Steiner Forests

J. Yancey

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 18, 2014

Outline

Outline

Outline

Outline

Problem Definition

Problem Statement

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

$$r(u, v) = \begin{cases}$$

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

$$r(u, v) = \begin{cases} 1 & \text{if } u \text{ and } v \text{ belong to the same set } S_i \end{cases}$$

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

$$r(u, v) = \begin{cases} 1 & \text{if } u \text{ and } v \text{ belong to the same set } S_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

$$r(u, v) = \begin{cases} 1 & \text{if } u \text{ and } v \text{ belong to the same set } S_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4. An equivalent formulation of the Steiner forest problem is to find a minimum cost subgraph $H \subseteq G$ such that if $r(u, v) = 1$, then there exists a path from u to v in H,

Problem Definition

Problem Statement

1. We are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{Q}^+$, and a collection of disjoint subsets of V, $\{S_1, \cdots, S_k\}$
2. Our goal is to find a minimum cost subgraph in which each pair of vertices belonging to the same set S_i are connected.
3. We define a *connectivity requirement function* $r : V \times V \to \{0, 1\}$ as follows:

$$r(u, v) = \begin{cases} 1 & \text{if } u \text{ and } v \text{ belong to the same set } S_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4. An equivalent formulation of the Steiner forest problem is to find a minimum cost subgraph $H \subseteq G$ such that if $r(u, v) = 1$, then there exists a path from u to v in H, or in other words, if $r(u, v) = 1$, then u and v are in the same connected component of H.

Outline

Integer Programming Formulation

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0, 1\}$ as follows:

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0, 1\}$ as follows:

$$f(S) = \begin{cases}$$

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0,1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u,v) = 1 \end{cases}$$

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0, 1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \to \{0, 1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$.

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0, 1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \to \{0, 1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.
3. For each edge $e \in E$ we introduce the variable x_e defined below:

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \to \{0, 1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.
3. For each edge $e \in E$ we introduce the variable x_e defined below:

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \rightarrow \{0,1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u,v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.
3. For each edge $e \in E$ we introduce the variable x_e defined below:

$$x_e = \left\{ \right.$$

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \to \{0,1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u,v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.
3. For each edge $e \in E$ we introduce the variable x_e defined below:

$$x_e = \begin{cases} 1 & \text{if } e \text{ is picked in the subgraph} \\ & \end{cases}$$

Integer Programming Formulation

1. We define a function on all cuts in G, $f : 2^V \to \{0,1\}$ as follows:

$$f(S) = \begin{cases} 1 & \text{if } \exists\ u \in S \text{ and } v \in \bar{S} \text{ such that } r(u,v) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

2. For $S \subseteq V$, let $\delta(S)$ denote the set of edges crossing the cut $(S, \bar{S})$. f can be viewed as a function on all cuts $(S, \bar{S})$ in G, which specifies the minimum value of $\delta(S)$ in any feasible solution.
3. For each edge $e \in E$ we introduce the variable x_e defined below:

$$x_e = \begin{cases} 1 & \text{if } e \text{ is picked in the subgraph} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Integer Program

Integer Program

- Our integer program is:

Integer Program

- Our integer program is:

$$\left\{ \right.$$

Integer Program

- Our integer program is:

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \end{cases}$$

Integer Program

- Our integer program is:

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e : e \in \delta(S)} x_e \geq f(S), S \subseteq V \end{cases}$$

Integer Program

- Our integer program is:

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e : e \in \delta(S)} x_e \geq f(S), S \subseteq V \\ & x_e \in \{0,1\}, e \in E \end{cases} \tag{4}$$

Outline

LP Relaxation

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

$$\left\{ \right.$$

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \end{cases}$$

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e : e \in \delta(S)} x_e \geq f(S), S \subseteq V \end{cases}$$

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e: e \in \delta(S)} x_e \geq f(S), S \subseteq V \\ & x_e \geq 0, e \in E \end{cases} \tag{5}$$

LP Relaxation

Following our LP relaxation we drop the redundant conditions $x_e \leq 1$.

$$\begin{cases} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \sum_{e: e \in \delta(S)} x_e \geq f(S), S \subseteq V \\ & x_e \geq 0, e \in E \end{cases} \tag{5}$$

Dual Program

Dual Program

Our dual program is:

$$\begin{cases}$$

Dual Program

Our dual program is:

$$\begin{cases} \text{maximize} & \sum_{S \subseteq V} f(S) \cdot y_S \\ & \\ & \end{cases}$$

Dual Program

Our dual program is:

$$\begin{cases} \text{maximize} & \sum_{S \subseteq V} f(S) \cdot y_S \\ \text{subject to} & \sum_{S : e \in \delta(S)} y_S \leq c_e, e \in E \end{cases}$$

Dual Program

Our dual program is:

$$\begin{cases} \text{maximize} & \sum_{S \subseteq V} f(S) \cdot y_S \\ \text{subject to} & \sum_{S: e \in \delta(S)} y_S \leq c_e, e \in E \\ & y_S \geq 0, S \subseteq V \end{cases} \tag{6}$$

Outline

Terminology

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$
- at any iteration, set S is *unsatisfied* if $f(S) = 1$ but no picked edge crosses the cut $(S, \bar{S})$

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$
- at any iteration, set S is *unsatisfied* if $f(S) = 1$ but no picked edge crosses the cut $(S, \bar{S})$
- set S is *active* if it is a minimal unsatisfied set in the current iteration.

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$
- at any iteration, set S is *unsatisfied* if $f(S) = 1$ but no picked edge crosses the cut $(S, \bar{S})$
- set S is *active* if it is a minimal unsatisfied set in the current iteration.

Note

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$
- at any iteration, set S is *unsatisfied* if $f(S) = 1$ but no picked edge crosses the cut $(S, \bar{S})$
- set S is *active* if it is a minimal unsatisfied set in the current iteration.

Note

- Notice that raising a set S for which $f(S) = 0$ does not affect the dual objective function.

Terminology

- set S has been *raised* in a dual solution if $y_S > 0$
- edge e is *tight* if $\sum_{S:e\in\delta(S)} y_S = c_e$
- at any iteration, set S is *unsatisfied* if $f(S) = 1$ but no picked edge crosses the cut $(S, \bar{S})$
- set S is *active* if it is a minimal unsatisfied set in the current iteration.

Note

- Notice that raising a set S for which $f(S) = 0$ does not affect the dual objective function. So we can assume that such sets are never raised.

Outline

Primal-dual Schema

Primal Conditions

Primal-dual Schema

Primal Conditions

1. For each $e \in E_i, x_e = 1$ implies $\sum_{i:e \in \delta(S)} y_S = c_e$

Primal-dual Schema

Primal Conditions

1. For each $e \in E_i, x_e = 1$ implies $\sum_{i:e \in \delta(S)} y_S = c_e$
2. In other words, every picked edge must be *tight*.

Primal-dual Schema

Primal Conditions

1. For each $e \in E_i, x_e = 1$ implies $\sum_{i:e \in \delta(S)} y_S = c_e$
2. In other words, every picked edge must be *tight*.
3. The algorithm starts with null primal and dual solutions.

Primal-dual Schema

Primal Conditions

1. For each $e \in E_i, x_e = 1$ implies $\sum_{i:e \in \delta(S)} y_S = c_e$
2. In other words, every picked edge must be *tight*.
3. The algorithm starts with null primal and dual solutions. At any given iteration, the current primal solution indicates which sets need to be raised

Primal-dual Schema

Primal Conditions

1. For each $e \in E_i, x_e = 1$ implies $\sum_{i:e \in \delta(S)} y_S = c_e$
2. In other words, every picked edge must be *tight*.
3. The algorithm starts with null primal and dual solutions. At any given iteration, the current primal solution indicates which sets need to be raised and the current dual solution indicates which edge needs to be picked.

Lemma 1

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.
- So S must be a union of connected components.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.
- So S must be a union of connected components.
- Since $f(S) = 1$, there must be a vertices $u \in S, v \in \bar{S}$ such that $r(u, v) = 1$.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.
- So S must be a union of connected components.
- Since $f(S) = 1$, there must be a vertices $u \in S, v \in \bar{S}$ such that $r(u, v) = 1$.
- Let S' be the connected component that contains u, then S' is also unsatisfied.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.
- So S must be a union of connected components.
- Since $f(S) = 1$, there must be a vertices $u \in S, v \in \bar{S}$ such that $r(u, v) = 1$.
- Let S' be the connected component that contains u, then S' is also unsatisfied.
- Since S is minimal, we must have $S' = S$.

Lemma 1

1. Set S is *active* iff it is a connected component in the currently picked forest and $f(S) = 1$.

Proof

- Let S be an active set.
- If S contains part of a connected component, then there would already be an edge crossing the cut $(S, \bar{S})$ which would contradict the fact that S is active.
- So S must be a union of connected components.
- Since $f(S) = 1$, there must be a vertices $u \in S, v \in \bar{S}$ such that $r(u, v) = 1$.
- Let S' be the connected component that contains u, then S' is also unsatisfied.
- Since S is minimal, we must have $S' = S$.
- The reverse follows by the definition of an active set.

Outline

Description of Algorithm

Description

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.
2. The algorithm will then raise the dual variables of these sets in a synchronized manner until some edge goes tight.

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.
2. The algorithm will then raise the dual variables of these sets in a synchronized manner until some edge goes tight.
3. A tight edge is then picked arbitrarily and the iteration terminates.

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.
2. The algorithm will then raise the dual variables of these sets in a synchronized manner until some edge goes tight.
3. A tight edge is then picked arbitrarily and the iteration terminates.
4. This process ends when a primal feasible solution F is found.

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.
2. The algorithm will then raise the dual variables of these sets in a synchronized manner until some edge goes tight.
3. A tight edge is then picked arbitrarily and the iteration terminates.
4. This process ends when a primal feasible solution F is found.
5. However, F might contain *redundant* edges, that is, edges without which F would still be a feasible solution.

Description of Algorithm

Description

1. By lemma 1, all active sets can be easily found in the current iteration.
2. The algorithm will then raise the dual variables of these sets in a synchronized manner until some edge goes tight.
3. A tight edge is then picked arbitrarily and the iteration terminates.
4. This process ends when a primal feasible solution F is found.
5. However, F might contain *redundant* edges, that is, edges without which F would still be a feasible solution.
6. So the algorithm drops all redundant edges and returns newly *pruned* F.

Outline

Algorithm

2-Approximation Steiner Forest

1. (**Initialization**)
   $F \leftarrow \emptyset$
   for each $S \subseteq V$, $y_S \leftarrow 0$.
2. (**Edge augmentation**)
   while there exists an unsatisfied set do:
   simultaneously raise y_S for each active set S, until some edge e goes tight: $F \leftarrow F \cup \{e\}$.
3. (**Pruning**)
   return $F' = \{e \in F | F' - \{e\}$ is primal unfeasible $\}$

Outline

Example

Example

- Consider the graph in figure 1.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u, a) and (v, b) go tight.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u, a) and (v, b) go tight.
- One of them, say (u, a) is picked, and the iteration ends.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u, a) and (v, b) go tight.
- One of them, say (u, a) is picked, and the iteration ends.
- In the second iteration, $\{u, a\}$ replaces $\{u\}$ as an active set.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u,v) = 1$ and $r(s,t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u,a) and (v,b) go tight.
- One of them, say (u,a) is picked, and the iteration ends.
- In the second iteration, $\{u,a\}$ replaces $\{u\}$ as an active set.
- However, in this iteration, there is no need to raise duals, since there is already a tight edge, (v,b).

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u,v) = 1$ and $r(s,t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u,a) and (v,b) go tight.
- One of them, say (u,a) is picked, and the iteration ends.
- In the second iteration, $\{u,a\}$ replaces $\{u\}$ as an active set.
- However, in this iteration, there is no need to raise duals, since there is already a tight edge, (v,b).
- This edge is picked, and the iteration terminates.

Example

- Consider the graph in figure 1. Costs of edges are marked, and the only nonzero connectivity requirements are $r(u, v) = 1$ and $r(s, t) = 1$. Thick edges indicate an optimal solution of cost 45.
- In the first iteration, the following four singleton sets are active: $\{s\}, \{t\}, \{u\}$, and $\{v\}$.
- When their dual variables are raised to 6 each, edge (u, a) and (v, b) go tight.
- One of them, say (u, a) is picked, and the iteration ends.
- In the second iteration, $\{u, a\}$ replaces $\{u\}$ as an active set.
- However, in this iteration, there is no need to raise duals, since there is already a tight edge, (v, b).
- This edge is picked, and the iteration terminates.
- The primal and dual solutions at this point are show in Figure 2, with picked edges marked thick.

Example cont.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.
- When they are raised by 1 each, edge (b, t) goes tight and is picked.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.
- When they are raised by 1 each, edge (b, t) goes tight and is picked.
- Figure 3 shows the current situation.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.
- When they are raised by 1 each, edge (b, t) goes tight and is picked.
- Figure 3 shows the current situation.
- In the fifth iteration, the active sets are $\{a, s, u\}$ and $\{b, v, t\}$.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.
- When they are raised by 1 each, edge (b, t) goes tight and is picked.
- Figure 3 shows the current situation.
- In the fifth iteration, the active sets are $\{a, s, u\}$ and $\{b, v, t\}$.
- When they are raised by 1 each, (u, v) goes tight, and we now have a primal feasible solution shown in figure 4.

Example cont.

- In the third iteration, $\{v, b\}$ replaces $\{v\}$ as an active set.
- When the active sets are raised by 2 each, edge (u, s) goes tight and is picked.
- In the fourth iteration, the active sets are $\{u, s, a\}, \{v\}$, and $\{t\}$.
- When they are raised by 1 each, edge (b, t) goes tight and is picked.
- Figure 3 shows the current situation.
- In the fifth iteration, the active sets are $\{a, s, u\}$ and $\{b, v, t\}$.
- When they are raised by 1 each, (u, v) goes tight, and we now have a primal feasible solution shown in figure 4.
- In the pruning step, edge (u, a) is deleted, and we obtain the following solution of cost 54.

Outline

Analysis

Lemma 2

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.
5. Now, when an edge goes tight, the current iteration terminates and active sets are redefined, so no edge is overtightened.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.
5. Now, when an edge goes tight, the current iteration terminates and active sets are redefined, so no edge is overtightened.
6. This means that y is a feasible solution for the dual program.

Analysis

Lemma 2

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.
5. Now, when an edge goes tight, the current iteration terminates and active sets are redefined, so no edge is overtightened.

Analysis

Lemma 2

At the end of the algorithm, F' and y are primal and dual feasible solutions, respectively.

Proof

1. At the end of Step 2, F satisfies all connectivity requirements (else another iteration would follow).
2. In each iteration, only connected sets are raised, so no edge between two vertices in the same component can go tight, and so F is acyclic, .e., it is a forest.
3. Hence, if $r(u, v) = 1$, there is a unique path from u to v in F, and each edge on this path is nonredundant and will not be removed in Step 3.
4. Thus, F' is a feasible solution.
5. Now, when an edge goes tight, the current iteration terminates and active sets are redefined, so no edge is overtightened.
6. This means that y is a feasible solution for the dual program.

Analysis

Lemma 3

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

1. Suppose, for the sake of contradiction, that$f(C) = 0$, and $deg_{F'}(C) = 1$.

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

1. Suppose, for the sake of contradiction, that$f(C) = 0$, and $deg_{F'}(C) = 1$.
2. Let e be the unique edge of F' crossing the cut $(C, \bar{C})$.

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

1. Suppose, for the sake of contradiction, that$f(C) = 0$, and $deg_{F'}(C) = 1$.
2. Let e be the unique edge of F' crossing the cut $(C, \bar{C})$.
3. Since $e \in F'$, we know that e is not redundant (every edge in F' is nonredundent), so there exists vertices, u, v, such that $r(u, v) = 1$ and e is part of the unique path between them.

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

1. Suppose, for the sake of contradiction, that$f(C) = 0$, and $deg_{F'}(C) = 1$.
2. Let e be the unique edge of F' crossing the cut $(C, \bar{C})$.
3. Since $e \in F'$, we know that e is not redundant (every edge in F' is nonredundent), so there exists vertices, u, v, such that $r(u, v) = 1$ and e is part of the unique path between them.
4. Since e is the only edge crossing the cut $(C, \bar{C})$, it must be the case that one of these vertices is in C and the other in $\bar{C}$.

Analysis

Lemma 3

Consider any iteration of the algorithm, and let C be a component w.r.t. the currently picked edges. If $f(C) = 0$ then $deg_{F'}(C) \neq 1$.

Proof

1. Suppose, for the sake of contradiction, that$f(C) = 0$, and $deg_{F'}(C) = 1$.
2. Let e be the unique edge of F' crossing the cut $(C, \bar{C})$.
3. Since $e \in F'$, we know that e is not redundant (every edge in F' is nonredundent), so there exists vertices, u, v, such that $r(u, v) = 1$ and e is part of the unique path between them.
4. Since e is the only edge crossing the cut $(C, \bar{C})$, it must be the case that one of these vertices is in C and the other in $\bar{C}$.
5. But we know $r(u, v) = 1$, so $f(C) = 1$, which is a contradiction.

Analysis

Lemma 4

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S \tag{8}$$

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S \tag{8}$$

2. If we change the order of the summations, we get

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S \tag{8}$$

2. If we change the order of the summations, we get

$$\sum_{e \in F'} c_e = \sum_{S \subseteq V} \sum_{e \in \delta(S) \cap F'} y_S = \sum_{S \subseteq V} deg_{F'}(S) \cdot y_S \tag{9}$$

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S \tag{8}$$

2. If we change the order of the summations, we get

$$\sum_{e \in F'} c_e = \sum_{S \subseteq V} \sum_{e \in \delta(S) \cap F'} y_S = \sum_{S \subseteq V} deg_{F'}(S) \cdot y_S \tag{9}$$

3. It is therefore sufficient to show that

Analysis

Lemma 4

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{7}$$

Proof

1. Since every picked edge is tight, so

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S : e \in \delta(S)} y_S \tag{8}$$

2. If we change the order of the summations, we get

$$\sum_{e \in F'} c_e = \sum_{S \subseteq V} \sum_{e \in \delta(S) \cap F'} y_S = \sum_{S \subseteq V} deg_{F'}(S) \cdot y_S \tag{9}$$

3. It is therefore sufficient to show that

$$\sum_{S \subseteq V} deg_{F'}(S) \cdot y_S \leq 2 \sum_{S \subseteq V} y_S \tag{10}$$

Proof cont.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{S \text{active}} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{S \text{active}} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration. Shrink each one of these components to a node and call the new graph H'.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration. Shrink each one of these components to a node and call the new graph H'. Let V' and E' denote the set of vertices and edges of H', respectively.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{S \text{active}} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{S \text{active}} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration. Shrink each one of these components to a node and call the new graph H'. Let V' and E' denote the set of vertices and edges of H', respectively. Notice that all vertices picked before the current iteration have been shrunk.

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration. Shrink each one of these components to a node and call the new graph H'. Let V' and E' denote the set of vertices and edges of H', respectively. Notice that all vertices picked before the current iteration have been shrunk. Further notice that if the node $s \in V'$ corresponds to the connected component S in H, then

Proof cont.

- We will prove that in each iteration, the change in the left hand side is at most the change in the right hand side.
- Consider an iteration and let $\triangle$ be the amount by which the dual variables were raised in this iteration. We want to prove that:

$$\triangle \times \sum_{Sactive} deg_{F'}(S) \leq 2\triangle \times (\sharp \text{ active sets}) \tag{11}$$

- or equivalently,

$$\frac{\sum_{Sactive} deg_{F'}(S)}{\sharp \text{ active sets}} \leq 2 \tag{12}$$

- Let $H = (V, F')$ and consider the set of connected components w.r.t F at the current iteration. Shrink each one of these components to a node and call the new graph H'. Let V' and E' denote the set of vertices and edges of H', respectively. Notice that all vertices picked before the current iteration have been shrunk. Further notice that if the node $s \in V'$ corresponds to the connected component S in H, then

$$deg_{F'}(S) = deg_{H'}(s) \tag{13}$$

Proof cont.

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise.

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

$$\frac{\sum_{S \text{ active}} deg_{H'}(s)}{\sharp \text{ active nodes in } H'} \leq 2 \tag{14}$$

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

$$\frac{\sum_{S \text{ active}} deg_{H'}(s)}{\sharp \text{ active nodes in } H'} \leq 2 \tag{14}$$

- or equivalently, that

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

$$\frac{\sum_{S \text{ active}} deg_{H'}(s)}{\sharp \text{ active nodes in } H'} \leq 2 \tag{14}$$

- or equivalently, that

$$\sum_{s \in V'} deg_{H'}(s) - \sum_{s \text{ inactive}} deg_{H'}(s) \leq 2 \times (\sharp \text{ active nodes in } H') \tag{15}$$

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

$$\frac{\sum_{S \text{ active}} deg_{H'}(s)}{\sharp \text{ active nodes in } H'} \leq 2 \tag{14}$$

- or equivalently, that

$$\sum_{s \in V'} deg_{H'}(s) - \sum_{s \text{ inactive}} deg_{H'}(s) \leq 2 \times (\sharp \text{ active nodes in } H') \tag{15}$$

- Notice that if $s \in V'$ is an active node, then $deg_{H'}(s) \geq 0$ because there must be an edge incident to it to satisfy connectivity requirements.

Proof cont.

- Call a node $s \in H'$ *active* if it corresponds to an active set in H and *inactive* otherwise. We want to prove that:

$$\frac{\sum_{S \text{ active}} deg_{H'}(s)}{\sharp \text{ active nodes in } H'} \leq 2 \tag{14}$$

- or equivalently, that

$$\sum_{s \in V'} deg_{H'}(s) - \sum_{s \text{ inactive}} deg_{H'}(s) \leq 2 \times (\sharp \text{ active nodes in } H') \tag{15}$$

- Notice that if $s \in V'$ is an active node, then $deg_{H'}(s) \geq 0$ because there must be an edge incident to it to satisfy connectivity requirements.
- If s is inactive then by lemma 3 we know that $deg_{H'}(s) \neq 1$, so either $deg_{H'}(s) = 0$ or $deg_{H'}(s) \geq 2$.

Proof cont.

Proof cont.

- By the handshaking lemma we know that

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H';

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H'; notice that any vertex we remove is inactive and that the inactive nodes that remain have degree greater than or equal to 2.

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H'; notice that any vertex we remove is inactive and that the inactive nodes that remain have degree greater than or equal to 2. Thus

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H'; notice that any vertex we remove is inactive and that the inactive nodes that remain have degree greater than or equal to 2. Thus

$$\sum_{s \text{ inactive}} deg_{H'}(s) \geq 2 \times (\sharp \text{ inactive nodes in } H') \tag{17}$$

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H'; notice that any vertex we remove is inactive and that the inactive nodes that remain have degree greater than or equal to 2. Thus

$$\sum_{s \text{ inactive}} deg_{H'}(s) \geq 2 \times (\sharp \text{ inactive nodes in } H') \tag{17}$$

- This means that

Proof cont.

- By the handshaking lemma we know that

$$\sum_{s \in V'} deg_{H'}(s) = 2|E'| = 2|V'| - 2 \times (\sharp \text{ connected components of } H') \leq 2|V'| \tag{16}$$

- This inequality still holds if we remove isolated vertices from V' because each time we remove an isolated vertex, both $|V'|$ and the number of connected components decrease by 1.
- So remove any isolated vertices from H'; notice that any vertex we remove is inactive and that the inactive nodes that remain have degree greater than or equal to 2. Thus

$$\sum_{s \text{ inactive}} deg_{H'}(s) \geq 2 \times (\sharp \text{ inactive nodes in } H') \tag{17}$$

- This means that

$$\sum_{s \in V'} deg_{H'}(s) - \sum_{s \text{ inactive}} deg_{H'}(s) \leq 2|V'| - 2 \times (\sharp \text{ of inactive nodes in } H') \tag{18}$$

$$= 2 \times (\sharp \text{ active nodes in } H') \tag{19}$$

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively.

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively. By lemma 3, we know that $\sum_{e \in F'} c_e \leq 2 \sum_{s \subseteq V} y_S$.

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively. By lemma 3, we know that $\sum_{e \in F'} c_e \leq 2 \sum_{s \subseteq V} y_S$. Since no set S for which $f(S) = 0$ was ever raised, we know that $2 \sum_{S \subseteq V} y_S = \sum_{S \subseteq V} f(S) \cdot y_S$. This is the dual objective function, which means that if OPT_f is the optimal solution to the dual program, then

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively. By lemma 3, we know that $\sum_{e \in F'} c_e \leq 2 \sum_{s \subseteq V} y_S$. Since no set S for which $f(S) = 0$ was ever raised, we know that $2 \sum_{S \subseteq V} y_S = \sum_{S \subseteq V} f(S) \cdot y_S$. This is the dual objective function, which means that if OPT_f is the optimal solution to the dual program, then

$$\sum_{S \subseteq V} f(S) \cdot y_S \leq OPT_f \tag{20}$$

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively. By lemma 3, we know that $\sum_{e \in F'} c_e \leq 2 \sum_{s \subseteq V} y_S$. Since no set S for which $f(S) = 0$ was ever raised, we know that $2 \sum_{S \subseteq V} y_S = \sum_{S \subseteq V} f(S) \cdot y_S$. This is the dual objective function, which means that if OPT_f is the optimal solution to the dual program, then

$$\sum_{S \subseteq V} f(S) \cdot y_S \leq OPT_f \tag{20}$$

But we know that the optimal solution to the fractional program is at most the optimal solution to the integer program, so

Theorem

The algorithm given above is a 2-factor approximation algorithm to the Steiner forest problem.

Proof.

By lemma 2 we know that F' and y are primal and dual feasible solutions, respectively. By lemma 3, we know that $\sum_{e \in F'} c_e \leq 2 \sum_{s \subseteq V} y_S$. Since no set S for which $f(S) = 0$ was ever raised, we know that $2 \sum_{S \subseteq V} y_S = \sum_{S \subseteq V} f(S) \cdot y_S$. This is the dual objective function, which means that if OPT_f is the optimal solution to the dual program, then

$$\sum_{S \subseteq V} f(S) \cdot y_S \leq OPT_f \tag{20}$$

But we know that the optimal solution to the fractional program is at most the optimal solution to the integer program, so

$$2 \sum_{e \in F'} c_e \leq 2 \cdot OPT_f \leq 2 \cdot OPT \tag{21}$$

□