, v \in V$

$$0 \leq f(u, v) \leq c(u, v).$$

- **Flow conservation**: For all $u \in V - \{s, t\}$, we have that

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

Flow Properties

Definition

A **flow** in $G = (V, E)$ is a real valued function $f : V \times V \rightarrow \mathbb{R}$ satisfying the following properties:

Properties

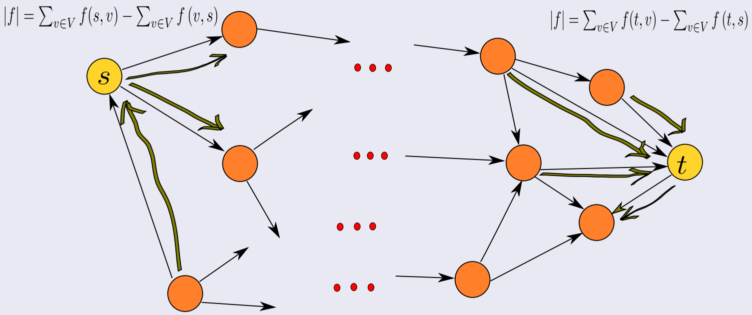
- **Capacity Constraint**: For all $u, v \in V$
$$0 \leq f(u, v) \leq c(u, v).$$
- **Flow conservation**: For all $u \in V - \{s, t\}$, we have that
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$

Outline

Cinvestav

Thus

We need to describe the concept of flow able to leave the source



Net Flow and Value of a Flow f

Definition of **net flow**

- The value of a **net flow** is defined as

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

- Which is the same as the total flow from any vertices to the sink?

Net Flow and Value of a Flow f

Definition of **net flow**

- The value of a **net flow** is defined as

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

This can bee seen as

- The total flow from **source** s to any other vertices.
- Which is the same as the total flow from any vertices to the **sink** t.

Net Flow and Value of a Flow f

Definition of **net flow**

- The value of a **net flow** is defined as

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

This can bee seen as

- The total flow from **source** s to any other vertices.
- Which is the same as the total flow from any vertices to the **sink** t.

Net Flow and Value of a Flow f

Definition of **net flow**

- The value of a **net flow** is defined as

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

This can bee seen as

- The total flow from **source** s to any other vertices.
- Which is the same as the total flow from any vertices to the **sink** t.

Cinvestav

Example

We have the following graph

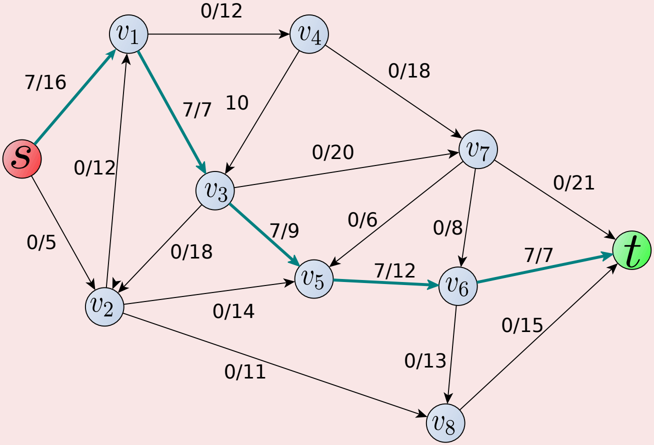


Figure: A flow f in G with net flow value $|f| = 7$

Outline

Cinvestav

Maximum Flow Problem

Definition

Given a flow network G with source s and sink t, it is necessary to find a flow of maximum value from s to t.

Question

How we solve this in an efficient manner?

Maximum Flow Problem

Definition

Given a flow network G with source s and sink t, it is necessary to find a flow of maximum value from s to t.

Question

How we solve this in an efficient manner?

Outline

Cinvestav

The Ford-Fulkerson Method

Observations

- Not exactly an algorithm, but several implementations with different running times.
- It depends on three fundamental ideas: **Residual Networks, Augmenting Paths and Cuts**.

The Ford-Fulkerson Method

Observations

- Not exactly an algorithm, but several implementations with different running times.
- It depends on three fundamental ideas: **Residual Networks, Augmenting Paths and Cuts**.

Pseudo-Code

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

1. Initialize flow f to 0
2. **while** there exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

Cinvestav

The Ford-Fulkerson Method

Observations

- Not exactly an algorithm, but several implementations with different running times.
- It depends on three fundamental ideas: **Residual Networks, Augmenting Paths and Cuts**.

Pseudo-Code

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

1. Initialize flow f to 0
2. **while** there exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

Cinvestav

Outline

First, the Intuition

First

- The residual network G_f consists of edges with capacities representing the change in the flow on edges of G.

Thus

- 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.

$$c_f(u,v) = c(u,v) - f(u,v) \text{ Residual Capacity}$$

First, the Intuition

First

- The residual network G_f consists of edges with capacities representing the change in the flow on edges of G.

Thus

- 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.

$$c_f(u,v) = c(u,v) - f(u,v) \text{ Residual Capacity}$$

Case I

The edges of G that are in G_f are those that can admit more flow

- i.e. $c(u,v) - f(u,v) > 0$

When you can add more flow to G

- Then $c_f(u,v) = c(u,v) - f(u,v)$

Case I

The edges of G that are in G_f are those that can admit more flow

- i.e. $c(u,v) - f(u,v) > 0$

When you can add more flow to G

- Then $c_f(u,v) = c(u,v) - f(u,v)$

Case II

If they have that $c(u,v) - f(u,v) = 0$

- Then $c_f(u,v) = 0$.

Basically

- Remove the edge in G_f given no more flow can be added to it!!!

Case II

If they have that $c(u, v) - f(u, v) = 0$

- Then $c_f(u, v) = 0$.

Basically

- Remove the edge in G_f given no more flow can be added to it!!!

Case III

First, the Intuition

- As an algorithm manipulates the flow to increase its total value, it might need to decrease the flow on a particular edge.

Case III

First, the Intuition

- As an algorithm manipulates the flow to increase its total value, it might need to decrease the flow on a particular edge.

To represent a possible decrease of a positive flow $f(u,v)$

- We place an edge (v,u) in G_f with residual capacity $c_f(v,u) = f(u,v)$.

Comments

This 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 sent along an edge.

Cinvestav

Comments

This 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 sent along an edge.

Residual Capacity

Given a flow network and a flow

- The **residual network** consists of edges that can admit more net flow.

IMPORTANT: Because of our initial assumption if $(u, v) \in E$ implies that $(v, u) \notin E$, thus only one case applies.

Residual Capacity

Given a flow network and a flow

- The **residual network** consists of edges that can admit more net flow.

It is based in the **residual capacity function**

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

IMPORTANT: Because of our initial assumption if $(u,v) \in E$ implies that $(v,u) \notin E$, thus only one case applies.

Cinvestav

Residual Capacity

Given a flow network and a flow

- The **residual network** consists of edges that can admit more net flow.

It is based in the residual capacity function

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

IMPORTANT: Because of our initial assumption if $(u,v) \in E$ implies that $(v,u) \notin E$, thus only one case applies.

Cinvestav

Residual Edges

Definition

Given a flow f, the residual network of G induced by f is $G_f = (V, E_f)$ where the set of residual edges E_f is defined as

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

Note: $|E_f| \leq 2|E|$ This is clear because the definition of capacity.

- Other than that it has the same properties: Capacity Constraint and Flow Conservation.

Cinvestav

Residual Edges

Definition

Given a flow f, the residual network of G induced by f is $G_f = (V, E_f)$ where the set of residual edges E_f is defined as

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

Note: $|E_f| \leq 2|E|$ This is clear because the definition of capacity.

Observations

- The residual network is not a flow network because there may contain both edges (u, v) and (v, u).
- Other than that it has the same properties: Capacity Constraint and Flow Conservation

Residual Edges

Definition

Given a flow f, the residual network of G induced by f is $G_f = (V, E_f)$ where the set of residual edges E_f is defined as

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

Note: $|E_f| \leq 2|E|$ This is clear because the definition of capacity.

Observations

- The residual network is not a flow network because there may contain both edges (u, v) and (v, u).
- Other than that it has the same properties: Capacity Constraint and Flow Conservation

Residual Edges

Definition

Given a flow f, the residual network of G induced by f is $G_f = (V, E_f)$ where the set of residual edges E_f is defined as

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

Note: $|E_f| \leq 2|E|$ This is clear because the definition of capacity.

Observations

- The residual network is not a flow network because there may contain both edges (u, v) and (v, u).
- Other than that it has the same properties: Capacity Constraint and Flow Conservation

Residual Edges

Definition

Given a flow f, the residual network of G induced by f is $G_f = (V, E_f)$ where the set of residual edges E_f is defined as

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

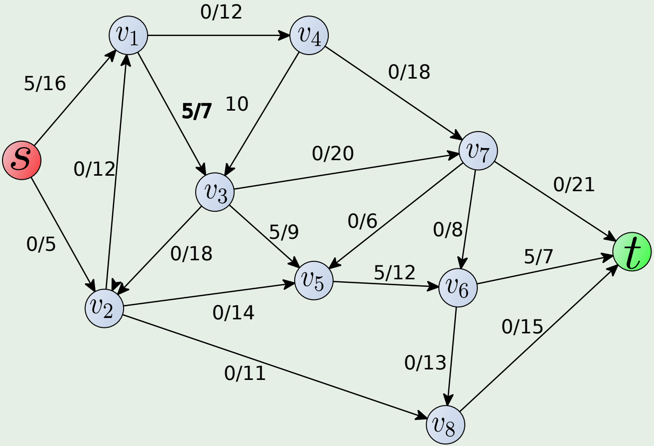
Note: $|E_f| \leq 2|E|$ This is clear because the definition of capacity.

Observations

- The residual network is not a flow network because there may contain both edges (u, v) and (v, u).
- Other than that it has the same properties: Capacity Constraint and Flow Conservation

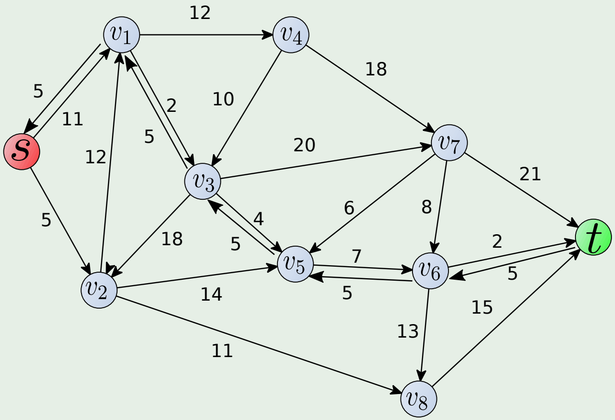
Example

Graph with FLOW/CAPACITY



Example

Its Residual Graph



Outline

Cinvestav

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Augmentation

If f is a flow in G and f' is a flow in the corresponding residual network, we define the augmentation of a flow f by f' as

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

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Augmentation

If f is a flow in G and f' is a flow in the corresponding residual network, we define the augmentation of a flow f by f' as

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

The Idea Behind Augmenting

- You can imagine augmentation as increase in the flow in a certain edge minus the reversal possible flow in the same edge.
- Looks like a **cancellation** of some sort!!!
- Actual pushing flow on the reverse edge in the residual network is also known as **cancellation**.

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Augmentation

If f is a flow in G and f' is a flow in the corresponding residual network, we define the augmentation of a flow f by f' as

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

The Idea Behind Augmenting

- You can imagine augmentation as increase in the flow in a certain edge minus the reversal possible flow in the same edge.
- Looks like a **cancellation** of some sort!!!
- Actual pushing flow on the reverse edge in the residual network is also known as **cancellation**.

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Augmentation

If f is a flow in G and f' is a flow in the corresponding residual network, we define the augmentation of a flow f by f' as

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

The Idea Behind Augmenting

- You can imagine augmentation as increase in the flow in a certain edge minus the reversal possible flow in the same edge.
- Looks like a **cancellation** of some sort!!!
- Actual pushing flow on the reverse edge in the residual network is also known as **cancellation**.

Residual Networks: Augmentation

Observation

Defining the residual flow allows to define augmentation.

Augmentation

If f is a flow in G and f' is a flow in the corresponding residual network, we define the augmentation of a flow f by f' as

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

The Idea Behind Augmenting

- You can imagine augmentation as increase in the flow in a certain edge minus the reversal possible flow in the same edge.
- Looks like a **cancellation** of some sort!!!
- Actual pushing flow on the reverse edge in the residual network is also known as **cancellation**.

Outline

Cinvestav

Augmentation Lemma

Lemma 26.1

Let $G = (V, E)$ be a flow network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f , and let f' be a flow in G_f. Then the function $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

Proof:

First, we verify that $f \uparrow f'$ obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s,t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq (f \uparrow f')(u,v) \leq c(u,v)$.

Flow conservation at each vertex in $V - \{s,t\}$

- For all $u \in V - \{s,t\} \Rightarrow \sum_{v \in V} (f \uparrow f')(v,u) = \sum_{v \in V} (f \uparrow f')(u,v)$.

Proof:

First, we verify that $f \uparrow f'$ obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s, t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq (f \uparrow f')(u, v) \leq c(u, v)$.

Flow conservation at each vertex in $V - \{s, t\}$

- For all $u \in V - \{s, t\} \Rightarrow \sum_{v \in V} (f \uparrow f')(v, u) = \sum_{v \in V} (f \uparrow f')(u, v)$.

Cinvestav

Proof:

First, we verify that $f \uparrow f'$ obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s, t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq (f \uparrow f')(u, v) \leq c(u, v)$.

Flow conservation at each vertex in $V - \{s, t\}$

- For all $u \in V - \{s, t\} \Rightarrow \sum_{v \in V} (f \uparrow f')(v, u) = \sum_{v \in V} (f \uparrow f')(u, v)$.

Proof:

Capacity Constraint

1. If $(u, v) \in E$ then $c_f(v, u) = f(u, v)$.
2. Therefore $f'(v, u) \leq c_f(v, u) = f(u, v)$.

Proof:

Capacity Constraint

1. If $(u, v) \in E$ then $c_f(v, u) = f(u, v)$.
2. Therefore $f'(v, u) \leq c_f(v, u) = f(u, v)$.

Hence

$$
\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\geq f(u, v) + f'(u, v) - f(u, v) \\
&= f'(u, v) \\
&\geq 0
\end{aligned}
$$

Proof:

Capacity Constraint

1. If $(u, v) \in E$ then $c_f(v, u) = f(u, v)$.
2. Therefore $f'(v, u) \leq c_f(v, u) = f(u, v)$.

Hence

$$
\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\geq f(u, v) + f'(u, v) - f(u, v) \\
&= f'(u, v) \\
&\geq 0
\end{aligned}
$$

Cinvestav

Proof:

Capacity Constraint

1. If $(u,v) \in E$ then $c_f(v,u) = f(u,v)$.
2. Therefore $f'(v,u) \leq c_f(v,u) = f(u,v)$.

Hence

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\geq f(u,v) + f'(u,v) - f(u,v) \\
&= f'(u,v) \\
&\geq 0
\end{aligned}
$$

Proof:

Capacity Constraint

1. If $(u, v) \in E$ then $c_f(v, u) = f(u, v)$.
2. Therefore $f'(v, u) \leq c_f(v, u) = f(u, v)$.

Hence

$$\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\geq f(u, v) + f'(u, v) - f(u, v) \\
&= f'(u, v) \\
&\geq 0
\end{aligned}$$

Proof

Second

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\leq f(u,v) + f'(u,v) \\
&\leq f(u,v) + c_f(u,v) \\
&\leq f(u,v) + c(u,v) - f(u,v) \\
&= c(u,v)
\end{aligned}
$$

Proof

Second

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\leq f(u,v) + f'(u,v) \\
&\leq f(u,v) + c_f(u,v) \\
&\leq f(u,v) + c(u,v) - f(u,v) \\
&= c(u,v)
\end{aligned}
$$

Thus

$$
0 \leq (f \uparrow f')(u,v) \leq c(u,v)
$$

Cinvestav

Proof

Second

$$
\begin{aligned}
\left(f \uparrow f'\right)(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\leq f(u,v) + f'(u,v) \\
&\leq f(u,v) + c_f(u,v) \\
&\leq f(u,v) + c(u,v) - f(u,v) \\
&= c(u,v)
\end{aligned}
$$

Thus

$$
0 \leq \left(f \uparrow f'\right)(u,v) \leq c(u,v)
$$

Proof

Second

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\leq f(u,v) + f'(u,v) \\
&\leq f(u,v) + c_f(u,v) \\
&\leq f(u,v) + c(u,v) - f(u,v) \\
&= c(u,v)
\end{aligned}
$$

Thus

$$
0 \leq (f \uparrow f')(u,v) \leq c(u,v)
$$

Proof

Second

$$
\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\leq f(u, v) + f'(u, v) \\
&\leq f(u, v) + c_f(u, v) \\
&\leq f(u, v) + c(u, v) - f(u, v) \\
&= c(u, v)
\end{aligned}
$$

Thus

$$0 \leq (f \uparrow f')(u, v) \leq c(u, v)$$

Cinvestav

Proof

Second

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) \\
&\leq f(u,v) + f'(u,v) \\
&\leq f(u,v) + c_f(u,v) \\
&\leq f(u,v) + c(u,v) - f(u,v) \\
&= c(u,v)
\end{aligned}
$$

Thus

$$0 \leq (f \uparrow f')(u,v) \leq c(u,v)$$

Cinvestav

Now, We prove the Flow Conservation

Therefore

$$\sum_{v \in V} (f \uparrow f')(u,v) = \sum_{v \in V} \left[f(u,v) + f'(u,v) - f'(v,u) \right]$$

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

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

$$= \sum_{v \in V} \left[f(v,u) + f'(v,u) - f'(u,v) \right]$$

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

- where the third line follows from the second by flow conservation in f and f'.

Now, We prove the Flow Conservation

Therefore

$$
\begin{aligned}
\sum_{v \in V} \left(f \uparrow f'\right)(u,v) &= \sum_{v \in V} \left[f(u,v) + f'(u,v) - f'(v,u)\right] \\
&= \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u) \\
&= \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(v,u) - \sum_{v \in V} f'(u,v) \\
&= \sum_{v \in V} \left[f(v,u) + f'(v,u) - f'(u,v)\right] \\
&= \sum_{v \in V} \left(f \uparrow f'\right)(v,u)
\end{aligned}
$$

- where the third line follows from the second by flow conservation in f and f'.

Now, We prove the Flow Conservation

Therefore

$$
\begin{aligned}
\sum_{v \in V} \left(f \uparrow f'\right)(u,v) &= \sum_{v \in V} \left[f(u,v) + f'(u,v) - f'(v,u)\right] \\
&= \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u) \\
&= \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(v,u) - \sum_{v \in V} f'(u,v) \\
&= \sum_{v \in V} \left[f(v,u) + f'(v,u) - f'(u,v)\right] \\
&= \sum_{v \in V} \left(f \uparrow f'\right)(v,u)
\end{aligned}
$$

- where the third line follows from the second by flow conservation in f and f'.

Now, We prove the Flow Conservation

Therefore

$$
\begin{aligned}
\sum_{v \in V} (f \uparrow f')(u,v) &= \sum_{v \in V} \left[f(u,v) + f'(u,v) - f'(v,u) \right] \\
&= \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u) \\
&= \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(v,u) - \sum_{v \in V} f'(u,v) \\
&= \sum_{v \in V} \left[f(v,u) + f'(v,u) - f'(u,v) \right] \\
&= \sum_{v \in V} (f \uparrow f')(v,u)
\end{aligned}
$$

- where the third line follows from the second by flow conservation in f and f'.

Now, We prove the Flow Conservation

Therefore

$$\begin{aligned}
\sum_{v \in V} \left(f \uparrow f'\right)(u,v) &= \sum_{v \in V} \left[f(u,v) + f'(u,v) - f'(v,u)\right] \\
&= \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) - \sum_{v \in V} f'(v,u) \\
&= \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(v,u) - \sum_{v \in V} f'(u,v) \\
&= \sum_{v \in V} \left[f(v,u) + f'(v,u) - f'(u,v)\right] \\
&= \sum_{v \in V} \left(f \uparrow f'\right)(v,u)
\end{aligned}$$

- where the third line follows from the second by flow conservation in f and f'.

Now, we need to prove that $|f \uparrow f'| = |f| + |f'|$.

Recall

- We disallow anti-parallel edges in G, but not in G_f.
  - For each vertex $v \in V$, we know that there can an edge (s, v) or (v, s) but never both.

Now, we define with respect to the source

- $V_1 = \{v | (s, v) \in E\}$
- $V_2 = \{v | (v, s) \in E\}$

Cinvestav

Now, we need to prove that $|f \uparrow f'| = |f| + |f'|$.

Recall

- We disallow anti-parallel edges in G, but not in G_f.
  - For each vertex $v \in V$, we know that there can an edge (s, v) or (v, s) but never both.

Now, we define with respect to the source

- $V_1 = \{v | (s, v) \in E\}$
- $V_2 = \{v | (v, s) \in E\}$

Cinvestav

Remember the definition of Net Flow

Definition of **net flow**

- The value of a **net flow** is defined as

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

Now

We have the following properties

- $V_1 \cup V_2 \subseteq V$.
- $V_1 \cap V_2 = \emptyset$ given not anti-parallel edges.

We can compute then

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V} \left(f \uparrow f'\right)(s, v) - \sum_{v \in V} \left(f \uparrow f'\right)(v, s) \\
&= \sum_{v \in V_1} \left(f \uparrow f'\right)(s, v) - \sum_{v \in V_1} \left(f \uparrow f'\right)(v, s)
\end{aligned}
$$

- Given that $\left(f \uparrow f'\right)(s, v) = 0$ if $(s, v) \notin E$

Now

We have the following properties

- $V_1 \cup V_2 \subseteq V$.
- $V_1 \cap V_2 = \emptyset$ given not anti-parallel edges.

We can compute then

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V} (f \uparrow f')(s, v) - \sum_{v \in V} (f \uparrow f')(v, s) \\
&= \sum_{v \in V_1} (f \uparrow f')(s, v) - \sum_{v \in V_1} (f \uparrow f')(v, s)
\end{aligned}
$$

- Given that $(f \uparrow f')(s, v) = 0$ if $(s, v) \notin E$

Then

Reordering some of the terms, we have

$$
\begin{aligned}
\left|f \uparrow f'\right| &= \sum_{v \in V_1} \left[f(s,v) + f'(s,v) - f'(v,s)\right] \\
&\quad - \sum_{v \in V_2} \left[f(v,s) + f'(v,s) - f'(s,v)\right] \\
&= \sum_{v \in V_1} f(s,v) + \sum_{v \in V_1} f'(s,v) - \sum_{v \in V_1} f'(v,s) \\
&\quad - \sum_{v \in V_2} f(v,s) - \sum_{v \in V_2} f'(v,s) + \sum_{v \in V_2} f'(s,v)
\end{aligned}
$$

Then

Reordering some of the terms, we have

$$
\begin{aligned}
\left|f \uparrow f'\right| &= \sum_{v \in V_1} \left[f(s,v) + f'(s,v) - f'(v,s)\right] \\
&\quad - \sum_{v \in V_2} \left[f(v,s) + f'(v,s) - f'(s,v)\right] \\
&= \sum_{v \in V_1} f(s,v) + \sum_{v \in V_1} f'(s,v) - \sum_{v \in V_1} f'(v,s) \\
&\quad - \sum_{v \in V_2} f(v,s) - \sum_{v \in V_2} f'(v,s) + \sum_{v \in V_2} f'(s,v)
\end{aligned}
$$

Cinvestav

Then, we have that

After some Reordering

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1} f'(s,v) + \sum_{v \in V_2} f'(s,v) \\
&- \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s) \\
&= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s) \\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s) \\
&= |f| + |f'|
\end{aligned}
$$

Then, we have that

After some Reordering

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1} f'(s,v) + \sum_{v \in V_2} f'(s,v) \\
&\quad - \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s) \\
&= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s) \\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s) \\
&= |f| + |f'|
\end{aligned}
$$

Cinvestav

Then, we have that

After some Reordering

$$
\begin{aligned}
|f \uparrow f'| &= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1} f'(s,v) + \sum_{v \in V_2} f'(s,v) \\
&- \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s) \\
&= \sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s) \\
&= \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s) \\
&= |f| + |f'|
\end{aligned}
$$

Outline

Cinvestav

Augmenting Paths

Augmenting Path

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

Augmenting Paths

Augmenting Path

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

Residual Capacity

- Residual capacity is the maximum amount by which we can increase the flow without violating capacity

$$c_f(p) = \min\left\{c_f(u,v) \,\middle|\, (u,v) \text{ is on } p\right\}.$$

Augmenting Paths

Augmenting Path

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

Residual Capacity

- Residual capacity is the maximum amount by which we can increase the flow without violating capacity

$$c_f(p) = \min\left\{c_f(u,v) \mid (u,v) \text{ is on } p\right\}.$$

Cinvestav

Example

Example of an augmented path (Shaded) given a flow graph G and flow f

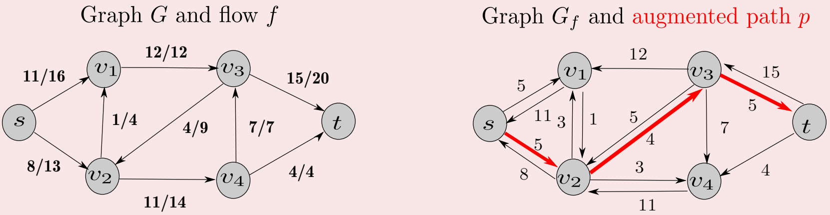


Figure: Residual Capacity of shaded path is 5

The new flow Graph

Here, we have

New Flow Graph G and flow f

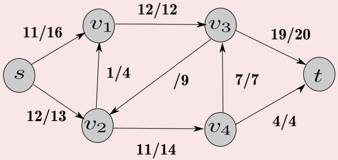


Figure: Result after Augmentation

Lemma 26.2

Lemma 26.2

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

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

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

Lemma 26.2

Lemma 26.2

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

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

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

Lemma 26.2

Lemma 26.2

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

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

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

Proof:

First, we verify that f_p obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s,t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq f_p(u,v) \leq c(u,v)$.

Flow conservation at each vertex in $V - \{s,t\}$

- For all $u \in V - \{s,t\} \Rightarrow \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$.

Proof:

First, we verify that f_p obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s,t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq f_p(u,v) \leq c(u,v)$.

Flow conservation at each vertex in $V - \{s,t\}$

- For all $u \in V - \{s,t\} \Rightarrow \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$.

Cinvestav

Proof:

First, we verify that f_p obeys

1. The capacity constraint for each edge in E.
2. Flow conservation at each vertex in $V - \{s, t\}$.

The capacity constraint for each edge in E.

- For all $u, v \in V \Rightarrow 0 \leq f_p(u, v) \leq c(u, v)$.

Flow conservation at each vertex in $V - \{s, t\}$

- For all $u \in V - \{s, t\} \Rightarrow \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

Cinvestav

Proof

The capacity constraints

- It follows from the definition...

Proof

The capacity constraints

- It follows from the definition...

The flow conservation, for all $u \in V - \{s, t\}$

$$\sum_{v \in V} f_p(v, u) = \sum_{v \in V, (v,u) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0$$

$$= c_f(p) + \sum_{v \in V, (v,u) \notin p} 0$$

$$= \sum_{v \in V, (u,v) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0$$

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

Proof

The capacity constraints

- It follows from the definition...

The flow conservation, for all $u \in V - \{s, t\}$

$$
\begin{aligned}
\sum_{v \in V} f_p(v,u) &= \sum_{v \in V, (v,u) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= \sum_{v \in V, (u,v) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= \sum_{v \in V} f_p(u,v)
\end{aligned}
$$

Proof

The capacity constraints

- It follows from the definition...

The flow conservation, for all $u \in V - \{s, t\}$

$$
\begin{aligned}
\sum_{v \in V} f_p(v, u) &= \sum_{v \in V, (v,u) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= \sum_{v \in V, (u,v) \in p} c_f(p) + \sum_{v \in V, (v,u) \notin p} 0 \\
&= \sum_{v \in V} f_p(u, v)
\end{aligned}
$$

Remember the definition of Net Flow

Given the definition

$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) = c_f(p) - 0 = c_f(p) > 0$$

Finally

Corollary 26.3

- Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f . Suppose that we augment f by f_p. Then the function $f \uparrow f_p$ is a flow in G with value $|f \uparrow f_p| = |f| + |f_p| > |f|$.

Proof. Immediate from previous lemmas.

Finally

Corollary 26.3

- Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f . Suppose that we augment f by f_p. Then the function $f \uparrow f_p$ is a flow in G with value $|f \uparrow f_p| = |f| + |f_p| > |f|$.

  Proof Immediate from previous lemmas.

Outline

Cinvestav

Ford-Fulkerson

Basic Process

Augment repeatedly the flow along augmenting paths

How do we stop?

Ah!! Here, we will use the concept of **cut**.

A cut (S, T)

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

Net flow $f(S,T)$

If f is flow, then a net flow is defined as

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

Ford-Fulkerson

Basic Process

Augment repeatedly the flow along augmenting paths

How do we stop?

Ah!! Here, we will use the concept of **cut.**

A cut (S,T)

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

Net flow $f(S,T)$

If f is flow, then a net flow is defined as

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

Ford-Fulkerson

Basic Process

Augment repeatedly the flow along augmenting paths

How do we stop?

Ah!! Here, we will use the concept of **cut.**

A cut (S, T)

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

Net flow $f(S, T)$

If f is flow, then a net flow is defined as

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

Ford-Fulkerson

Basic Process

Augment repeatedly the flow along augmenting paths

How do we stop?

Ah!! Here, we will use the concept of **cut.**

A cut (S, T)

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

Net flow $f(S, T)$

If f is flow, then a net flow is defined as

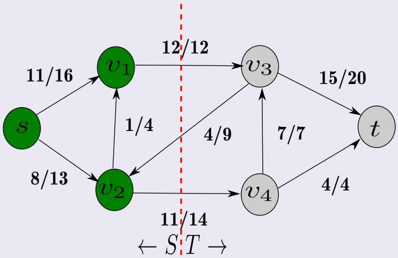
$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u).$$

Outline

Cinvestav

Minimum Cut

The net flow across the cut $f(S,T) = 19$ and the capacity is $c(S,T) = 26$

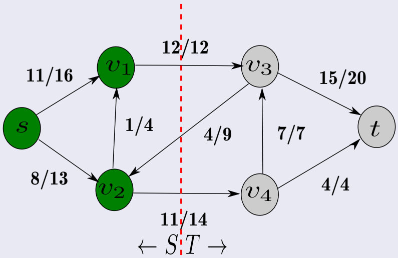


Therefore

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

Minimum Cut

The net flow across the cut $f(S,T) = 19$ and the capacity is $c(S,T) = 26$



Therefore

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

Important

First

The asymmetry between the definitions of flow and capacity of a cut is intentional and important.

Important

First

The asymmetry between the definitions of flow and capacity of a cut is intentional and important.

Why?

1. For capacity, we count only the capacities of edges going from S to T , ignoring edges in the reverse direction.
2. For flow, we consider the flow going from S to T minus the flow going in the reverse direction from T to S.

Important

First

The asymmetry between the definitions of flow and capacity of a cut is intentional and important.

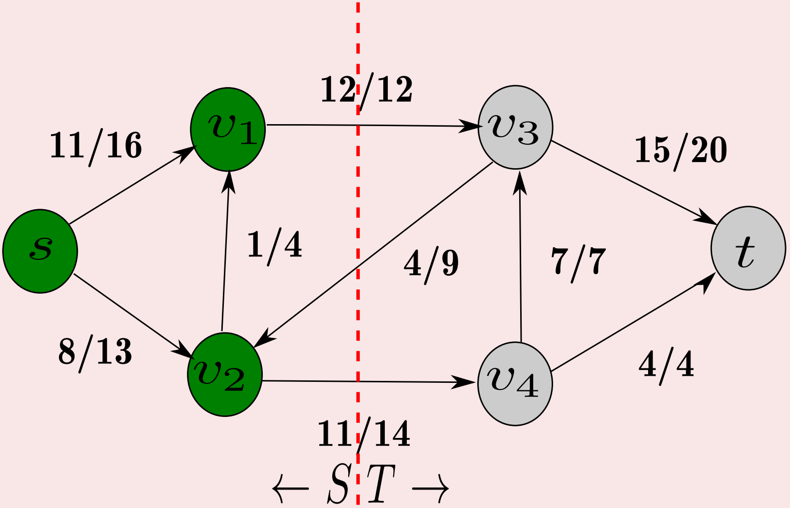
Why?

1. For capacity, we count only the capacities of edges going from S to T , ignoring edges in the reverse direction.
2. For flow, we consider the flow going from S to T minus the flow going in the reverse direction from T to S.

Cinvestav

Example

We have



Outline

Cinvestav

The Net Flow across any cut is the same

Lemma 26.4

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 the net flow across (S, T) is $f(S, T) = |f|$.

Proof

- We have from flow-conservation

$$\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = 0$$

The Net Flow across any cut is the same

Lemma 26.4

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 the net flow across (S, T) is $f(S, T) = |f|$.

Proof

- We have from flow-conservation

$$\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = 0$$

Cinvestav

Now

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

$$|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) + \sum_{u\in S-\{s\}} \left(\sum_{v\in V} f(u,v) - \sum_{v\in V} f(v,u) \right)$$

Regrouping Terms

$$|f| = \sum_{v\in V} \left(f(s,v) - \sum_{u\in S-\{s\}} f(u,v) \right) - \sum_{v\in V} \left(f(v,s) + \sum_{u\in S-\{s\}} f(v,u) \right)$$

Now

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

$$|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) + \sum_{u\in S-\{s\}} \left(\sum_{v\in V} f(u,v) - \sum_{v\in V} f(v,u) \right)$$

Regrouping Terms

$$|f| = \sum_{v\in V} \left(f(s,v) - \sum_{u\in S-\{s\}} f(u,v) \right) - \sum_{v\in V} \left(f(v,s) + \sum_{u\in S-\{s\}} f(v,u) \right)$$

Finally, we have

The following equation

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

Because $V = S \cup T$ and $S \cap T = \emptyset$

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

Finally, we have

The following equation

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

Because $V = S \cup T$ and $S \cap T = \emptyset$

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

Cinvestav

Therefore

We have

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

Cinvestav

Bounding the value of a flow.

Corollary 26.5

- The value of any flow f in a flow network G is bounded from above by the **capacity of any cut** of G.

Cinvestav

Proof

Let (S,T) any cut of G and f be any flow

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

Proof

Let (S, T) any cut of G and f be any flow

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

Cinvestav

Proof

Let (S, T) any cut of G and f be any flow

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

Proof

Let (S, T) any cut of G and f be any flow

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

Cinvestav

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

Theorem 26.6 (Max-Flow Min-Cut Theorem) - Stopping Condition

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.

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

Theorem 26.6 (Max-Flow Min-Cut Theorem) - Stopping Condition

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.

Cinvestav

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

Theorem 26.6 (Max-Flow Min-Cut Theorem) - Stopping Condition

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.

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

Theorem 26.6 (Max-Flow Min-Cut Theorem) - Stopping Condition

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

$(1) \implies (2)$

- Assume the following for a contradiction f is a maximum flow and G_f has an augmenting path p.

Therefore, we can augment f by f_p

$$|f \uparrow f_p| > |f|$$

Problem $f \uparrow f_p$ is a flow

- Contradiction!!! Thus G_f does not contain any augmenting path.

Cinvestav

Proof

$(1) \implies (2)$

- Assume the following for a contradiction f is a maximum flow and G_f has an augmenting path p.

Therefore, we can augment f by f_p

$$|f \uparrow f_p| > |f|$$

Problem $f \uparrow f_p$ is a flow

- Contradiction!!! Thus G_f does not contain any augmenting path.

Proof

$(1) \implies (2)$

- Assume the following for a contradiction f is a maximum flow and G_f has an augmenting path p.

Therefore, we can augment f by f_p

$$|f \uparrow f_p| > |f|$$

Problem $f \uparrow f_p$ is a flow

- Contradiction!!! Thus G_f does not contain any augmenting path.

Cinvestav

Proof

$(2) \implies (3)$

- Suppose that G_f does not contain augmenting path... no path from s to t.

Define

$$S \{v \in V | \exists \text{ path } s \rightsquigarrow t\} \text{ and } T = V - S$$

We have $s \in S$ trivially and $t \notin S$ given not path from s to t in G_f

- We have several cases...

Proof

$(2) \implies (3)$

- Suppose that G_f does not contain augmenting path... no path from s to t.

Define

$$S\left\{v \in V | \exists \text{ path } s \rightsquigarrow t\right\} \text{ and } T = V - S$$

We have $s \in S$ trivially and $t \notin S$ given not path from s to t in G_f

- We have several cases...

Proof

$(2) \implies (3)$

- Suppose that G_f does not contain augmenting path... no path from s to t.

Define

$$S\left\{v \in V | \exists \text{ path } s \rightsquigarrow t\right\} \text{ and } T = V - S$$

We have $s \in S$ trivially and $t \notin S$ given not path from s to t in G_f

- We have several cases...

Therefore

If $(u, v) \in E$

- We have $f(u, v) = c(u, v)$.
  - Because in any other case $(u, v) \in E_f$ which will place $v \in S$.

If $(v, u) \in E$

- We must have $f(v, u) = 0$
  - Otherwise $c_f(u, v) = f(v, u)$ would be positive and we would have $(u, v) \in E_f$ again placing $v \in S$

Therefore

If $(u, v) \in E$

- We have $f(u, v) = c(u, v)$.
  - Because in any other case $(u, v) \in E_f$ which will place $v \in S$.

If $(v, u) \in E$

- We must have $f(v, u) = 0$
  - Otherwise $c_f(u, v) = f(v, u)$ would be positive and we would have $(u, v) \in E_f$ again placing $v \in S$

Finally

If neither (u, v) nor (v, u) is in E

- Then $f(u, v) = f(v, u) = 0$

We have then

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

Cinvestav

Finally

If neither (u, v) nor (v, u) is in E

- Then $f(u, v) = f(v, u) = 0$

We have then

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

Finally

We have because of Lemma 26.4

$$|f| = f(S,T) = c(S,T)$$

.

Now from $(3) \Longrightarrow (2)$

- By Corollary 26.5 $|f| \leq c(S,T)$ for all cuts (S,T)

Thus, the condition $|f| = c(S,T)$

- It implies that f is a maximum flow...

Finally

We have because of Lemma 26.4

$$|f| = f(S, T) = c(S, T)$$

.

Now from $(3) \implies (2)$

- By Corollary 26.5 $|f| \leq c(S, T)$ for all cuts (S, T)

Thus, the condition $|f| = c(S, T)$

- It implies that f is a maximum flow...

Finally

We have because of Lemma 26.4

$$|f| = f(S,T) = c(S,T)$$

.

Now from $(3) \implies (2)$

- By Corollary 26.5 $|f| \leq c(S,T)$ for all cuts (S,T)

Thus, the condition $|f| = c(S,T)$

- It implies that f is a maximum flow...

Cinvestav

Outline

Cinvestav

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
   1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
    1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
   1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
    1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Cinvestav

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
   1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
   1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Meaning of All This

Procedure

Init We start with flow f zero.

1. We construct the residual graph.
2. We find a path p between s and t in the residual Graph.
3. If The residual network G_f contains no augmenting paths.
   1. The f is the maximum flow!!! and exit
4. We find $C_f(p)$
5. We augment the flow in the original graph.
6. Repeat to 1

Cinvestav

Outline

Cinvestav

Now

For the algorithm, if $(u, v) \in E$

We make $(u, v).f = (u, v).f + c_f(p)$ because you can add flow.

If $(u, v) \notin E$

You have that (u, v) is a reversal in G_f, then $(v, u) \in E$:

$$(u, v).f = (u, v).f - c_f(p)$$

Meaning

- Do not add flow but remove to do the cancellation.

Now

For the algorithm, if $(u, v) \in E$

We make $(u, v).f = (u, v).f + c_f(p)$ because you can add flow.

If $(u, v) \notin E$

You have that (u, v) is a reversal in G_f, then $(v, u) \in E$:

$$(u, v).f = (u, v).f - c_f(p)$$

Meaning

- Do not add flow but remove to do the cancellation.

Cinvestav

Now

For the algorithm, if $(u, v) \in E$

We make $(u, v).f = (u, v).f + c_f(p)$ because you can add flow.

If $(u, v) \notin E$

You have that (u, v) is a reversal in G_f, then $(v, u) \in E$:

$$(u, v).f = (u, v).f - c_f(p)$$

Meaning

- Do not add flow but remove to do the cancellation.

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

1. **for** each edge $(u, v) \in G.E$
2. $\quad (u, v).f = 0$
3. **while** there exists a path p form s to t in the residual network G_f
4. $\quad c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
5. $\quad$ **for** each edge (u, v) in p
6. $\quad\quad$ if $(u, v) \in E$
7. $\quad\quad\quad (u, v).f = (u, v).f + c_f(p)$
8. $\quad\quad$ else $(v, u).f = (v, u).f - c_f(p)$

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p form s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

```
for each edge (u, v) ∈ G.E
        (u, v).f = 0
while there exists a path p form 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)
```

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p form s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$

Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t)

for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p form s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$

Ford-Fulkerson Algorithm

Explanation

- Line 1-2 initialize flows to 0.
- Line 3-8 are executed as long as a path exist in G_f between s to t:
  - Line 4 finds the $c_f(p)$.

Ford-Fulkerson Algorithm

Explanation

- Line 1-2 initialize flows to 0.
- Line 3-8 are executed as long as a path exist in G_f between s to t:
  - Line 4 finds the $c_f(p)$.

Ford-Fulkerson Algorithm

Explanation

- Line 1-2 initialize flows to 0.
- Line 3-8 are executed as long as a path exist in G_f between s to t:
  - Line 4 finds the $c_f(p)$.

Ford-Fulkerson Algorithm

OBSERVATION: Each residual edge in path p is either an edge in the original network or the reversal

- Thus, Line 6-8 basically are an equilibrium act:
  - If the edge exist add flow to it.
  - If not remove flow otherwise from the reverse edge.

Ford-Fulkerson Algorithm

OBSERVATION: Each residual edge in path p is either an edge in the original network or the reversal

- Thus, Line 6-8 basically are an equilibrium act:
  - If the edge exist add flow to it.
  - If not remove flow otherwise from the reverse edge.

Ford-Fulkerson Algorithm

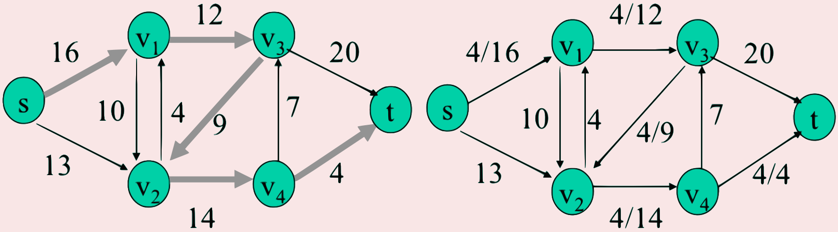
OBSERVATION: Each residual edge in path p is either an edge in the original network or the reversal

- Thus, Line 6-8 basically are an equilibrium act:
  - If the edge exist add flow to it.
  - If not remove flow otherwise from the reverse edge.

Outline

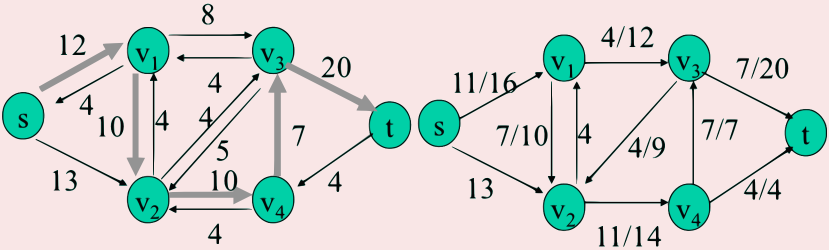
Example of Ford-Fulkerson

First Augmentation Path



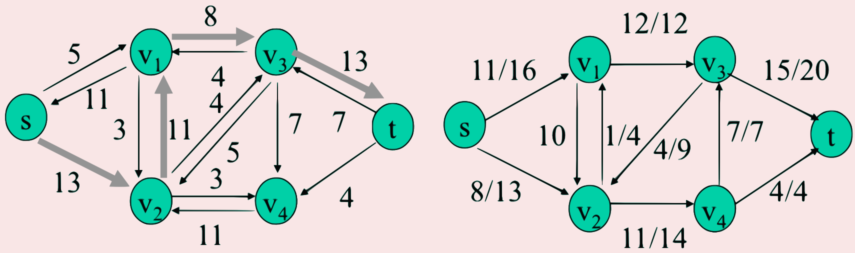
Example

Second Augmentation Path



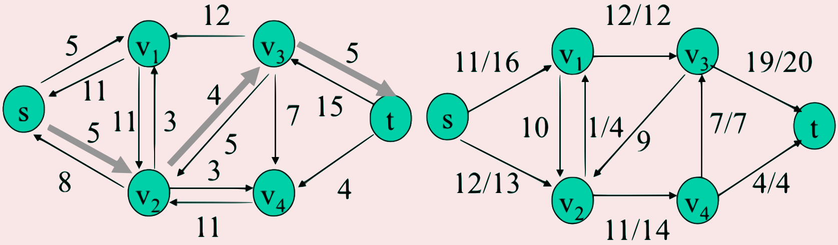
Example

Then



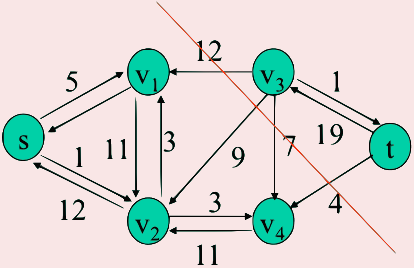
Example

Example



Example

Example



Outline

Ford-Fulkerson Algorithm

Final Code

Ford-Fulkerson(G, s, t)

for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p form s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
    for each edge (u, v) in p
        if $(u, v) \in E$
            $(u, v).f = (u, v).f + c_f(p)$
        else $(v, u).f = (v, u).f - c_f(p)$

Cinvestav

Complexity I

Complexity

- **Note: Be careful a bad implementation will not converge because we need to choose p.**
- Ford-Fulkerson works for integer numbers, but rational numbers can be transformed into integers by scaling (Real can be approximated by rational numbers).
- Imagine that after that transformation, we have f^* the maximum flow of a transformed network.
  - **while** loop of lines 3-8 are bounded by $|f^*|$ since the flow value increases by at least one unit at each iteration.

Complexity I

Complexity

- **Note: Be careful a bad implementation will not converge because we need to choose p.**
- Ford-Fulkerson works for integer numbers, but rational numbers can be transformed into integers by scaling (Real can be approximated by rational numbers).
- Imagine that after that transformation, we have f^* the maximum flow of a transformed network.
    - **while** loop of lines 3-8 are bounded by $|f^*|$ since the flow value increases by at least one unit at each iteration.

Complexity I

Complexity

- **Note: Be careful a bad implementation will not converge because we need to choose p.**
- Ford-Fulkerson works for integer numbers, but rational numbers can be transformed into integers by scaling (Real can be approximated by rational numbers).
- Imagine that after that transformation, we have f^* the maximum flow of a transformed network.
  - **while** loop of lines 3-8 are bounded by $|f^*|$ since the flow value increases by at least one unit at each iteration.

Complexity I

Complexity

- **Note: Be careful a bad implementation will not converge because we need to choose p.**
- Ford-Fulkerson works for integer numbers, but rational numbers can be transformed into integers by scaling (Real can be approximated by rational numbers).
- Imagine that after that transformation, we have f^* the maximum flow of a transformed network.
  - **while** loop of lines 3-8 are bounded by $|f^*|$ since the flow value increases by at least one unit at each iteration.

Cinvestav

Complexity II

Using BFS or DFS

- Complexity of finding a path is $O(V + E') = O(E)$ (Line 3 While Loop)
- Final complexity time of the Ford-Fulkerson Algorithm is $O(E|f^*|)$

Complexity II

Using BFS or DFS

- Complexity of finding a path is $O\left(V + E'\right) = O\left(E\right)$ (Line 3 While Loop)
- Final complexity time of the Ford-Fulkerson Algorithm is $O\left(E\left|f^{*}\right|\right)$

Now

What if if $c_f(p) = 1$ each time?

Complexity II

Using BFS or DFS

- Complexity of finding a path is $O(V + E') = O(E)$ (Line 3 While Loop)
- Final complexity time of the Ford-Fulkerson Algorithm is $O(E|f^*|)$

Now

What if if $c_f(p) = 1$ each time?

Outline

Cinvestav

Example where the situation is not so Good

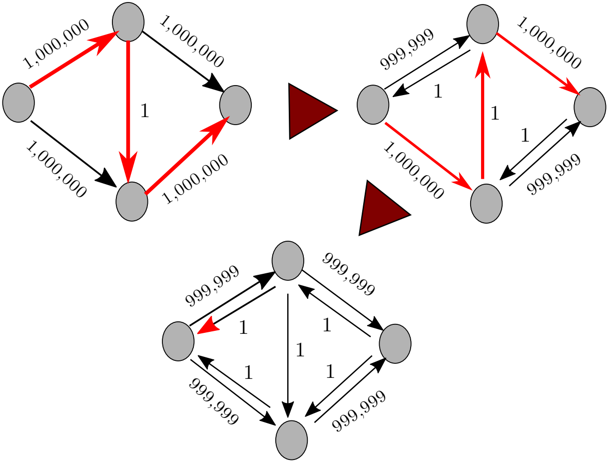


Figure: An example where complexity can be a killer when selecting the central path all the time

Outline

Edmond-Karp Algorithm

Observation Edmond-Karp

- Edmond-Karp is Ford-Fulkerson with shortest path in the residual network, $\delta_f(u, v)$, where each edge has unit distance (weight).
- Basically use BFS.

Edmond-Karp Algorithm

Observation Edmond-Karp

- Edmond-Karp is Ford-Fulkerson with shortest path in the residual network, $\delta_f(u, v)$, where each edge has unit distance (weight).
- Basically use BFS.

Lemma 26.7

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest-path distance $\delta_f(u, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Theorem 26.8

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

Edmond-Karp Algorithm

Observation Edmond-Karp

- Edmond-Karp is Ford-Fulkerson with shortest path in the residual network, $\delta_f(u,v)$, where each edge has unit distance (weight).
- Basically use BFS.

Lemma 26.7

If the Edmonds-Karp algorithm is run on a flow network $G=(V,E)$ with source s and sink t, then for all vertices $v \in V - \{s,t\}$, the shortest-path distance $\delta_f(u,v)$ in the residual network G_f increases monotonically with each flow augmentation.

Theorem 26.8

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

Edmond-Karp Algorithm

Observation Edmond-Karp

- Edmond-Karp is Ford-Fulkerson with shortest path in the residual network, $\delta_f(u, v)$, where each edge has unit distance (weight).
- Basically use BFS.

Lemma 26.7

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then for all vertices $v \in V - \{s, t\}$, the shortest-path distance $\delta_f(u, v)$ in the residual network G_f increases monotonically with each flow augmentation.

Theorem 26.8

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

Outline

Cinvestav

Complexity

Complexity Edmond-Karp

- Each iteration of Ford-Fulkerson can be implemented in $O(E)$.
- The Complexity of Edmond-Karp is $O(VE^2)$.

- Don't Panic, It is beyond this class!!!

Complexity

Complexity Edmond-Karp

- Each iteration of Ford-Fulkerson can be implemented in $O(E)$.
- The Complexity of Edmond-Karp is $O(VE^2)$.

Better Complexity

- The Generic Push-Relabel by Golberg for max-flow has complexity $O(V^2E)$.
- Don't Panic, It is beyond this class!!!

Cinvestav

Complexity

Complexity Edmond-Karp

- Each iteration of Ford-Fulkerson can be implemented in $O(E)$.
- The Complexity of Edmond-Karp is $O(VE^2)$.

Better Complexity

- The Generic Push-Relabel by Golberg for max-flow has complexity $O\left(V^2E\right)$.
- Don't Panic, It is beyond this class!!!

Cinvestav

Complexity

Complexity Edmond-Karp

- Each iteration of Ford-Fulkerson can be implemented in $O(E)$.
- The Complexity of Edmond-Karp is $O(VE^2)$.

Better Complexity

- The Generic Push-Relabel by Golberg for max-flow has complexity $O\left(V^2E\right)$.
- Don't Panic, It is beyond this class!!!

Cinvestav

Outline

Cinvestav

The Maximum-Bipartite-Matching Problem

The Bipartite Graph

A graph $G = (V, E)$, where $V = L \cup R$ s.t. $L \cap R = \emptyset$, and for every $(u, v) \in E$, $u \in L$ and $v \in R$.

Matching

Given an undirected graph $G = (V, E)$, a matching is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v.

Maximum Matching

A maximum matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have: $|M'| \leq |M|$.

The Maximum-Bipartite-Matching Problem

The Bipartite Graph

A graph $G = (V, E)$, where $V = L \cup R$ s.t. $L \cap R = \emptyset$, and for every $(u, v) \in E$, $u \in L$ and $v \in R$.

Matching

Given an undirected graph $G = (V, E)$, a matching is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v.

Maximum Matching

A maximum matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have: $|M'| \leq |M|$.

The Maximum-Bipartite-Matching Problem

The Bipartite Graph

A graph $G = (V, E)$, where $V = L \cup R$ s.t. $L \cap R = \emptyset$, and for every $(u, v) \in E$, $u \in L$ and $v \in R$.

Matching

Given an undirected graph $G = (V, E)$, a matching is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v.

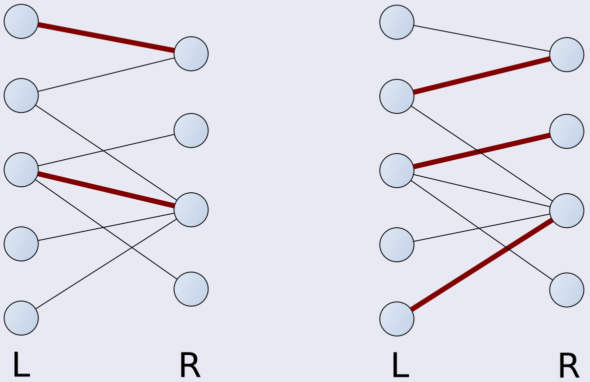
Maximum Matching

A maximum matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have: $|M'| \leq |M|$.

Cinvestav

Example

Two examples of matching



Corresponding Flow Network

Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \mid u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3|E|$
- $|E'| = \Theta(E)$
- Make for any $(u, v) \in E'$, $w(u, v) = 1$

Corresponding Flow Network

Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \,|u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3\,|E|$
- $|E'| = \Theta\,(E)$
- Make for any $(u, v) \in E'$, $w\,(u, v) = 1$

Cinvestav

Corresponding Flow Network

Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \,|u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3|E|$
- $|E'| = \Theta(E)$
- Make for any $(u, v) \in E'$, $w(u, v) = 1$

Corresponding Flow Network

Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \, | u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3 \, |E|$
- $|E'| = \Theta(E)$
- Make for any $(u, v) \in E'$, $w(u, v) = 1$

Corresponding Flow Network

Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \,|u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3\,|E|$
- $|E'| = \Theta\,(E)$
- Make for any $(u, v) \in E'$, $w\,(u, v) = 1$

Corresponding Flow Network

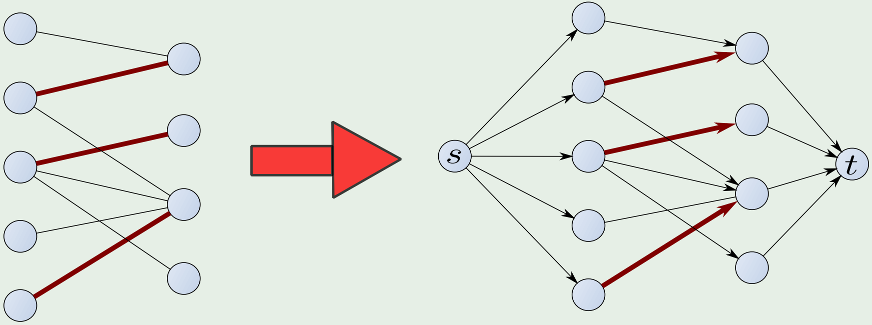
Build

A graph $G' = (V', E')$ is a **corresponding flow network** from a bipartite graph G:

- $V' = V \cup \{s, t\}$
- $E' = \{(s, u) \, | u \in L\} \cup E \cup \{(v, t) \in E'\}$
- $|E| \leq |E'| = |E| + |V| \leq 3 \, |E|$
- $|E'| = \Theta(E)$
- Make for any $(u, v) \in E'$, $w(u, v) = 1$

Example

Add new source s and sink t



Outline

Cinvestav

Corresponding Flow Network

Ok, What do we do?

- Basically, you run Edmond-Karp on the Graph G'.

- Then look at the following lemma, theorem and corollary!!!

Corresponding Flow Network

Ok, What do we do?

- Basically, you run Edmond-Karp on the Graph G'.

How do you see that this is correct?

- First introduce the concept: f is a flow on a flow network $G = (V, E)$ is integer-valued if $f(u, v)$ is an integer for all $(u, v) \in V \times V$.
- Then look at the following lemma, theorem and corollary!!!

Cinvestav

Corresponding Flow Network

Ok, What do we do?

- Basically, you run Edmond-Karp on the Graph G'.

How do you see that this is correct?

- First introduce the concept: f is a flow on a flow network $G = (V, E)$ is integer-valued if $f(u, v)$ is an integer for all $(u, v) \in V \times V$.
- Then look at the following lemma, theorem and corollary!!!

Cinvestav

Proving Correctness

Lemma 26.9

- Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' 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 f is an integer-valued flow in G' , then there is a matching M in G with cardinality $|f| = |M|$.

Integrality Theorem

- If the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer.
- Moreover, for all vertices u and v, the value of $f(u, v)$ is an integer.

Proving Correctness

Lemma 26.9

- Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' 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 f is an integer-valued flow in G' , then there is a matching M in G with cardinality $|f| = |M|$.

Integrality Theorem

- If the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer.
- Moreover, for all vertices u and v, the value of $f(u, v)$ is an integer.

Cinvestav

Finally

Corollary 26.11

- The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in its corresponding flow network G'.

Outline

Cinvestav

Complexity

Using G'

Thus, given G, you build G' and run Ford-Fulkerson method. Then, use the max flow f to build the maximum matching by using:

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

- In addition every time the residual graph is build the candidate flow is augmented in one.
- Thus, $O(VE') = O(VE)$.

Complexity

Using G'

Thus, given G, you build G' and run Ford-Fulkerson method. Then, use the max flow f to build the maximum matching by using:

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

Complexity

- Because we know the $|M| \leq \min\{L, R\} = O(V)$ thus the value of the maximum flow in G' is $O(V)$
- In addition every time the residual graph is build the candidate flow is augmented in one.
- Thus, $O(VE') = O(VE)$.

Complexity

Using G'

Thus, given G, you build G' and run Ford-Fulkerson method. Then, use the max flow f to build the maximum matching by using:

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

Complexity

- Because we know the $|M| \leq \min\{L, R\} = O(V)$ thus the value of the maximum flow in G' is $O(V)$
- In addition every time the residual graph is build the candidate flow is augmented in one.
- Thus, $O(VE') = O(VE)$.

Complexity

Using G'

Thus, given G, you build G' and run Ford-Fulkerson method. Then, use the max flow f to build the maximum matching by using:

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

Complexity

- Because we know the $|M| \leq \min\{L, R\} = O(V)$ thus the value of the maximum flow in G' is $O(V)$
- In addition every time the residual graph is build the candidate flow is augmented in one.
- Thus, $O(VE') = O(VE)$.

Complexity

Using G'

Thus, given G, you build G' and run Ford-Fulkerson method. Then, use the max flow f to build the maximum matching by using:

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

Complexity

- Because we know the $|M| \leq \min\{L, R\} = O(V)$ thus the value of the maximum flow in G' is $O(V)$
- In addition every time the residual graph is build the candidate flow is augmented in one.
- Thus, $O(VE') = O(VE)$.

Outline

Exercises

1. 26.1-2
2. 26.1-4
3. 26.1-6
4. 26.2-3
5. 26.2-5
6. 26.2-8
7. 26.2-11