Multiway Cut

Hrishee Shastri

CS 441

Spring 2021

Cuts

- Input: A connected, undirected graph $G = (V, E)$ with weights to edges $w : E \to \mathbb{R}^+$.

Cuts

- Input: A connected, undirected graph $G = (V, E)$ with weights to edges $w : E \to \mathbb{R}^+$.
- A **cut** is defined as a partition of V into two sets V' and $V \setminus V'$, and consists of all edges that have an endpoint in both sets.

Cuts

- Input: A connected, undirected graph $G = (V, E)$ with weights to edges $w : E \rightarrow \mathbb{R}^+$.
- A **cut** is defined as a partition of V into two sets V' and $V \setminus V'$, and consists of all edges that have an endpoint in both sets.
- Given two nodes $s, t \in V$, an s-t **cut** is a partition of V that disconnects s and t (i.e. $s \in V'$ and $t \in V \setminus V'$ or vice versa).

Cuts

- Input: A connected, undirected graph $G = (V, E)$ with weights to edges $w : E \to \mathbb{R}^+$.
- A **cut** is defined as a partition of V into two sets V' and $V \setminus V'$, and consists of all edges that have an endpoint in both sets.
- Given two nodes $s, t \in V$, an s-t **cut** is a partition of V that disconnects s and t (i.e. $s \in V'$ and $t \in V \setminus V'$ or vice versa).
- The minimum-weight s-t cut can be found in polynomial time using the max flow algorithm.

Example

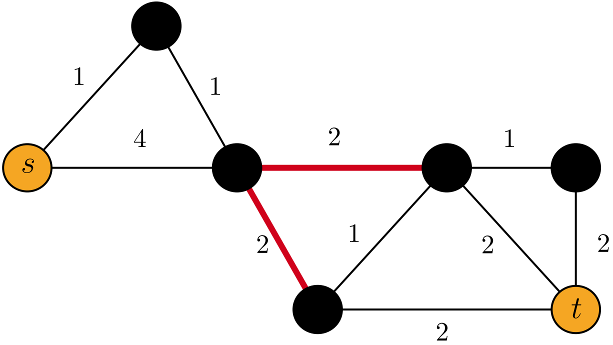


Figure: Minimum weight s-t cut.

- Edges in red are a minimum-weight $s - t$ cut because their removal disconnects s from t in G.

Multiway Cut

- Generalization of s-t cut.

Multiway Cut

- Generalization of s-t cut.
- Given a set of terminals $S = \{s_1, s_2, ..., s_k\}$, a **multiway cut** is a set of edges whose removal disconnects every terminal from one another.

Multiway Cut

- Generalization of s-t cut.
- Given a set of terminals $S = \{s_1, s_2, ..., s_k\}$, a **multiway cut** is a set of edges whose removal disconnects every terminal from one another.
- **The Multiway Cut Problem** requires us to find the multiway cut with minimum-weight.

Multiway Cut

- Generalization of s-t cut.
- Given a set of terminals $S = \{s_1, s_2, ..., s_k\}$, a **multiway cut** is a set of edges whose removal disconnects every terminal from one another.
- **The Multiway Cut Problem** requires us to find the multiway cut with minimum-weight.
- Multiway Cut is NP-Hard for $k \geq 3$.

Example

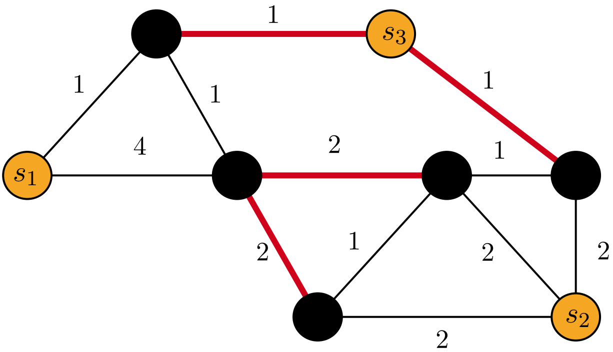


Figure: Minimum weight multiway cut.

- Edges in red are the minimum-weight multiway cut because their removal disconnects every pair of s_1, s_2 and s_3.

Example

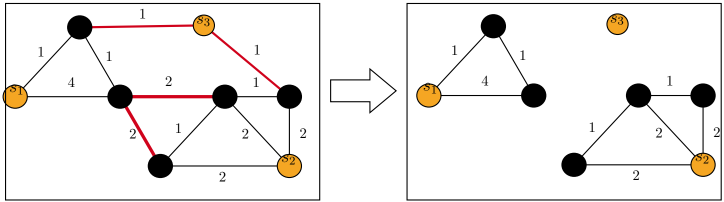


Figure: Removing the multiway cut creates k connected components.

A $2 - 2/k$ Approximation

- An **isolating cut** for terminal s_i is a set of edges whose removal disconnects s_i from all other terminals in G.

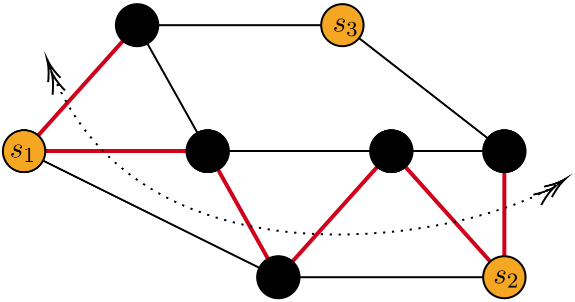


Figure: An isolating cut for terminal s_3.

A $2 - 2/k$ Approximation

- Algorithm 1

A $2 - 2/k$ Approximation

- Algorithm 1
    1. For each $i = 1, ..., k$, compute C_i, the minimum-weight isolating cut for terminal s_i.

A $2 - 2/k$ Approximation

- Algorithm 1
  1. For each $i = 1, ..., k$, compute C_i, the minimum-weight isolating cut for terminal s_i.
  2. Output C, the union of the $k - 1$ least expensive isolating cuts.

A $2 - 2/k$ Approximation

- Algorithm 1
  1. For each $i = 1, ..., k$, compute C_i, the minimum-weight isolating cut for terminal s_i.
  2. Output C, the union of the $k-1$ least expensive isolating cuts.
- Computing C_i can be done in polynomial time by collapsing terminals $S \setminus \{s_i\}$ into one node t and then finding the minimum weight s_i-t cut in G.

A $2 - 2/k$ Approximation

- Algorithm 1
  1. For each $i = 1, ..., k$, compute C_i, the minimum-weight isolating cut for terminal s_i.
  2. Output C, the union of the $k-1$ least expensive isolating cuts.
- Computing C_i can be done in polynomial time by collapsing terminals $S \setminus \{s_i\}$ into one node t and then finding the minimum weight s_i-t cut in G.
- C is a multiway cut because it contains an isolating cut for $k-1$ terminals, and thus the k^{th} terminal must be isolated as well.

Computing a Minimum-Weight Isolating Cut

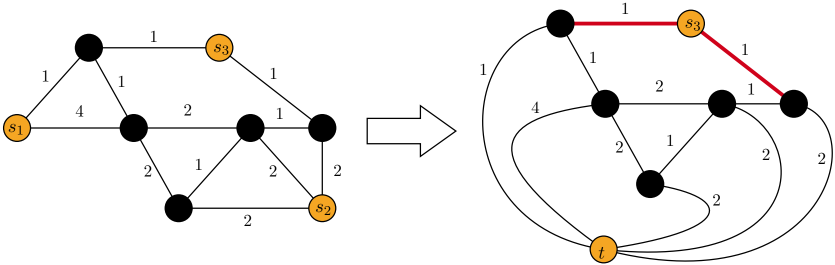


Figure: Computing C_3, an isolating cut for s_3, is done by collapsing s_1, s_2 into a node t and then outputting the minimum-weight s_3-t cut. C_3 comprises of the edges in red.

Proof of $2 - 2/k$ Approximation

- Let A be an optimal (minimum-weight) multiway cut in G.

Proof of $2 - 2/k$ Approximation

- Let A be an optimal (minimum-weight) multiway cut in G.
- Then removing the edges in A from G creates k connected components, each of which has exactly one terminal.

Proof of $2 - 2/k$ Approximation

- Let A be an optimal (minimum-weight) multiway cut in G.
- Then removing the edges in A from G creates k connected components, each of which has exactly one terminal.
- Write $A = \bigcup_{i=1}^{k} A_i$, where $A_i \subset A$ is the cut that disconnects terminal s_i from all other terminals.

Proof of $2 - 2/k$ Approximation

- Let A be an optimal (minimum-weight) multiway cut in G.
- Then removing the edges in A from G creates k connected components, each of which has exactly one terminal.
- Write $A = \bigcup_{i=1}^{k} A_i$, where $A_i \subset A$ is the cut that disconnects terminal s_i from all other terminals.
- Each edge in A has an endpoint in two connected components, so each edge in A will be in two different A_i.

Proof of $2 - 2/k$ Approximation

- Let A be an optimal (minimum-weight) multiway cut in G.
- Then removing the edges in A from G creates k connected components, each of which has exactly one terminal.
- Write $A = \bigcup_{i=1}^{k} A_i$, where $A_i \subset A$ is the cut that disconnects terminal s_i from all other terminals.
- Each edge in A has an endpoint in two connected components, so each edge in A will be in two different A_i.
  - This means $\sum_{i=1}^{k} w(A_i) = 2w(A)$, where $w(A)$ is the total weight of the edges in cut A.

Proof of $2 - 2/k$ Approximation (cont.)

- Note that $w(C_i) \leq w(A_i)$ because C_i is a minimum-weight isolating cut for terminal s_i, and A_i is some isolating cut for s_i.

Proof of $2 - 2/k$ Approximation (cont.)

- Note that $w(C_i) \leq w(A_i)$ because C_i is a minimum-weight isolating cut for terminal s_i, and A_i is some isolating cut for s_i.
- Recall that C, the multiway cut returned by our algorithm, is obtained by taking the union of all C_i except the heaviest one C_j'. This gives

$$w(C) \leq \left(\sum_{i=1}^{k} w(C_i) \right) - w(C_j'). \tag{1}$$

Proof of $2 - 2/k$ Approximation (cont.)

- Note that $w(C_i) \leq w(A_i)$ because C_i is a minimum-weight isolating cut for terminal s_i, and A_i is some isolating cut for s_i.
- Recall that C, the multiway cut returned by our algorithm, is obtained by taking the union of all C_i except the heaviest one C'_j. This gives

$$w(C) \leq \left(\sum_{i=1}^{k} w(C_i) \right) - w(C'_j). \tag{1}$$

- It must be that $\frac{\sum_{i=1}^{k} w(C_i)}{k} \leq w(C'_j)$, because the heaviest isolating cut is at least the average. This gives

$$w(C) \leq \left(\sum_{i=1}^{k} w(C_i) \right) - \frac{\sum_{i=1}^{k} w(C_i)}{k}$$

Proof of $2 - 2/k$ Approximation (cont.)

- And so

$$\begin{aligned} w(C) &\le \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} w(C_i) \\ &\le \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} w(A_i) \\ &= 2\left(1 - \frac{1}{k}\right) w(A) \\ &= \left(2 - \frac{2}{k}\right) OPT. \end{aligned}$$

A Tight Example for the $2 - 2/k$ Approximation

- Consider a graph G on $2k$ nodes that is a k-cycle (unit edge weights) with k terminals each attached via an edge (weight 2) to a distinct node in the k-cycle .

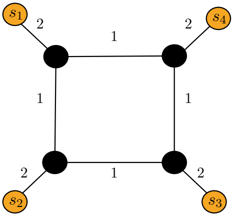


Figure: Tight example for $k = 4$.

A Tight Example for the $2 - 2/k$ Approximation

- Consider a graph G on $2k$ nodes that is a k-cycle (unit edge weights) with k terminals each attached via an edge (weight 2) to a distinct node in the k-cycle .

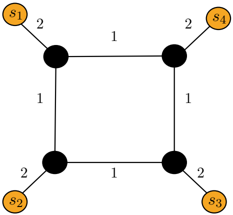


Figure: Tight example for $k = 4$.

- The min-weight isolating cut for each s_i is of weight 2, so $w(ALG) = 2(k-1) = 2k-2$. But the optimal multiway cut is the k-cycle, so $w(OPT) = k$.

A Randomized $3/2$-approximation Linear Programming Algorithm

The Unit Simplex

- Let Δ_k denote the unit $k-1$ dimensional simplex in $\mathbb{R}^k$

The Unit Simplex

- Let Δ_k denote the unit $k-1$ dimensional simplex in $\mathbb{R}^k$
  - i.e. $\Delta_k = \{\mathbf{x} \in \mathbb{R}^k \mid \mathbf{x} \geq 0$ and $\sum_i x^i = 1\}$, where x^i is the i^{th} coordinate of $\mathbf{x}$.

The Unit Simplex

- Let Δ_k denote the unit $k-1$ dimensional simplex in $\mathbb{R}^k$
  - i.e. $\Delta_k = \{\mathbf{x} \in \mathbb{R}^k \mid \mathbf{x} \geq 0 \text{ and } \sum_i x^i = 1\}$, where x^i is the i^{th} coordinate of $\mathbf{x}$.

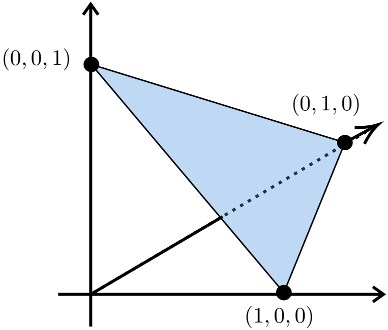


Figure: Δ_3, the 2-dimensional simplex in $\mathbb{R}^3$

Preliminaries

- Idea: every node of $G = (V, E)$ maps to a point in Δ_k.

Preliminaries

- Idea: every node of $G = (V, E)$ maps to a point in Δ_k.
- Each of the k terminals in $S = \{s_1, ..., s_k\}$ maps to a unit vector $e_i \in \mathbb{R}^k$.

Preliminaries

- Idea: every node of $G = (V, E)$ maps to a point in Δ_k.
- Each of the k terminals in $S = \{s_1, ..., s_k\}$ maps to a unit vector $e_i \in \mathbb{R}^k$.
- Let x_v be the point in Δ_k that node $v \in V$ maps to. (Note that $0 \leq x_v^i \leq 1$ for all i)

Preliminaries

- Idea: every node of $G = (V, E)$ maps to a point in Δ_k.
- Each of the k terminals in $S = \{s_1, ..., s_k\}$ maps to a unit vector $e_i \in \mathbb{R}^k$.
- Let x_v be the point in Δ_k that node $v \in V$ maps to. (Note that $0 \leq x_v^i \leq 1$ for all i)
- Let the length of an edge $(u, v) \in E$ be half the Manhattan distance between x_u and x_v, i.e. $d(u, v) = \frac{1}{2} \sum_{i=1}^{k} |x_u^i - x_v^i|$.

The LP Relaxation

- The relaxation is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{subject to} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, && \forall (u,v) \in E \\
& x_v \in \Delta_k, && \forall v \in V \\
& x_{s_i} = e_i, && \forall s_i \in S.
\end{aligned}
$$

The LP Relaxation

- The relaxation is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{subject to} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, && \forall (u,v) \in E \\
& x_v \in \Delta_k, && \forall v \in V \\
& x_{s_i} = e_i, && \forall s_i \in S.
\end{aligned}
$$

- But... is this even an LP? (Suspend your disbelief for a moment)

LP Relaxation Intuition

- **Intuition**: An integral solution to this LP maps every node in G to one of the k unit vectors

$$
\begin{aligned}
\text{min} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{s.t.} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \\
& \qquad \forall (u,v) \in E \\
& x_v \in \Delta_k, \forall v \in V \\
& x_{s_i} = e_i, \forall s_i \in S.
\end{aligned}
$$

LP Relaxation Intuition

- **Intuition**: An integral solution to this LP maps every node in G to one of the k unit vectors
  - This corresponds to an assignment of nodes in G to the k connected components (where each connected component has a single terminal s_i).

$$
\begin{aligned}
\text{min} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{s.t.} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \\
& \qquad \forall (u,v) \in E \\
& x_v \in \Delta_k, \forall v \in V \\
& x_{s_i} = e_i, \forall s_i \in S.
\end{aligned}
$$

LP Relaxation Intuition

- $d(u,v) = 0 \implies u, v$ map to the same unit vector

$$
\begin{aligned}
\text{min} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{s.t.} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \\
& \qquad\qquad\qquad\qquad \forall (u,v) \in E \\
& x_v \in \Delta_k, \forall v \in V \\
& x_{s_i} = e_i, \forall s_i \in S.
\end{aligned}
$$

LP Relaxation Intuition

- $d(u,v) = 0 \implies u, v$ map to the same unit vector
- $d(u,v) = 1 \implies u, v$ map to different unit vectors

$$\begin{aligned}
\text{min} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{s.t.} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \\
& \qquad\qquad \forall (u,v) \in E \\
& x_v \in \Delta_k, \forall v \in V \\
& x_{s_i} = e_i, \forall s_i \in S.
\end{aligned}$$

LP Relaxation Intuition

- $d(u,v) = 0 \implies u, v$ map to the same unit vector
- $d(u,v) = 1 \implies u, v$ map to different unit vectors
- So $d(u,v) = 1 \implies (u,v)$ is in the multiway cut

$$\begin{aligned}
\text{min} \quad & \sum_{(u,v)\in E} w(u,v)d(u,v) \\
\text{s.t.} \quad & d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \\
& \qquad\qquad\qquad\qquad \forall (u,v) \in E \\
& x_v \in \Delta_k, \forall v \in V \\
& x_{s_i} = e_i, \forall s_i \in S.
\end{aligned}$$

LP Relaxation Intuition

- $d(u,v) = 0 \implies u, v$ map to the same unit vector
- $d(u,v) = 1 \implies u, v$ map to different unit vectors
- So $d(u,v) = 1 \implies (u,v)$ is in the multiway cut
- Integral solution that minimizes the objective function thus corresponds to the minimum-weight multiway cut.

$$\text{min} \quad \sum_{(u,v)\in E} w(u,v)d(u,v)$$

$$\text{s.t.} \quad d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$$

$$x_v \in \Delta_k, \forall v \in V$$

$$x_{s_i} = e_i, \forall s_i \in S.$$

Integral Solution Determines a Multiway Cut

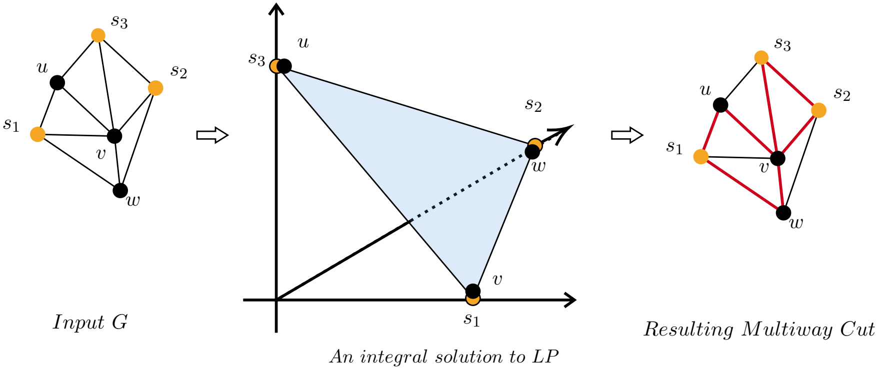


Figure: An integral solution to the LP naturally determines a multiway cut.

What About a Non-Integral Solution?

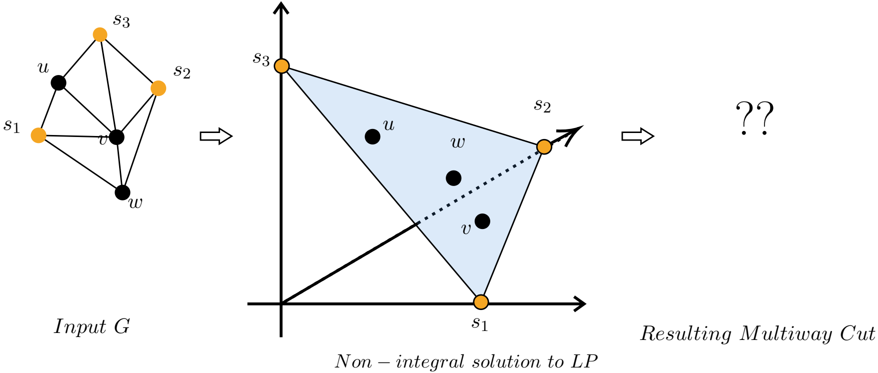


Figure: A non-integral solution to the LP. How do we round?

LP Relaxation is a valid LP

- The first constraint $d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$ is not linear

LP Relaxation is a valid LP

- The first constraint $d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$ is not linear
- We can make it linear by replacing it with the constraints

$$
\begin{aligned}
x_{uv}^i &\geq x_u^i - x_v^i, & 1 \leq i \leq k \quad (2)\\
x_{uv}^i &\geq x_v^i - x_u^i, & 1 \leq i \leq k \quad (3)\\
d(u,v) &= \frac{1}{2}\sum_{i=1}^{k} x_{uv}^i & \forall (u,v) \in E. \quad (4)
\end{aligned}
$$

LP Relaxation is a valid LP

- The first constraint $d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$ is not linear
- We can make it linear by replacing it with the constraints

$$x_{uv}^i \geq x_u^i - x_v^i, \qquad 1 \leq i \leq k \tag{2}$$

$$x_{uv}^i \geq x_v^i - x_u^i, \qquad 1 \leq i \leq k \tag{3}$$

$$d(u,v) = \frac{1}{2}\sum_{i=1}^{k} x_{uv}^i \qquad \forall (u,v) \in E. \tag{4}$$

- This means a feasible solution will have $x_{uv}^i \geq |x_u^i - x_v^i|$.

LP Relaxation is a valid LP

- The first constraint $d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$ is not linear
- We can make it linear by replacing it with the constraints

$$x_{uv}^i \geq x_u^i - x_v^i, \qquad 1 \leq i \leq k \tag{2}$$

$$x_{uv}^i \geq x_v^i - x_u^i, \qquad 1 \leq i \leq k \tag{3}$$

$$d(u,v) = \frac{1}{2}\sum_{i=1}^{k} x_{uv}^i \qquad \forall (u,v) \in E. \tag{4}$$

- This means a feasible solution will have $x_{uv}^i \geq |x_u^i - x_v^i|$.
- But an optimal solution will obey $x_{uv}^i = |x_u^i - x_v^i|$ because we are minimizing the objective.

LP Relaxation is a valid LP

- The first constraint $d(u,v) = \frac{1}{2}\sum_{i=1}^{k} |x_u^i - x_v^i|, \quad \forall (u,v) \in E$ is not linear
- We can make it linear by replacing it with the constraints

$$x_{uv}^i \geq x_u^i - x_v^i, \qquad 1 \leq i \leq k \tag{2}$$
$$x_{uv}^i \geq x_v^i - x_u^i, \qquad 1 \leq i \leq k \tag{3}$$
$$d(u,v) = \frac{1}{2}\sum_{i=1}^{k} x_{uv}^i \qquad \forall (u,v) \in E. \tag{4}$$

- This means a feasible solution will have $x_{uv}^i \geq |x_u^i - x_v^i|$.
- But an optimal solution will obey $x_{uv}^i = |x_u^i - x_v^i|$ because we are minimizing the objective.
  - To see this: If disobeys, then x_{uv}^i can be made smaller, so solution can be made smaller, contradicting its optimality

A Helpful Lemma

Lemma (Two-coordinate Lemma)

For an optimal solution to the LP relaxation, we can assume (w.l.o.g) that for each edge $(u, v) \in E$, x_u and x_v differ in at most two coordinates.

- Note that x_u and x_v can never differ in exactly one coordinate, because both vectors must sum to 1

Proof of Two-coordinate Lemma

- Consider an edge (u, v) where x_u and x_v differ in > 2 coordinates

Proof of Two-coordinate Lemma

- Consider an edge (u, v) where x_u and x_v differ in > 2 coordinates
- Make a new node y and replace (u, v) with (u, y) and (y, v), setting both their edge weights to be $w(u, v)$

Proof of Two-coordinate Lemma

- Consider an edge (u, v) where x_u and x_v differ in > 2 coordinates
- Make a new node y and replace (u, v) with (u, y) and (y, v), setting both their edge weights to be $w(u, v)$
  - This means the cost of the optimal integral solution is unchanged because either $d(u, y) = 0$ or $d(y, v) = 0$.

Proof of Two-coordinate Lemma

- Consider an edge (u, v) where x_u and x_v differ in > 2 coordinates
- Make a new node y and replace (u, v) with (u, y) and (y, v), setting both their edge weights to be $w(u, v)$
  - This means the cost of the optimal integral solution is unchanged because either $d(u, y) = 0$ or $d(y, v) = 0$.
- Consider the optimal fractional solution

Proof of Two-coordinate Lemma

- Consider an edge (u, v) where x_u and x_v differ in > 2 coordinates
- Make a new node y and replace (u, v) with (u, y) and (y, v), setting both their edge weights to be $w(u, v)$
  - This means the cost of the optimal integral solution is unchanged because either $d(u, y) = 0$ or $d(y, v) = 0$.
- Consider the optimal fractional solution
  - Note that $d(u, y) + d(y, v) \geq d(u, v)$ since d is a valid distance function, so adding y does not improve the optimal solution

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.
  - This means $x_y \in \Delta_k$

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.
  - This means $x_y \in \Delta_k$
  - And $d(u, y) + d(y, v) = d(u, v)$, so cost of optimal solution does not change

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.
  - This means $x_y \in \Delta_k$
  - And $d(u, y) + d(y, v) = d(u, v)$, so cost of optimal solution does not change
- x_y and x_v differ in exactly two coordinates

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.
  - This means $x_y \in \Delta_k$
  - And $d(u, y) + d(y, v) = d(u, v)$, so cost of optimal solution does not change
- x_y and x_v differ in exactly two coordinates
- x_y and x_u differ in fewer coordinates than x_u and x_v

Proof of Two-coordinate Lemma

- Out of all coordinates where x_u, x_v differ, let i be the coordinate in which the difference is minimal
  - w.l.o.g let $x_u^i < x_v^i$ and let $\alpha = x_v^i - x_u^i$
- Then there must be a coordinate j s.t. $x_u^j - x_v^j \geq \alpha$ (w.l.o.g $x_u^j > x_v^j$)
- Now define x_y to be the point where $x_y^i = x_u^i$, $x_y^j = x_v^j + \alpha$, and remaining coordinates of x_y identical to those of x_v.
  - This means $x_y \in \Delta_k$
  - And $d(u, y) + d(y, v) = d(u, v)$, so cost of optimal solution does not change
- x_y and x_v differ in exactly two coordinates
- x_y and x_u differ in fewer coordinates than x_u and x_v
  - Iteratively doing this process will eventually guarantee the two-coordinate property while maintaining the cost of the optimal solution

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.
  - i.e. $E_i = \{(u, v) \in E \mid x_u^i \neq x_v^i\}$

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.
  - i.e. $E_i = \{(u, v) \in E \mid x_u^i \neq x_v^i\}$
  - Every $(u, v) \in E$ with $d(u, v) > 0$ will lie in two different such sets

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.
  - i.e. $E_i = \{(u, v) \in E \mid x_u^i \neq x_v^i\}$
  - Every $(u, v) \in E$ with $d(u, v) > 0$ will lie in two different such sets
- Let $W_i = \sum_{e \in E_i} w(e)d(e)$ and w.l.o.g let W_k be the largest of them

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.
  - i.e. $E_i = \{(u, v) \in E \mid x_u^i \neq x_v^i\}$
  - Every $(u, v) \in E$ with $d(u, v) > 0$ will lie in two different such sets
- Let $W_i = \sum\limits_{e \in E_i} w(e)d(e)$ and w.l.o.g let W_k be the largest of them
  - i.e. W_i is the cost contributed by E_i to the optimal solution

Randomized Rounding Algorithm Preliminaries

- Let $\{x_u \mid u \in V\}$ be the optimal solution to the LP relaxation (with cost OPT) such that x_u and x_v differ in at most two coordinates for all $(u, v) \in E$
- Define E_i to be the set of all edges whose endpoints differ at the i^{th} coordinate.
  - i.e. $E_i = \{(u, v) \in E \mid x_u^i \neq x_v^i\}$
  - Every $(u, v) \in E$ with $d(u, v) > 0$ will lie in two different such sets
- Let $W_i = \sum_{e \in E_i} w(e)d(e)$ and w.l.o.g let W_k be the largest of them
  - i.e. W_i is the cost contributed by E_i to the optimal solution
- Define $B(s_i, \rho) = \{v \in V \mid x_v^i \geq \rho\}$ for $\rho \in (0, 1)$

Randomized Rounding Algorithm

- Algorithm 2:

Randomized Rounding Algorithm

- Algorithm 2:
    1. Compute an optimal solution to the LP relaxation

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random
  5. For $i = 1$ to $k - 1$:

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random
  5. For $i = 1$ to $k - 1$:
     - Set $V_{\sigma(i)} = B(s_i, \rho) \setminus \bigcup_{j<i} V_{\sigma(j)}$

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random
  5. For $i = 1$ to $k-1$:
     - Set $V_{\sigma(i)} = B(s_i, \rho) \setminus \bigcup_{j<i} V_{\sigma(j)}$
     - #Put all remaining nodes who's i^{th} coordinate is at least ρ into set $V_{\sigma(i)}$

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random
  5. For $i = 1$ to $k - 1$:
     - Set $V_{\sigma(i)} = B(s_i, \rho) \setminus \bigcup_{j<i} V_{\sigma(j)}$
     - #Put all remaining nodes who's i^{th} coordinate is at least ρ into set $V_{\sigma(i)}$
  6. Set $V_k =$ all nodes that remain

Randomized Rounding Algorithm

- Algorithm 2:
  1. Compute an optimal solution to the LP relaxation
  2. Renumber terminals so that W_k is greater than W_i for all $1 \leq i < k$.
  3. Choose $\rho \in (0, 1)$ uniformly at random
  4. Choose $\sigma \in \{(1, 2, ..., k-1, k), (k-1, k-2, ..., 2, 1, k)\}$ uniformly at random
  5. For $i = 1$ to $k - 1$:
     - Set $V_{\sigma(i)} = B(s_i, \rho) \setminus \bigcup_{j<i} V_{\sigma(j)}$
     - #Put all remaining nodes who's i^{th} coordinate is at least ρ into set $V_{\sigma(i)}$
  6. Set V_k = all nodes that remain
  7. Output C, the set of edges that run between sets in the partition $V_1, ..., V_k$

Example

$\sigma \ = \ (1, 2, 3), \ \rho \ < 1/2$

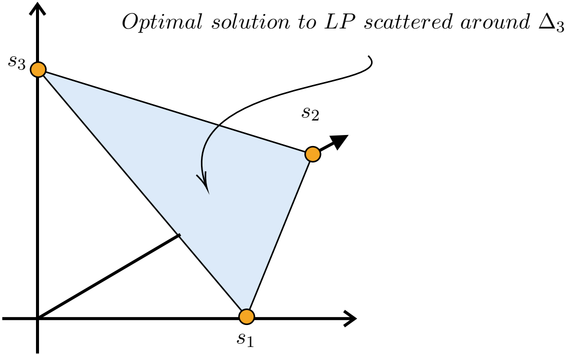


Figure: Initialize the randomized rounding procedure: compute LP solution, relabel the terminals s_i s.t. W_3 is largest, and choose σ and ρ

Example

$$V_1 = \{v \in V \mid x_v^1 \geqslant \rho\}$$

$$\sigma = (1,2,3)$$

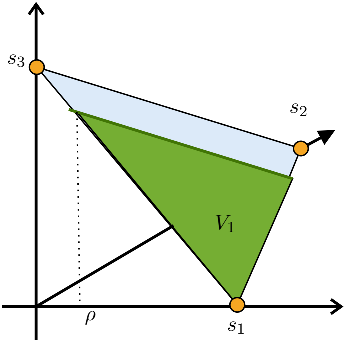


Figure: Compute $V_{\sigma(1)} = V_1$

Example

$\sigma \;=\; (1,2,3)$

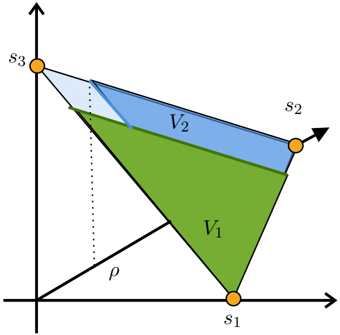


Figure: Compute $V_{\sigma(2)} = V_2$

$$V_1 \;= \{v \;\in V \;|x_v^1 \geqslant \rho\}$$

$$V_2 \;= \{v \;\in V \;|x_v^2 \geqslant \rho\} \setminus V_1$$

Example

$\sigma \;=\; (1, 2, 3)$

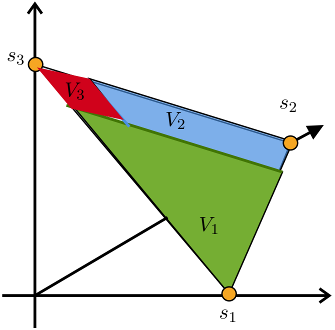


Figure: Compute V_3

$$
\begin{aligned}
V_1 &= \{v \in V \mid x_v^1 \geqslant \rho\} \\
V_2 &= \{v \in V \mid x_v^2 \geqslant \rho\} \setminus V_1 \\
V_3 &= V \setminus V_1 \cup V_2
\end{aligned}
$$

Example

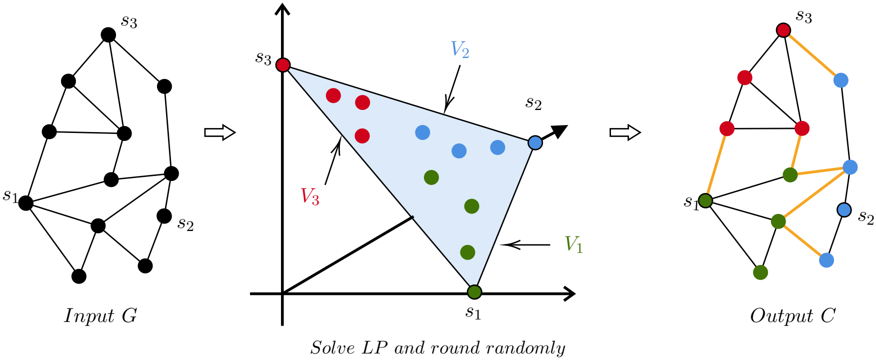


Figure: Voila! Output multiway cut C comprises of the orange edges.

Approximation Guarantee

- Recall E_k is the set of all edges whose endpoints differ in the k^{th} coordinate.
- First, we prove two Lemmas:

Lemma (A)

$e \in E_k \implies Pr[e \in C] \leq d(e)$

Lemma (B)

$e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates
  - This must mean $|x_u^i - x_v^i| = |x_u^k - x_v^k|$ so coordinates of each point sum to 1

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates
  - This must mean $|x_u^i - x_v^i| = |x_u^k - x_v^k|$ so coordinates of each point sum to 1
- In Algorithm 2, V_k contains all leftover points without considering their coordinates

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates
  - This must mean $|x_u^i - x_v^i| = |x_u^k - x_v^k|$ so coordinates of each point sum to 1
- In Algorithm 2, V_k contains all leftover points without considering their coordinates
- Thus, only way for $(u, v) \in C$ is if exactly one of u or v is in V_i.

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates
  - This must mean $|x_u^i - x_v^i| = |x_u^k - x_v^k|$ so coordinates of each point sum to 1
- In Algorithm 2, V_k contains all leftover points without considering their coordinates
- Thus, only way for $(u, v) \in C$ is if exactly one of u or v is in V_i.
- This occurs when ρ is somewhere between x_u^i and x_v^i, which occurs with probability $|x_u^i - x_v^i|$.

Proof of $e \in E_k \implies Pr[e \in C] \leq d(e)$

- Since $(u, v) \in E_k$, x_u and x_v differ in coordinate k and some other coordinate i, and are identical in all other coordinates
  - This must mean $|x_u^i - x_v^i| = |x_u^k - x_v^k|$ so coordinates of each point sum to 1
- In Algorithm 2, V_k contains all leftover points without considering their coordinates
- Thus, only way for $(u, v) \in C$ is if exactly one of u or v is in V_i.
- This occurs when ρ is somewhere between x_u^i and x_v^i, which occurs with probability $|x_u^i - x_v^i|$.
- This probability is equal to $|x_u^i - x_v^i| = \frac{|x_u^i - x_v^i| + |x_u^k - x_v^k|}{2} = d(u, v)$.

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
  - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
    - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
    - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$
- Given all this, in what cases is it true that $(u, v) \in C$?

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
  - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$
- Given all this, in what cases is it true that $(u, v) \in C$?
  - Case I: $\rho \in \beta$ and $\sigma = (....j, ..., i, ...k)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
  - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$
- Given all this, in what cases is it true that $(u, v) \in C$?
  - Case I: $\rho \in \beta$ and $\sigma = (....j, ..., i, ...k)$
  - Case II: $\rho \in \beta$ and $\sigma = (....i, ..., j, ...k)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

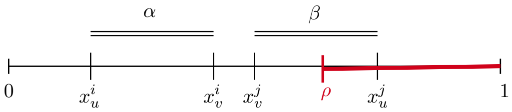
- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
  - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$
- Given all this, in what cases is it true that $(u, v) \in C$?
  - Case I: $\rho \in \beta$ and $\sigma = (....j, ..., i, ...k)$
  - Case II: $\rho \in \beta$ and $\sigma = (....i, ..., j, ...k)$
  - Case III: $\rho \in \alpha$ and $\sigma = (....j, ..., i, ...k)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- If $(u, v) \in E \setminus E_k$, then x_u and x_v differ in two coordinates i, j s.t. $i \neq k, j \neq k$
- $(u, v) \in C$ if the algorithm puts u and v in different sets
  - Only need to consider V_i, V_j, V_k since all other coordinates are identical, and V_k is the "overflow" bin
- Let intervals $\alpha = [x_u^i, x_v^i]$ and $\beta = [x_v^j, x_u^j]$
- Given all this, in what cases is it true that $(u, v) \in C$?
  - Case I: $\rho \in \beta$ and $\sigma = (....j, ..., i, ...k)$
  - Case II: $\rho \in \beta$ and $\sigma = (....i, ..., j, ...k)$
  - Case III: $\rho \in \alpha$ and $\sigma = (....j, ..., i, ...k)$
  - Case IV: $\rho \in \alpha$ and $\sigma = (....i, ..., j, ...k)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- Case I: $\rho \in \beta$ and $\sigma = (....j, ..., i, ...k)$



$\sigma(j) < \sigma(i)$

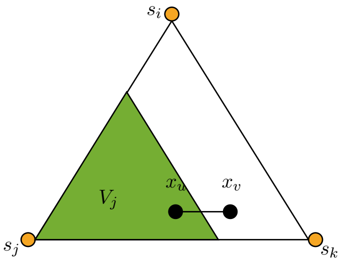
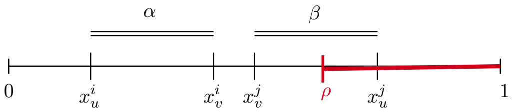


Figure: $(u, v) \in C$ is TRUE

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- Case II: $\rho \in \beta$ and $\sigma = (....i, ..., j, ...k)$



$\sigma(i) \; < \; \sigma(j)$

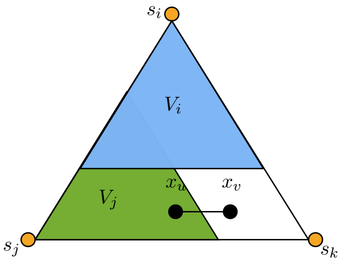
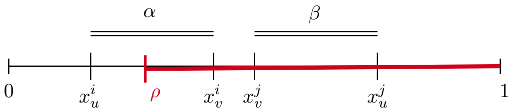


Figure: $(u, v) \in C$ is TRUE

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- Case III: $\rho \in \alpha$ and $\sigma = (....j, ..., i, ...k)$



$\sigma(j) < \sigma(i)$

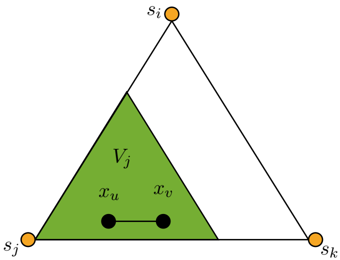
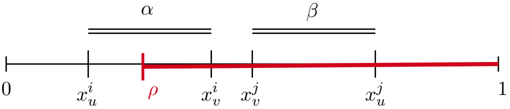


Figure: $(u, v) \in C$ is FALSE

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- Case IV: $\rho \in \alpha$ and $\sigma = (....i, ..., j, ...k)$



$\sigma(i) \; < \; \sigma(j)$

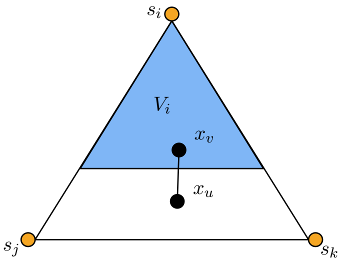


Figure: $(u, v) \in C$ is TRUE

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1
  - So $d(u, v) = |x_u^i - x_v^i| = |x_u^j - x_v^j| = |\alpha| = |\beta|$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1
  - So $d(u, v) = |x_u^i - x_v^i| = |x_u^j - x_v^j| = |\alpha| = |\beta|$
- Therefore $Pr[(u, v) \in C] = \frac{3}{2}d(u, v)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or $(\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k))$.
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1
  - So $d(u, v) = |x_u^i - x_v^i| = |x_u^j - x_v^j| = |\alpha| = |\beta|$
- Therefore $Pr[(u, v) \in C] = \frac{3}{2}d(u, v)$
  - if α does not overlap with β

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1
  - So $d(u, v) = |x_u^i - x_v^i| = |x_u^j - x_v^j| = |\alpha| = |\beta|$
- Therefore $Pr[(u, v) \in C] = \frac{3}{2}d(u, v)$
  - if α does not overlap with β
- $Pr[(u, v) \in C] \leq \frac{3}{2}d(u, v)$

Proof of $e \in E \setminus E_k \implies Pr[e \in C] \leq 1.5d(e)$

- By our case analysis, we have shown that $(u, v) \in C$ when $\rho \in \beta$ or ($\rho \in \alpha$ and $\sigma = (...i, ..., j, ...k)$).
  - I.e. in Case I, Case II, and Case IV
- $Pr[(u, v) \in C] = Pr[\rho \in \beta] + Pr[\rho \in \alpha \wedge \sigma = (...i, ..., j, ...k)]$
- $Pr[(u, v) \in C] = |\beta| + |\alpha| \cdot \frac{1}{2}$
- Now note $|\alpha| = |\beta| = d(u, v)$. Why?
  - x_u and x_v only differ in coordinates i and j
  - So $d(u, v) = \frac{1}{2}|x_u^i - x_v^i| + \frac{1}{2}|x_u^j - x_v^j|$
  - But $|x_u^j - x_v^j| = |x_u^i - x_v^i|$ because coordinates of x_v, x_u must sum to 1
  - So $d(u, v) = |x_u^i - x_v^i| = |x_u^j - x_v^j| = |\alpha| = |\beta|$
- Therefore $Pr[(u, v) \in C] = \frac{3}{2}d(u, v)$
  - if α does not overlap with β
- $Pr[(u, v) \in C] \leq \frac{3}{2}d(u, v)$
  - if α does overlap with β

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

- Let OPT be cost of optimal solution to the LP relaxation, so $OPT = \sum_{e \in E} w(e)d(e)$.

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

- Let OPT be cost of optimal solution to the LP relaxation, so $OPT = \sum_{e \in E} w(e)d(e)$.
- Recall $W_i = \sum_{e \in E_i} w(e)d(e)$.

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

- Let OPT be cost of optimal solution to the LP relaxation, so $OPT = \sum_{e \in E} w(e)d(e)$.
- Recall $W_i = \sum_{e \in E_i} w(e)d(e)$.
- Note that $\sum_{i=1}^{k} W_i = 2OPT$ since every edge e with $d(e) > 0$ belongs in two of the E_i

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

- Let OPT be cost of optimal solution to the LP relaxation, so $OPT = \sum_{e \in E} w(e)d(e)$.
- Recall $W_i = \sum_{e \in E_i} w(e)d(e)$.
- Note that $\sum_{i=1}^{k} W_i = 2OPT$ since every edge e with $d(e) > 0$ belongs in two of the E_i
- Then $W_k \geq \frac{2}{k}OPT$ since we made W_k the maximum such W_i, and the maximum is at least the average

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

- Let OPT be cost of optimal solution to the LP relaxation, so $OPT = \sum_{e \in E} w(e)d(e)$.
- Recall $W_i = \sum_{e \in E_i} w(e)d(e)$.
- Note that $\sum_{i=1}^{k} W_i = 2OPT$ since every edge e with $d(e) > 0$ belongs in two of the E_i
- Then $W_k \geq \frac{2}{k}OPT$ since we made W_k the maximum such W_i, and the maximum is at least the average
- ...

Proof of (randomized) $3/2 - \frac{1}{k}$ Approximation

$$
\begin{aligned}
\mathbb{E}[w(C)] &= \sum_{e \in E} w(e) Pr[e \in C] \\
&= \sum_{e \in E_k} w(e) Pr[e \in C] + \sum_{e \in E \setminus E_k} w(e) Pr[e \in C] \\
&\leq \sum_{e \in E_k} w(e) d(e) + 1.5 \sum_{e \in E \setminus E_k} w(e) d(e) \qquad \text{Lemmas A and B} \\
&= 1.5 \sum_{e \in E} w(e) d(e) - 0.5 \sum_{e \in E_k} w(e) d(e) \\
&\leq 1.5 OPT - 0.5 \left(\frac{2}{k} OPT \right) \\
&= \left(\frac{3}{2} - \frac{1}{k} \right) OPT
\end{aligned}
$$

References

1. V. Vazirani. Approximation Algorithms. Berlin: Springer, 2003.
2. A. Groce. Linear Programming Algorithm for the Multiway Cut Problem. MIT, 2006.
3. S. Lowen. Multiway Cut. Rutgers, 2011.