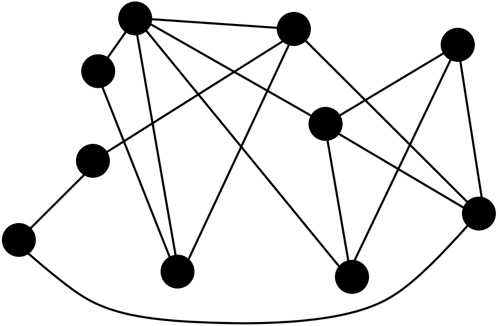
Approximation Algorithms for Multicommodity-Type Problems with Guarantees Independent of the Graph Size

Ankur Moitra, MIT

January 25, 2011

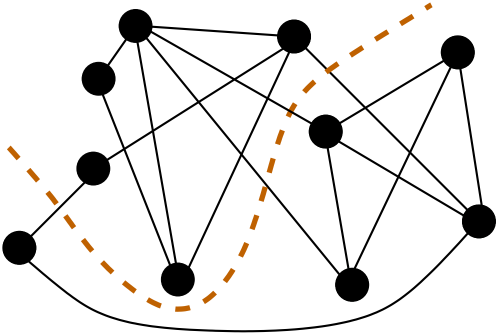
The Minimum Bisection Problem

Goal: Minimize cost of bisection



The Minimum Bisection Problem

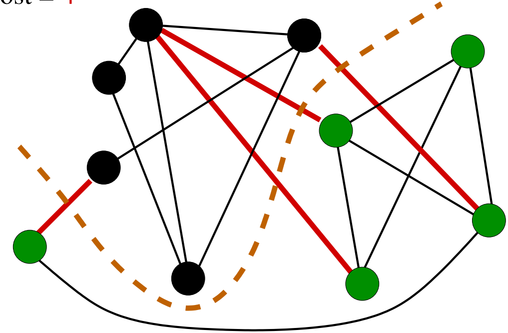
Goal: Minimize cost of bisection



The Minimum Bisection Problem

Goal: Minimize cost of bisection

cost = 4



History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995] $\frac{n}{2}$-approximation algorithm

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995] $\frac{n}{2}$-approximation algorithm
6. [Arora, Karger, Karpinski 1999] PTAS for dense graphs

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995] $\frac{n}{2}$-approximation algorithm
6. [Arora, Karger, Karpinski 1999] PTAS for dense graphs
7. [Feige, Krauthgamer 2001] $O(\log^{1.5} n)$-approximation algorithm

History of the Minimum Bisection Problem

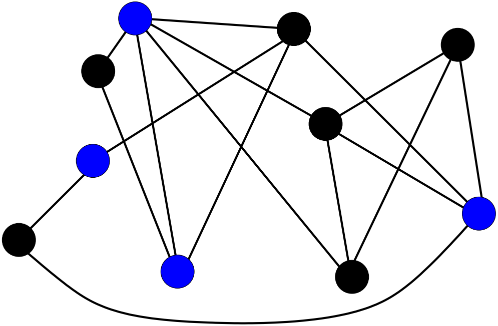
1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995] $\frac{n}{2}$-approximation algorithm
6. [Arora, Karger, Karpinski 1999] PTAS for dense graphs
7. [Feige, Krauthgamer 2001] $O(\log^{1.5} n)$-approximation algorithm
8. [Khot 2004] No PTAS, unless $P = NP$

History of the Minimum Bisection Problem

1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995] $\frac{n}{2}$-approximation algorithm
6. [Arora, Karger, Karpinski 1999] PTAS for dense graphs
7. [Feige, Krauthgamer 2001] $O(\log^{1.5} n)$-approximation algorithm
8. [Khot 2004] No PTAS, unless $P = NP$
9. [Räcke, 2008] $O(\log n)$-approximation algorithm

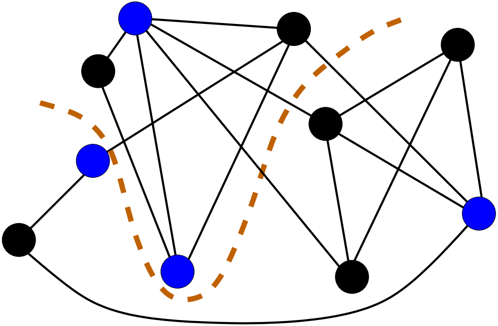
The Steiner Minimum Bisection Problem

Goal: Minimize cost of a bisection of the k blue nodes



The Steiner Minimum Bisection Problem

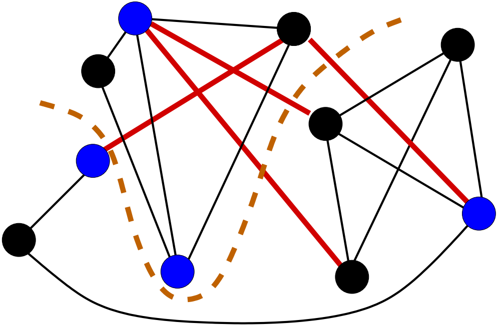
Goal: Minimize cost of a bisection of the k blue nodes



The Steiner Minimum Bisection Problem

Goal: Minimize cost of a bisection of the k blue nodes

cost = 4



Question

Can we find a $\mathrm{poly}(\log k)$*-approximation algorithm?*

Question

Can we find a poly$(\log k)$-approximation algorithm?

Some approximation guarantees can be made independent of the graph size:

Question

Can we find a poly$(\log k)$-approximation algorithm?

Some approximation guarantees can be made independent of the graph size:

1. $O(\log k)$ **generalized sparsest cut** for k commodities [Linial, London, Rabinovich 1995] and [Aumann, Rabani 1997]

Question

Can we find a $\mathit{poly}(\log k)$*-approximation algorithm?*

Some approximation guarantees can be made independent of the graph size:

1. $O(\log k)$ **generalized sparsest cut** for k commodities
[Linial, London, Rabinovich 1995] and [Aumann, Rabani 1997]
2. $O(\log k)$ **multicut** for k terminals
[Garg, Vazirani, Yannakakis 1996]

Question

Can we find a poly$(\log k)$*-approximation algorithm?*

Some approximation guarantees can be made independent of the graph size:

1. $O(\log k)$ **generalized sparsest cut** for k commodities
   [Linial, London, Rabinovich 1995] and [Aumann, Rabani 1997]
2. $O(\log k)$ **multicut** for k terminals
   [Garg, Vazirani, Yannakakis 1996]
3. $O(\frac{\log k}{\log \log k})$ 0-**extension** for k terminals
   [Fakcharoenphol, Harrelson, Rao, Talwar 2003]

A Meta Question

Given

A $\text{poly}(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for an optimization problem characterized by cuts or flows*

A Meta Question

Given

A $poly(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for an optimization problem characterized by cuts or flows*

Let k be the number of "interesting" nodes

A Meta Question

Given

A $poly(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for an optimization problem characterized by cuts or flows*

Let k be the number of "interesting" nodes

Meta Question

Can we give a $poly(\log k)$ *approximation algorithm (integrality gap or competitive ratio)?*

A Meta Question

Given

A poly$(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for an optimization problem characterized by cuts or flows*

Let k be the number of "interesting" nodes

Meta Question

Can we give a poly$(\log k)$ *approximation algorithm (integrality gap or competitive ratio)?*

Yes we can...

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first *poly*$(\log k)$ approximation algorithms (or competitive ratios) for:

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first *poly*$(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection,

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut,

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut,

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension,

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first *poly*(log k) approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing,

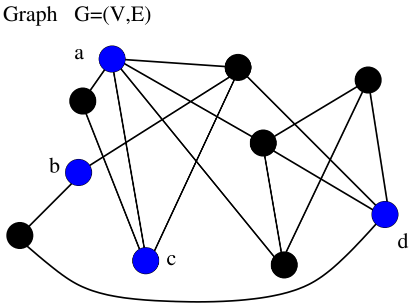
Highlights of Vertex Sparsification

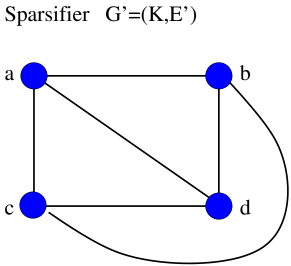
1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement,

Highlights of Vertex Sparsification

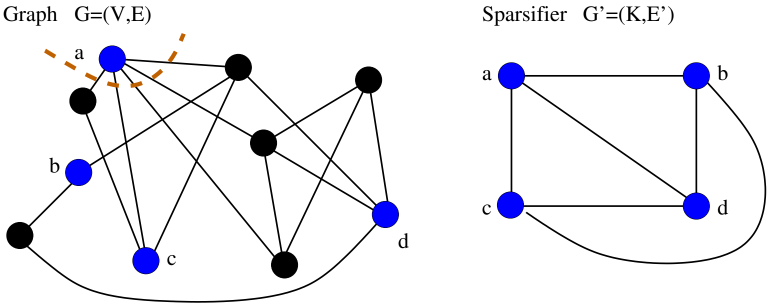
1. **Approximation Guarantees Independent of the Graph Size:** We give the first *poly*($\log k$) approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement, and minimum linear arrangement

General Approach: Cut Sparsifiers

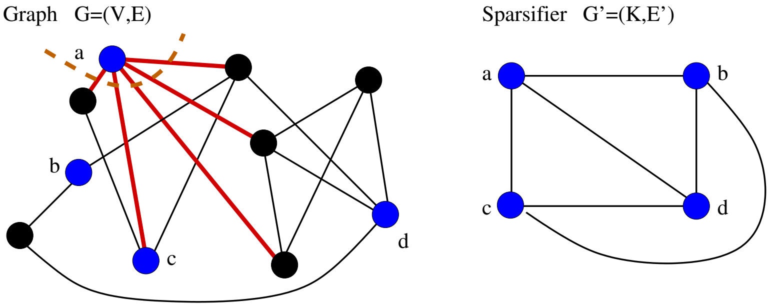




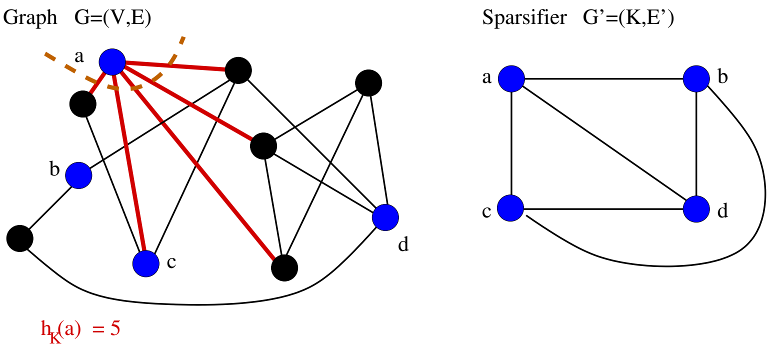
General Approach: Cut Sparsifiers



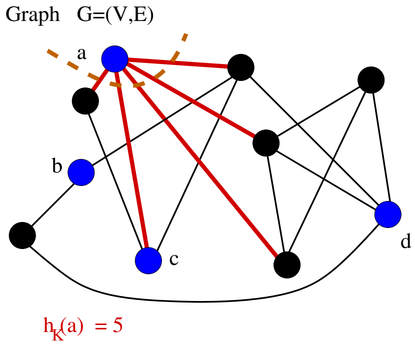
General Approach: Cut Sparsifiers

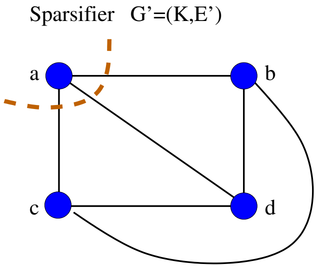


General Approach: Cut Sparsifiers

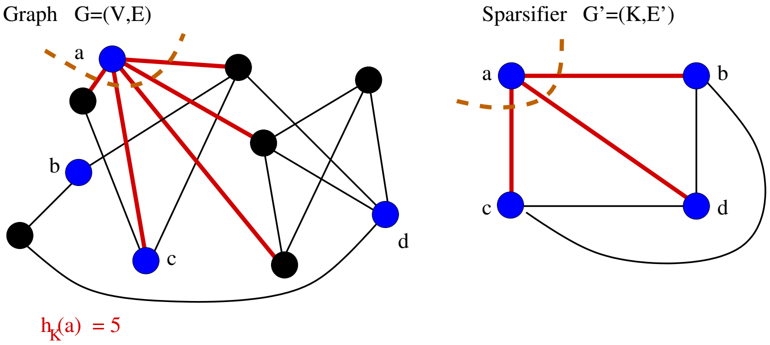


General Approach: Cut Sparsifiers

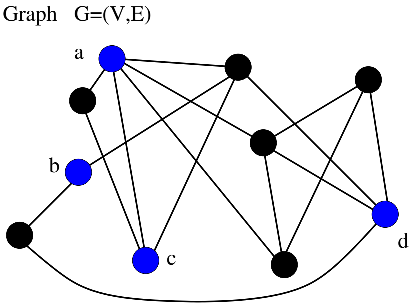


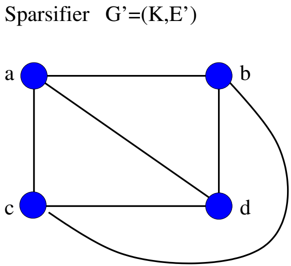


General Approach: Cut Sparsifiers

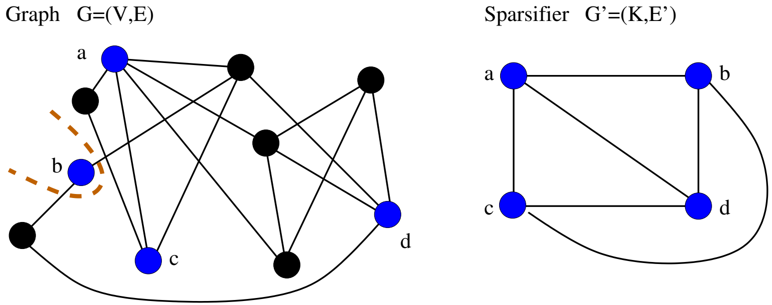


General Approach: Cut Sparsifiers

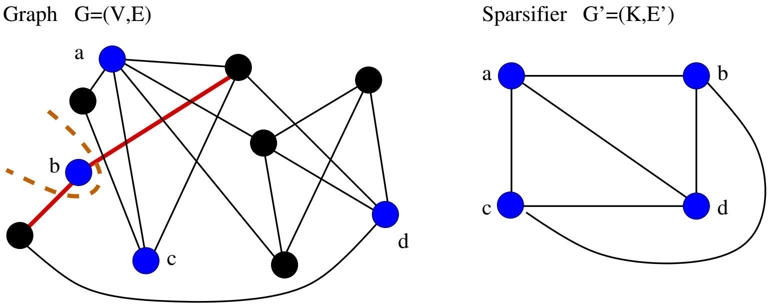




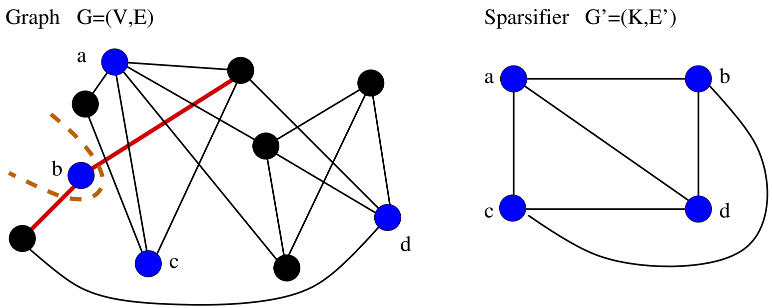
General Approach: Cut Sparsifiers



General Approach: Cut Sparsifiers

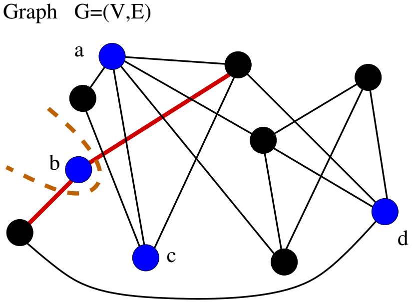


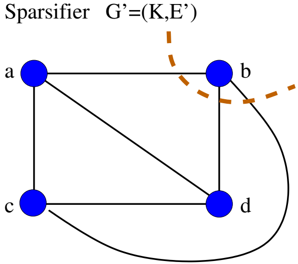
General Approach: Cut Sparsifiers



$h_K(b) = 2$

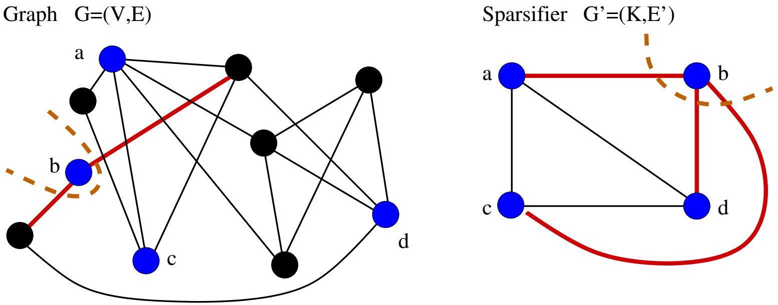
General Approach: Cut Sparsifiers





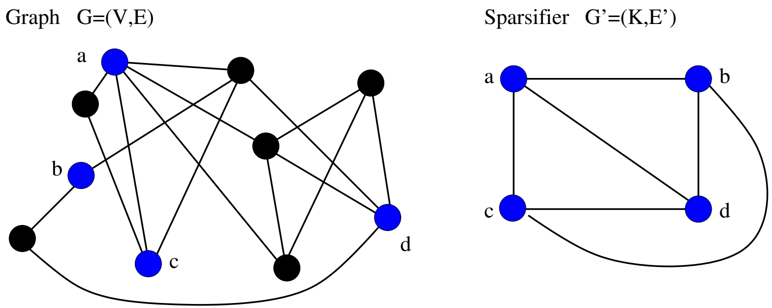
$h_K(b) = 2$

General Approach: Cut Sparsifiers

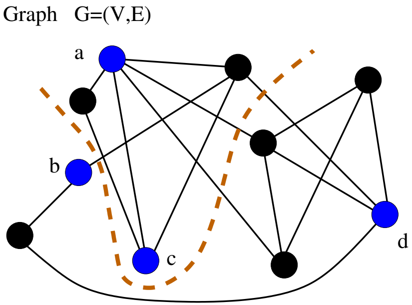


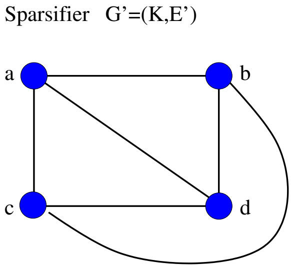
$h_K(b) = 2$

General Approach: Cut Sparsifiers

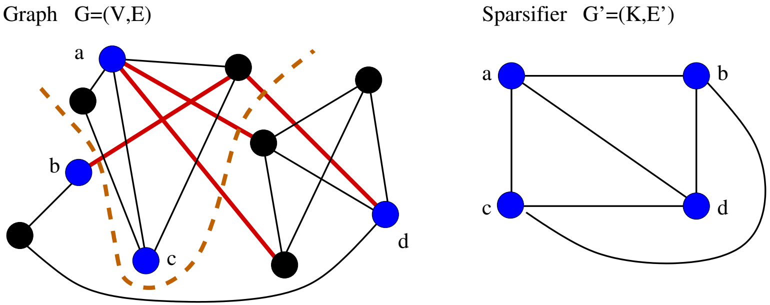


General Approach: Cut Sparsifiers

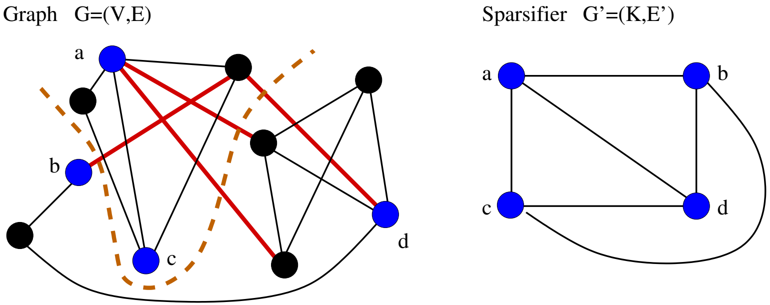




General Approach: Cut Sparsifiers

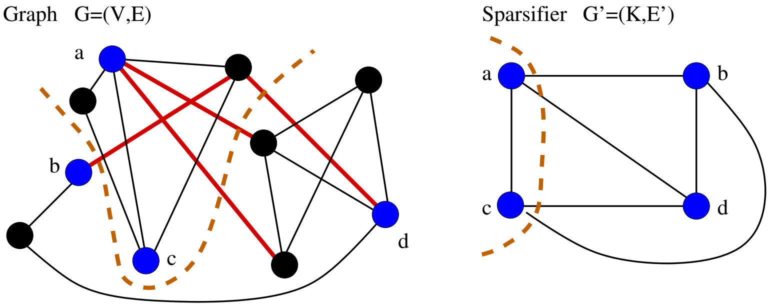


General Approach: Cut Sparsifiers



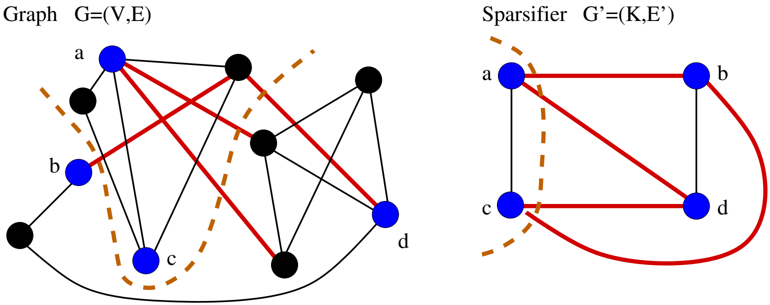
$h_K(ac) = 4$

General Approach: Cut Sparsifiers



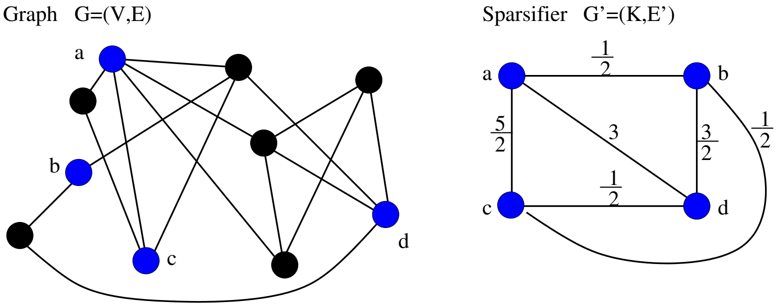
$h_K(ac) = 4$

General Approach: Cut Sparsifiers

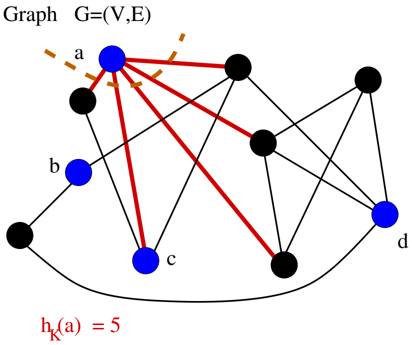


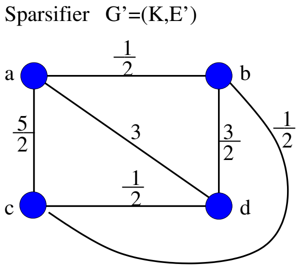
$h_K(ac) = 4$

General Approach: Cut Sparsifiers

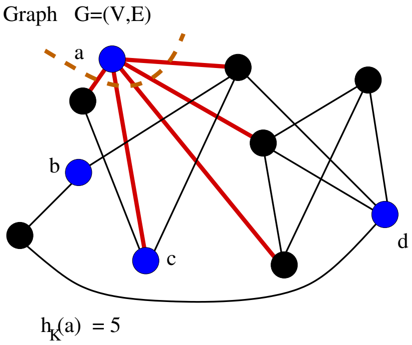


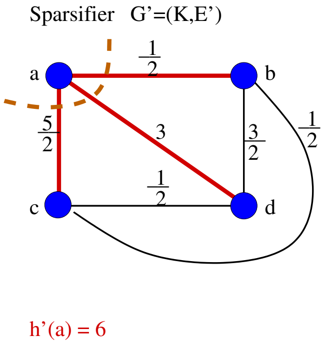
General Approach: Cut Sparsifiers





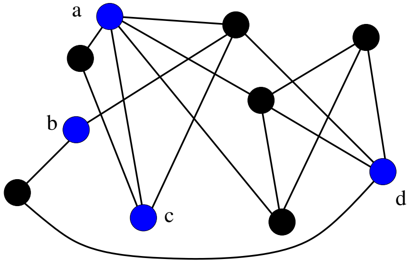
General Approach: Cut Sparsifiers





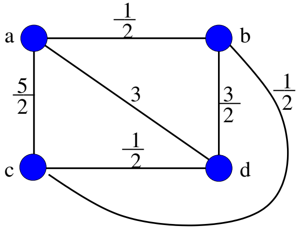
General Approach: Cut Sparsifiers

Graph G=(V,E)



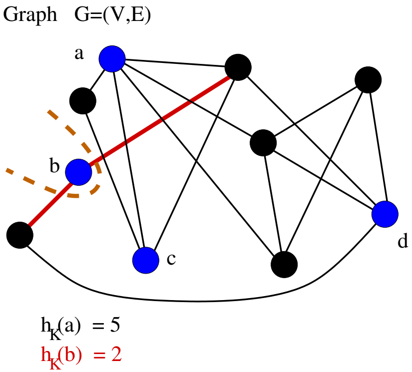
$h_K(a) = 5$

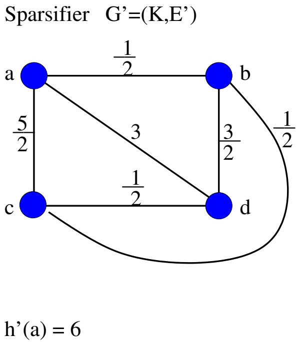
Sparsifier G'=(K,E')



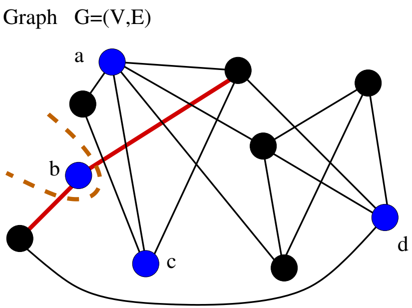
$h'(a) = 6$

General Approach: Cut Sparsifiers

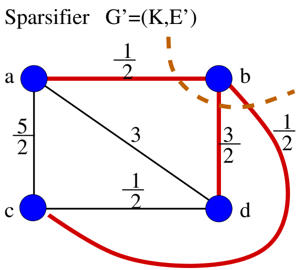




General Approach: Cut Sparsifiers



$h_K(a) = 5$
$h_K(b) = 2$

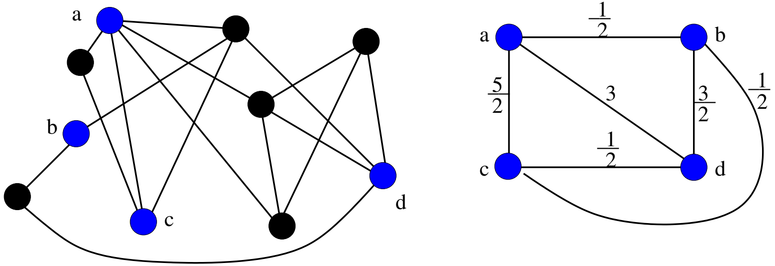


h'(a) = 6
h'(b) = 2.5

General Approach: Cut Sparsifiers

Graph G=(V,E)

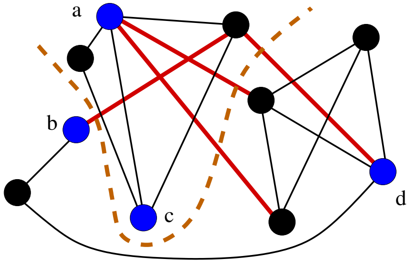
Sparsifier G'=(K,E')



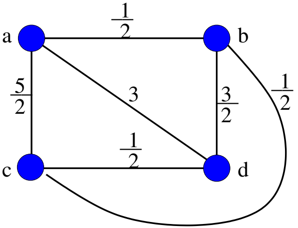
$h_K(a) = 5$
$h_K(b) = 2$

$h'(a) = 6$
$h'(b) = 2.5$

General Approach: Cut Sparsifiers



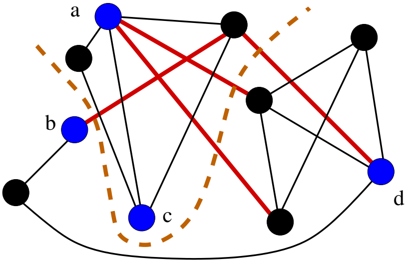
$h_K(a) = 5$
$h_K(b) = 2$
$h_K(ac) = 4$



$h'(a) = 6$
$h'(b) = 2.5$

General Approach: Cut Sparsifiers

Graph G=(V,E)

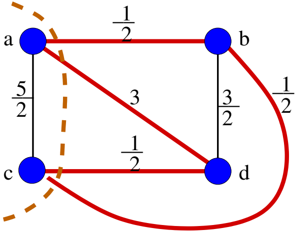


$h_K(a) = 5$

$h_K(b) = 2$

$h_K(ac) = 4$

Sparsifier G'=(K,E')



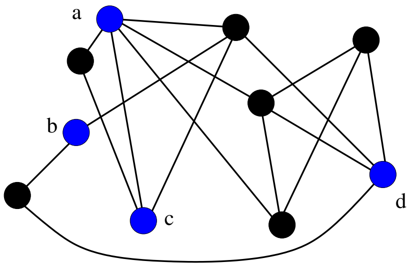
$h'(a) = 6$

$h'(b) = 2.5$

$h'(ac) = 4.5$

General Approach: Cut Sparsifiers

Graph G=(V,E)

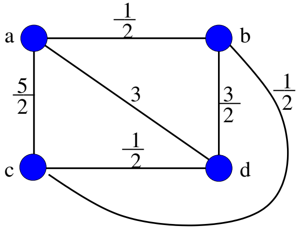


$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$

$h_K(b) = 2$ $h_K(ab) = 7$

$h_K(c) = 3$ $h_K(ac) = 4$

Sparsifier G'=(K,E')



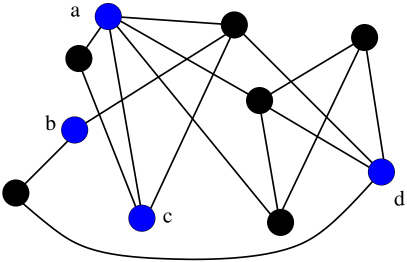
$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$

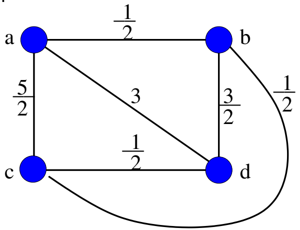
$h'(b) = 2.5$ $h'(ab) = 7.5$

$h'(c) = 3.5$ $h'(ac) = 4.5$

General Approach: Cut Sparsifiers

$$\text{Quality} = \frac{5}{4}$$





$h_K(a) = 5$	$h_K(d) = 4$	$h_K(ad) = 5$
$h_K(b) = 2$	$h_K(ab) = 7$	
$h_K(c) = 3$	$h_K(ac) = 4$	

$h'(a) = 6$	$h'(d) = 5$	$h'(ad) = 5$
$h'(b) = 2.5$	$h'(ab) = 7.5$	
$h'(c) = 3.5$	$h'(ac) = 4.5$	

Cut Sparsifiers, Informally

Definition

$G' = (K, E')$ is a **Cut Sparsifier** for $G = (V, E)$ if all cuts in G' are at least as large as the corresponding min-cut in G.

Cut Sparsifiers, Informally

Definition

$G' = (K, E')$ is a **Cut Sparsifier** for $G = (V, E)$ if all cuts in G' are at least as large as the corresponding min-cut in G.

Definition

The **Quality** of a Cut Sparsifier is the maximum ratio of a cut in G' to the corresponding min-cut in G.

Cut Sparsifiers

Good quality Cut Sparsifiers exist!

Cut Sparsifiers

Good quality Cut Sparsifiers exist! And such graphs can be computed efficiently!

Cut Sparsifiers

Good quality Cut Sparsifiers exist! And such graphs can be computed efficiently!

Theorem (Moitra, FOCS 2009)

For all (undirected) weighted graphs $G = (V, E)$, and all $K \subset V$ there is an (undirected) weighted graph $G' = (K, E')$ such that G' is a $O(\log k / \log \log k)$-quality Cut Sparsifier.

Cut Sparsifiers

Good quality Cut Sparsifiers exist! And such graphs can be computed efficiently!

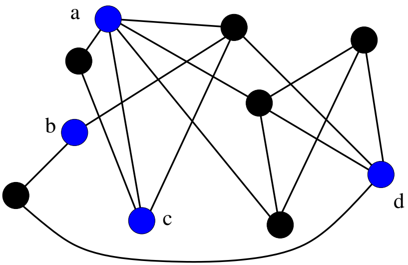
Theorem (Moitra, FOCS 2009)

For all (undirected) weighted graphs $G = (V, E)$, and all $K \subset V$ there is an (undirected) weighted graph $G' = (K, E')$ such that G' is a $O(\log k / \log\log k)$-quality Cut Sparsifier.

This bound improves to $O(1)$ if G is planar, or if G excludes any fixed minor!

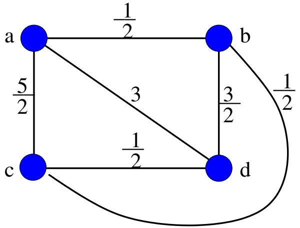
An Application to Steiner Minimum Bisection

Graph $G=(V,E)$



$h_K(a) = 5$ $\quad h_K(d) = 4$ $\quad h_K(ad) = 5$
$h_K(b) = 2$ $\quad h_K(ab) = 7$
$h_K(c) = 3$ $\quad h_K(ac) = 4$

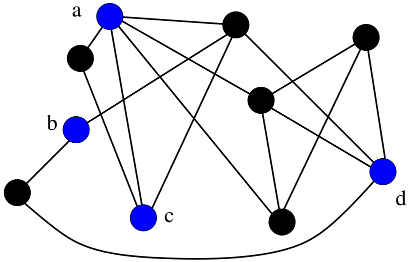
Sparsifier $G'=(K,E')$



$h'(a) = 6$ $\quad h'(d) = 5$ $\quad h'(ad) = 5$
$h'(b) = 2.5$ $\quad h'(ab) = 7.5$
$h'(c) = 3.5$ $\quad h'(ac) = 4.5$

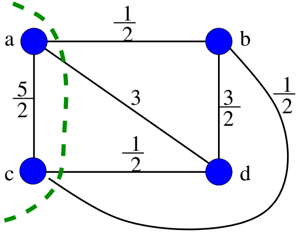
An Application to Steiner Minimum Bisection

Graph G=(V,E)



$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$

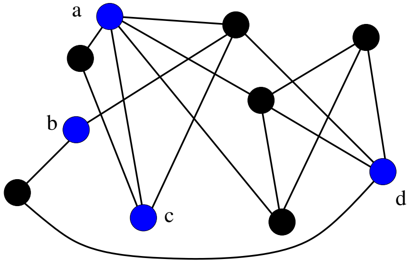
Sparsifier G'=(K,E')



$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

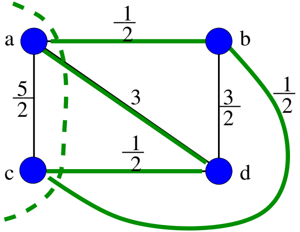
An Application to Steiner Minimum Bisection

Graph $G=(V,E)$



$h_K(a) = 5$ $\quad h_K(d) = 4$ $\quad h_K(ad) = 5$
$h_K(b) = 2$ $\quad h_K(ab) = 7$
$h_K(c) = 3$ $\quad h_K(ac) = 4$

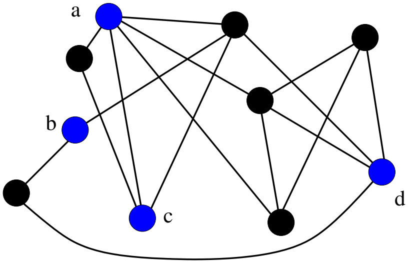
Sparsifier $G'=(K,E')$



$h'(a) = 6$ $\quad h'(d) = 5$ $\quad h'(ad) = 5$
$h'(b) = 2.5$ $\quad h'(ab) = 7.5$
$h'(c) = 3.5$ $\quad h'(ac) = 4.5$

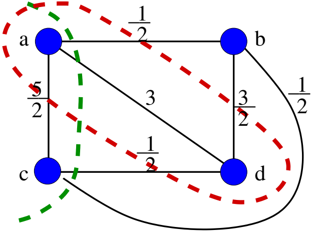
An Application to Steiner Minimum Bisection

Graph G=(V,E)



$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$

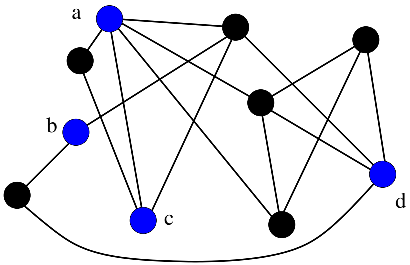
Sparsifier G'=(K,E')



$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

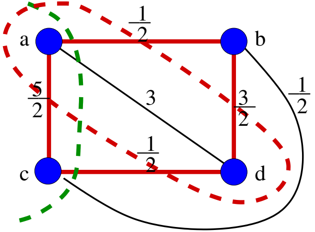
An Application to Steiner Minimum Bisection

Graph $G=(V,E)$



$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$

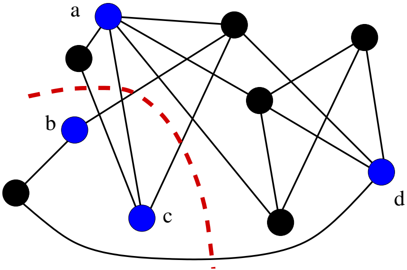
Sparsifier $G'=(K,E')$



$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

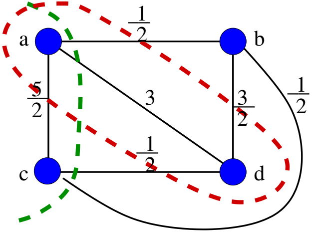
An Application to Steiner Minimum Bisection

Graph $G=(V,E)$



$h_K(a) = 5$ $\quad h_K(d) = 4$ $\quad h_K(ad) = 5$
$h_K(b) = 2$ $\quad h_K(ab) = 7$
$h_K(c) = 3$ $\quad h_K(ac) = 4$

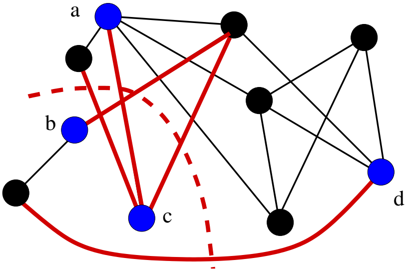
Sparsifier $G'=(K,E')$



$h'(a) = 6$ $\quad h'(d) = 5$ $\quad h'(ad) = 5$
$h'(b) = 2.5$ $\quad h'(ab) = 7.5$
$h'(c) = 3.5$ $\quad h'(ac) = 4.5$

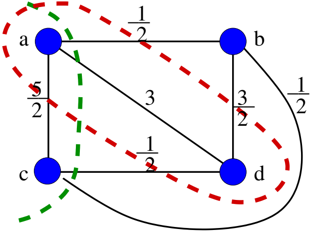
An Application to Steiner Minimum Bisection

Graph $G=(V,E)$



$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$

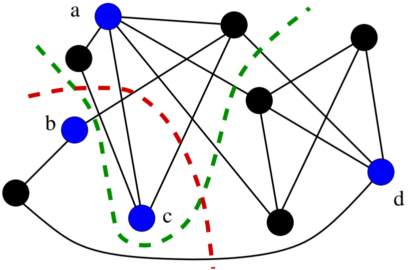
Sparsifier $G'=(K,E')$



$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

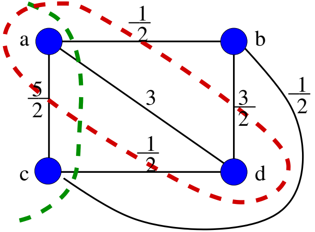
An Application to Steiner Minimum Bisection

Graph G=(V,E)



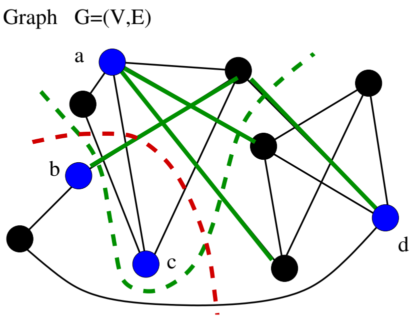
$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$

Sparsifier G'=(K,E')

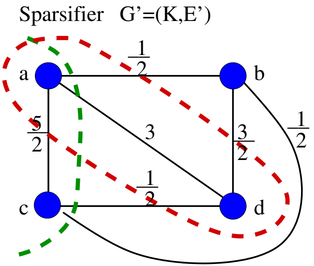


$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

An Application to Steiner Minimum Bisection



$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$
$h_K(b) = 2$ $h_K(ab) = 7$
$h_K(c) = 3$ $h_K(ac) = 4$



$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

This is a general strategy!

This is a general strategy!

For any problem characterized by cuts or flows:

This is a general strategy!

For any problem characterized by cuts or flows:

1. Construct G' so $OPT' \leq poly(\log k) OPT$

This is a general strategy!

For any problem characterized by cuts or flows:

1. Construct G' so $OPT' \leq poly(\log k) OPT$
2. Run approximation algorithm on G'

This is a general strategy!

For any problem characterized by cuts or flows:

1. Construct G' so $OPT' \leq poly(\log k)OPT$
2. Run approximation algorithm on G'
3. Map solution back to G

This is a general strategy!

For any problem characterized by cuts or flows:

1. Construct G' so $OPT' \leq poly(\log k) OPT$
2. Run approximation algorithm on G'
3. Map solution back to G

This will bootstrap a $poly(\log k)$ guarantee from a $poly(\log n)$ guarantee

Oblivious Reductions

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Question

What if we are asked to solve a routing problem on K, but we don't yet know the demands?

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Question

What if we are asked to solve a routing problem on K, but we don't yet know the demands?

1. Construct G'

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Question

What if we are asked to solve a routing problem on K, but we don't yet know the demands?

1. Construct G'

   Given G', there will be a **canonical** way to map flows in G' back to G

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Question

What if we are asked to solve a routing problem on K, but we don't yet know the demands?

1. Construct G'
   Given G', there will be a **canonical** way to map flows in G' back to G
2. Given demands, optimally solve on G'

Oblivious Reductions

This approach is useful even for efficiently solvable problems!

Question

What if we are asked to solve a routing problem on K, but we don't yet know the demands?

1. Construct G'
   Given G', there will be a **canonical** way to map flows in G' back to G
2. Given demands, optimally solve on G'
3. Map solution back to G

Highlights of Vertex Sparsification

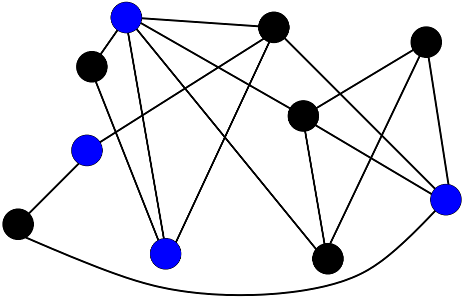
1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement, and minimum linear arrangement

Highlights of Vertex Sparsification

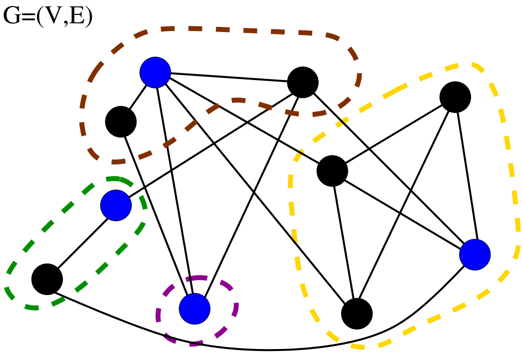
1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement, and minimum linear arrangement
2. **Oblivious Reductions:** All you need to know about the underlying communication network is its vertex sparsifier

Definition

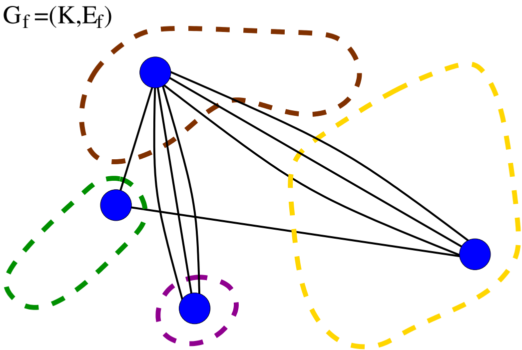
G=(V,E)



Definition

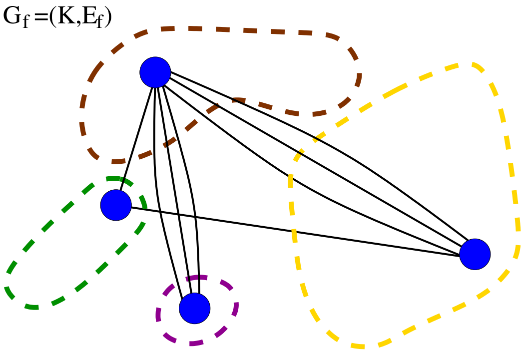


Definition



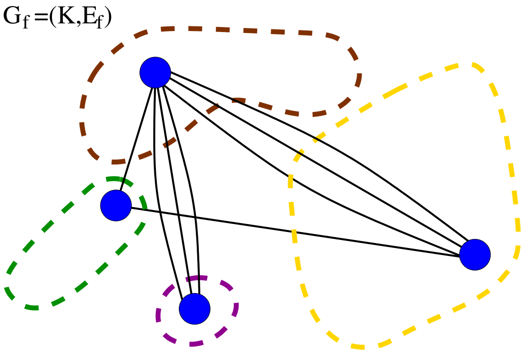
Definition

Let $f : V \rightarrow K$, is a 0-extension if for all $a \in K, f(a) = a$.



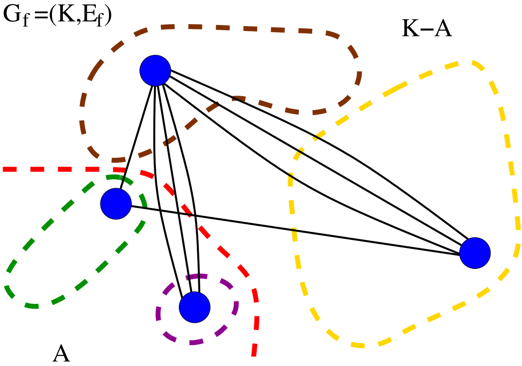
Lemma

G_f is a Cut Sparsifier



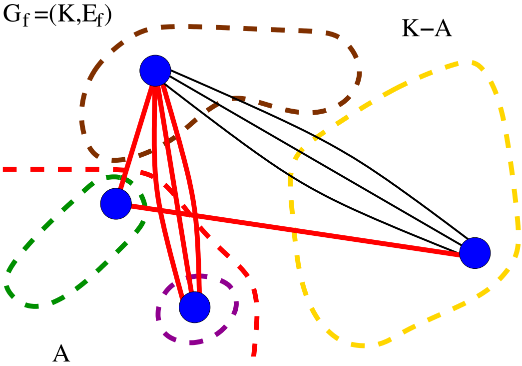
Lemma

G_f is a Cut Sparsifier



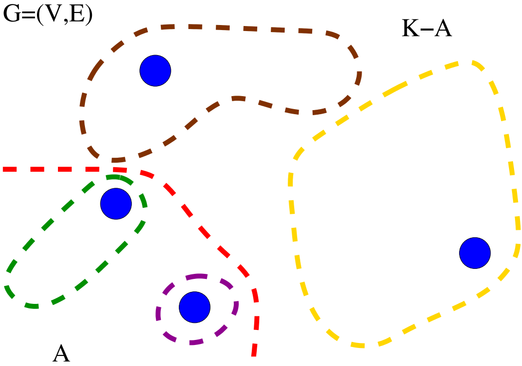
Lemma

G_f is a Cut Sparsifier



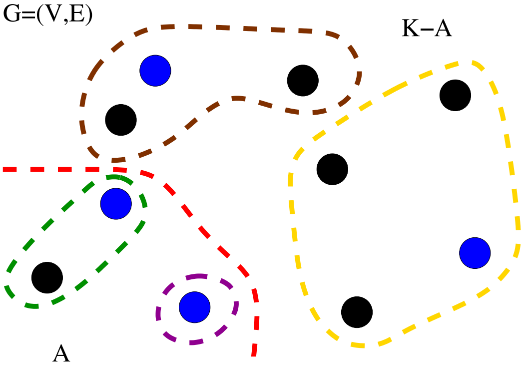
Lemma

G_f is a Cut Sparsifier



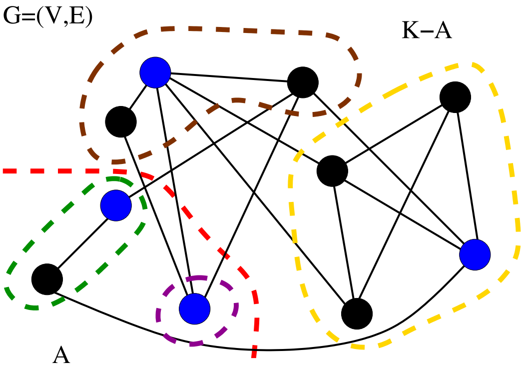
Lemma

G_f is a Cut Sparsifier



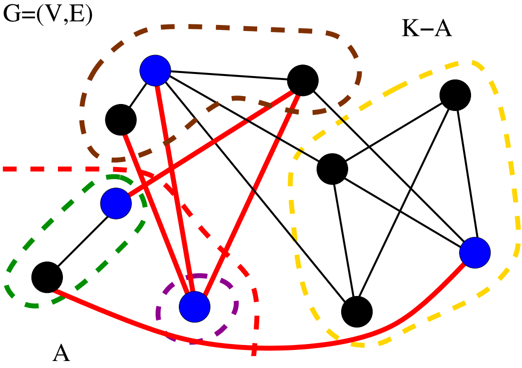
Lemma

G_f is a Cut Sparsifier

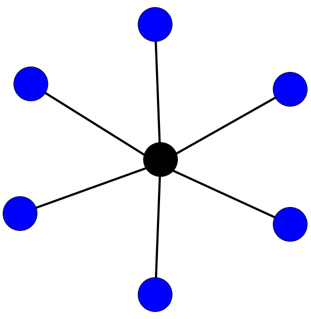


Lemma

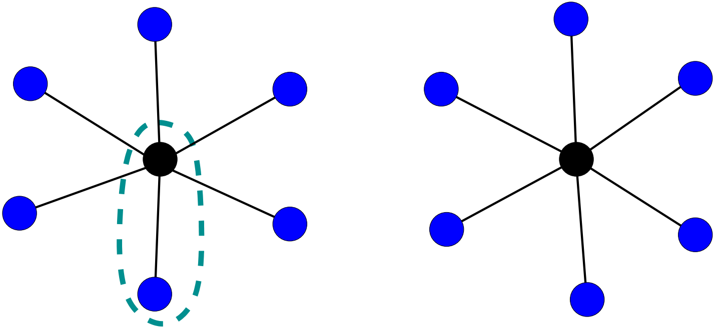
G_f is a Cut Sparsifier



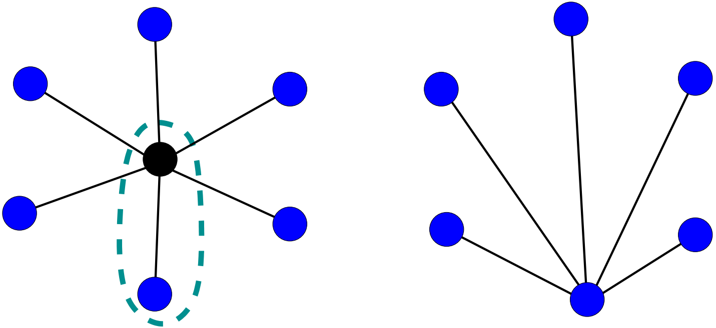
An Example



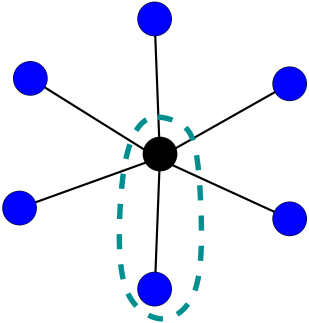
An Example



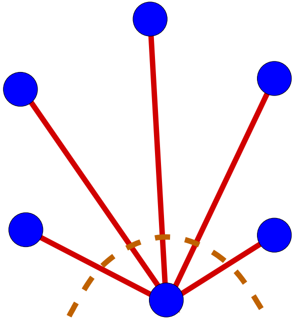
An Example



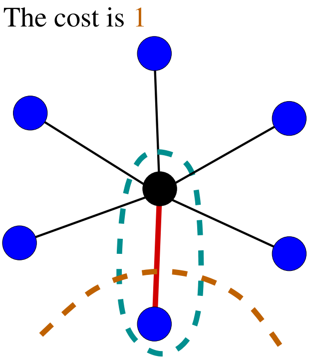
An Example

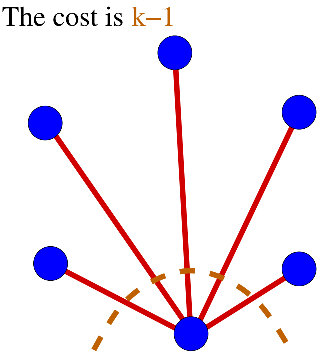


The cost is k−1

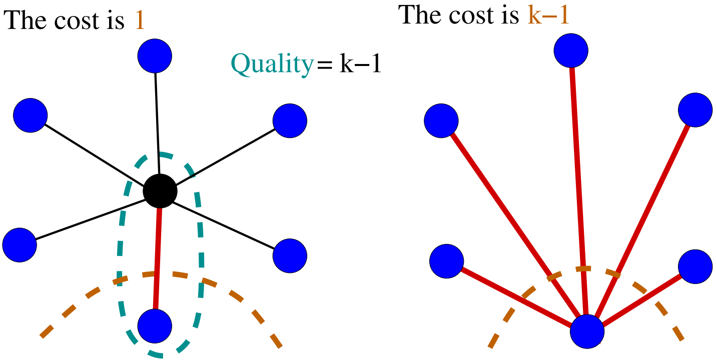


An Example

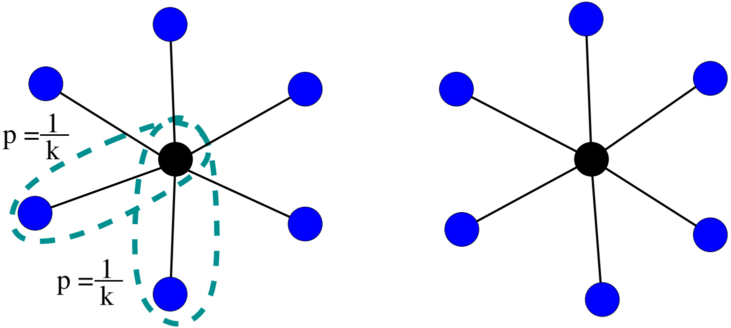




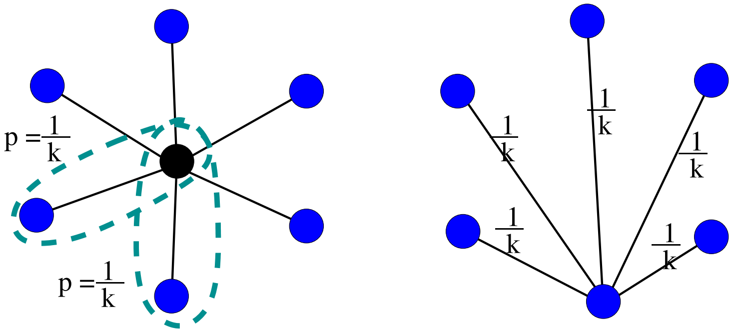
An Example



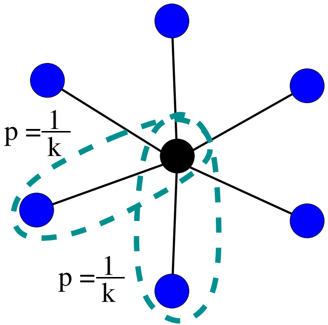
An Example: A Second Attempt

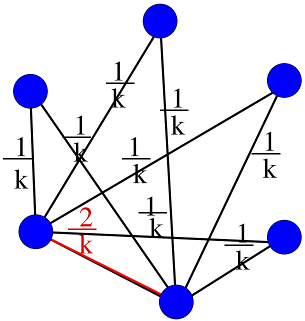


An Example: A Second Attempt

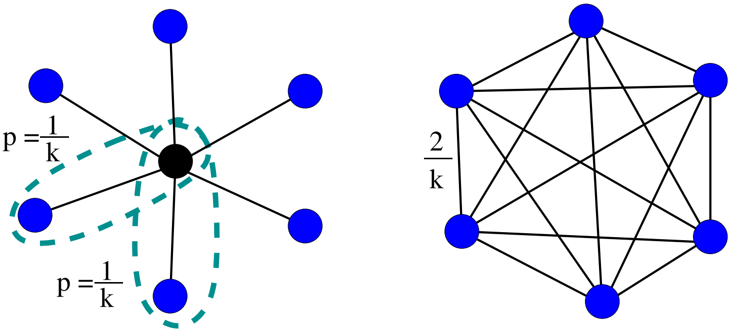


An Example: A Second Attempt

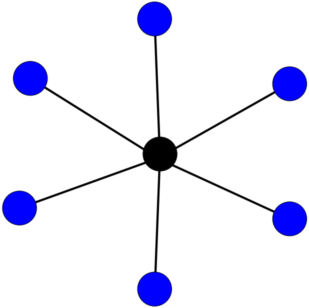


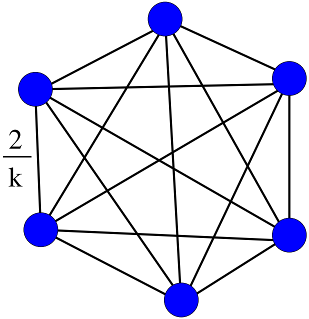


An Example: A Second Attempt

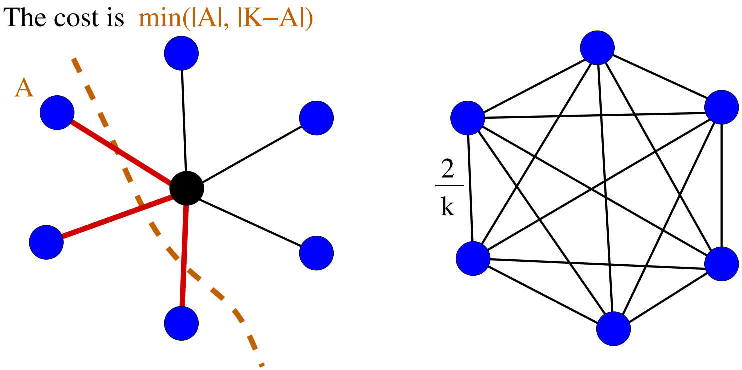


An Example: A Second Attempt

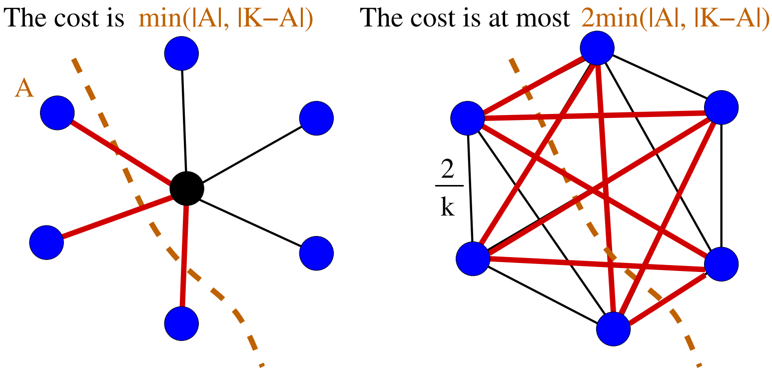




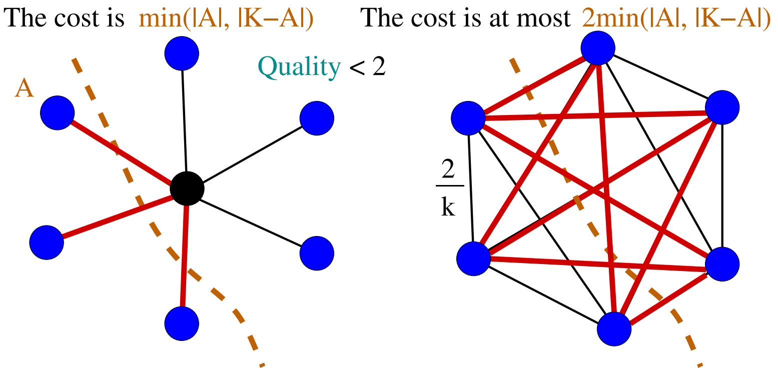
An Example: A Second Attempt



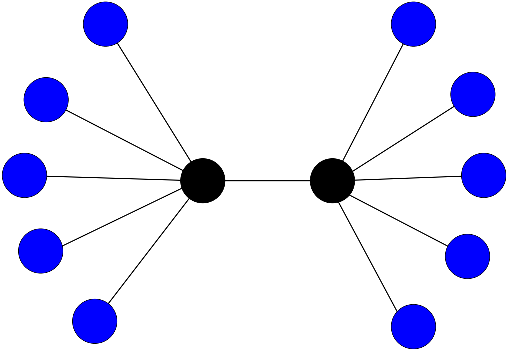
An Example: A Second Attempt



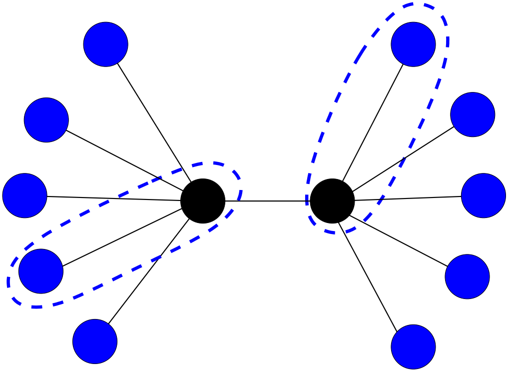
An Example: A Second Attempt



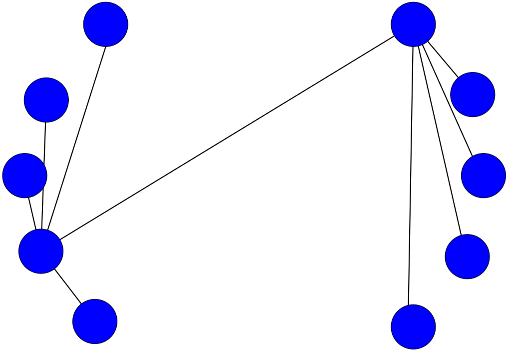
Another Example



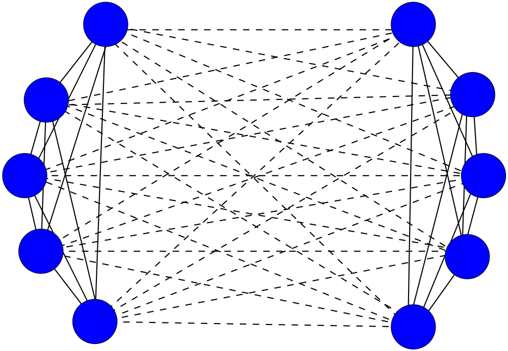
Another Example



Another Example



Another Example

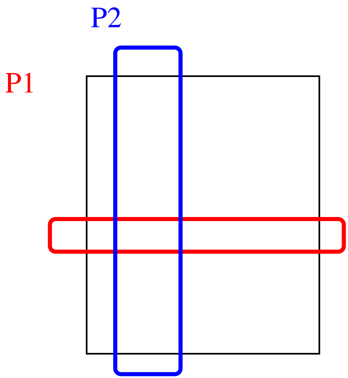


Proof Outline

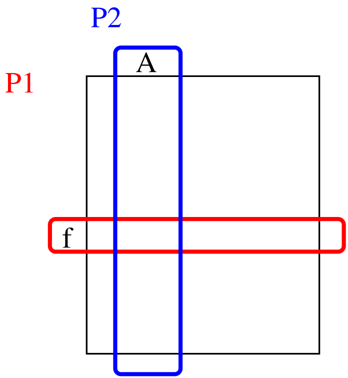
Proof Outline

1. Define a **Zero-Sum Game**

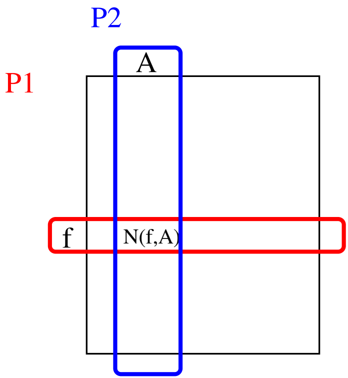
The Extension-Cut Game



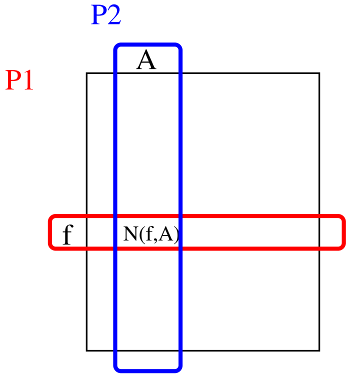
The Extension-Cut Game

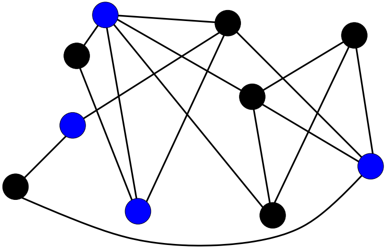


The Extension-Cut Game

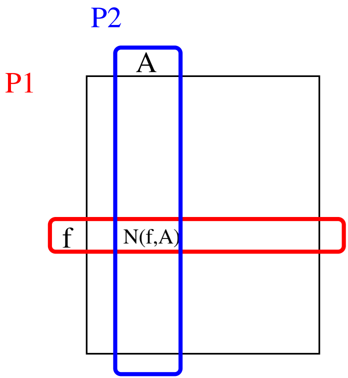


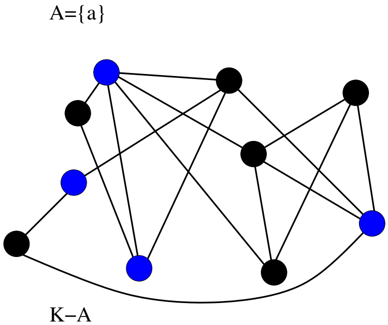
The Extension-Cut Game



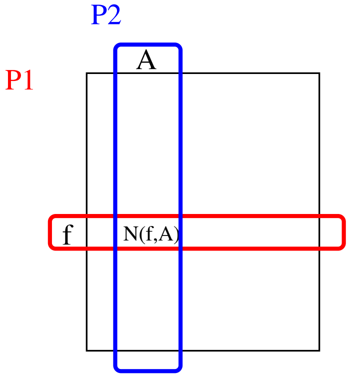


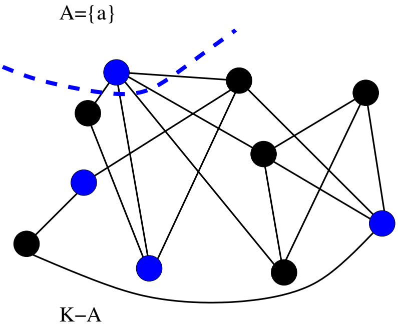
The Extension-Cut Game



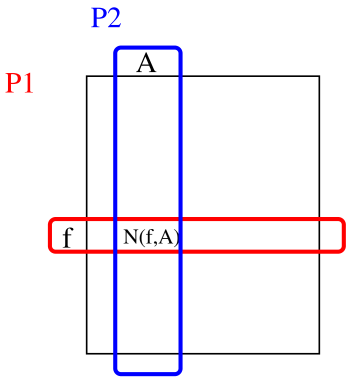


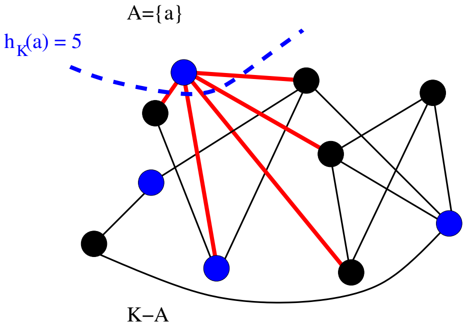
The Extension-Cut Game



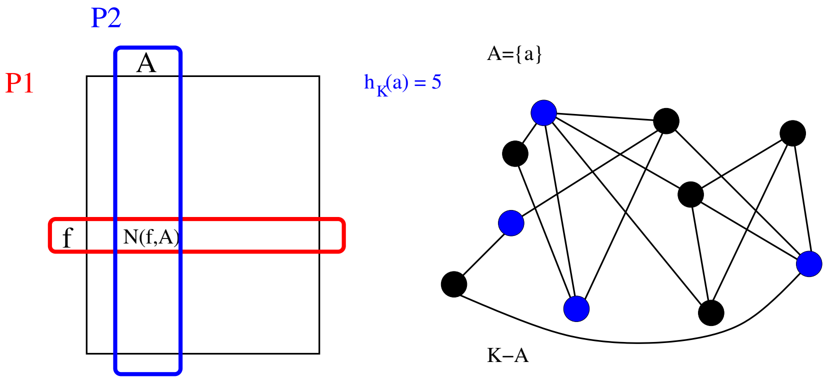


The Extension-Cut Game

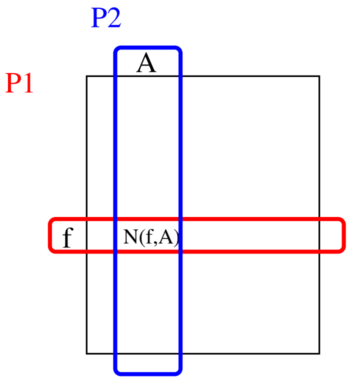


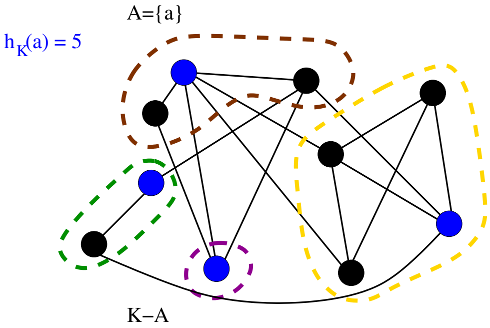


The Extension-Cut Game

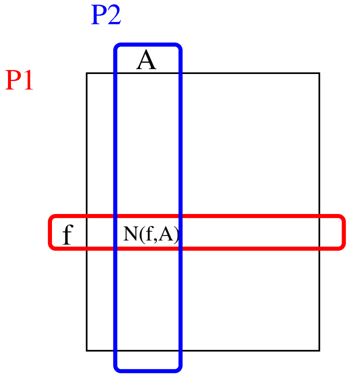


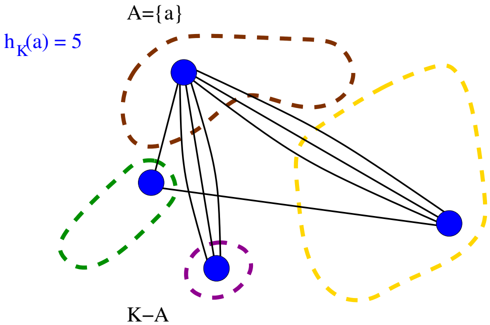
The Extension-Cut Game



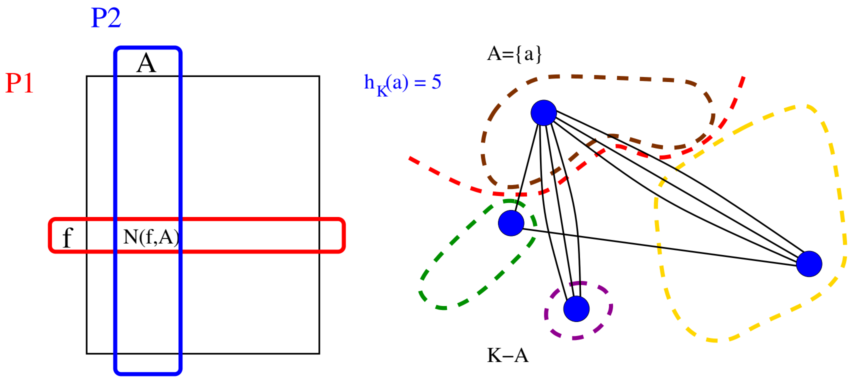


The Extension-Cut Game

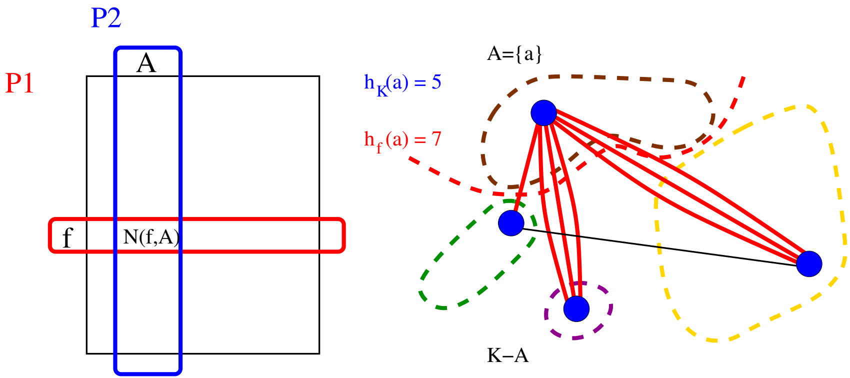




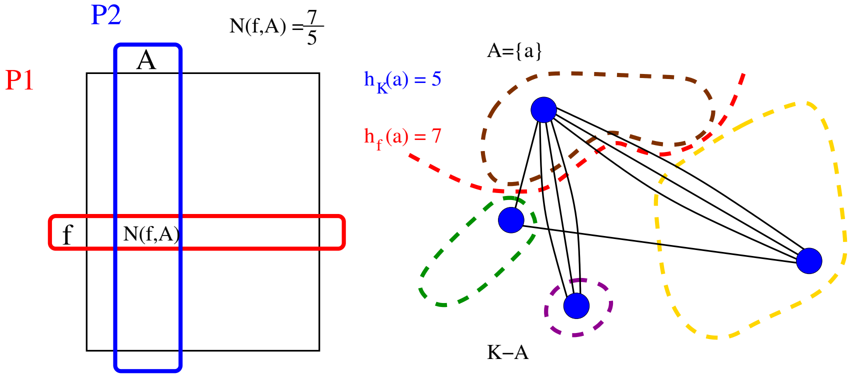
The Extension-Cut Game



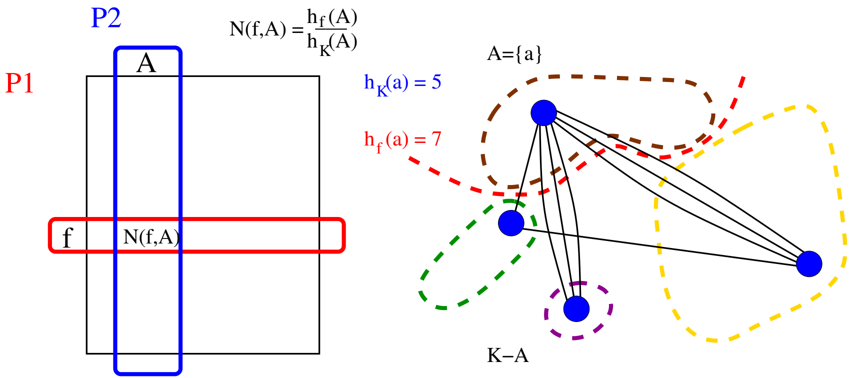
The Extension-Cut Game



The Extension-Cut Game



The Extension-Cut Game



Definition

Let ν denote the game value of the extension-cut game

Definition

Let ν denote the game value of the extension-cut game

So $\exists$ a distribution γ on 0-extensions s.t. for all $A \subset K$:

$$E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$$

Definition

Let ν denote the game value of the extension-cut game

So $\exists$ a distribution γ on 0-extensions s.t. for all $A \subset K$:

$$E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$$

Let $G' = \sum_f \gamma(f) G_f$. Then for all $A \subset K$:

$$h'(A) = \sum_f \gamma(f) h_f(A) = E_{f \leftarrow \gamma}[N(f, A)] h_K(A) \leq \nu h_K(A)$$

Proof Outline

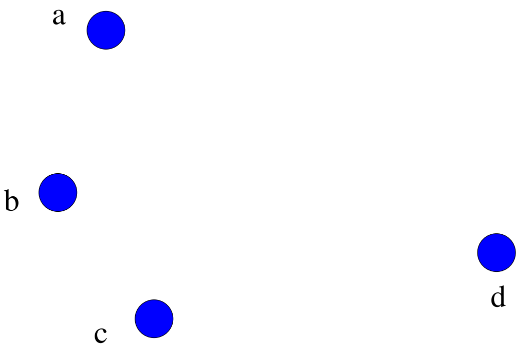
1. Define a **Zero-Sum Game**

Proof Outline

1. Define a **Zero-Sum Game**
2. The **Best Response** is a 0-Extension Problem

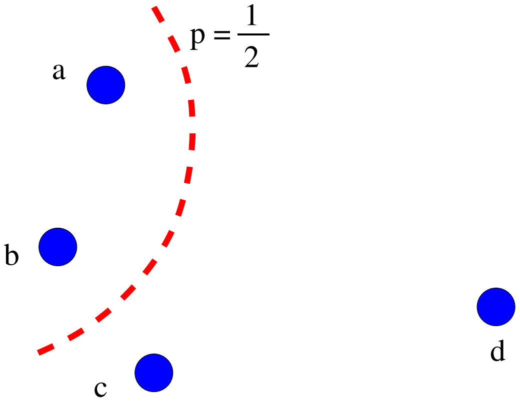
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



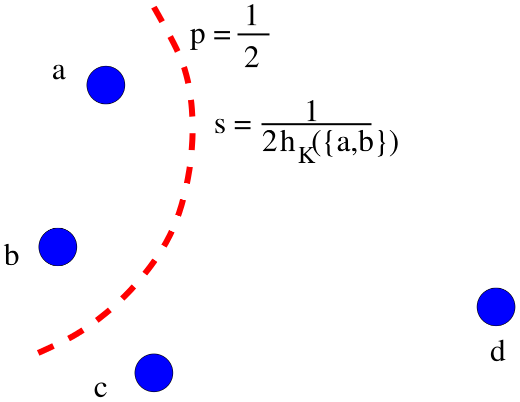
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



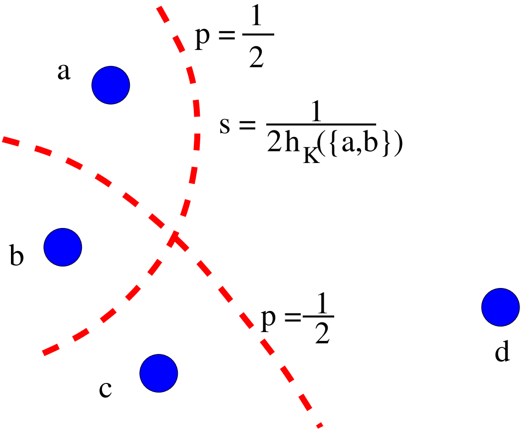
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



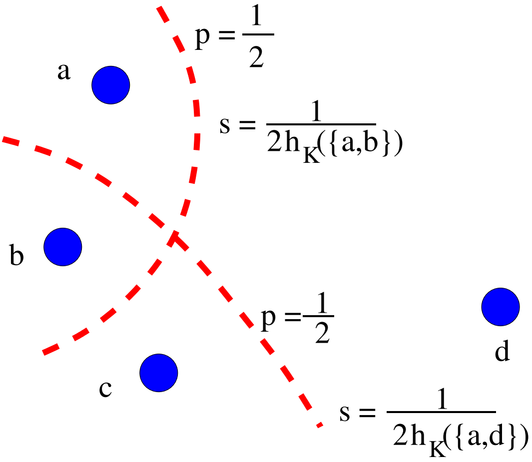
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



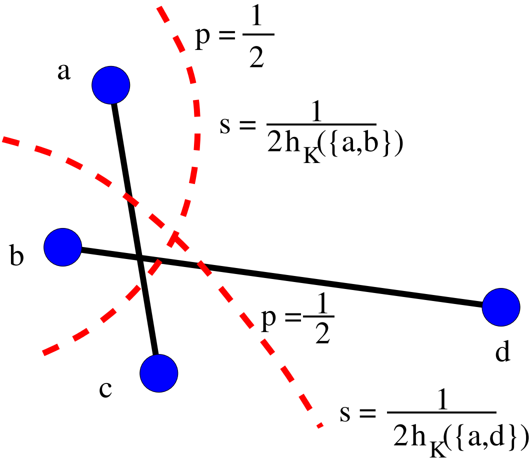
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



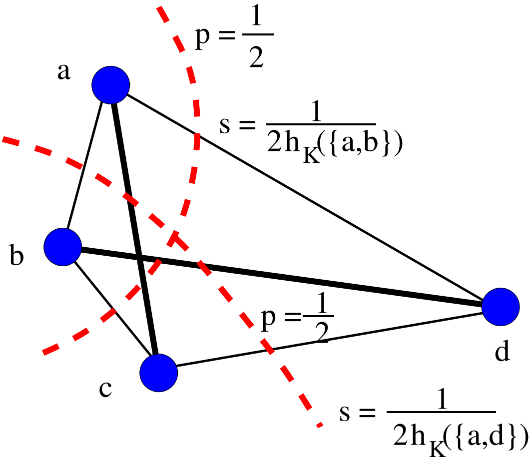
Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



Best Response?

Let $\mu = \frac{1}{2}\{a, b\} + \frac{1}{2}\{a, d\}$



Proof Outline

1. Define a **Zero-Sum Game**
2. The **Best Response** is a 0-Extension Problem

Proof Outline

1. Define a **Zero-Sum Game**
2. The **Best Response** is a 0-Extension Problem
3. Construct a **Feasible Solution** for the Linear Programming Relaxation

Proof Outline

1. Define a **Zero-Sum Game**
2. The **Best Response** is a 0-Extension Problem
3. Construct a **Feasible Solution** for the Linear Programming Relaxation
4. Round the solution to bound the **Game Value**
   [Fakcharoenphol, Harrelson, Rao, Talwar 2003]
   [Calinescu, Karloff, Rabani 2001]

Bounds on the Integrality Gap

($OPT^* =$ value of the LP)

Theorem (Fakcharoenphol, Harrelson, Rao, Talwar)

$$OPT \leq O(\frac{\log k}{\log \log k})OPT^*$$

Bounds on the Integrality Gap

(OPT^* = value of the LP)

Theorem (Fakcharoenphol, Harrelson, Rao, Talwar)

$$OPT \leq O\left(\frac{\log k}{\log \log k}\right) OPT^*$$

Theorem (Calinescu, Karloff, Rabani)

If G excludes any fixed minor,

$$OPT \leq O(1) OPT^*$$

Proof Outline

1. Define a **Zero-Sum Game**
2. The **Best Response** is a 0-Extension Problem
3. Construct a **Feasible Solution** for the Linear Programming Relaxation
4. Round the solution to bound the **Game Value**
   [Fakcharoenphol, Harrelson, Rao, Talwar 2003]
   [Calinescu, Karloff, Rabani 2001]

Limits to Cut Sparsification

Limits to Cut Sparsification

planar graphs,

Limits to Cut Sparsification

planar graphs, graphs excluding a fixed minor,

Limits to Cut Sparsification

planar graphs, graphs excluding a fixed minor, expanders

Limits to Cut Sparsification

planar graphs, graphs excluding a fixed minor, expanders all admit $O(1)$-quality Cut Sparsifiers

Limits to Cut Sparsification

planar graphs, graphs excluding a fixed minor, expanders all admit $O(1)$-quality Cut Sparsifiers

Question

Does every graph admit an $O(1)$-quality Cut Sparsifier?

Limits to Cut Sparsification

planar graphs, graphs excluding a fixed minor, expanders all admit $O(1)$-quality Cut Sparsifiers

Question

Does every graph admit an $O(1)$-quality Cut Sparsifier?

Theorem

There is an infinite family of graphs that admits no Cut Sparsifier of quality better than $\Omega(\log^{1/4} k)$

Independently proven in [Charikar, Leighton, Li, Moitra, FOCS 2010] and [Makarychev, Makarychev, FOCS 2010]

"Simple" Cut Sparsifiers

"Simple" Cut Sparsifiers

Question

Can we compute good Cut Sparsifiers on which our optimization problem is **easy***?*

"Simple" Cut Sparsifiers

Question

Can we compute good Cut Sparsifiers on which our optimization problem is **easy***?*

Question

Can we compute good Cut Sparsifiers that can be realized as a convex combination of (contraction-based) **trees***?*

"Simple" Cut Sparsifiers

Question

Can we compute good Cut Sparsifiers on which our optimization problem is **easy***?*

Question

Can we compute good Cut Sparsifiers that can be realized as a convex combination of (contraction-based) **trees***?*

Independently asked in [Englert, Gupta, Krauthgamer, Räcke, Talgam-Cohen, Talwar, APPROX 2010] with similar algorithmic implications...

Fractional Graph Partitioning Problems

Definition

We call an optimization problem a Fractional Graph Partitioning Problem if it can be written as

$$\begin{array}{rl} \min & \sum_{(u,v)\in E} c(u,v)d(u,v) \\ \text{s.t.} & \\ & d : V \times V \to \Re^{+} \text{ is a semi-metric} \\ & \ldots \end{array}$$

Fractional Graph Partitioning Problems

Definition

We call an optimization problem a Fractional Graph Partitioning Problem if it can be written as (for some monotone increasing function f):

$$\begin{array}{ll} \min & \sum_{(u,v)\in E} c(u,v)d(u,v) \\ \text{s.t.} & \\ & d : V \times V \rightarrow \Re^{+} \text{ is a semi-metric} \\ & f(d\big|_K) \geq 1 \end{array}$$

Examples

Consider the (standard) fractional relaxations for:

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:**

   Goal: Separate all pairs of demands, cutting few edges

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:** $f(d\Big|_K) = \min_i d(s_i, t_i)$
   Goal: Separate all pairs of demands, cutting few edges

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:** $f(d\Big|_K) = \min_i d(s_i, t_i)$
   Goal: Separate all pairs of demands, cutting few edges
2. **Sparsest Cut:**
   Goal: Find a cut with small ratio

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:** $f(d\Big|_K) = \min_i d(s_i, t_i)$
   Goal: Separate all pairs of demands, cutting few edges
2. **Sparsest Cut:** $f(d\Big|_K) = \sum_i dem(i) d(s_i, t_i)$
   Goal: Find a cut with small ratio

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:** $f(d\Big|_K) = \min_i d(s_i, t_i)$
   Goal: Separate all pairs of demands, cutting few edges
2. **Sparsest Cut:** $f(d\Big|_K) = \sum_i dem(i)d(s_i, t_i)$
   Goal: Find a cut with small ratio
3. **Requirement Cut:**
   Goal: Separate all sets R_i into at least p_i components, cutting few edges

Examples

Consider the (standard) fractional relaxations for:

1. **Multi-Cut:** $f(d\Big|_K) = \min_i d(s_i, t_i)$
   Goal: Separate all pairs of demands, cutting few edges
2. **Sparsest Cut:** $f(d\Big|_K) = \sum_i dem(i)d(s_i, t_i)$
   Goal: Find a cut with small ratio
3. **Requirement Cut:** $f(d\Big|_K) = \min_i \frac{\mathrm{MST}(R_i)}{p_i}$
   Goal: Separate all sets R_i into at least p_i components, cutting few edges

Theorem

(Charikar, Leighton, Li, Moitra, FOCS 2010) For any graph partitioning problem, the maximum integrality gap is at most $O(\log k)$ times the max integrality gap restricted to trees

Theorem

(Charikar, Leighton, Li, Moitra, FOCS 2010) For any graph partitioning problem, the maximum integrality gap is at most $O(\log k)$ times the max integrality gap restricted to trees

This encapsulates known integrality gaps for fractional graph partitioning problems

Theorem

(Charikar, Leighton, Li, Moitra, FOCS 2010) For any graph partitioning problem, the maximum integrality gap is at most $O(\log k)$ times the max integrality gap restricted to trees

This encapsulates known integrality gaps for fractional graph partitioning problems such as:

1. [Garg, Vazirani, Yannakakis]

Theorem

(Charikar, Leighton, Li, Moitra, FOCS 2010) For any graph partitioning problem, the maximum integrality gap is at most $O(\log k)$ times the max integrality gap restricted to trees

This encapsulates known integrality gaps for fractional graph partitioning problems such as:

1. [Garg, Vazirani, Yannakakis]
2. [Linial, Londan, Rabinovich], [Aumann, Rabani]

Theorem

(Charikar, Leighton, Li, Moitra, FOCS 2010) For any graph partitioning problem, the maximum integrality gap is at most $O(\log k)$ times the max integrality gap restricted to trees

This encapsulates known integrality gaps for fractional graph partitioning problems such as:

1. [Garg, Vazirani, Yannakakis]
2. [Linial, Londan, Rabinovich], [Aumann, Rabani]
3. [Gupta, Nagarajan, Ravi]
4. ...

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement, and minimum linear arrangement
2. **Oblivious Reductions:** All you need to know about the underlying communication network is its vertex sparsifier

Highlights of Vertex Sparsification

1. **Approximation Guarantees Independent of the Graph Size:** We give the first $poly(\log k)$ approximation algorithms (or competitive ratios) for: Steiner minimum bisection, requirement cut, l-multicut, oblivious 0-extension, and Steiner generalizations of oblivious routing, min-cut linear arrangement, and minimum linear arrangement
2. **Oblivious Reductions:** All you need to know about the underlying communication network is its vertex sparsifier
3. **Abstract Integrality Gaps:** We give $O(\log k)$ flow-cut gaps for any graph partitioning problem, if the integrality gap is constant on trees

Epilogue

Epilogue

1. Constructive results through **lifting**

Epilogue

1. Constructive results through **lifting**
2. Extensions to multicommodity flow – implications for network coding

Epilogue

1. Constructive results through **lifting**
2. Extensions to multicommodity flow – implications for network coding
3. Lower bounds via examples from **functional analysis**

Epilogue

1. Constructive results through **lifting**
2. Extensions to multicommodity flow – implications for network coding
3. Lower bounds via examples from **functional analysis**
4. Separations using **harmonic analysis** of Boolean functions

Thanks!

References

1. Moitra, "Approximation algorithms with guarantees independent of the graph size", FOCS 2009
2. Leighton, Moitra, "Extensions and limits to vertex sparsification", STOC 2010
3. Englert, Gupta, Krauthgamer, Räcke, Talgam-Cohen, Talwar, "Vertex sparsifiers: new results from old techniques", APPROX 2010
4. Makarychev, Makarychev, "Metric extension operators, vertex sparsifiers and lipschitz extendability", FOCS 2010
5. Charikar, Leighton, Li, Moitra, "Vertex sparsifiers and abstract rounding algorithms", FOCS 2010