Discrete Optimization
Lecture-13

Ngày 21 tháng 11 năm 2011

The Max-Flow Min-Cut problem

Discussion

The max-flow min-cut problem is a central problem in discrete optimization with many applications.

We shall study this problem, look at some applications, learn how it interacts with linear programming and explore the Ford Fulkerson algorithm.

The Max-Flow Min-Cut problem

Discussion

The max-flow min-cut problem is a central problem in discrete optimization with many applications.

We shall study this problem, look at some applications, learn how it interacts with linear programming and explore the Ford Fulkerson algorithm.

Definition

A **network** *is a directed graph $D(V, E)$ with a function $c : E \longrightarrow R^+$, called capacity and two specified vertices S, T (source and sink or terminal).*

The Max-Flow Min-Cut problem

Discussion

The max-flow min-cut problem is a central problem in discrete optimization with many applications.

We shall study this problem, look at some applications, learn how it interacts with linear programming and explore the Ford Fulkerson algorithm.

Definition

A **network** *is a directed graph $D(V, E)$ with a function $c : E \longrightarrow R^+$, called capacity and two specified vertices S, T (source and sink or terminal).*

The Max-Flow Min-Cut problem

Discussion

The max-flow min-cut problem is a central problem in discrete optimization with many applications.

We shall study this problem, look at some applications, learn how it interacts with linear programming and explore the Ford Fulkerson algorithm.

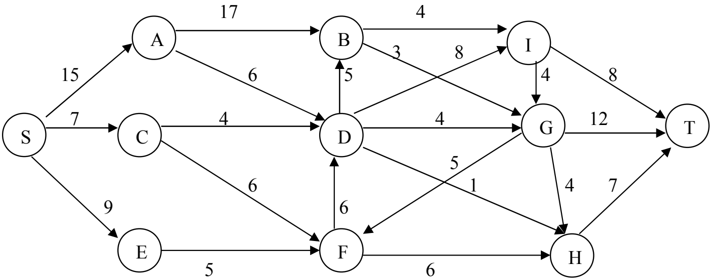
Definition

A **network** *is a directed graph $D(V, E)$ with a function $c : E \longrightarrow R^+$, called capacity and two specified vertices S, T (source and sink or terminal).*

We denote by $N^+(v)$ the edges in G of the form $v \to x$ and by $N^-(v)$ the edges $x \to v$.

Example-1

A network with integer capacities.



Preliminaries

Definition

A **flow** *is a function* $f : E \longrightarrow R^{+}$ *such that:*

Preliminaries

Definition

*A **flow** is a function $f : E \longrightarrow R^+$ such that:*

- *The amount of flow through each edge does not exceed its capacity ($f(e) \leq c(e)\ \forall e \in E$).*

Preliminaries

Definition

A **flow** *is a function* $f : E \longrightarrow R^+$ *such that:*

- *The amount of flow through each edge does not exceed its capacity* $(f(e) \leq c(e)\ \forall e \in E)$.
- *For every internal vertex (not S or T) "what comes in must go out."*

$$\sum_{e \in N^-(v)} f(e) = \sum_{e \in N^+(v)} f(e).$$

Preliminaries

Definition

A **flow** *is a function* $f : E \longrightarrow R^+$ *such that:*

- *The amount of flow through each edge does not exceed its capacity* ($f(e) \leq c(e)\ \forall e \in E$).
- *For every internal vertex (not* S *or* T*) "what comes in must go out."*

$$\sum_{e \in N^-(v)} f(e) = \sum_{e \in N^+(v)} f(e).$$

Preliminaries

Definition

A **flow** *is a function* $f : E \longrightarrow R^{+}$ *such that:*

- *The amount of flow through each edge does not exceed its capacity* ($f(e) \leq c(e)\ \forall e \in E$).
- *For every internal vertex (not* S *or* T*) "what comes in must go out."*

$$\sum_{e \in N^{-}(v)} f(e) = \sum_{e \in N^{+}(v)} f(e).$$

The size of a flow f is $\sum_{e \in N^{+}(S)} f(e)$.

Preliminaries

Definition

A **flow** *is a function* $f : E \longrightarrow R^+$ *such that:*

- *The amount of flow through each edge does not exceed its capacity* ($f(e) \leq c(e)\ \forall e \in E$).
- *For every internal vertex (not* S *or* T*) "what comes in must go out."*

$$\sum_{e \in N^-(v)} f(e) = \sum_{e \in N^+(v)} f(e).$$

The size of a flow f is $\sum_{e \in N^+(S)} f(e)$.

Observation

$$\sum_{e \in N^+(S)} f(e) = \sum_{e \in N^-(T)} f(e)$$

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let $x(e)$ $e \in E$ be variables.*

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let $x(e)$ $e \in E$ be variables.*
2. *maximize $\sum_{e \in N^+(S)} x(e)$ subject to:*

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let* $x(e)$ $e \in E$ *be variables.*
2. *maximize* $\sum_{e \in N^+(S)} x(e)$ *subject to:*
   - $x(e) \leq c(e)\ \forall e \in V(E)$

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let* $x(e)$ $e \in E$ *be variables.*
2. *maximize* $\sum_{e \in N^{+}(S)} x(e)$ *subject to:*
    - $x(e) \leq c(e)\ \forall e \in V(E)$
    - $\sum_{e \in N^{+}(v)} x(e) = \sum_{e \in N^{-}(v)} x(e)\ \forall v \notin \{S, T\}$

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let* $x(e)$ $e \in E$ *be variables.*
2. *maximize* $\sum_{e \in N^+(S)} x(e)$ *subject to:*
   - $x(e) \leq c(e) \; \forall e \in V(E)$
   - $\sum_{e \in N^+(v)} x(e) = \sum_{e \in N^-(v)} x(e) \; \forall v \notin \{S, T\}$
   - $x(e) \geq 0.$

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let $x(e)$ $e \in E$ be variables.*
2. *maximize $\sum_{e \in N^+(S)} x(e)$ subject to:*
   - $x(e) \leq c(e)\ \forall e \in V(E)$
   - $\sum_{e \in N^+(v)} x(e) = \sum_{e \in N^-(v)} x(e)\ \forall v \notin \{S, T\}$
   - $x(e) \geq 0$.

Question

Suppose all capacities are integers and the flows cannot be broken into fractions. Will the optimal solution be integers?

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let $x(e)$ $e \in E$ be variables.*
2. *maximize $\sum_{e \in N^+(S)} x(e)$ subject to:*
   - $x(e) \leq c(e)\ \forall e \in V(E)$
   - $\sum_{e \in N^+(v)} x(e) = \sum_{e \in N^-(v)} x(e)\ \forall v \notin \{S, T\}$
   - $x(e) \geq 0.$

Question

Suppose all capacities are integers and the flows cannot be broken into fractions. Will the optimal solution be integers?

The max-flow problem as an LP problem

Given a network, we wish to efficiently find a maximal flow .

Observation

The maximal flow in a network can be found by a linear program as follows:

1. *Let $x(e)$ $e \in E$ be variables.*
2. *maximize $\sum_{e \in N^+(S)} x(e)$ subject to:*
    - $x(e) \leq c(e)\ \forall e \in V(E)$
    - $\sum_{e \in N^+(v)} x(e) = \sum_{e \in N^-(v)} x(e)\ \forall v \notin \{S, T\}$
    - $x(e) \geq 0.$

Question

Suppose all capacities are integers and the flows cannot be broken into fractions. Will the optimal solution be integers?

Or do we need to add a constraint that $x(e)$ must be integers?

Integer Linear Programs: ILP

Question

Integer Linear Programs: ILP

Question

- *Are linear programs where an added constraint that all variables in the optimal solution be integers easier to solve?*

Integer Linear Programs: ILP

Question

- *Are linear programs where an added constraint that all variables in the optimal solution be integers easier to solve?*
- *After all, there might be only finitely many feasible solutions.*

Integer Linear Programs: ILP

Question

- *Are linear programs where an added constraint that all variables in the optimal solution be integers easier to solve?*
- *After all, there might be only finitely many feasible solutions.*
- *NO! ILP are among the most difficult computational problems to solve.*

Integer Linear Programs: ILP

Question

- *Are linear programs where an added constraint that all variables in the optimal solution be integers easier to solve?*
- *After all, there might be only finitely many feasible solutions.*
- *NO! ILP are among the most difficult computational problems to solve.*
- *But for some linear programs the simplex method is guaranteed to produce an integral solution.*

Integer Linear Programs: ILP

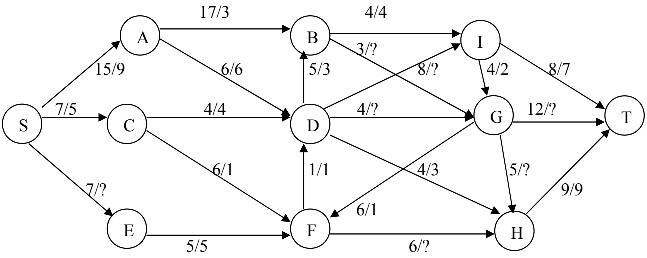
Question

- *Are linear programs where an added constraint that all variables in the optimal solution be integers easier to solve?*
- *After all, there might be only finitely many feasible solutions.*
- *NO! ILP are among the most difficult computational problems to solve.*
- *But for some linear programs the simplex method is guaranteed to produce an integral solution.*

Comment

As network flows in applications can be quite large, we seek an alternative, more effcicent algorithm for solving netwrok flow problems: preferably a combinatorial algorithm.

An example



In this network an entry of the form n/m represents a flow of size m through an edge with capacity n.

Let us try to fill in the "?" so that the flow represented by this diagram will be a proper flow.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

We shall first go through the algorithm, execute it on a sample and then identify its components analyse it and prove the min-cut max-flow theorem.

Definition (f-Augmenting Path)

An $S-T$ path P in a network N with a flow f is $f-augmenting$ if replacing the flow on every edge $e \in P$ by the amount on this edge in P increases the total flow.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:
  - If $u \to v$ and $c(u, v) > f(u, v)$ then $Label(v) = (u, +, w(v) = min\{c(u, v) - f(u, v), w(u)\})$.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:
  - If $u \to v$ and $c(u, v) > f(u, v)$ then $Label(v) = (u, +, w(v) = min\{c(u, v) - f(u, v), w(u)\})$.
  - If $u \leftarrow v$ and $f(v, u) > 0$ then $Label(v) = (u, -, min\{w(u), f(v, u)\})$ (flow can be decreased).

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:
  - If $u \rightarrow v$ and $c(u,v) > f(u,v)$ then $Label(v) = (u, +, w(v) = min\{c(u,v) - f(u,v), w(u)\})$.
  - If $u \leftarrow v$ and $f(v,u) > 0$ then $Label(v) = (u, -, min\{w(u), f(v,u)\})$ (flow can be decreased).
  - If T is labelled STOP. You have a flow augmenting path.

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:
  - If $u \rightarrow v$ and $c(u, v) > f(u, v)$ then $Label(v) = (u, +, w(v) = min\{c(u, v) - f(u, v), w(u)\})$.
  - If $u \leftarrow v$ and $f(v, u) > 0$ then $Label(v) = (u, -, min\{w(u), f(v, u)\})$ (flow can be decreased).
  - If T is labelled STOP. You have a flow augmenting path.
  - Augment the flow and execute the algorithm on the new flow.

Ford-Fulkerson's Max-Flow Min-Cut Algorithm

The algorithm has two procedures: "*Scan*" and "*Label*."

Ford-Fulkerson flow augmenting algorithm:

- Given a flow f on the network N.
- Label(S) = $(-\infty, +, +\infty)$.
- For a labelled vertex u do:
- Scan(u):
  - For every vertex v connected by an edge to u in any direction do:
  - If $u \rightarrow v$ and $c(u,v) > f(u,v)$ then $Label(v) = (u, +, w(v) = min\{c(u,v) - f(u,v), w(u)\})$.
  - If $u \leftarrow v$ and $f(v,u) > 0$ then $Label(v) = (u, -, min\{w(u), f(v,u)\})$ (flow can be decreased).
  - If T is labelled STOP. You have a flow augmenting path.
  - Augment the flow and execute the algorithm on the new flow.
- If T is not labelled the flow is a maximum flow.

Example

Let N be the network:

$S \to A: \; 3 \quad S \to C: \; 7$

$\mathrm{A} \to B: \; 5 \quad A \to D: \; 4 \quad A \to C: \; 2$

$\mathrm{B} \to D: \; 2 \quad B \to T: \; 8$

$\mathrm{C} \to B: \; 1 \quad C \to D: \; 4$

$\mathrm{D} \to T: \; 3$

Example

Let N be the network:

$$
\begin{array}{lll}
S \rightarrow A: \ 3 & S \rightarrow C: \ 7 & \\
\mathrm{A} \rightarrow B: \ 5 & A \rightarrow D: \ 4 & A \rightarrow C: \ 2 \\
\mathrm{B} \rightarrow D: \ 2 & B \rightarrow T: \ 8 & \\
\mathrm{C} \rightarrow B: \ 1 & C \rightarrow D: \ 4 & \\
\mathrm{D} \rightarrow T: \ 3 & &
\end{array}
$$

- $Label(S) = (-\infty, +, +\infty)$

Example

Let N be the network:

$S \to A: \ 3 \quad S \to C: \ 7$
$\mathrm{A} \to B: \ 5 \quad A \to D: \ 4 \quad A \to C: \ 2$
$\mathrm{B} \to D: \ 2 \quad B \to T: \ 8$
$\mathrm{C} \to B: \ 1 \quad C \to D: \ 4$
$\mathrm{D} \to T: \ 3$

- $Label(S) = (-\infty, +, +\infty)$
- $Label(C) = (S, +, 7) \quad Label(A) = (S, +, 3)$

Example

Let N be the network:

$$
\begin{array}{lll}
S \to A: \ 3 & S \to C: \ 7 & \\
\mathrm{A} \to B: \ 5 & A \to D: \ 4 & A \to C: \ 2 \\
\mathrm{B} \to D: \ 2 & B \to T: \ 8 & \\
\mathrm{C} \to B: \ 1 & C \to D: \ 4 & \\
\mathrm{D} \to T: \ 3 & &
\end{array}
$$

- $Label(S) = (-\infty, +, +\infty)$
- $Label(C) = (S, +, 7)$ $\quad Label(A) = (S, +, 3)$
- $Scan(C)$:
  Label(B) = (C, +, 1) Label(D) = (C, +, 4)

Example

Let N be the network:

$S \to A: \ 3 \quad S \to C: \ 7$
A $\to B: \ 5 \quad A \to D: \ 4 \quad A \to C: \ 2$
B $\to D: \ 2 \quad B \to T: \ 8$
C $\to B: \ 1 \quad C \to D: \ 4$
D $\to T: \ 3$

- $Label(S) = (-\infty, +, +\infty)$
- $Label(C) = (S, +, 7) \quad Label(A) = (S, +, 3)$
- $Scan(C)$:
  Label(B) = (C, +, 1)   Label(D) = (C, +, 4)
- $Scan(B)$:
  Label(T) = (B, +, 1)

Example

Let N be the network:

$S \to A: \ 3 \quad S \to C: \ 7$
A $\to B: \ 5 \quad A \to D: \ 4 \quad A \to C: \ 2$
B $\to D: \ 2 \quad B \to T: \ 8$
C $\to B: \ 1 \quad C \to D: \ 4$
D $\to T: \ 3$

- $Label(S) = (-\infty, +, +\infty)$
- $Label(C) = (S, +, 7) \quad Label(A) = (S, +, 3)$
- $Scan(C)$:
  Label(B) = (C, +, 1) Label(D) = (C, +, 4)
- $Scan(B)$:
  Label(T) = (B, +, 1)
- STOP! You have a flow augmenting path.

Using backflow: an example

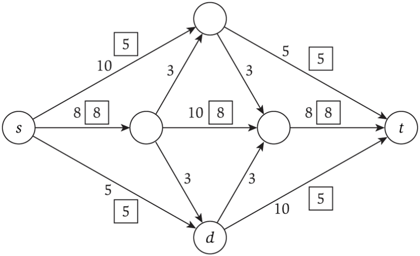


Figure 7.26 What is the value of the depicted flow? Is it a maximum flow? What is the minimum cut?

Analysis

Question

How bad can the Ford-Fulkerson algorithm be?

Analysis

Question

How bad can the Ford-Fulkerson algorithm be?

Answer

If you are not careful with the slection of your augmenting path, the algorithm can behave very bad indeed..

Analysis

Question

How bad can the Ford-Fulkerson algorithm be?

Answer

If you are not careful with the slection of your augmenting path, the algorithm can behave very bad indeed..

Analysis

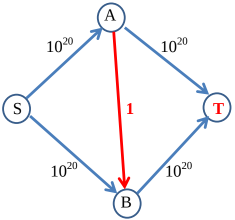
Question

How bad can the Ford-Fulkerson algorithm be?

Answer

If you are not careful with the slection of your augmenting path, the algorithm can behave very bad indeed..

Be careful how you select your augmenting paths.



Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of “scans” in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*
- *How can we use this observation to bound the number of iterations?*

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*
- *How can we use this observation to bound the number of iterations?*
- *At any iteration we have three types of edges:*

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*
- *How can we use this observation to bound the number of iterations?*
- *At any iteration we have three types of edges:*
  - *Edges e for which $f(e) < c(e)$.*

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*
- *How can we use this observation to bound the number of iterations?*
- *At any iteration we have three types of edges:*
    - *Edges e for which $f(e) < c(e)$.*
    - *Edges e for which $f(e) > 0$*

Analysis

Question

So how efficient is the Ford-Fulkerson algorithm? The size of the network should not depend on the capacity values. Can we estimate the number of "scans" in term of the number of edges and vertices?

Answer

- *A closer study of the algorithm shows that at every augmentation of a flow f either for some edge $e,\ f'(e) = c(e)$ or for some edge e with $f(e) > 0,\ f'(e) = 0$.*
- *How can we use this observation to bound the number of iterations?*
- *At any iteration we have three types of edges:*
  - *Edges e for which $f(e) < c(e)$.*
  - *Edges e for which $f(e) > 0$*
  - *Edges e for which $f(e) = c(e)$.*

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x, y)) < c((x, y))$ then $(x, y) \in E_f$.

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x, y)) < c((x, y))$ then $(x, y) \in E_f$.
- If $f((x, y)) > 0$ then $(y, x) \in E_f$

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x,y)) < c((x,y))$ then $(x,y) \in E_f$.
- If $f((x,y)) > 0$ then $(y,x) \in E_f$
- Note that if $0 < f((x,y)) < c((x,y))$ then both (x,y) *and* $(y,x) \in E_f$.

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x,y)) < c((x,y))$ then $(x,y) \in E_f$.
- If $f((x,y)) > 0$ then $(y,x) \in E_f$
- Note that if $0 < f((x,y)) < c((x,y))$ then both (x,y) *and* $(y,x) \in E_f$.

Observation

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x,y)) < c((x,y))$ then $(x,y) \in E_f$.
- If $f((x,y)) > 0$ then $(y,x) \in E_f$
- Note that if $0 < f((x,y)) < c((x,y))$ then both (x,y) *and* $(y,x) \in E_f$.

Observation

- *Let $\alpha(N)$ be the length of the shortest $S - T$ path in the network N.*

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x, y)) < c((x, y))$ then $(x, y) \in E_f$.
- If $f((x, y)) > 0$ then $(y, x) \in E_f$
- Note that if $0 < f((x, y)) < c((x, y))$ then both (x, y) *and* $(y, x) \in E_f$.

Observation

- *Let $\alpha(N)$ be the length of the shortest $S - T$ path in the network N.*
- *Let $\mathcal{A}(N) = \{e \in E \mid e \in$ some shortest $S - T$ path$\}$.*

Analysis, continued

We define a network $N_f = (V, E_f)$ as follows:

- If $f((x,y)) < c((x,y))$ then $(x,y) \in E_f$.
- If $f((x,y)) > 0$ then $(y,x) \in E_f$
- Note that if $0 < f((x,y)) < c((x,y))$ then both (x,y) *and* $(y,x) \in E_f$.

Observation

- *Let $\alpha(N)$ be the length of the shortest $S-T$ path in the network N.*
- *Let $\mathcal{A}(N) = \{e \in E \mid e \in$ some shortest $S-T$ path$\}$.*
- *If $N' = (V, E \cup \{(y,x)\})$ $(x,y) \in E$ then $\alpha(N') = \alpha(N)$ and $\mathcal{A}(N') = \mathcal{A}(N)$.*

Final remarks

If f_1 is an augmentation of the flow f along a shortest path in the network N then N_{f_1} is a subgraph of the netwrok $N_f \cup \{(y,x) \mid (x,y) \in N_f$.

Final remarks

If f_1 is an augmentation of the flow f along a shortest path in the network N then N_{f_1} is a subgraph of the netwrok $N_f \cup \{(y, x) \mid (x, y) \in N_f$.

Therefore $\alpha(N_{f_1}) = \alpha(N_f)$ but $\mathcal{A}(N_{f-1}) \subsetneq \mathcal{A}(N_f)$.

Final remarks

If f_1 is an augmentation of the flow f along a shortest path in the network N then N_{f_1} is a subgraph of the netwrok $N_f \cup \{(y,x) \mid (x,y) \in N_f$.

Therefore $\alpha(N_{f_1}) = \alpha(N_f)$ but $\mathcal{A}(N_{f-1}) \subsetneq \mathcal{A}(N_f)$.

So by choosing an augmentating path along a shortest path we guarantee that regardless of the capacities sizes, the number of edges in N_f is reduced.

Final remarks

If f_1 is an augmentation of the flow f along a shortest path in the network N then N_{f_1} is a subgraph of the netwrok $N_f \cup \{(y, x) \mid (x, y) \in N_f$.

Therefore $\alpha(N_{f_1}) = \alpha(N_f)$ but $\mathcal{A}(N_{f-1}) \subsetneq \mathcal{A}(N_f)$.

So by choosing an augmentating path along a shortest path we guarantee that regardless of the capacities sizes, the number of edges in N_f is reduced.

When the number of edges in N_f is zero we have a maximum flow.

Final remarks

If f_1 is an augmentation of the flow f along a shortest path in the network N then N_{f_1} is a subgraph of the netwrok $N_f \cup \{(y, x) \mid (x, y) \in N_f\}$.

Therefore $\alpha(N_{f_1}) = \alpha(N_f)$ but $\mathcal{A}(N_{f-1}) \subsetneq \mathcal{A}(N_f)$.

So by choosing an augmentating path along a shortest path we guarantee that regardless of the capacities sizes, the number of edges in N_f is reduced.

When the number of edges in N_f is zero we have a maximum flow.

So at each augmentation the algorithm will construct first all shortest paths (Dijkstra's algorithm) then scan all edges. So the running time of this algorithm will be $c \cdot |V| \cdot |E|^2$.

Definition

An $S - T$ **cut** *is a set of edges in a netwrok whose removal disconnects S from T.*

Definition

*An $S - T$ **cut** is a set of edges in a netwrok whose removal disconnects S from T.*

Example

The edges $(S, A), (S, C), (S, E)$ form a cut in Example-1.

Definition

*An $S - T$ **cut** is a set of edges in a netwrok whose removal disconnects S from T.*

Example

The edges $(S, A), (S, C), (S, E)$ form a cut in Example-1.

Theorem

If C is a cut in the network $N(V, E)$ and $f : E \to R^+$ is a flow of value V_0 then $c(C)$ the capcity of the cut is $\leq V_0$

Definition

*An $S - T$ **cut** is a set of edges in a netwrok whose removal disconnects S from T.*

Example

The edges $(S, A), (S, C), (S, E)$ form a cut in Example-1.

Theorem

If C is a cut in the network $N(V, E)$ and $f : E \to R^+$ is a flow of value V_0 then $c(C)$ the capcity of the cut is $\leq \ V_0$

Corollary (Min Cut Max Flow Theorem)

In a netwrok $N(V, E)$ the largest flow and the minimum cut have the same size.

Circulation

Definition

A weighted directed graph $D(V, E)$ with a weight function $f(u, v) \rightarrow R$ in which for every vertex $v \in V$:

$$\sum_{u \in N^+(v)} f(v, u) = \sum_{u \in N^-(v)} f(u, v)$$

is called a **circulation**.

Circulation

Definition

A weighted directed graph $D(V, E)$ with a weight function $f(u, v) \to R$ in which for every vertex $v \in V$:

$$\sum_{u \in N^+(v)} f(v, u) = \sum_{u \in N^-(v)} f(u, v)$$

is called a **circulation***.*

Comment

1. In a circulation the weights do not have to be positive.

Circulation

Definition

A weighted directed graph $D(V, E)$ with a weight function $f(u, v) \to R$ in which for every vertex $v \in V$:

$$\sum_{u \in N^{+}(v)} f(v, u) = \sum_{u \in N^{-}(v)} f(u, v)$$

is called a **circulation***.*

Comment

1. In a circulation the weights do not have to be positive.

Circulation

Definition

A weighted directed graph $D(V, E)$ with a weight function $f(u, v) \rightarrow R$ in which for every vertex $v \in V$:

$$\sum_{u \in N^+(v)} f(v, u) = \sum_{u \in N^-(v)} f(u, v)$$

is called a **circulation**.

Comment

1. In a circulation the weights do not have to be positive.

2. A network can be converted to a circulation if the total coming into the sink T is equal to the total coming out of the source S

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.
- $\forall e = (x, y) \in D_1$ there must be an edge $e' = (y, x) \in D_1$

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.
- $\forall e = (x, y) \in D_1$ there must be an edge $e' = (y, x) \in D_1$
- D_1 contains a cycle C.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.
- $\forall e = (x, y) \in D_1$ there must be an edge $e' = (y, x) \in D_1$
- D_1 contains a cycle C.
- Let $g'(e) = g(e) + \alpha \; \forall e \in C$ we get a new weight function.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \ \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.
- $\forall e = (x, y) \in D_1$ there must be an edge $e' = (y, x) \in D_1$
- D_1 contains a cycle C.
- Let $g'(e) = g(e) + \alpha \ \forall e \in C$ we get a new weight function.
- Clearly α can be chosen so that $\lfloor f(e) \rfloor \leq g'(e) + \alpha \leq \lceil f(e) \rceil$ for at least one edge of the cycle will be an integer.

Circulations

Theorem

If f is a weight function on a circulation $D(V, E)$ then there is an N-circulation g on D such that $\lfloor f(e) \rfloor \leq g(e) \leq \lceil f(e) \rceil \; \forall e \in E$.

- Let $\lfloor f(e) \rfloor \geq g(e) \geq \lceil f(e) \rceil$ be a weight function that maximizes the number of edges for which $g(e) \in Z$.
- Let D_1 be the subgraph of D spanned by the edges for which $g(e)$ is not an integer.
- $\forall e = (x, y) \in D_1$ there must be an edge $e' = (y, x) \in D_1$
- D_1 contains a cycle C.
- Let $g'(e) = g(e) + \alpha \; \forall e \in C$ we get a new weight function.
- Clearly α can be chosen so that $\lfloor f(e) \rfloor \leq g'(e) + \alpha \leq \lceil f(e) \rceil$ for at least one edge of the cycle will be an integer.
- This contradicts the choice of g.

Summary

Remark

Remark

- *If all capacities in a netwrok $N(V, e)$ are integers then the maximum flow is also an integer.*

Summary

Remark

- *If all capacities in a netwrok* $N(V, e)$ *are integers then the maximum flow is also an integer.*
- *If all capacities are rational then the Ford-Fulkerson algorithm will terminate.*

Summary

Remark

- *If all capacities in a netwrok $N(V, e)$ are integers then the maximum flow is also an integer.*
- *If all capacities are rational then the Ford-Fulkerson algorithm will terminate.*
- *Curiously, there are small netwroks with irrational capacities for which the Ford-Fulkerson algorithm will run forever.*

Summary

Remark

- *If all capacities in a netwrok $N(V, e)$ are integers then the maximum flow is also an integer.*
- *If all capacities are rational then the Ford-Fulkerson algorithm will terminate.*
- *Curiously, there are small netwroks with irrational capacities for which the Ford-Fulkerson algorithm will run forever.*

Summary

Remark

- *If all capacities in a netwrok $N(V, e)$ are integers then the maximum flow is also an integer.*
- *If all capacities are rational then the Ford-Fulkerson algorithm will terminate.*
- *Curiously, there are small netwroks with irrational capacities for which the Ford-Fulkerson algorithm will run forever.*

Netwrokflows have numerous applications in discrete optimization. We shall attempt to see some in the exercises.