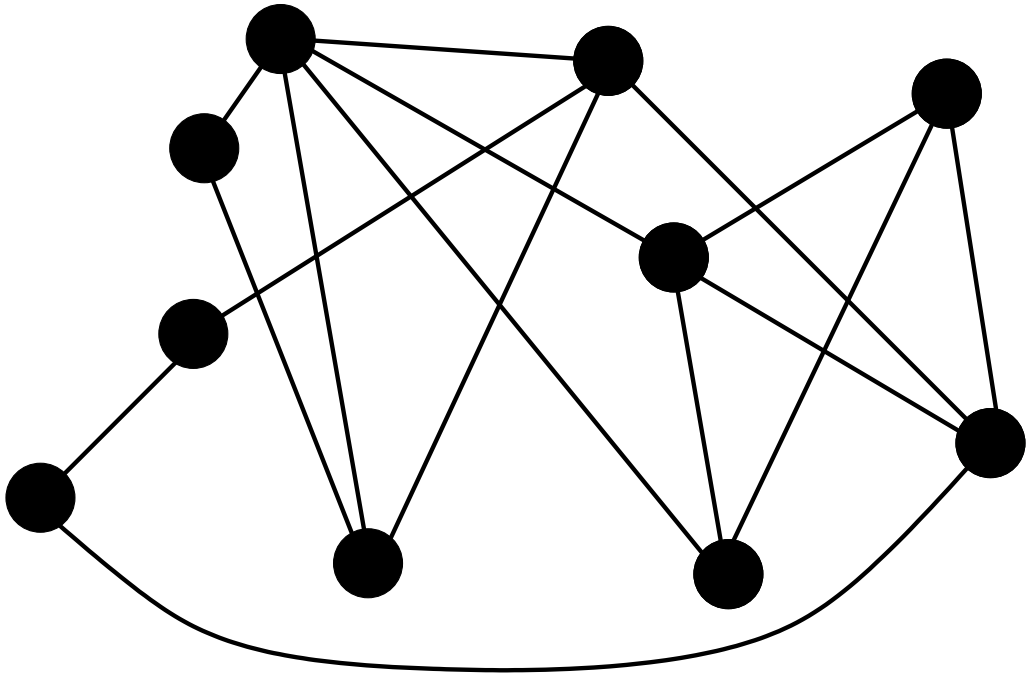
Approximation Algorithms for Multicommodity-Type Problems with Guarantees Independent of the Graph Size

Ankur Moitra, MIT

October 23, 2009

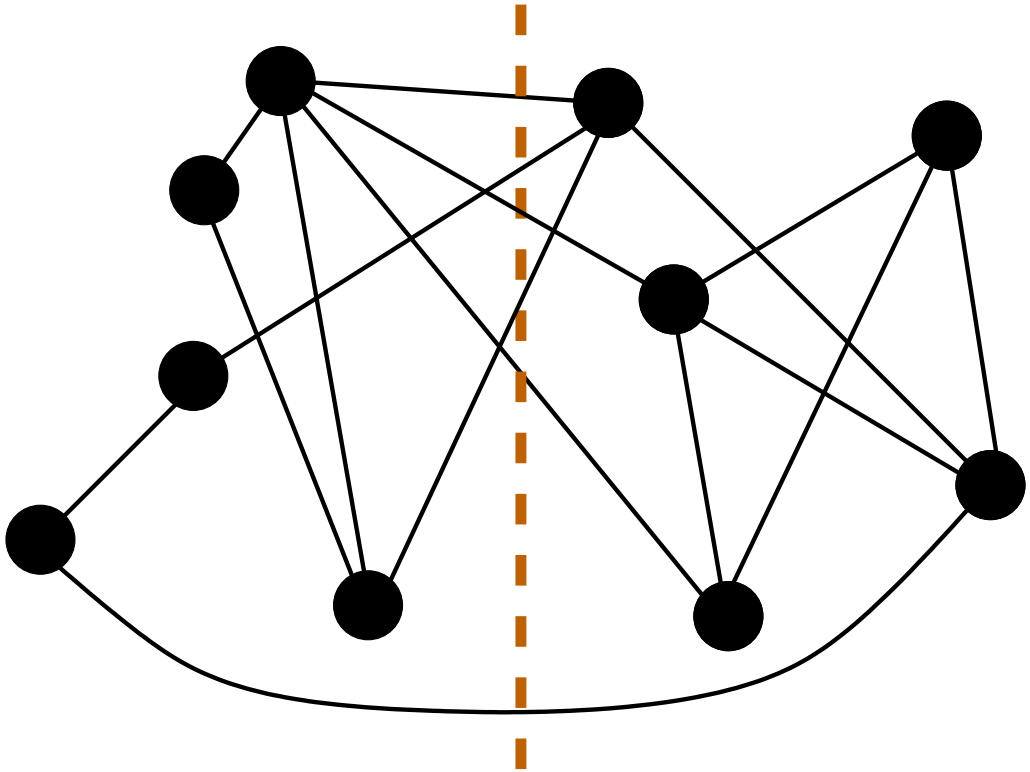
The Minimum Bisection Problem

Goal: Minimize cost of bisection



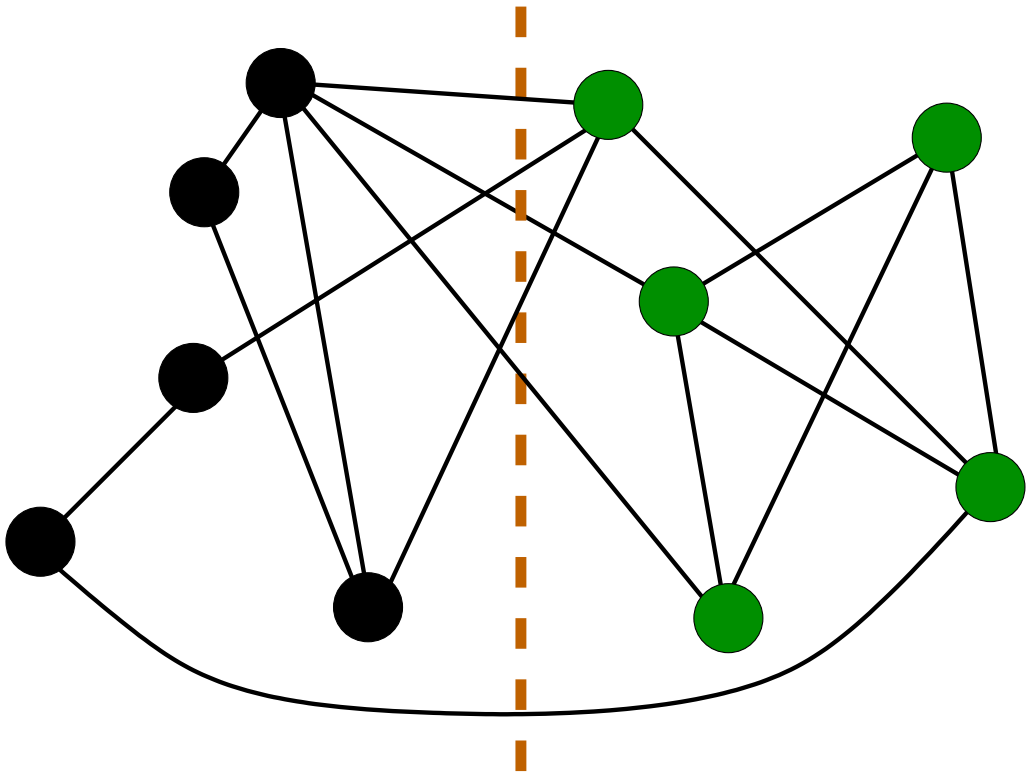
The Minimum Bisection Problem

Goal: Minimize cost of bisection



The Minimum Bisection Problem

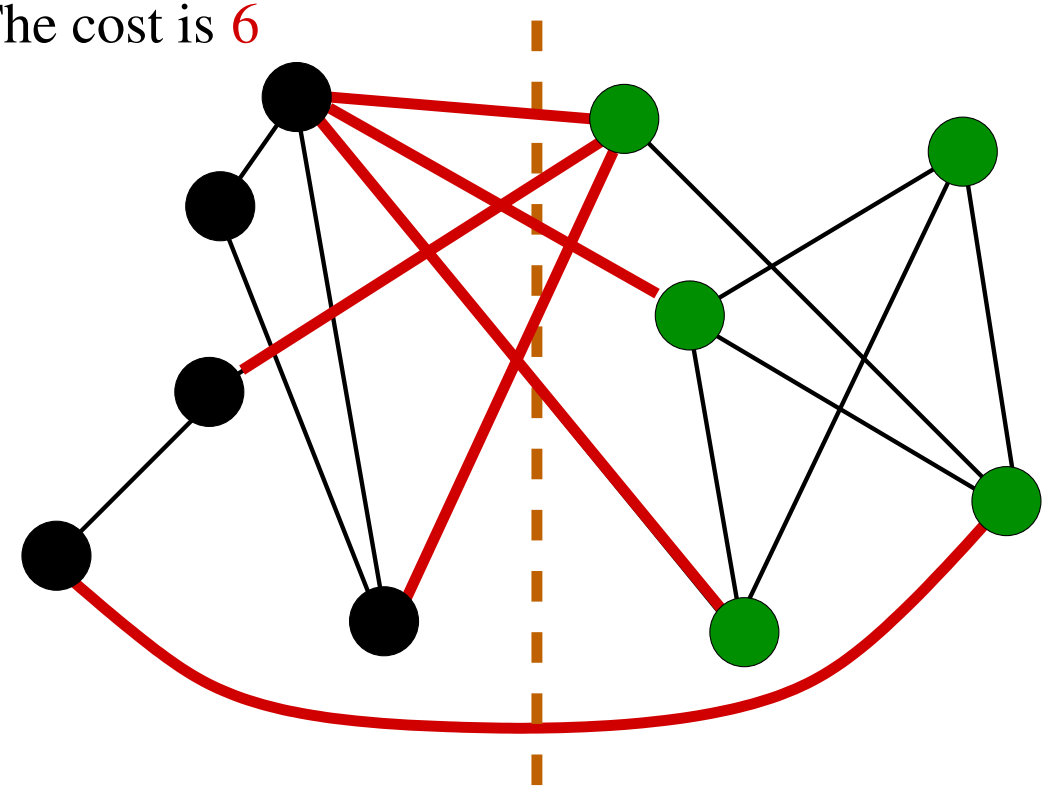
Goal: Minimize cost of bisection



The Minimum Bisection Problem

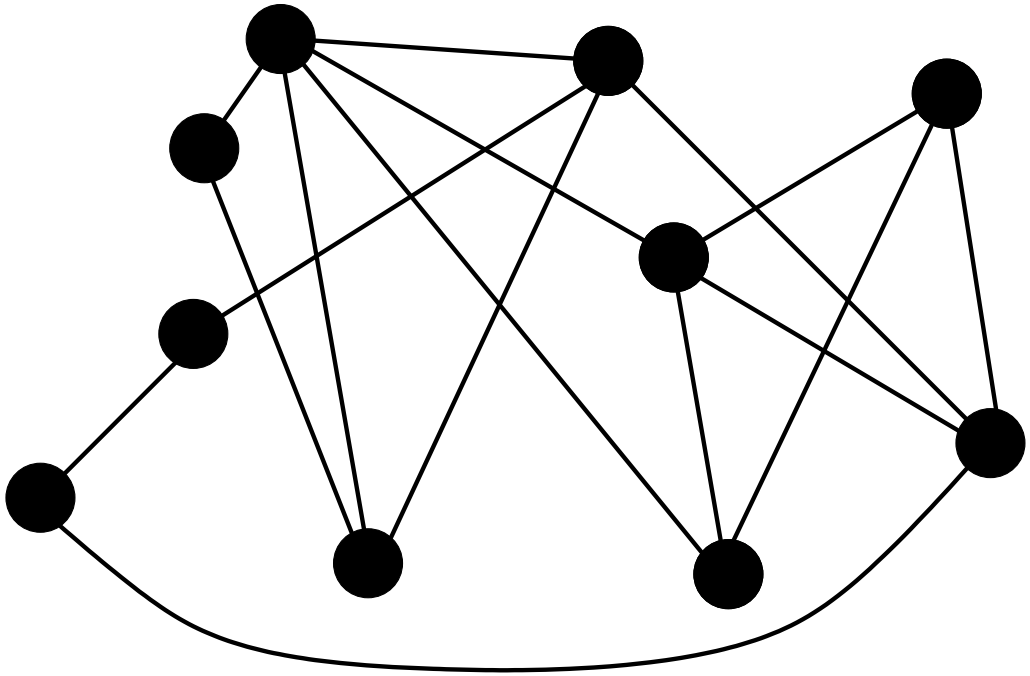
Goal: Minimize cost of bisection

The cost is 6



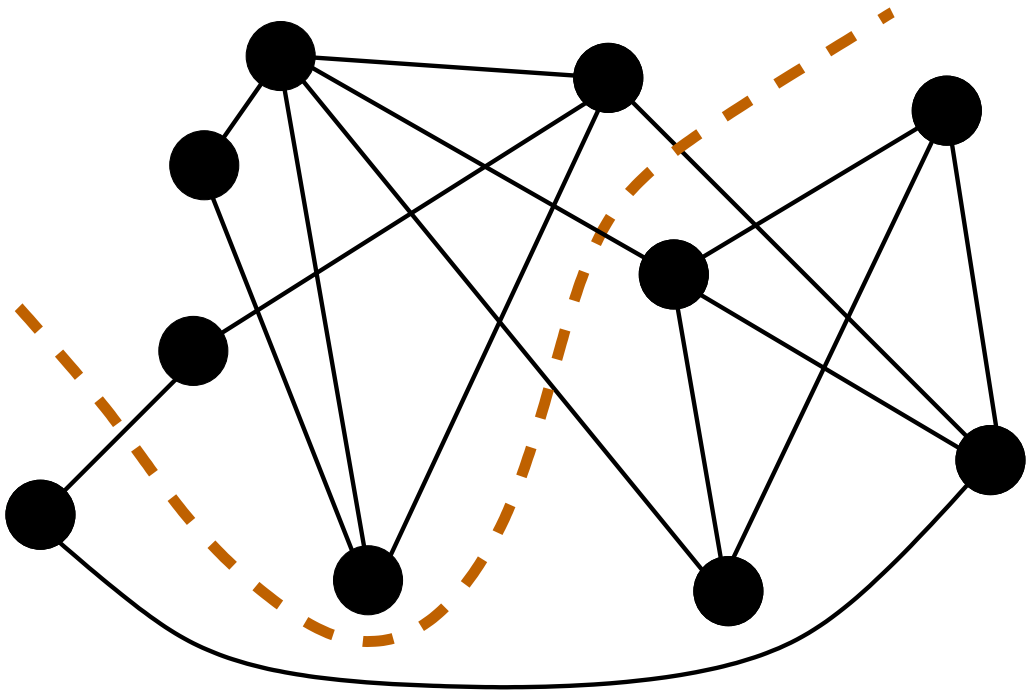
The Minimum Bisection Problem

Goal: Minimize cost of bisection



The Minimum Bisection Problem

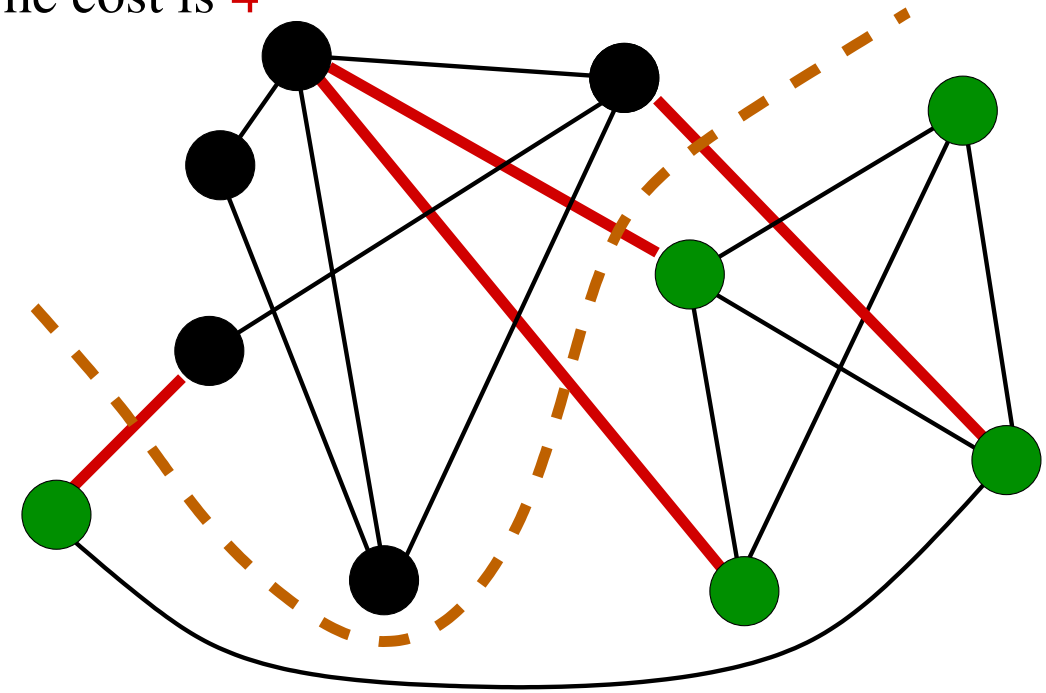
Goal: Minimize cost of bisection



The Minimum Bisection Problem

Goal: Minimize cost of bisection

The cost is 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

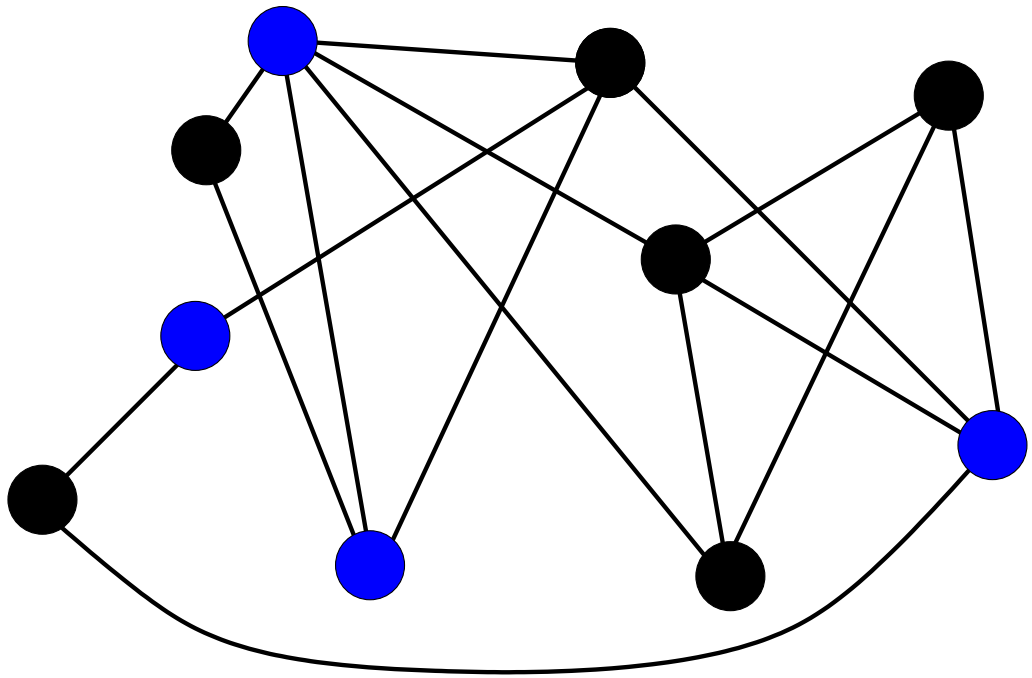
1. Applications through Divide-and-Conquer: VLSI design, sparse matrix computations, approximation algorithms
2. [Kernighan, Lin 1970] Local search heuristic
3. [Garey, Johnson, Stockmeyer 1976] *NP*-Complete
4. [Leighton, Rao 1988] $O(\log n)$ *approximate* minimum bisection
5. [Saran, Vazirani 1995]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

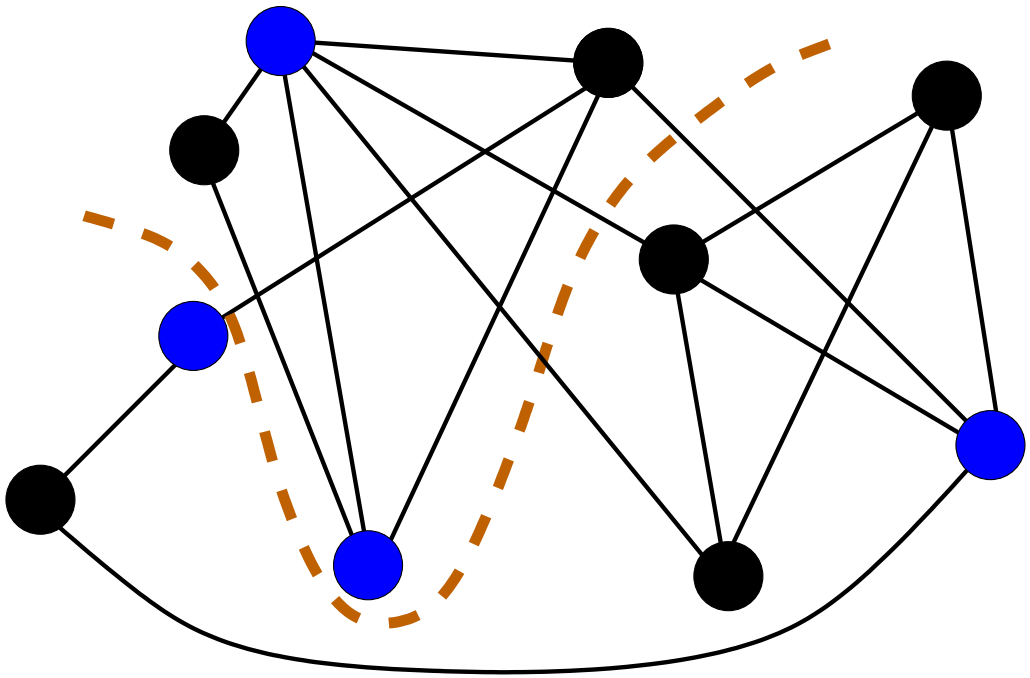
The Steiner Minimum Bisection Problem

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



The Steiner Minimum Bisection Problem

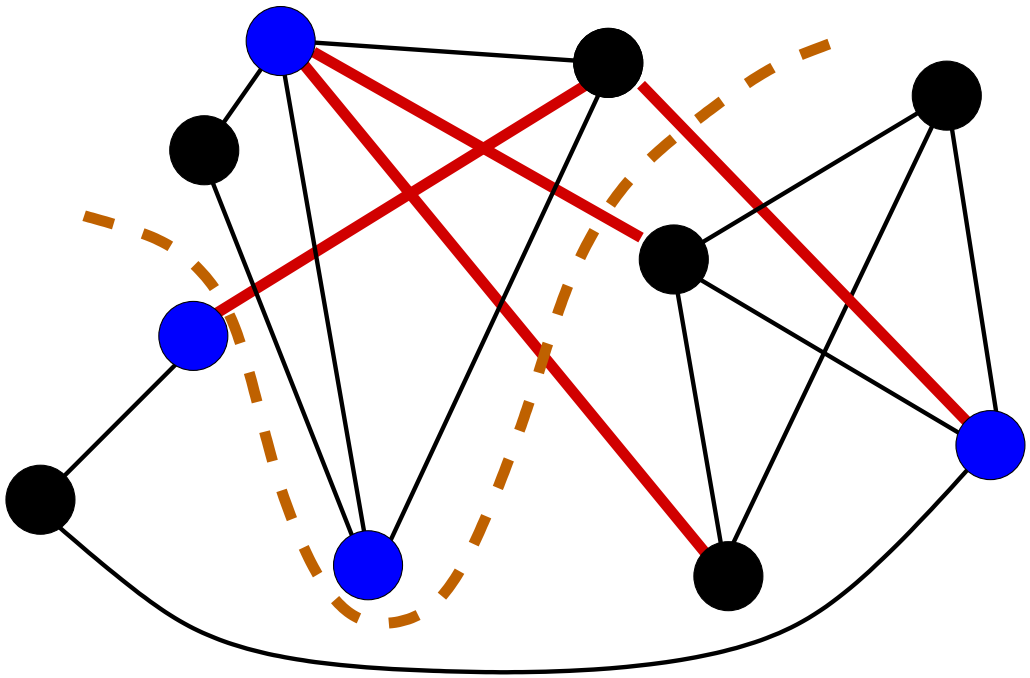
Goal: Minimize cost of a bisection of the k blue nodes



The Steiner Minimum Bisection Problem

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

The cost is 4



Question

Can we find a 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 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 poly$(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for a multicommodity-type problem*

A Meta Question

Given

A poly$(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for a multicommodity-type problem*

Let k be the number of "interesting" nodes

A Meta Question

Given

A poly$(\log n)$ *approximation algorithm (integrality gap or competitive ratio) for a multicommodity-type problem*

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 a multicommodity-type problem*

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, and ...

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut
3. l-multicut

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut
3. l-multicut
4. oblivious 0-extension

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut
3. l-multicut
4. oblivious 0-extension
   And Steiner generalizations of:
5. oblivious routing

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut
3. l-multicut
4. oblivious 0-extension
   And Steiner generalizations of:
5. oblivious routing
6. min-cut linear arrangement

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

1. Steiner minimum bisection
2. requirement cut
3. l-multicut
4. oblivious 0-extension

   And Steiner generalizations of:
5. oblivious routing
6. min-cut linear arrangement
7. minimum linear arrangement

Our Results

We give the first $poly(\log k)$-approximation algorithms (or competitive ratios) for:

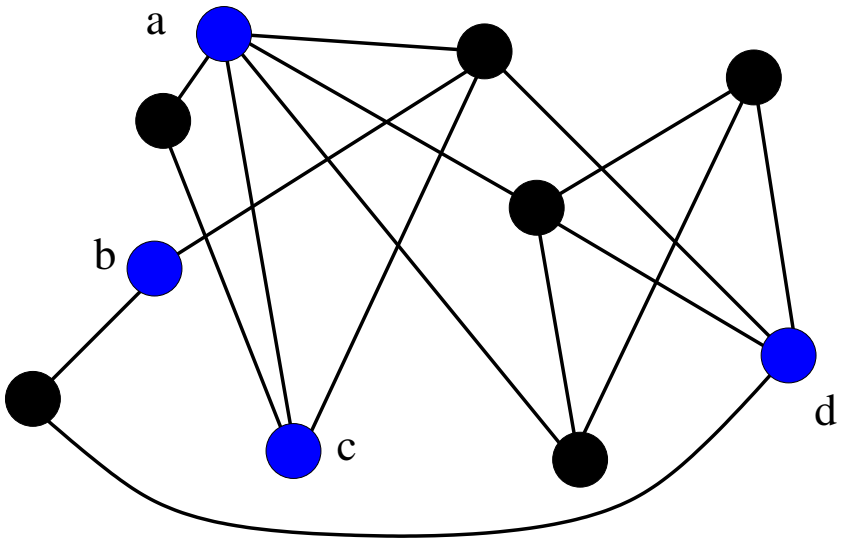
1. Steiner minimum bisection
2. requirement cut
3. l-multicut
4. oblivious 0-extension

   And Steiner generalizations of:
5. oblivious routing
6. min-cut linear arrangement
7. minimum linear arrangement

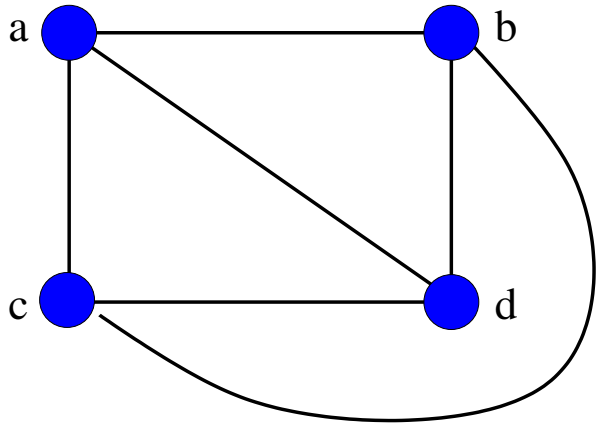
Our approach is to prove the existence of vertex sparsifiers that preserve cuts...

General Approach: Vertex Sparsifiers

Graph G=(V,E)

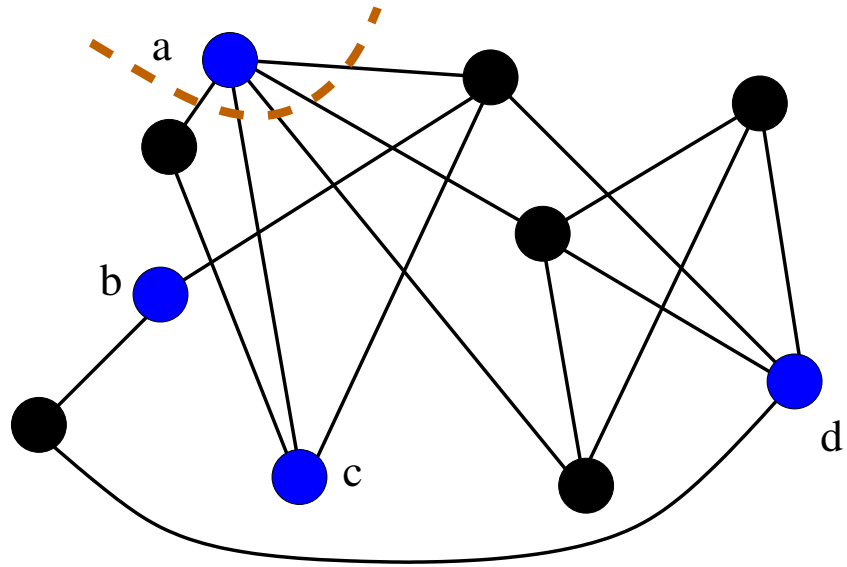


Sparsifier G'=(K,E')

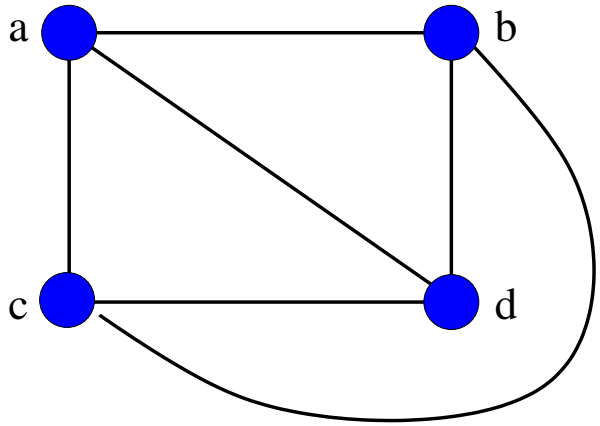


General Approach: Vertex Sparsifiers

Graph G=(V,E)

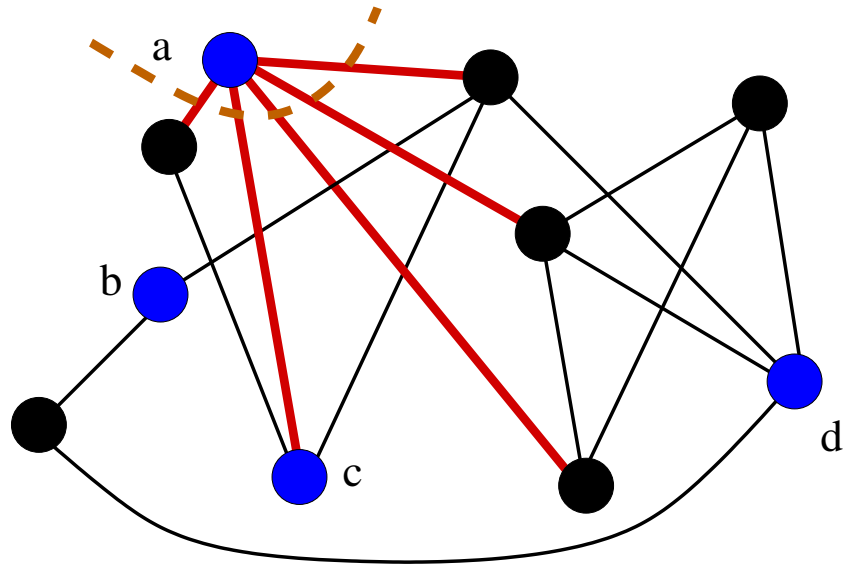


Sparsifier G'=(K,E')

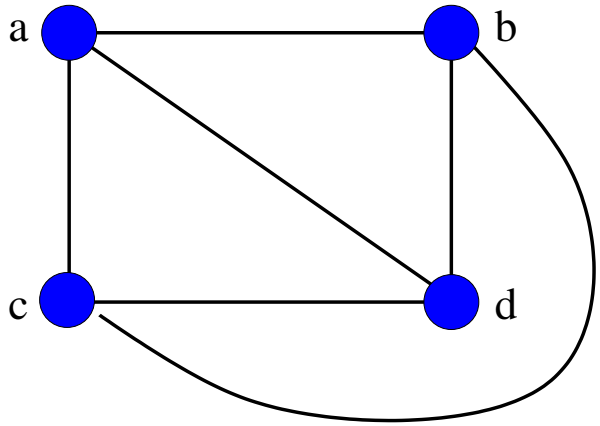


General Approach: Vertex Sparsifiers

Graph G=(V,E)

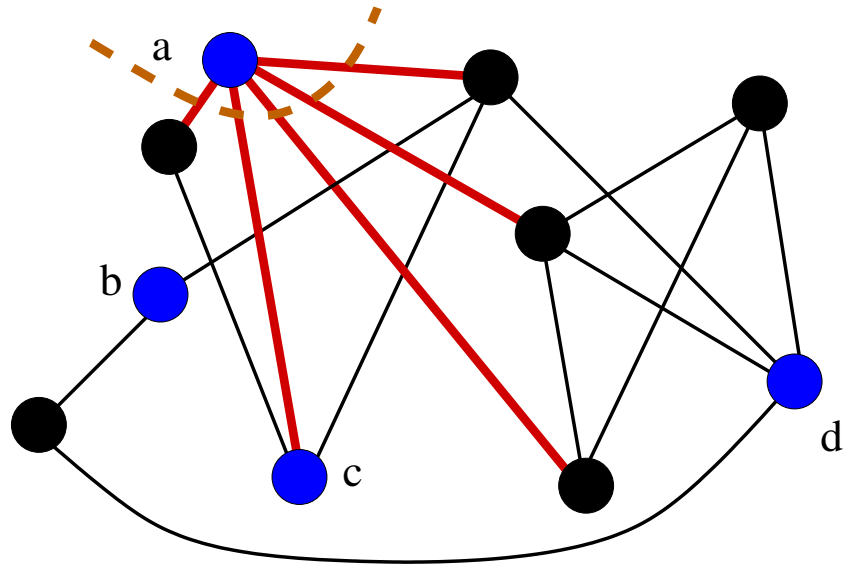


Sparsifier G'=(K,E')



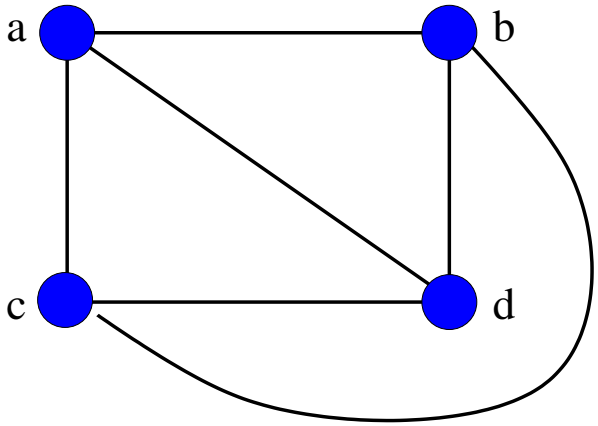
General Approach: Vertex Sparsifiers

Graph G=(V,E)

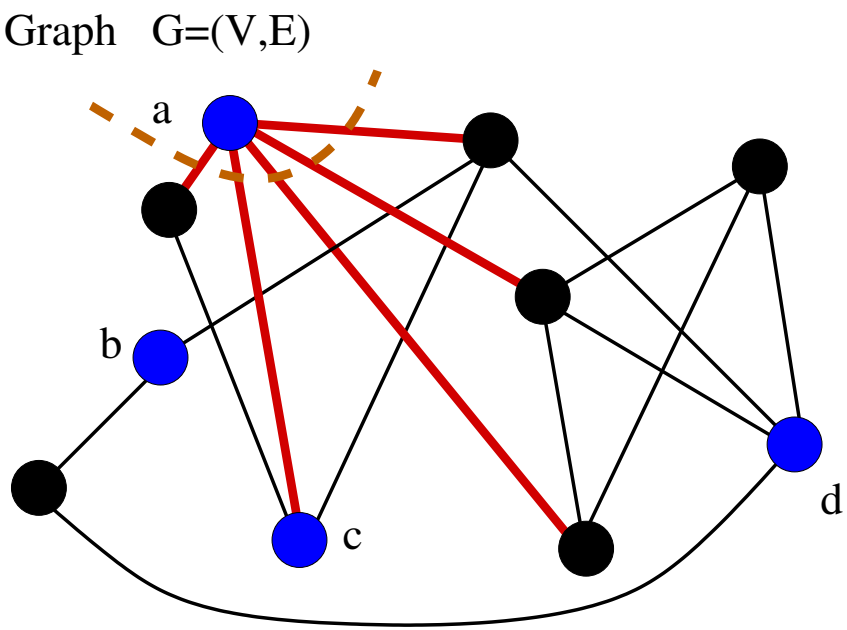


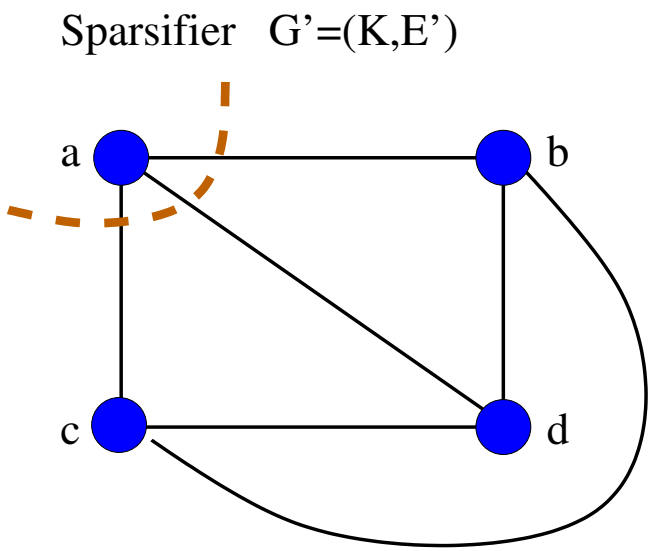
$h_K(a) = 5$

Sparsifier G'=(K,E')

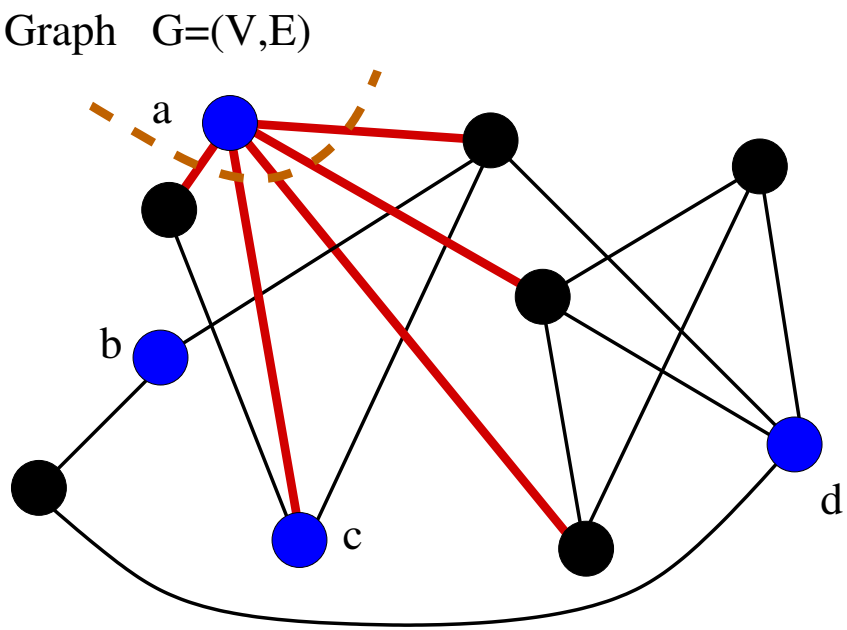


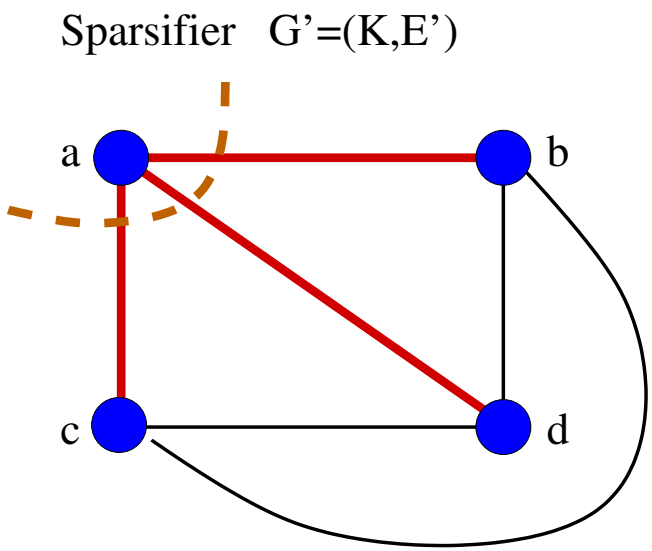
General Approach: Vertex Sparsifiers





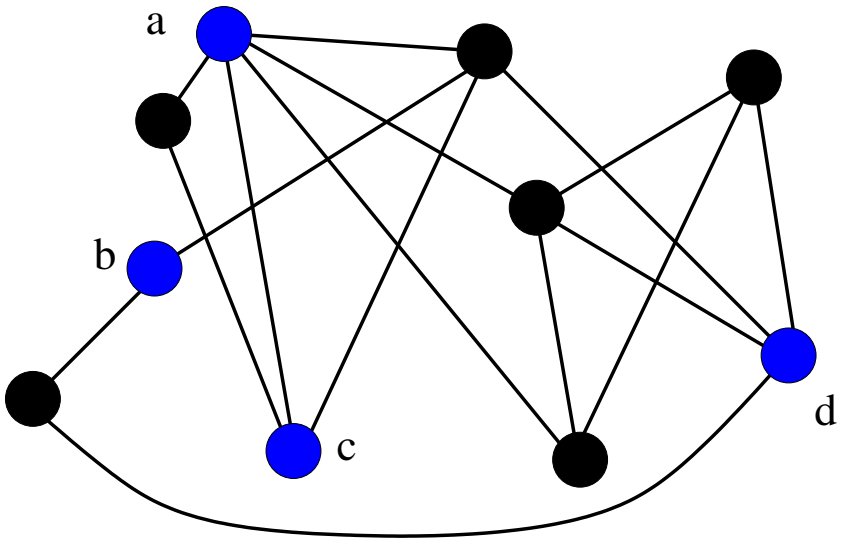
General Approach: Vertex Sparsifiers



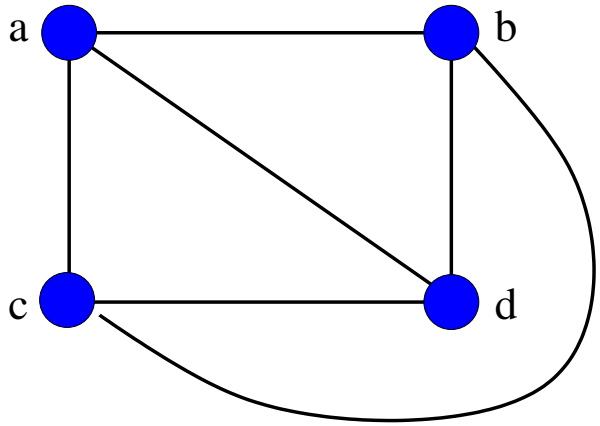


General Approach: Vertex Sparsifiers

Graph G=(V,E)

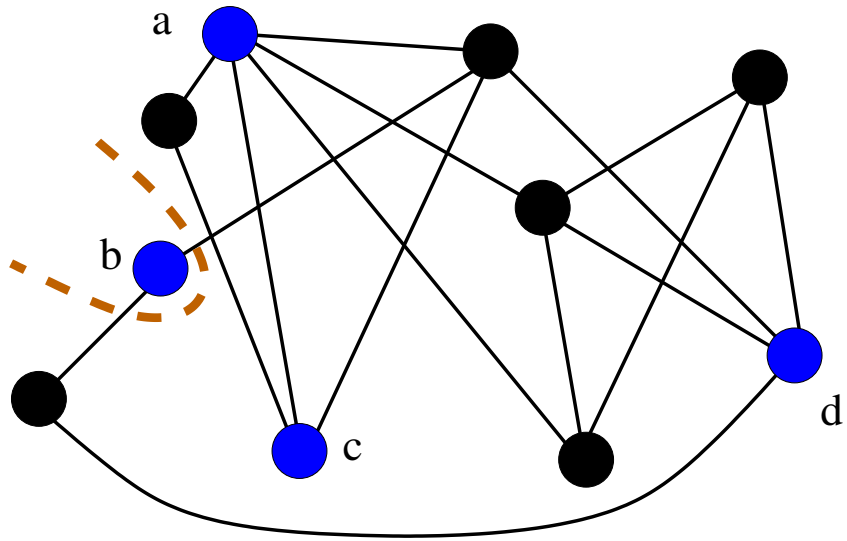


Sparsifier G'=(K,E')

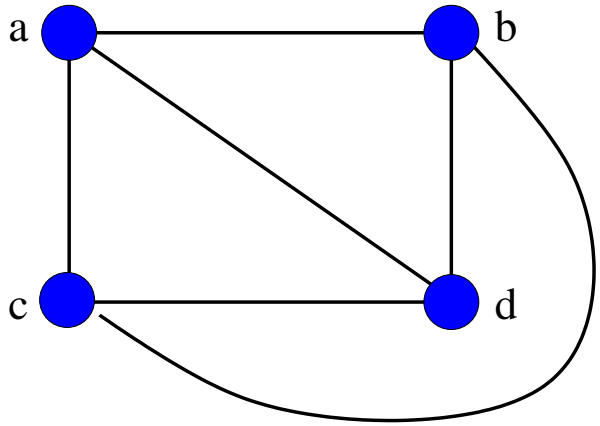


General Approach: Vertex Sparsifiers

Graph G=(V,E)

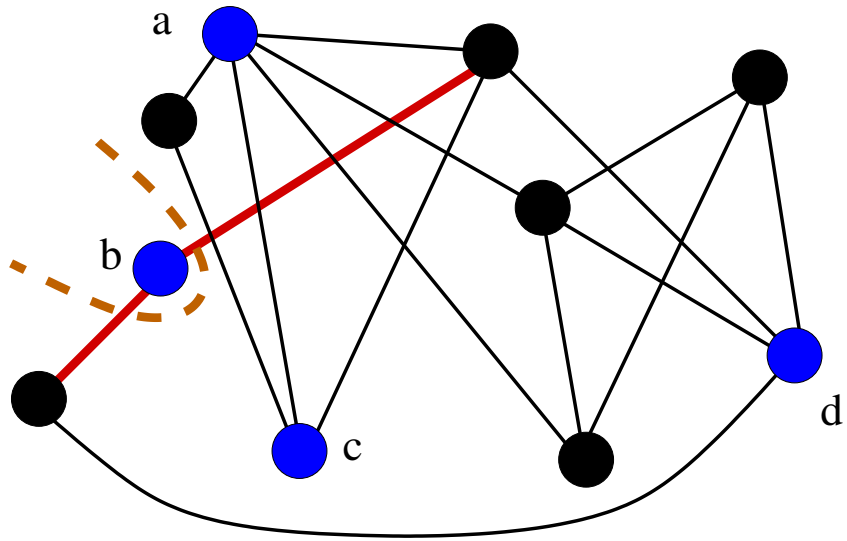


Sparsifier G'=(K,E')

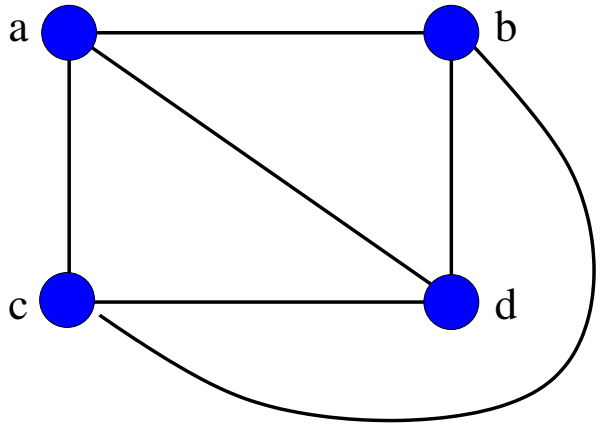


General Approach: Vertex Sparsifiers

Graph G=(V,E)

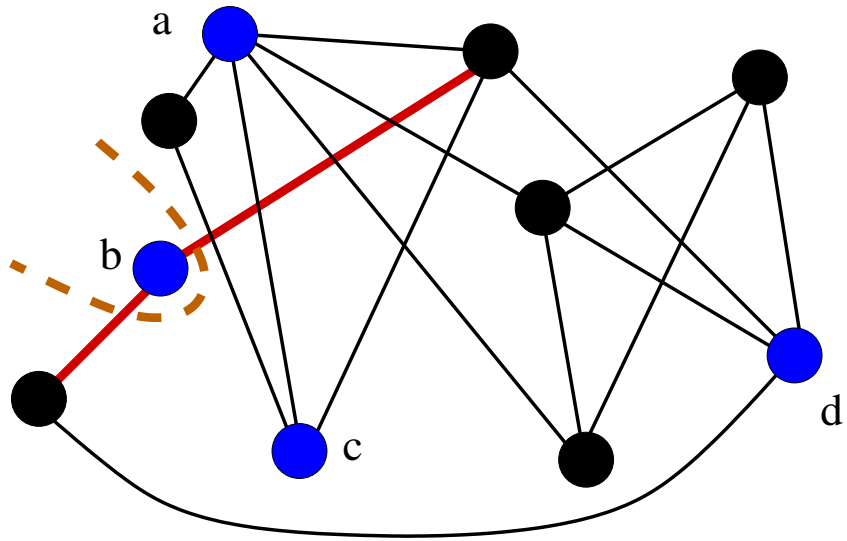


Sparsifier G'=(K,E')

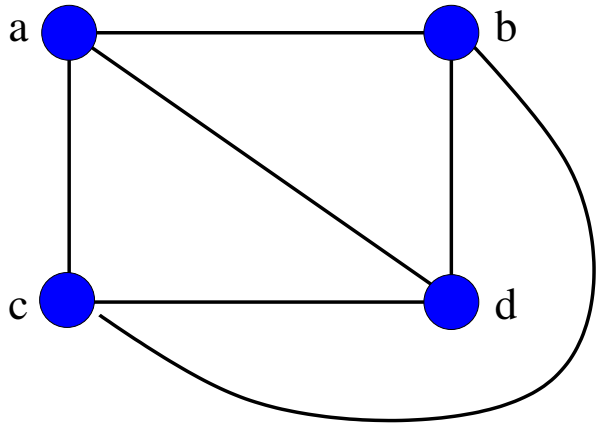


General Approach: Vertex Sparsifiers

Graph G=(V,E)



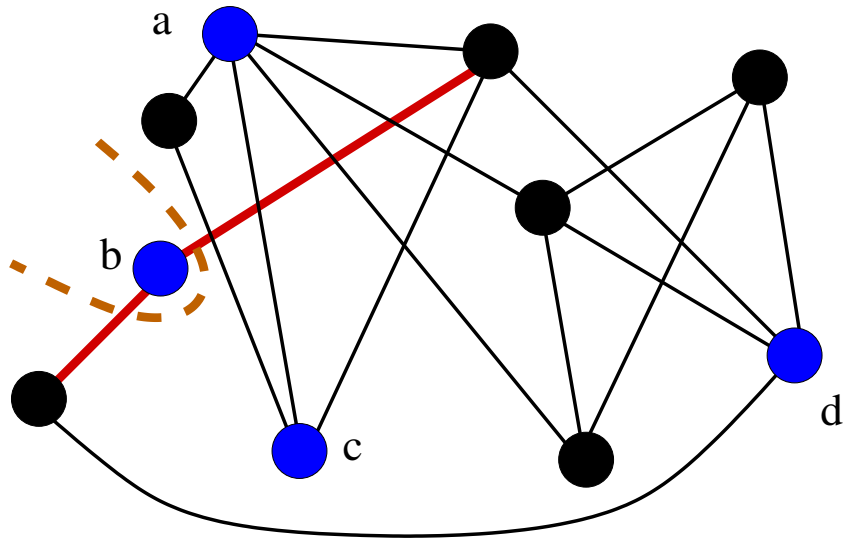
Sparsifier G'=(K,E')



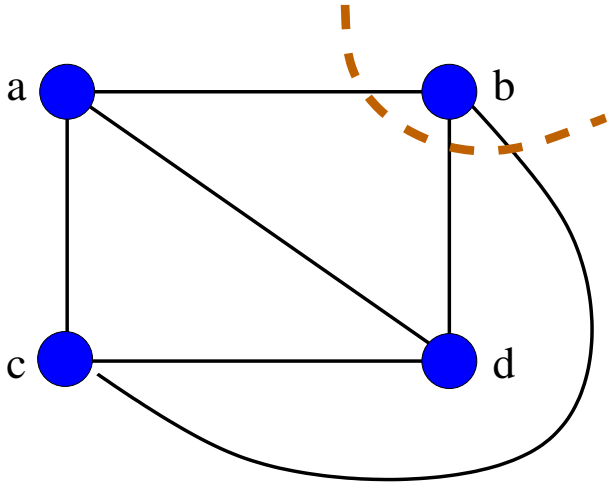
$h_K(b) = 2$

General Approach: Vertex Sparsifiers

Graph G=(V,E)



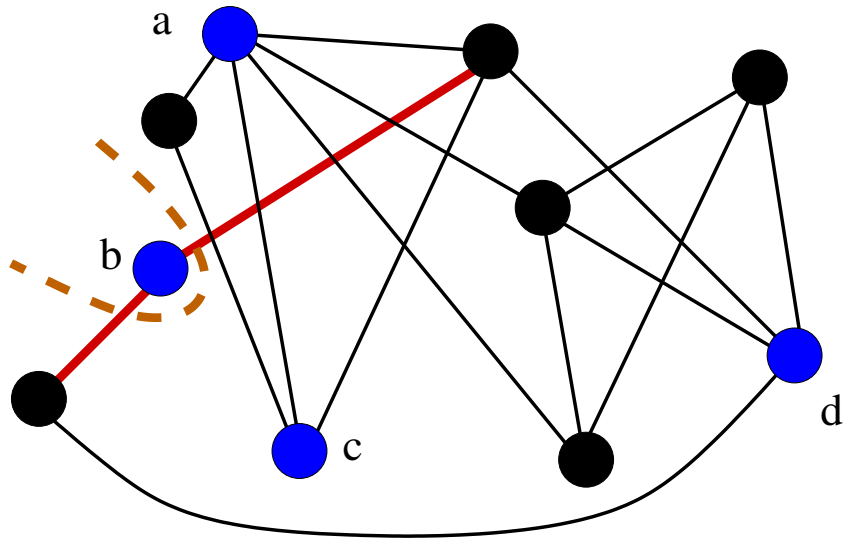
Sparsifier G'=(K,E')



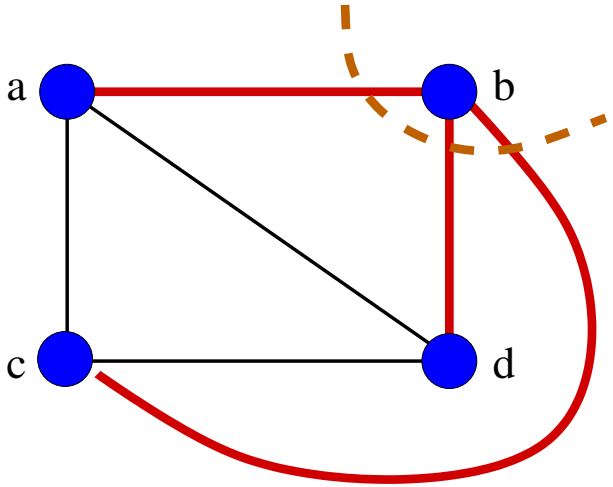
$h_K(b) = 2$

General Approach: Vertex Sparsifiers

Graph G=(V,E)



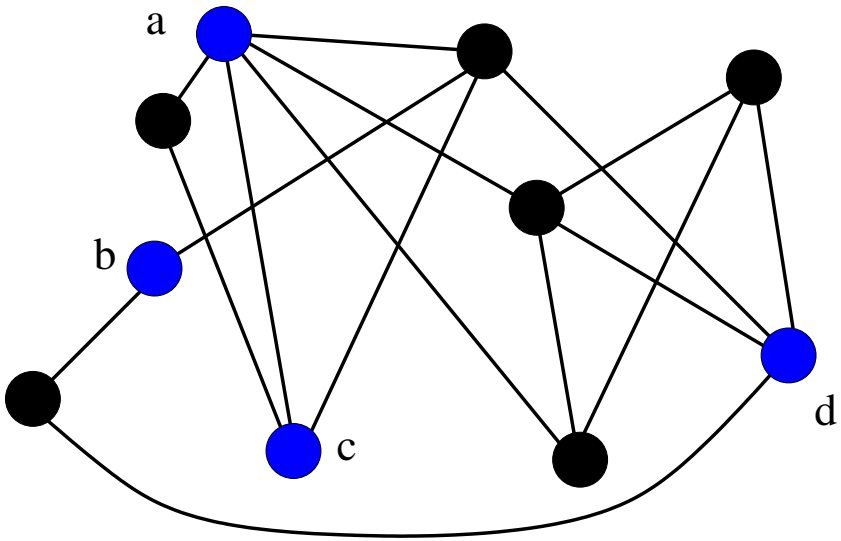
Sparsifier G'=(K,E')



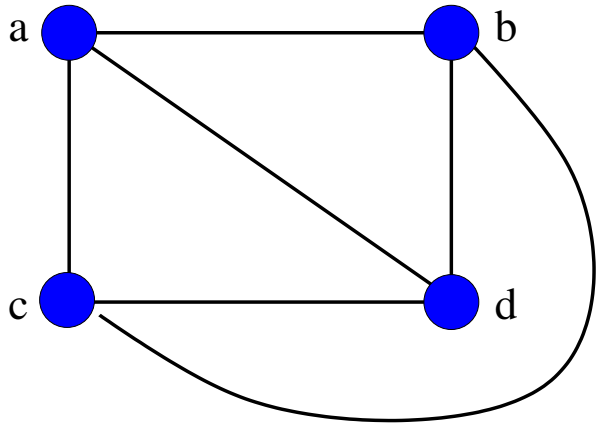
$h_K(b) = 2$

General Approach: Vertex Sparsifiers

Graph G=(V,E)

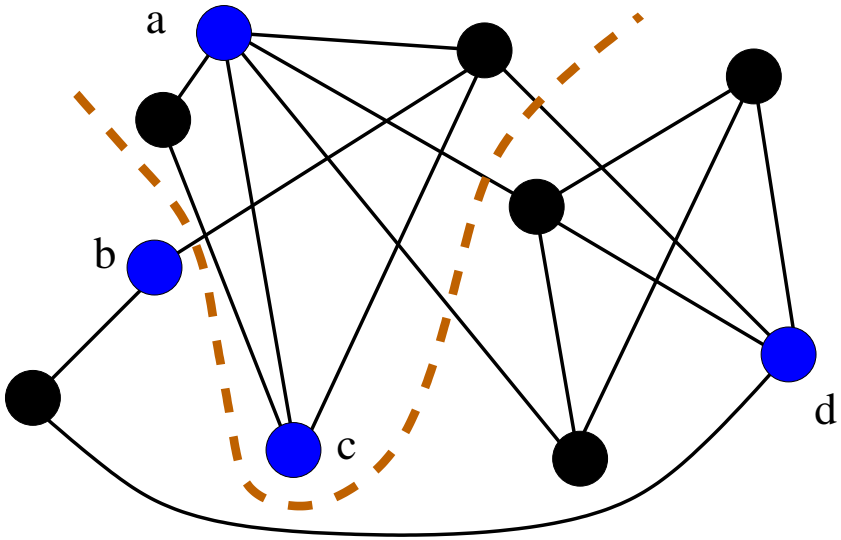


Sparsifier G'=(K,E')

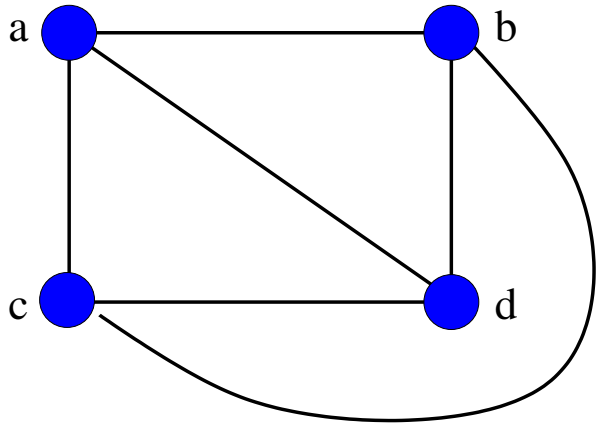


General Approach: Vertex Sparsifiers

Graph G=(V,E)

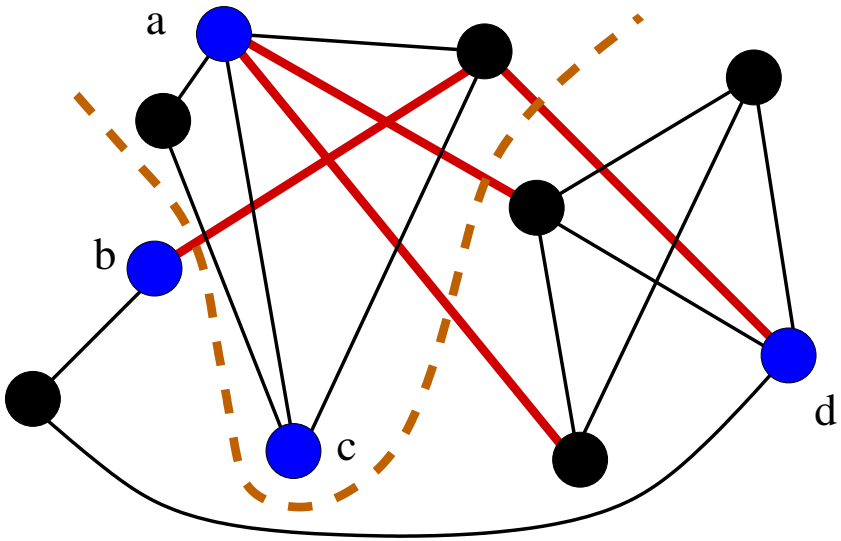


Sparsifier G'=(K,E')

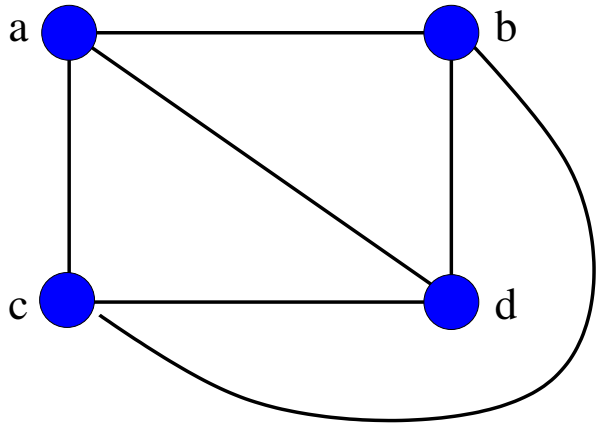


General Approach: Vertex Sparsifiers

Graph G=(V,E)

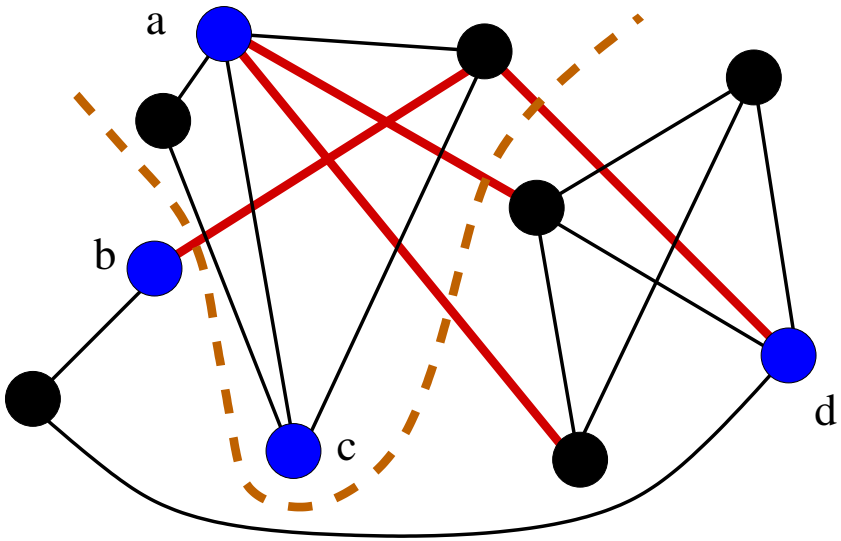


Sparsifier G'=(K,E')

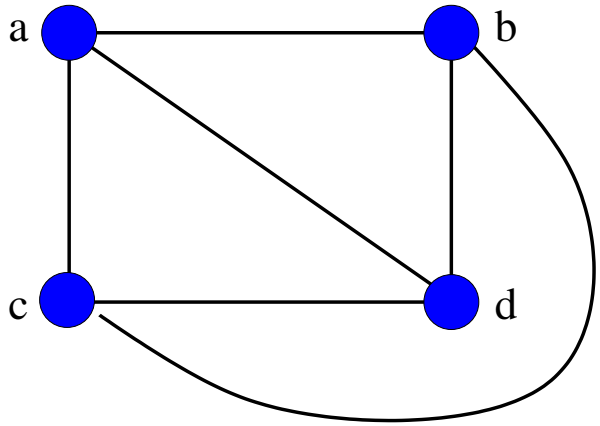


General Approach: Vertex Sparsifiers

Graph G=(V,E)



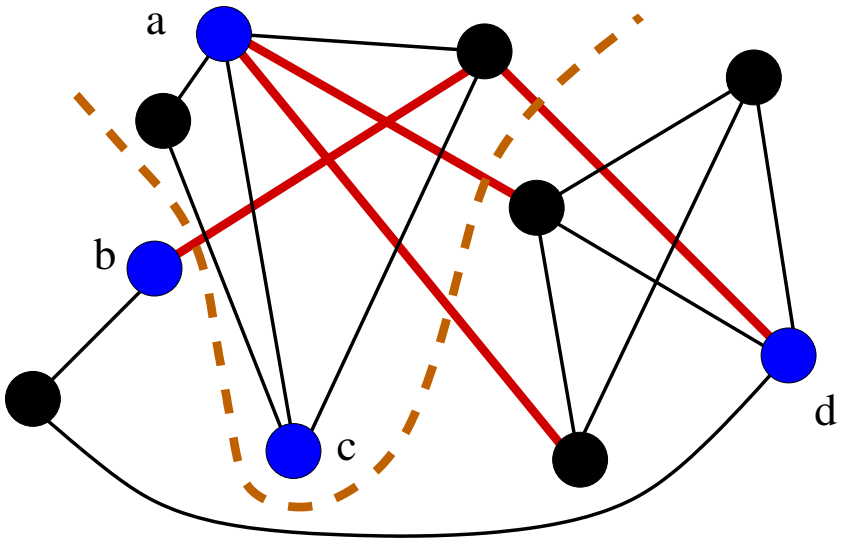
Sparsifier G'=(K,E')



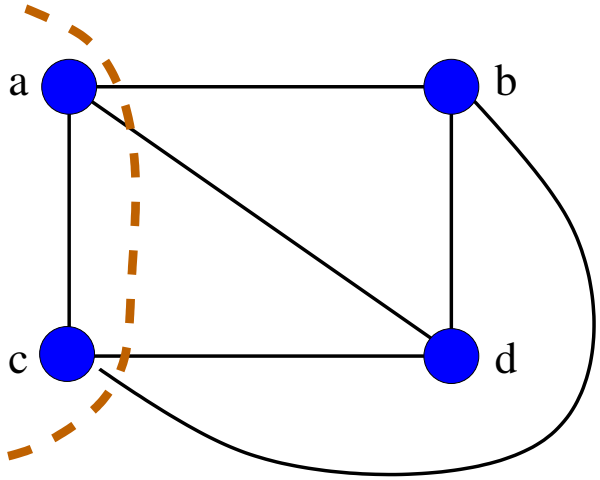
$h_K(ac) = 4$

General Approach: Vertex Sparsifiers

Graph G=(V,E)



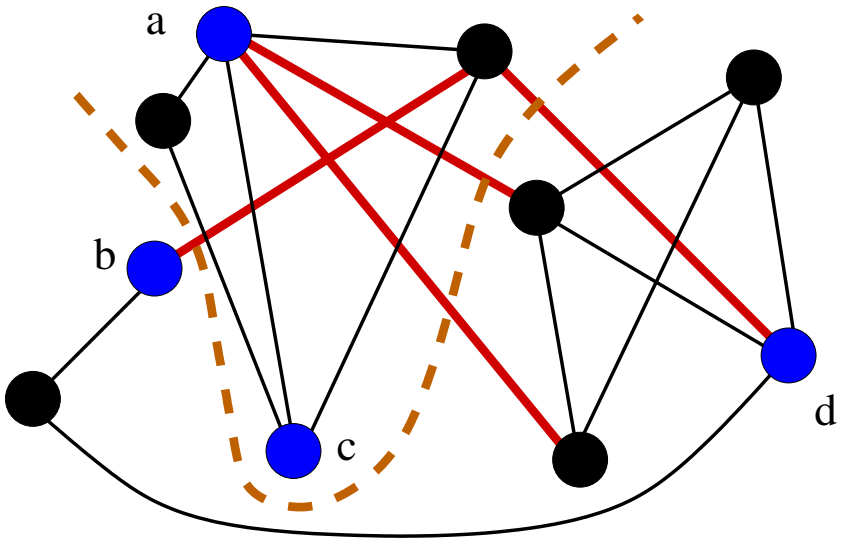
Sparsifier G'=(K,E')



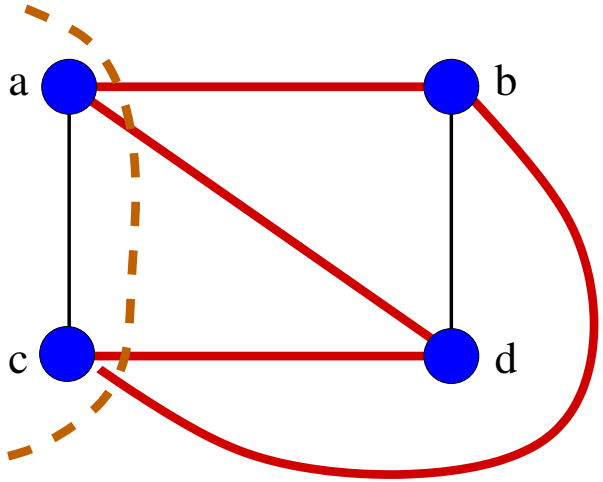
$h_K(ac) = 4$

General Approach: Vertex Sparsifiers

Graph G=(V,E)



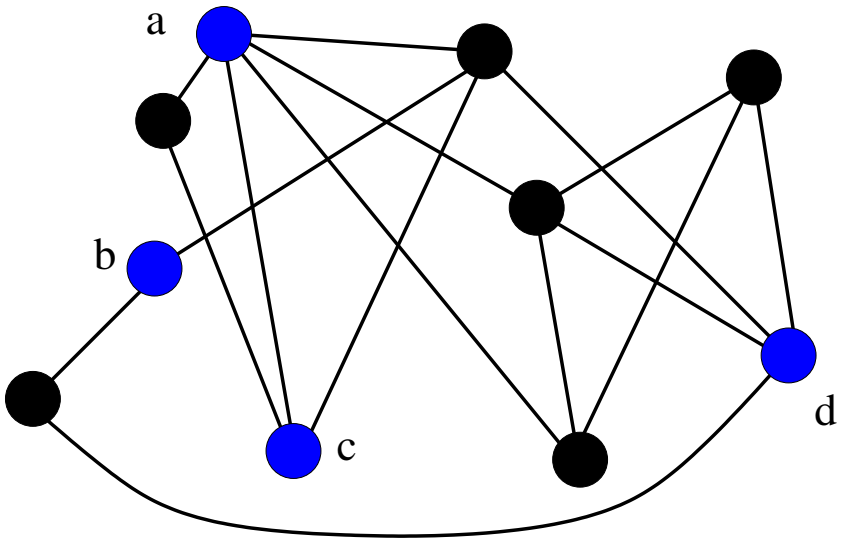
Sparsifier G'=(K,E')



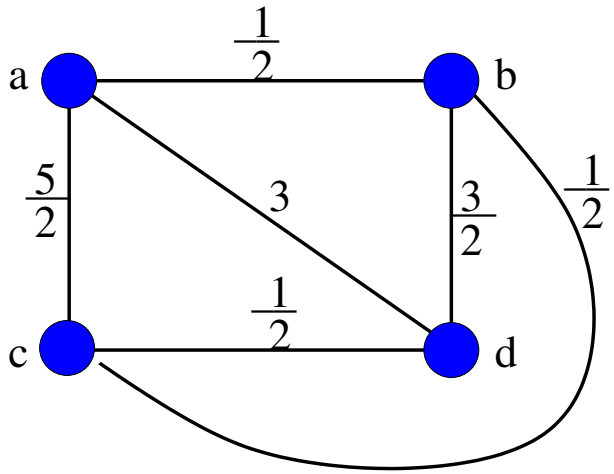
$h_K(ac) = 4$

General Approach: Vertex Sparsifiers

Graph G=(V,E)

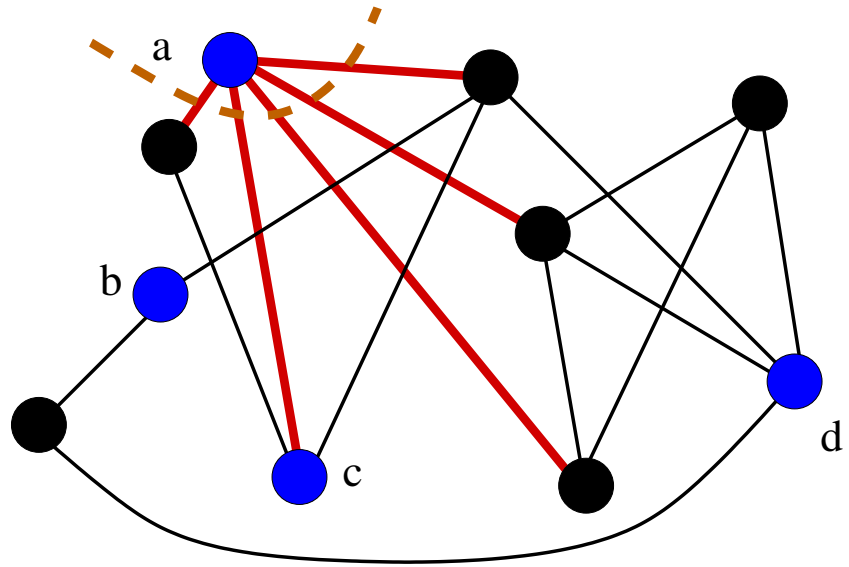


Sparsifier G'=(K,E')



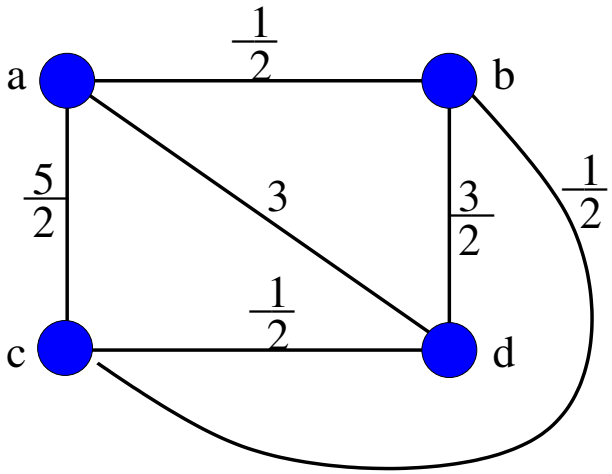
General Approach: Vertex Sparsifiers

Graph G=(V,E)



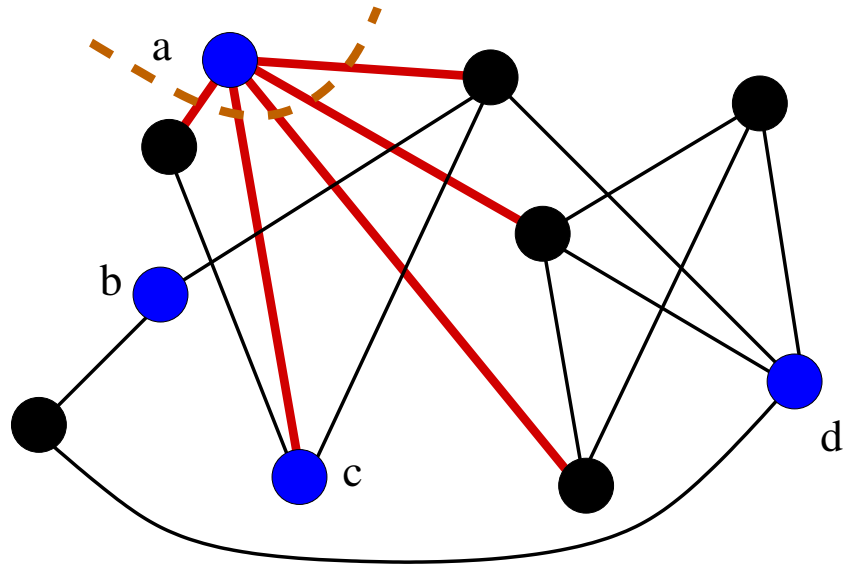
$h_K(a) = 5$

Sparsifier G'=(K,E')



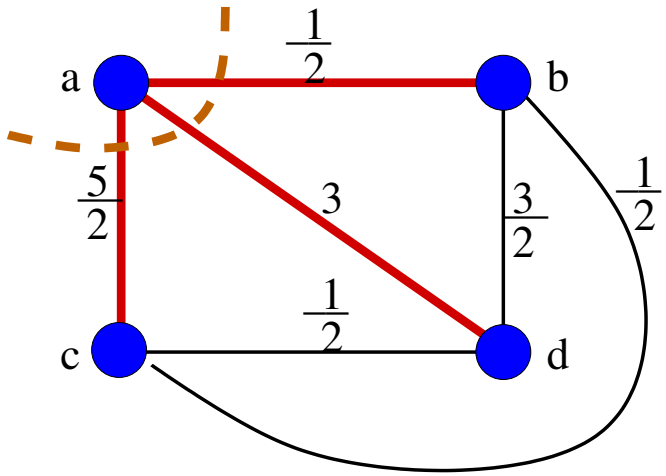
General Approach: Vertex Sparsifiers

Graph G=(V,E)



$h_K(a) = 5$

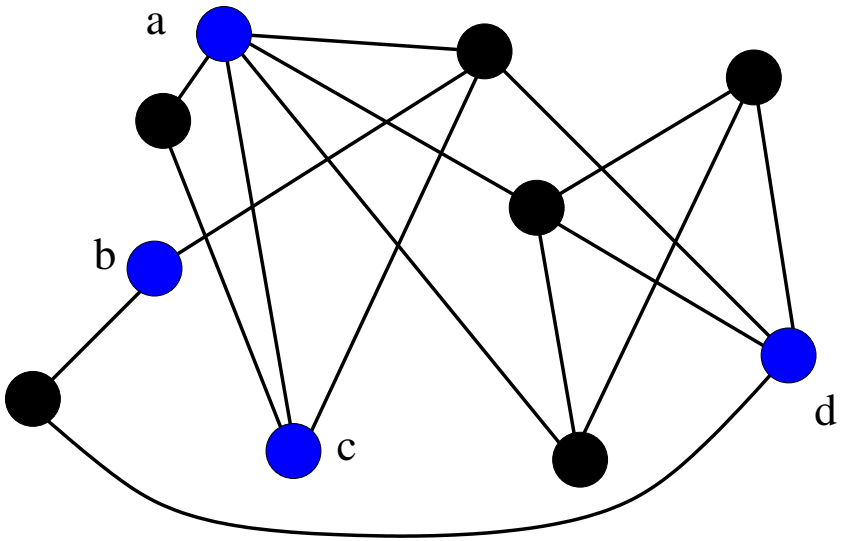
Sparsifier G'=(K,E')



$h'(a) = 6$

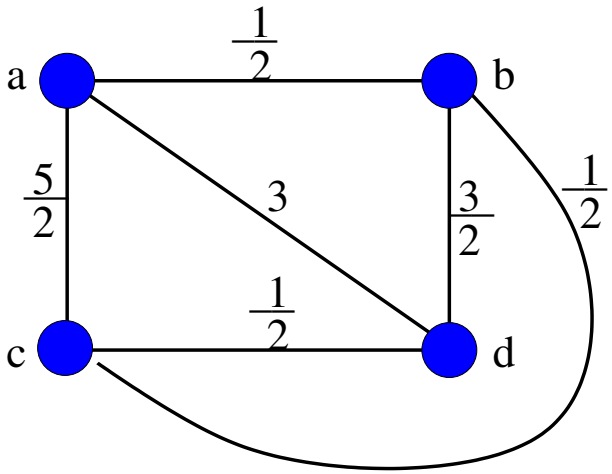
General Approach: Vertex Sparsifiers

Graph G=(V,E)



$h_K(a) = 5$

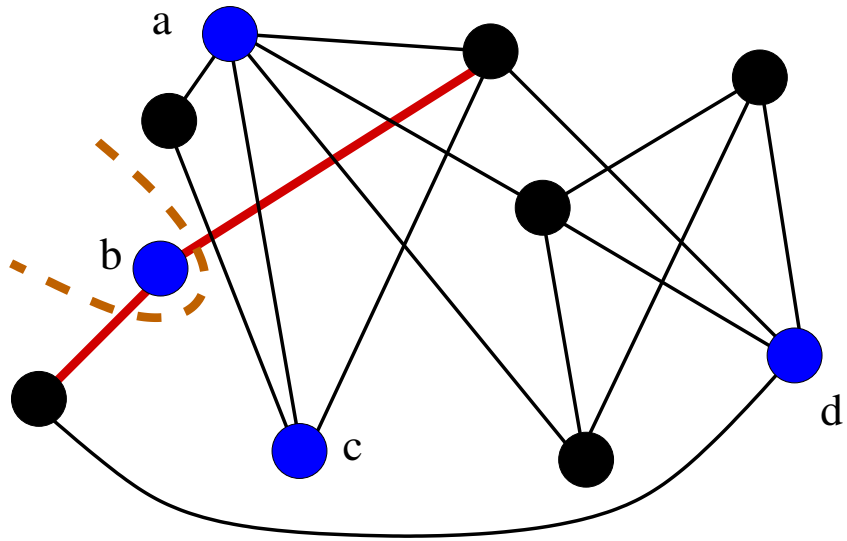
Sparsifier G'=(K,E')



$h'(a) = 6$

General Approach: Vertex Sparsifiers

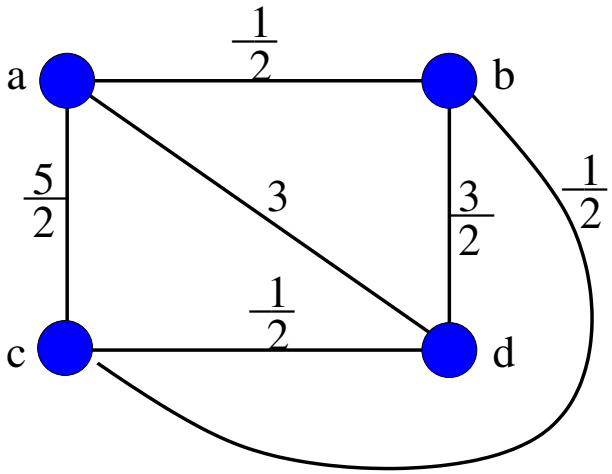
Graph G=(V,E)



$h_K(a) = 5$

$h_K(b) = 2$

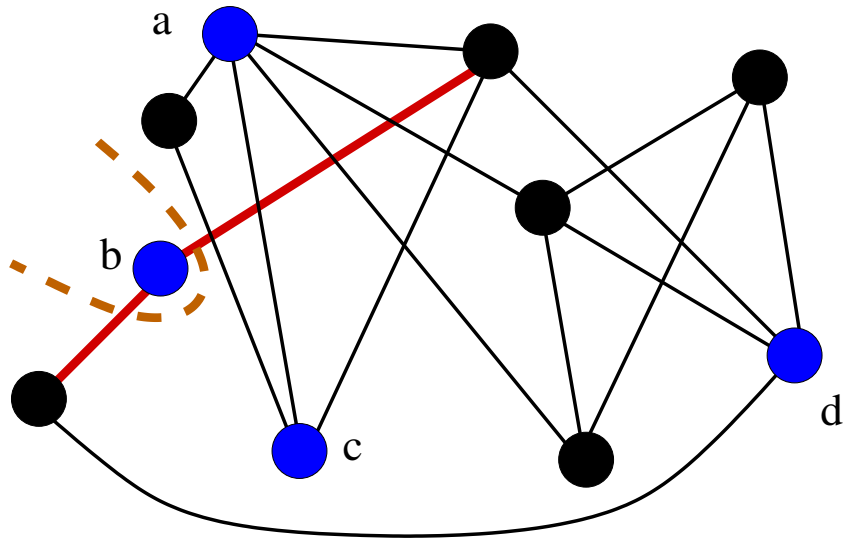
Sparsifier G'=(K,E')



$h'(a) = 6$

General Approach: Vertex Sparsifiers

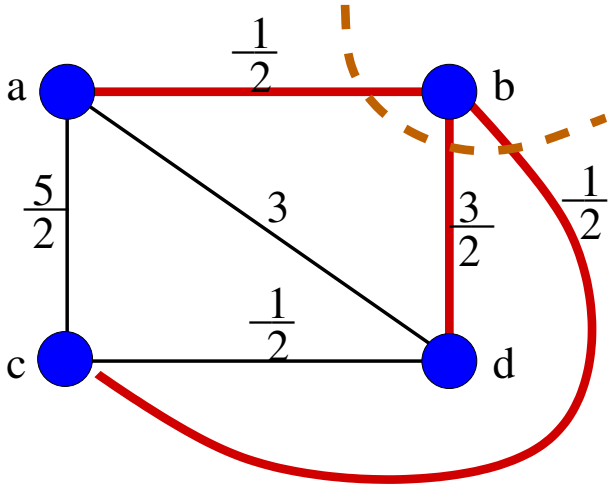
Graph G=(V,E)



$h_K(a) = 5$

$h_K(b) = 2$

Sparsifier G'=(K,E')

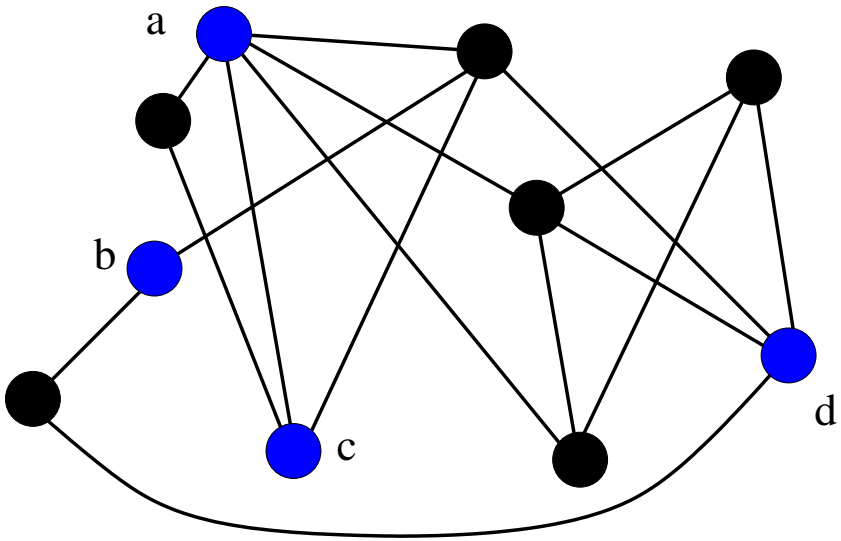


$h'(a) = 6$

$h'(b) = 2.5$

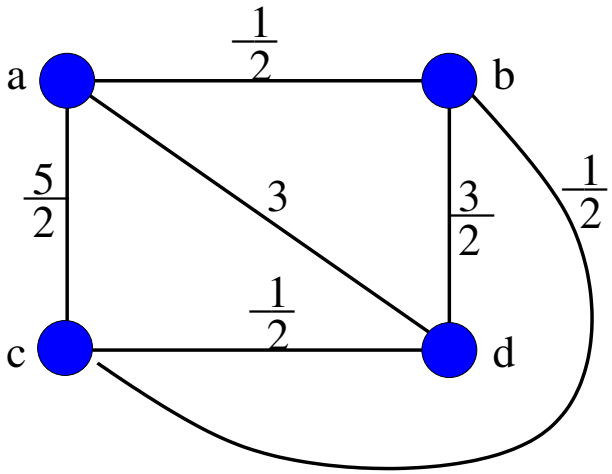
General Approach: Vertex Sparsifiers

Graph G=(V,E)



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

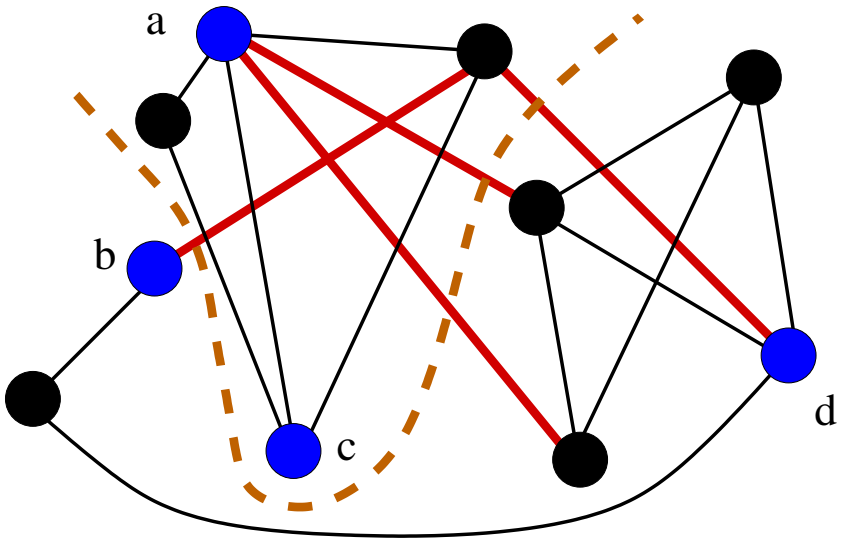
Sparsifier G'=(K,E')



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

General Approach: Vertex Sparsifiers

Graph G=(V,E)

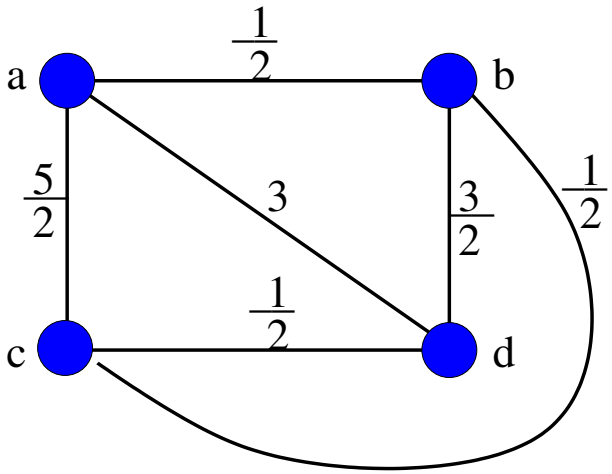


$h_K(a) = 5$

$h_K(b) = 2$

$h_K(ac) = 4$

Sparsifier G'=(K,E')

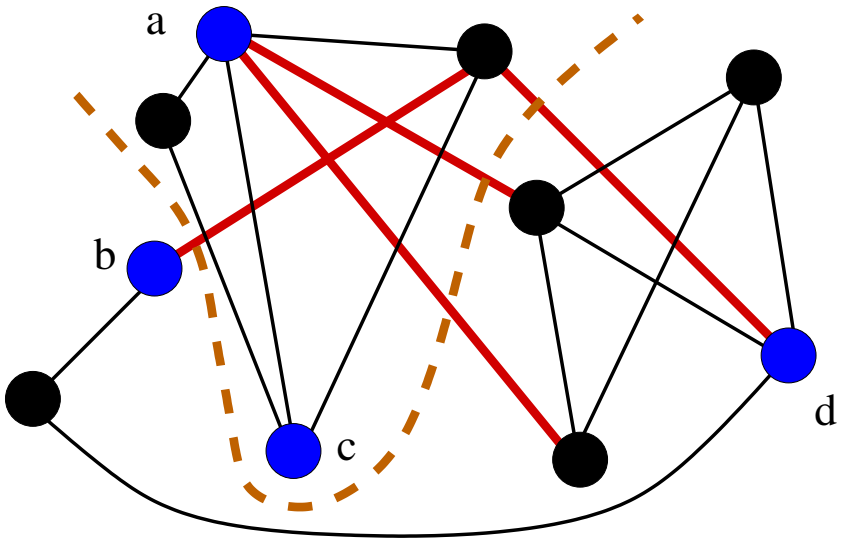


$h'(a) = 6$

$h'(b) = 2.5$

General Approach: Vertex Sparsifiers

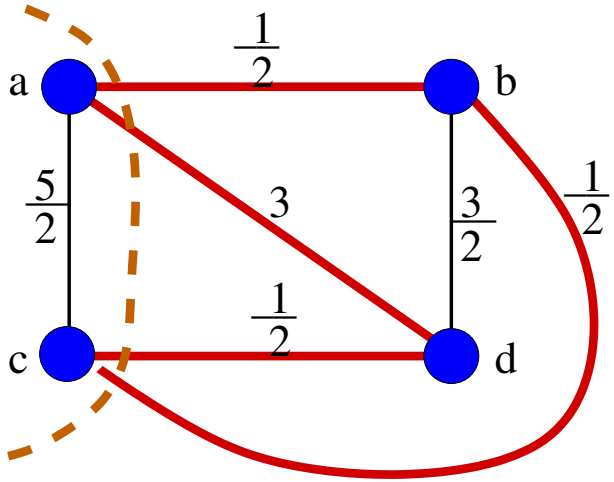
Graph G=(V,E)



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

$h_K(ac) = 4$

Sparsifier G'=(K,E')

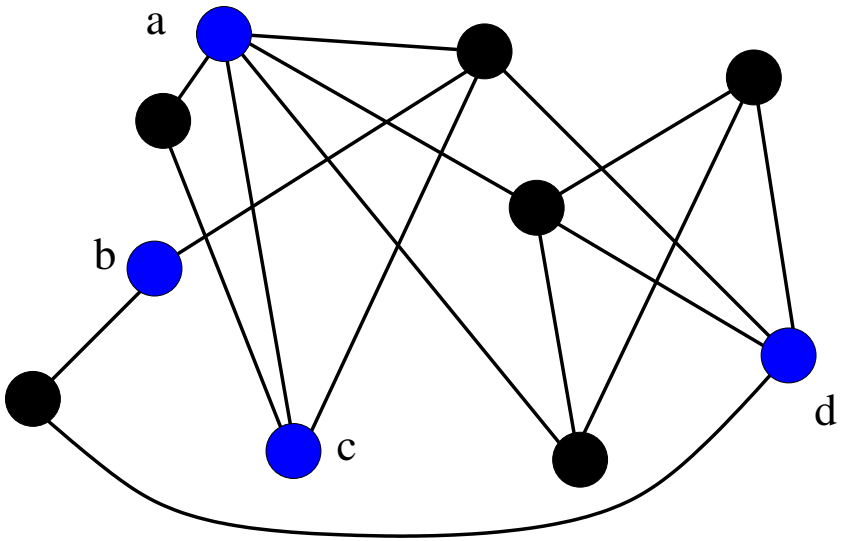


$h'(a) = 6$
$h'(b) = 2.5$

$h'(ac) = 4.5$

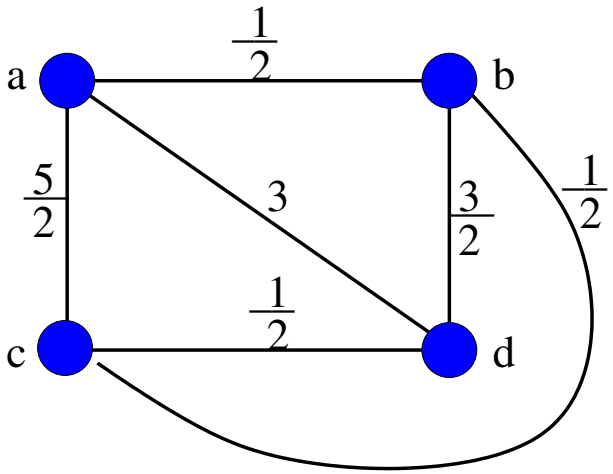
General Approach: Vertex Sparsifiers

Graph G=(V,E)



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

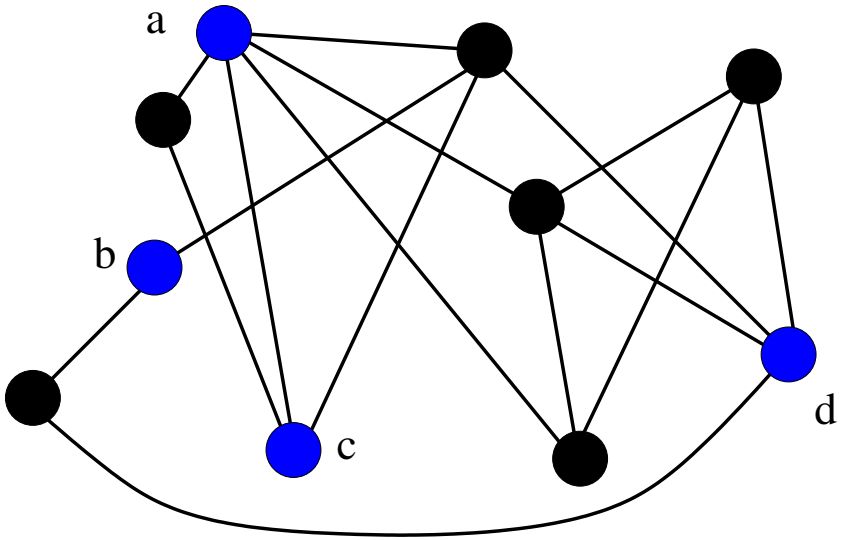
Sparsifier G'=(K,E')

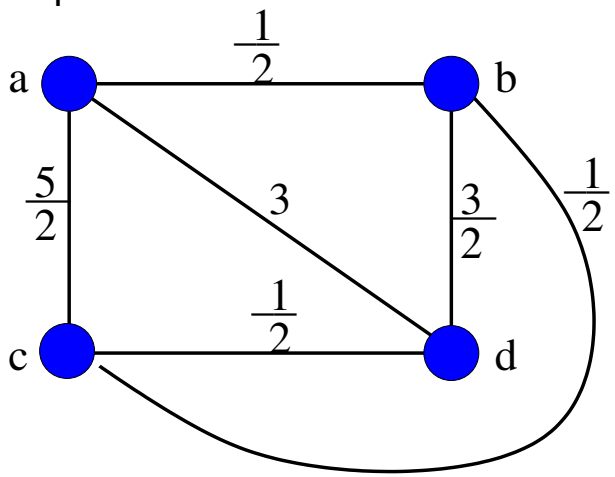


$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$
$h'(b) = 2.5$ $h'(ab) = 7.5$
$h'(c) = 3.5$ $h'(ac) = 4.5$

General Approach: Vertex Sparsifiers

The Quality of the Sparsifer is $\frac{5}{4}$





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

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

Vertex Sparsifiers, Informally

Definition

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

Vertex Sparsifiers, Informally

Definition

$G' = (K, E')$ is a **Vertex 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 Vertex Sparsifier is the maximum ratio of a cut in G' to the corresponding min-cut in G.

Vertex Sparsifiers

Good Vertex Sparsifiers exist!

Vertex Sparsifiers

Good Vertex Sparsifiers exist!

And can be computed in polynomial time!

Vertex Sparsifiers

Good Vertex Sparsifiers exist!

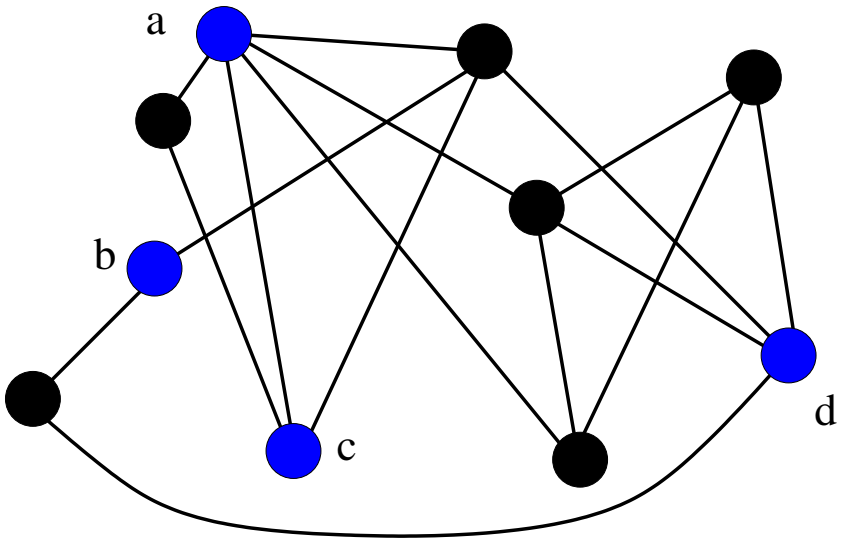
And can be computed in polynomial time!

Theorem

For all weighted graphs $G = (V, E)$, and all $K \subset V$ there is a weighted graph $G' = (K, E')$ such that G' is a poly$(\log k)$-quality Vertex Sparsifier and such a graph can be computed in time polynomial in n and k.

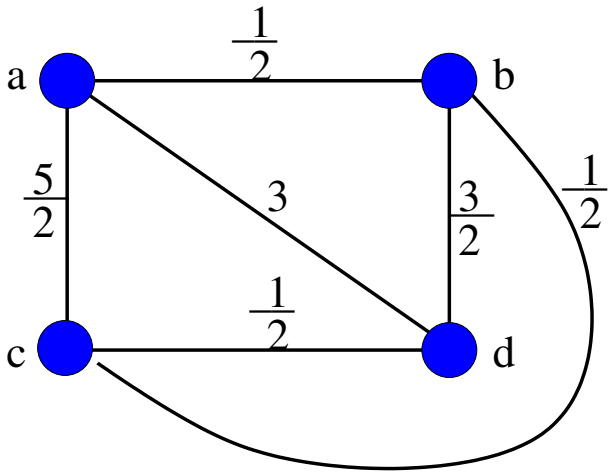
An Application to Steiner Minimum Bisection

Graph G=(V,E)



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

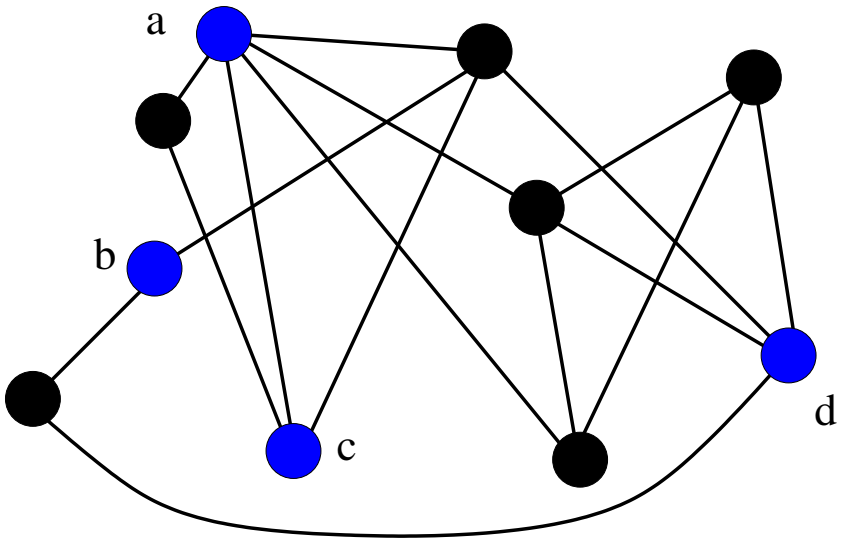
Sparsifier G'=(K,E')



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

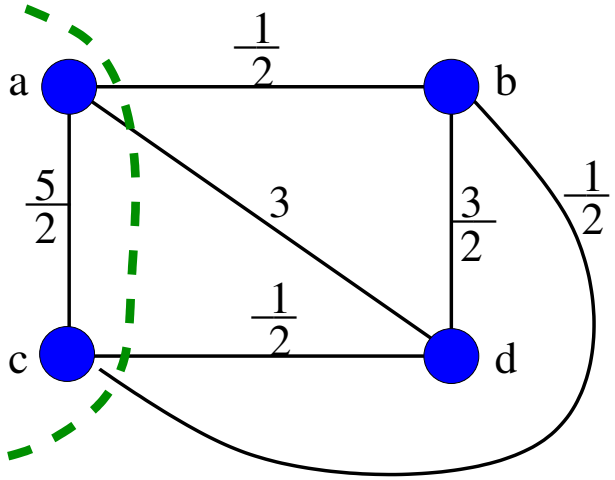
An Application to Steiner Minimum Bisection

Graph G=(V,E)



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

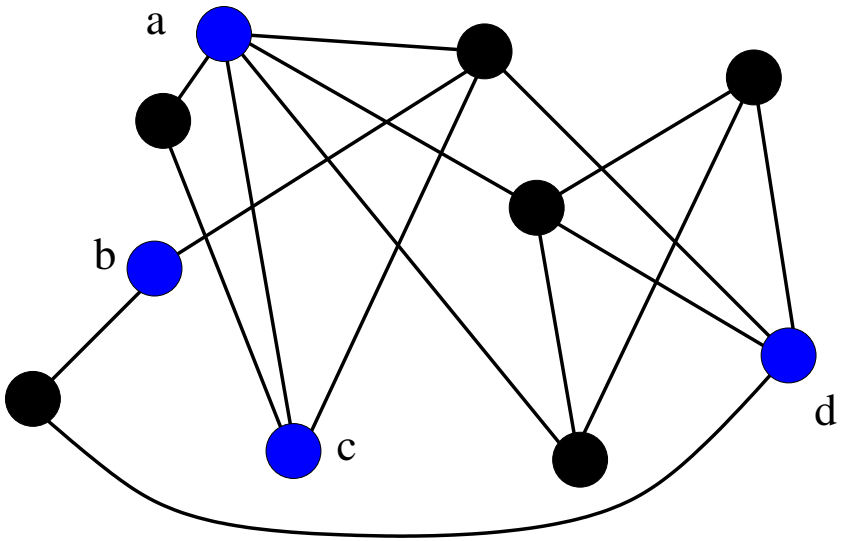
Sparsifier G'=(K,E')



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

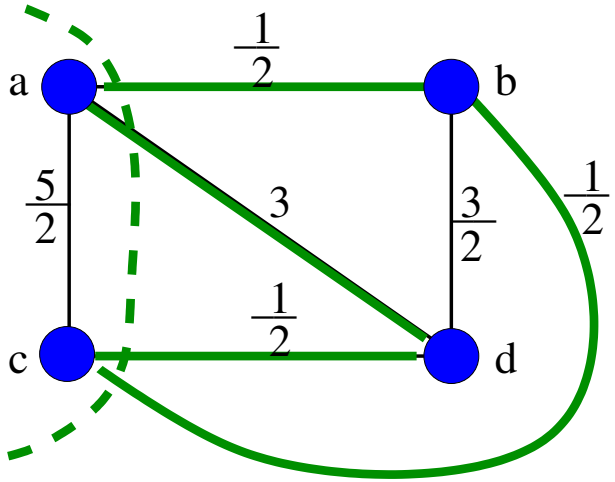
An Application to Steiner Minimum Bisection

Graph G=(V,E)



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

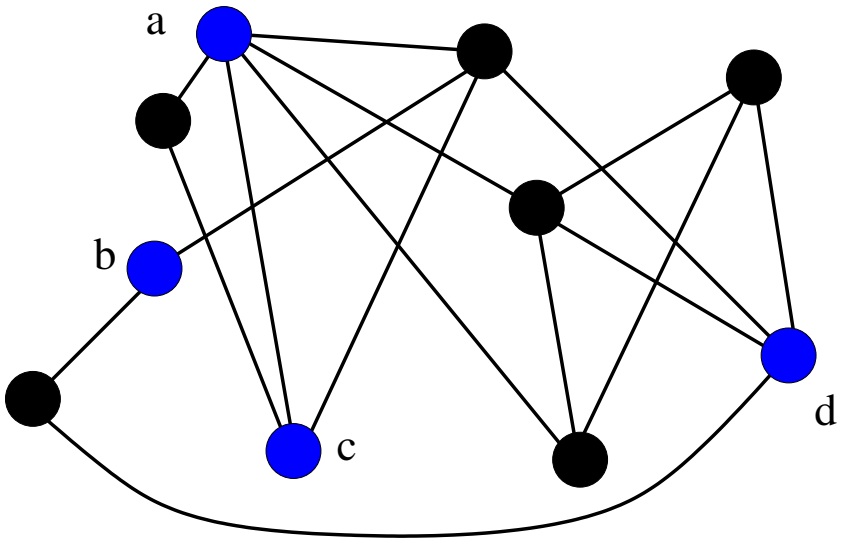
Sparsifier G'=(K,E')



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

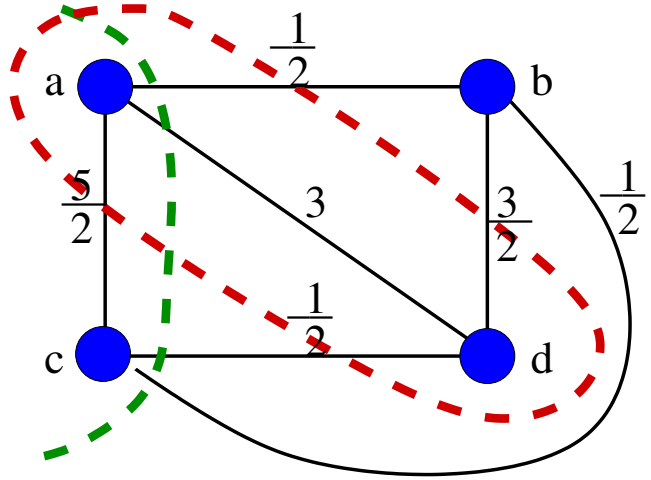
An Application to Steiner Minimum Bisection

Graph G=(V,E)



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

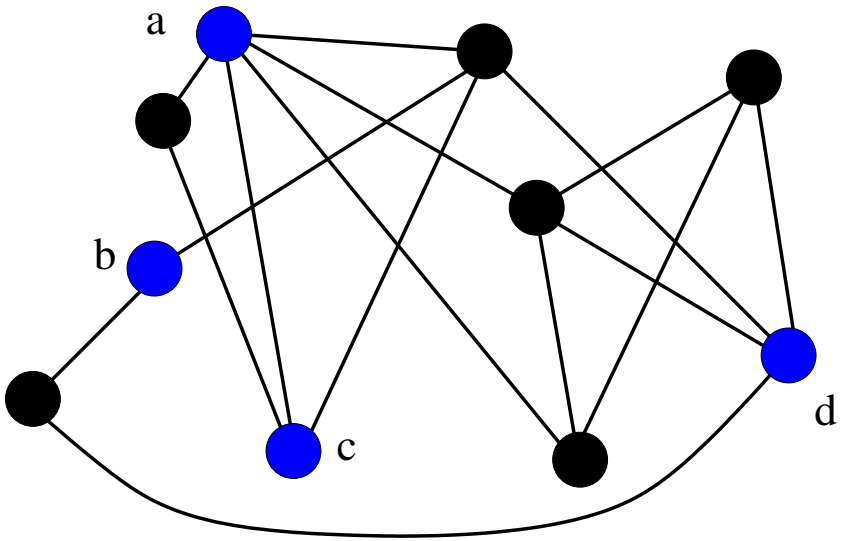
Sparsifier G'=(K,E')



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

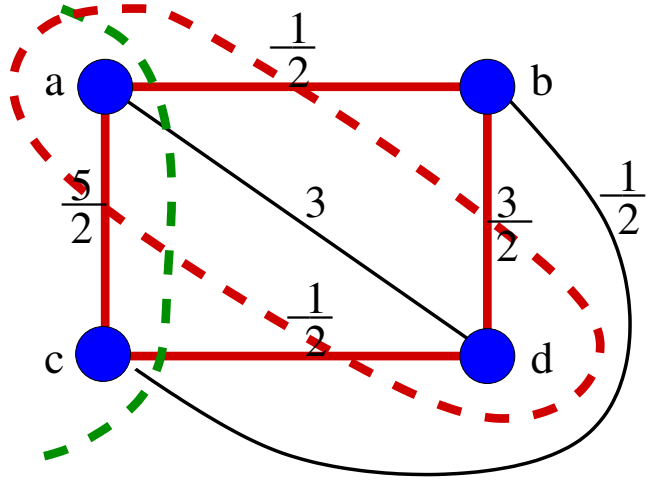
An Application to Steiner Minimum Bisection

Graph G=(V,E)



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

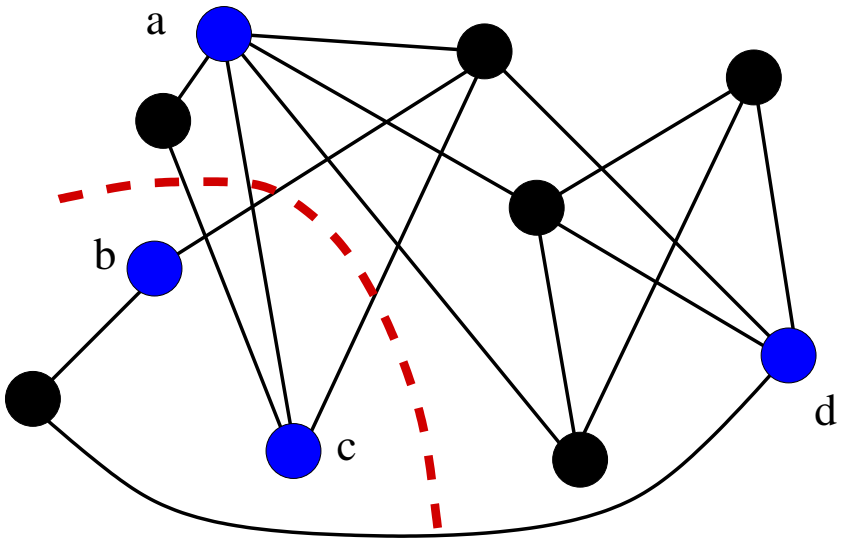
Sparsifier G'=(K,E')



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)

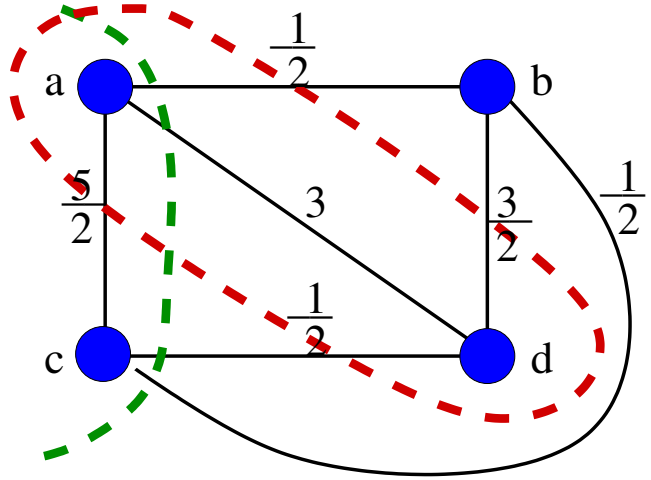


$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$

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

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

Sparsifier G'=(K,E')



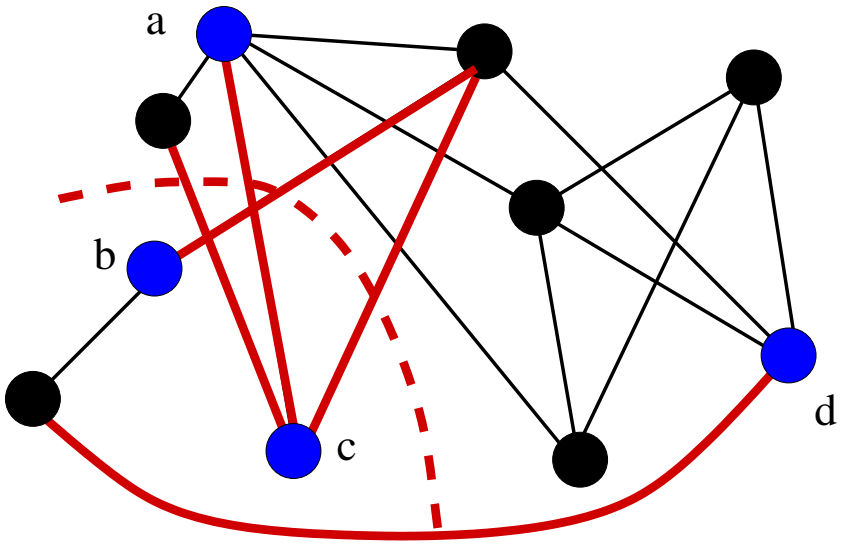
$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$

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

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

An Application to Steiner Minimum Bisection

Graph G=(V,E)

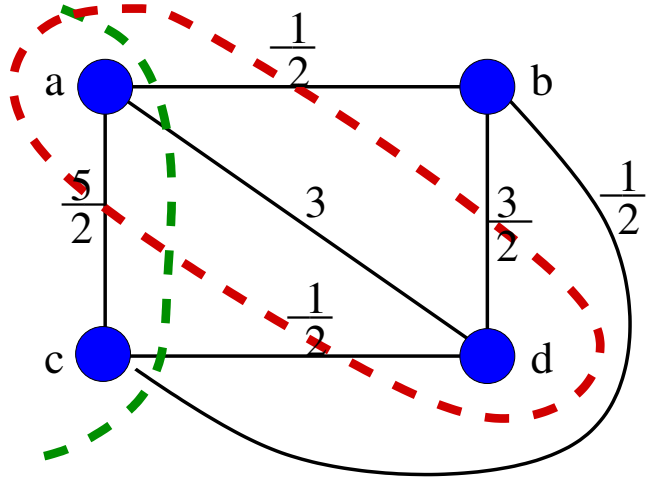


$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$

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

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

Sparsifier G'=(K,E')



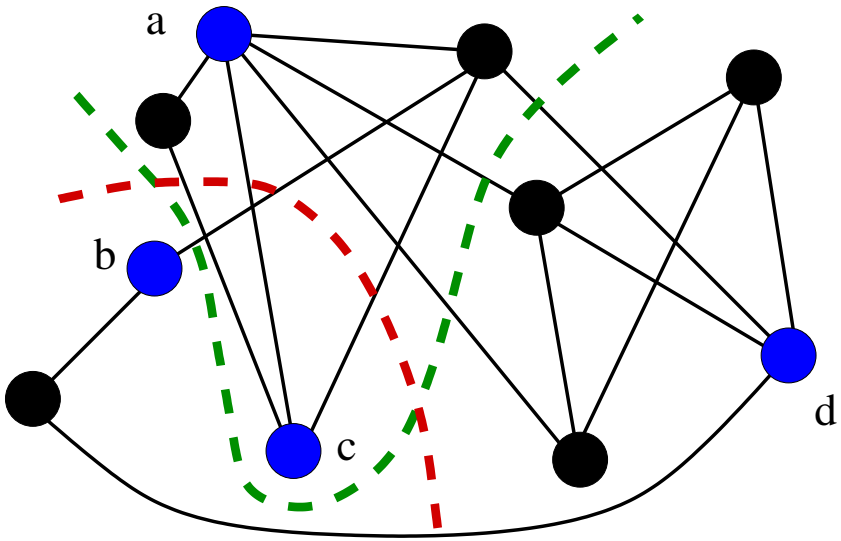
$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$

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

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

An Application to Steiner Minimum Bisection

Graph G=(V,E)

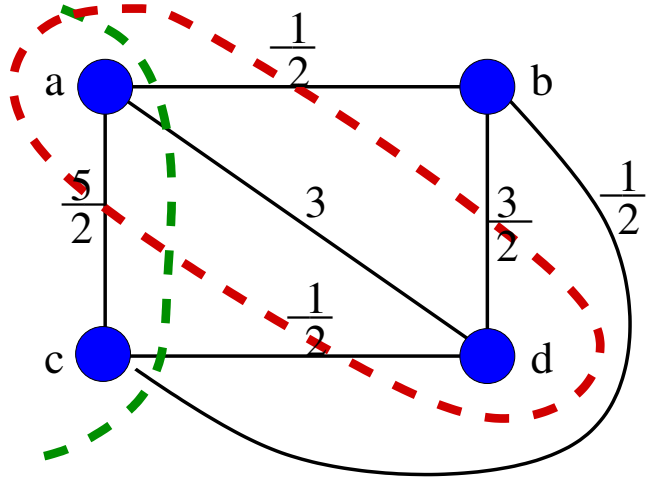


$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$

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

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

Sparsifier G'=(K,E')



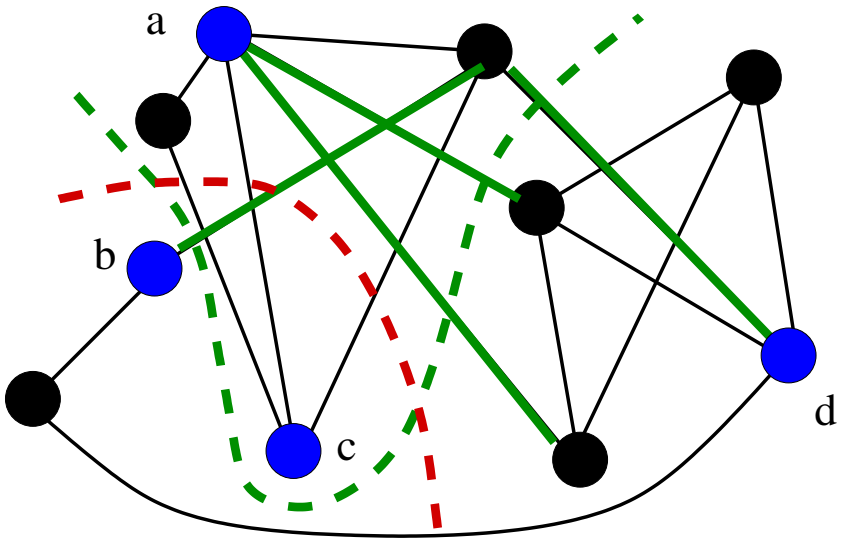
$h'(a) = 6$ $h'(d) = 5$ $h'(ad) = 5$

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

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

An Application to Steiner Minimum Bisection

Graph G=(V,E)

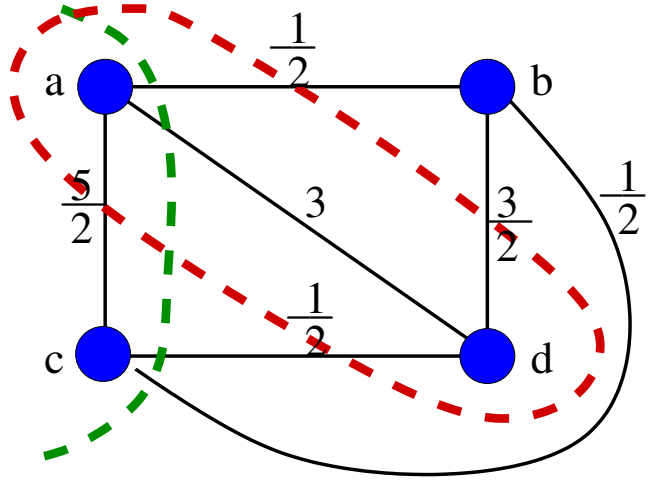


$h_K(a) = 5$ $h_K(d) = 4$ $h_K(ad) = 5$

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

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

Sparsifier G'=(K,E')



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

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

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

Reducibility in Multicommodity-Type Problems

This is a general strategy!

Reducibility in Multicommodity-Type Problems

This is a general strategy!

Reducibility for Multicommodity-Type Problems:

Reducibility in Multicommodity-Type Problems

This is a general strategy!

Reducibility for Multicommodity-Type Problems:

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

Reducibility in Multicommodity-Type Problems

This is a general strategy!

Reducibility for Multicommodity-Type Problems:

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

Reducibility in Multicommodity-Type Problems

This is a general strategy!

Reducibility for Multicommodity-Type Problems:

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

Reducibility in Multicommodity-Type Problems

This is a general strategy!

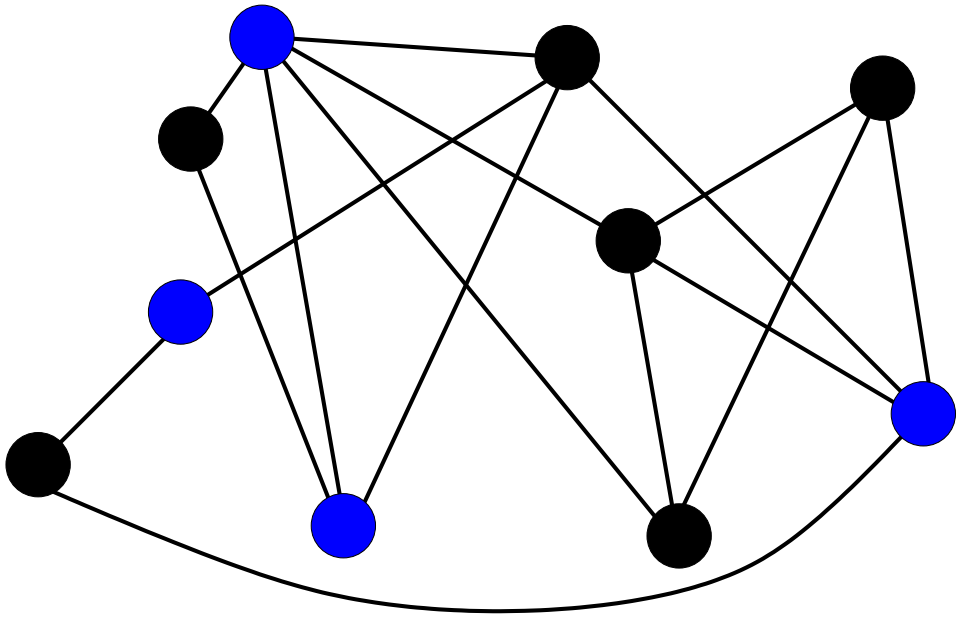
Reducibility for Multicommodity-Type Problems:

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

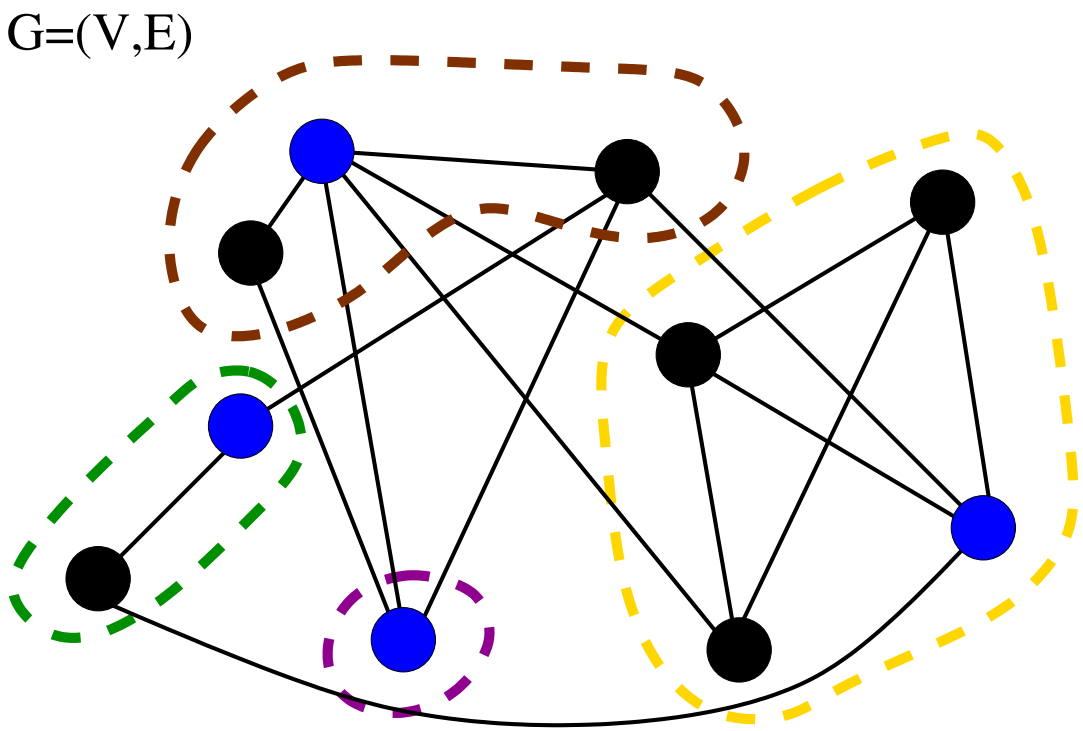
Informally: If the problem is Lipschitz w.r.t. h_K, then this is a general reduction!

Definition

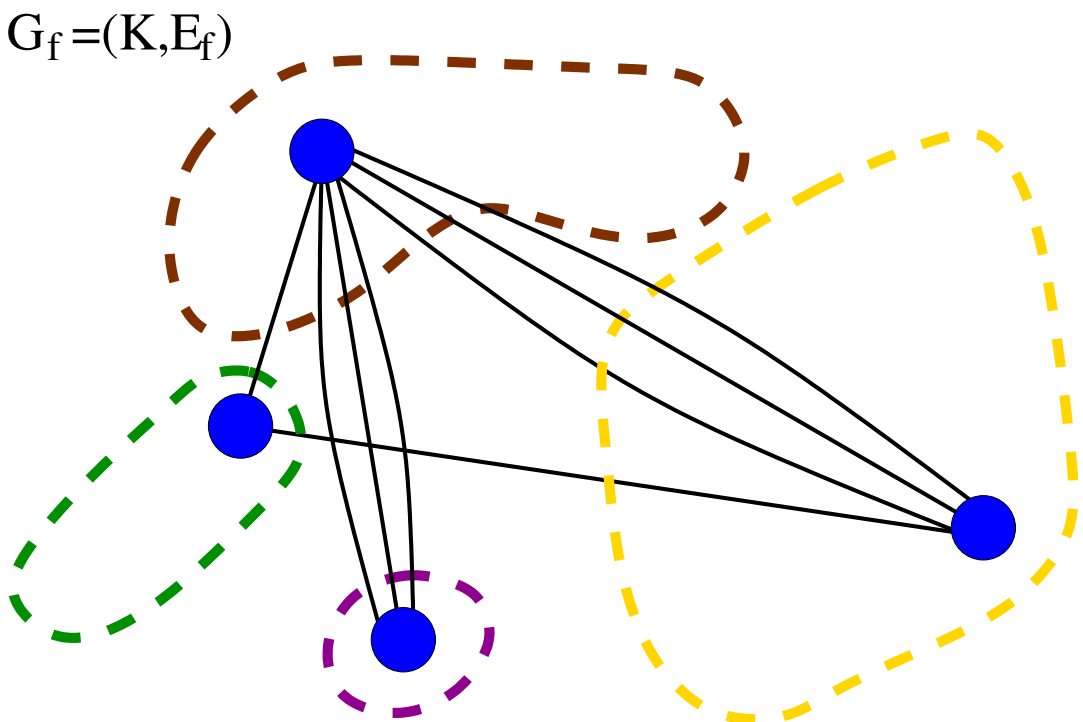
G=(V,E)



Definition

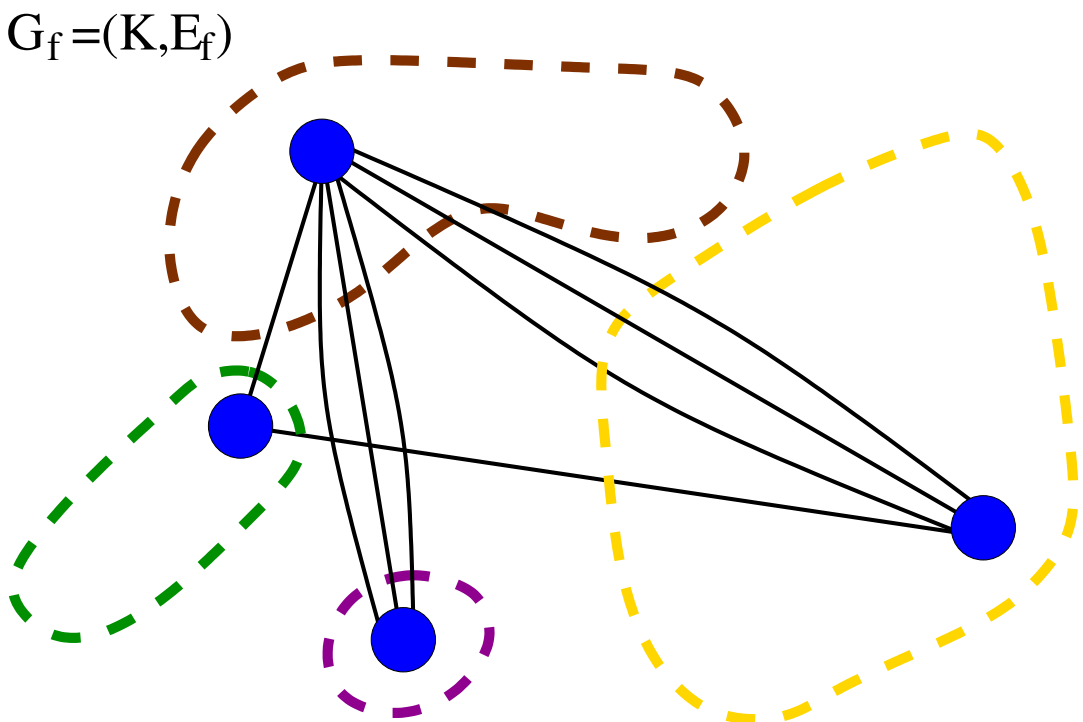


Definition



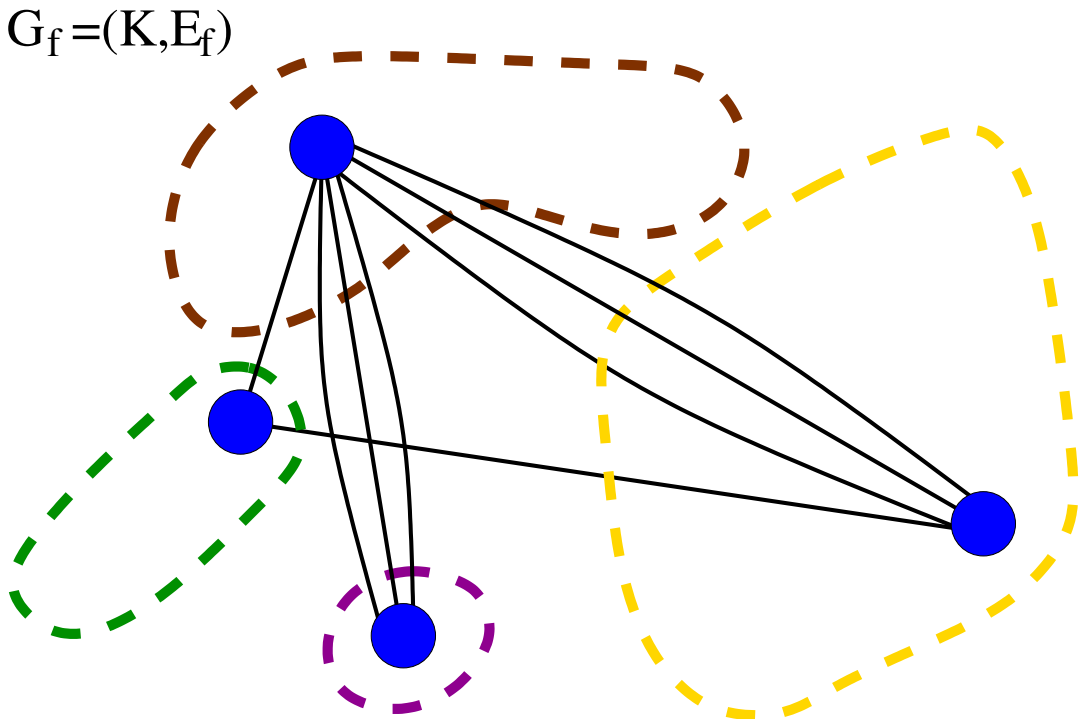
Definition

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



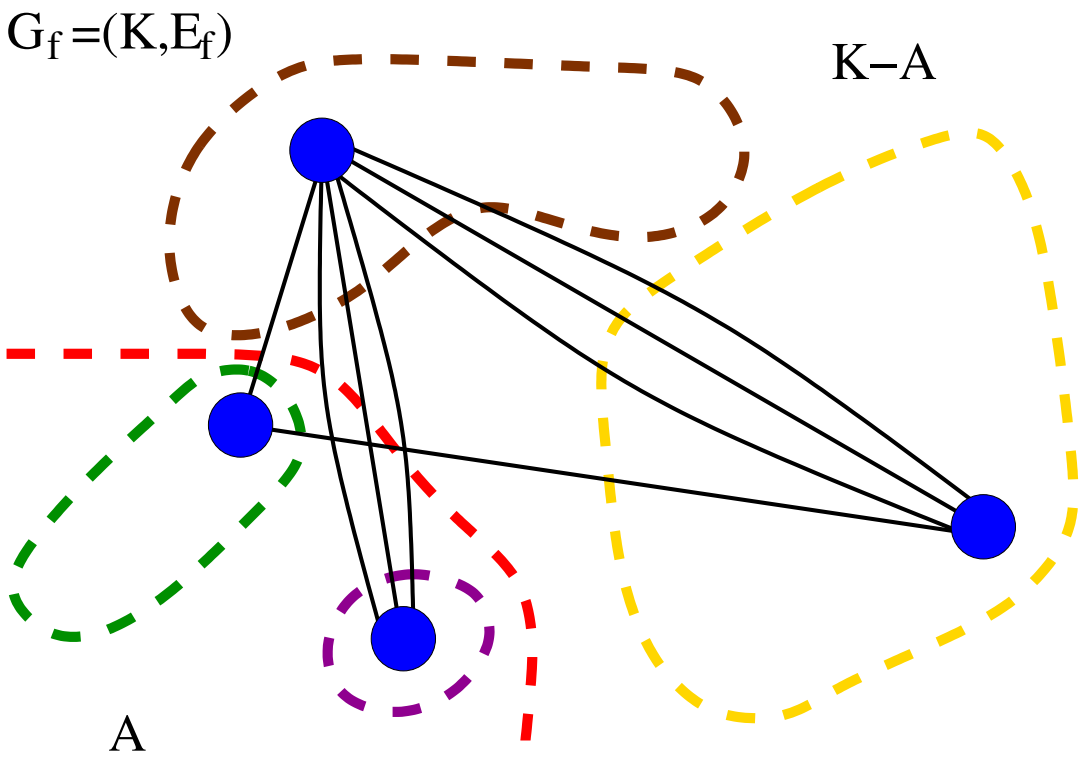
Lemma

G_f is a vertex sparsifier



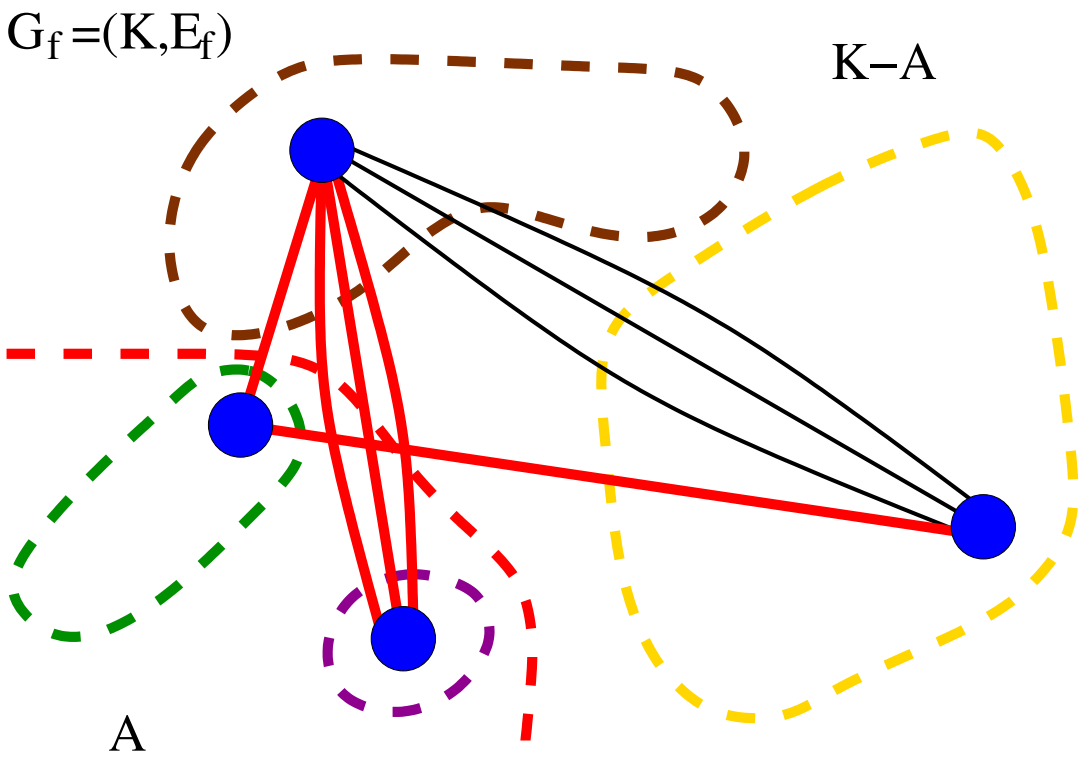
Lemma

G_f is a vertex sparsifier



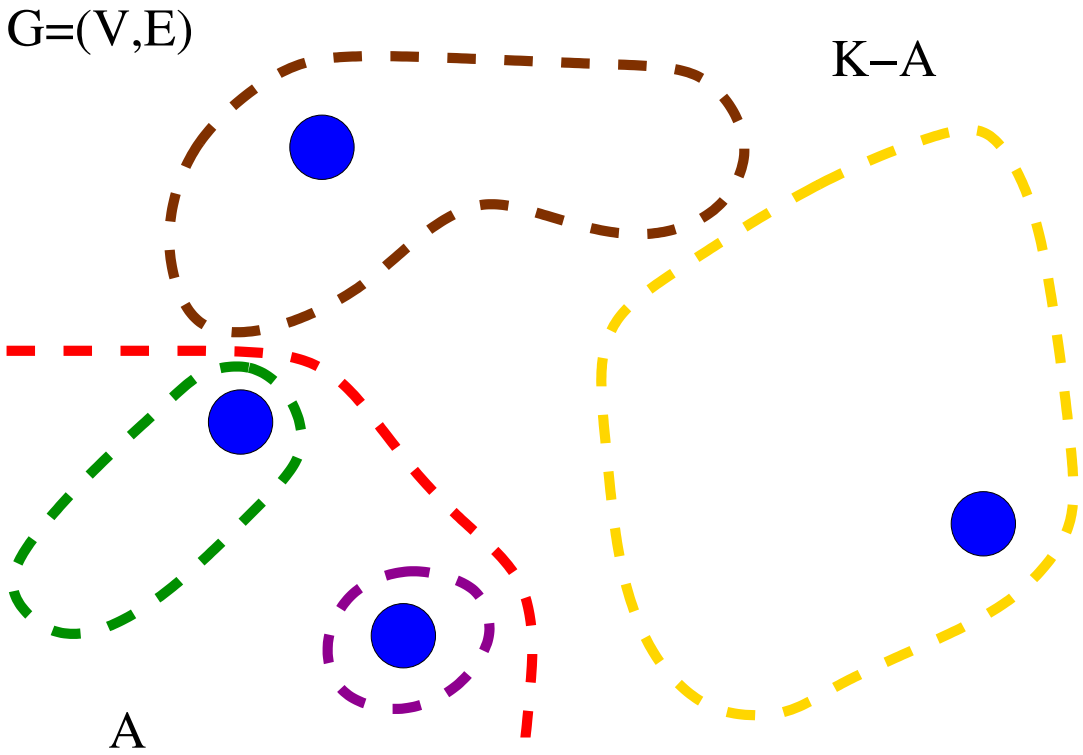
Lemma

G_f is a vertex sparsifier



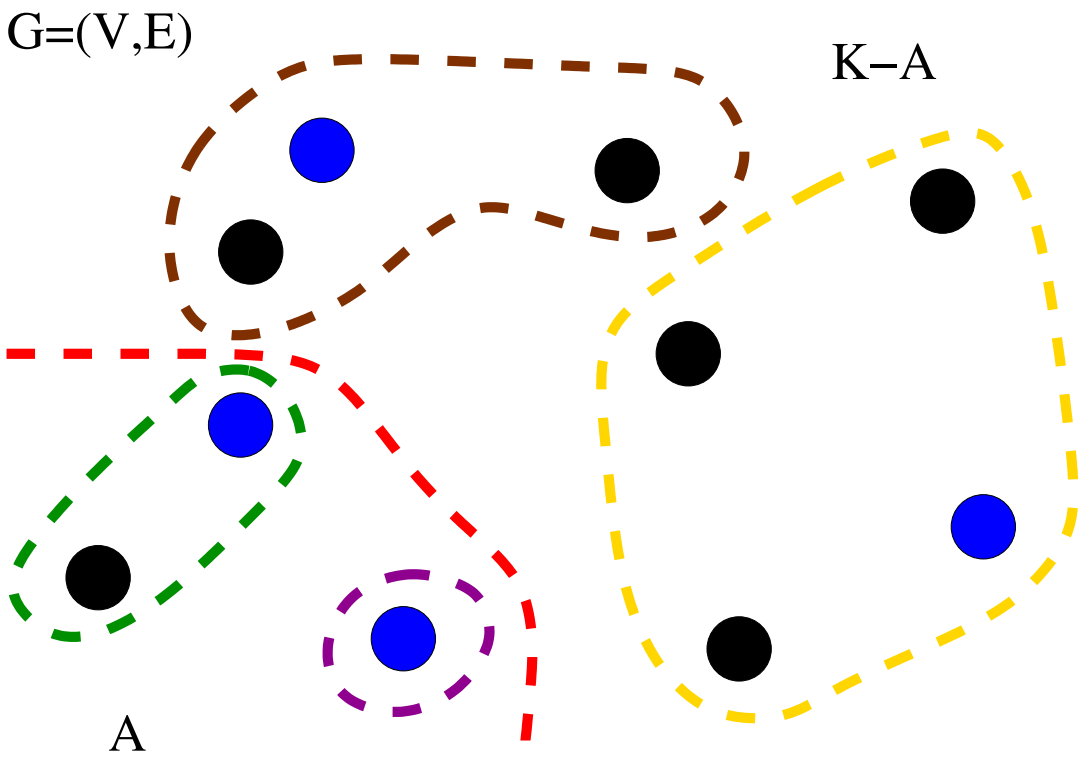
Lemma

G_f is a vertex sparsifier



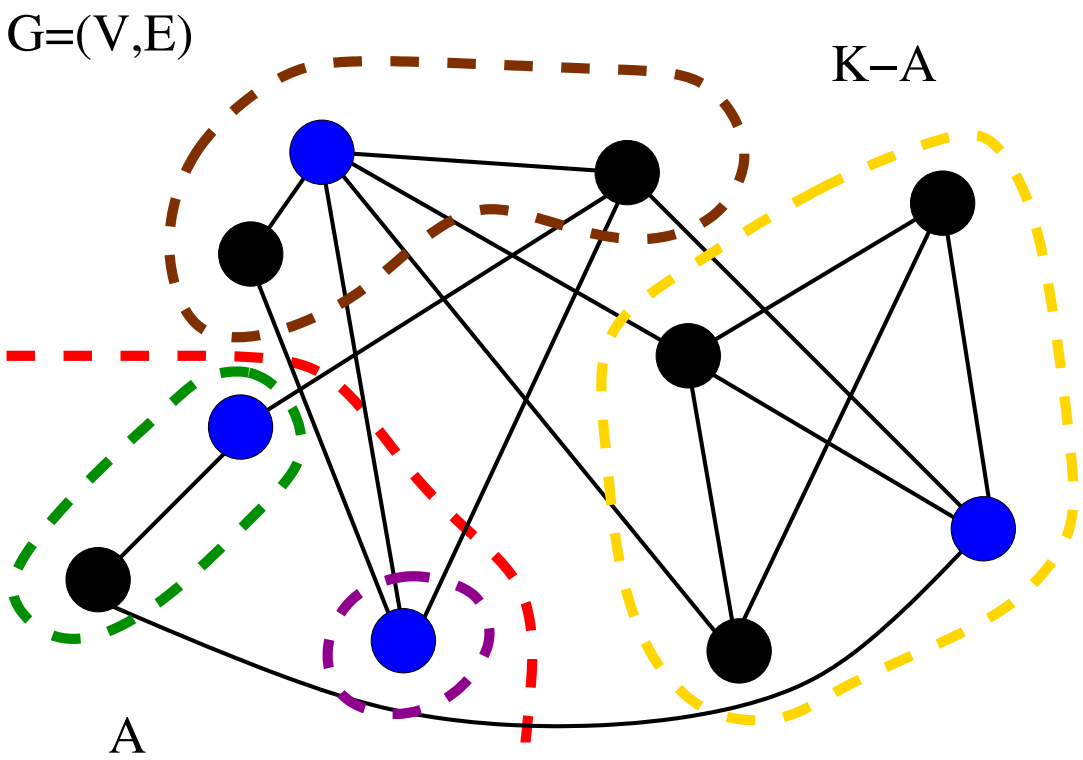
Lemma

G_f is a vertex sparsifier



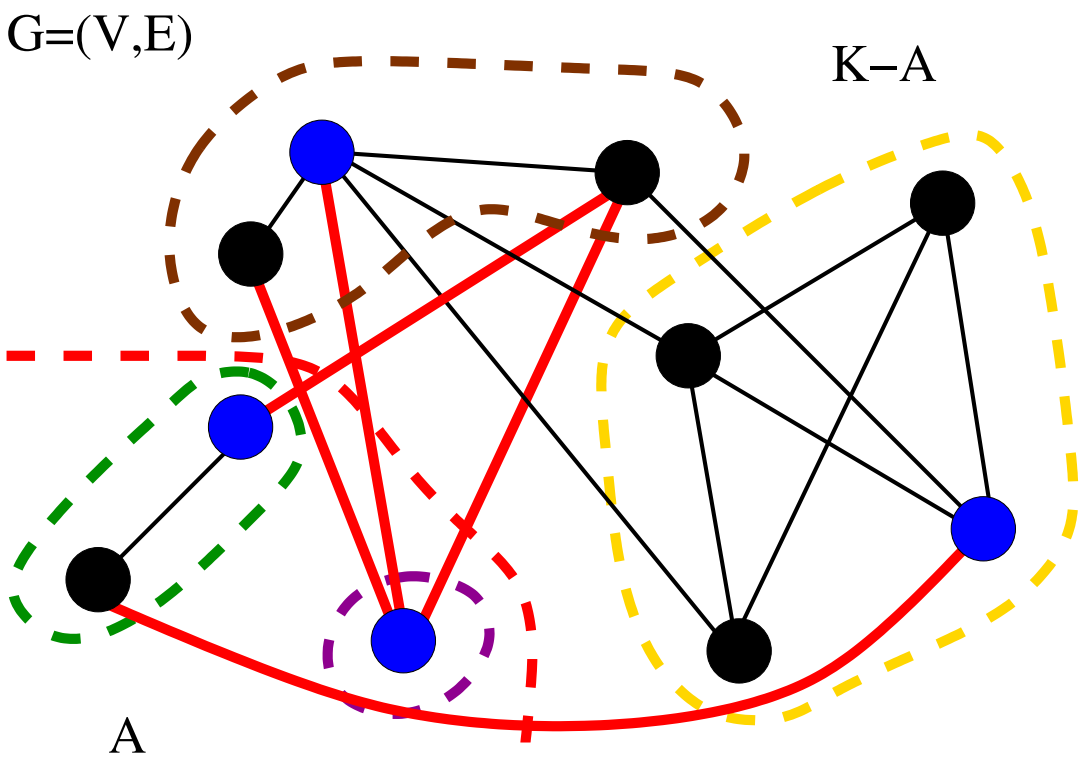
Lemma

G_f is a vertex sparsifier

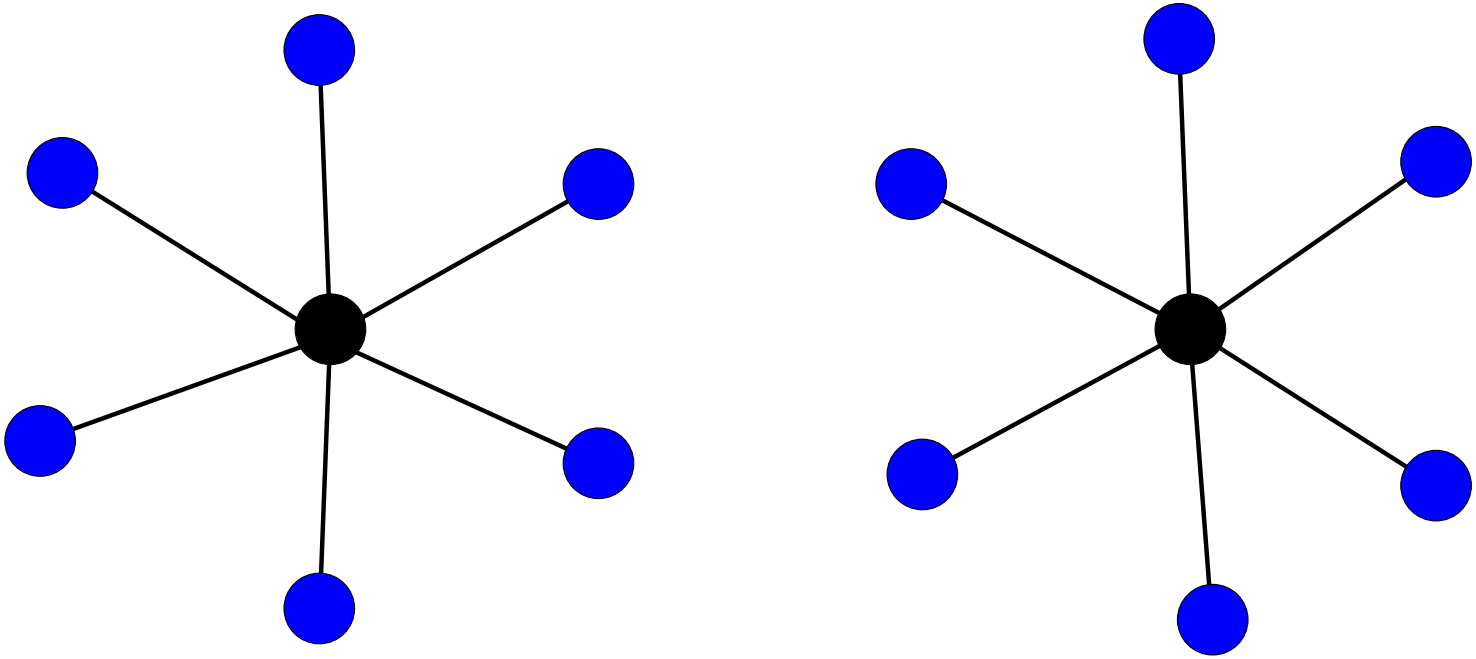


Lemma

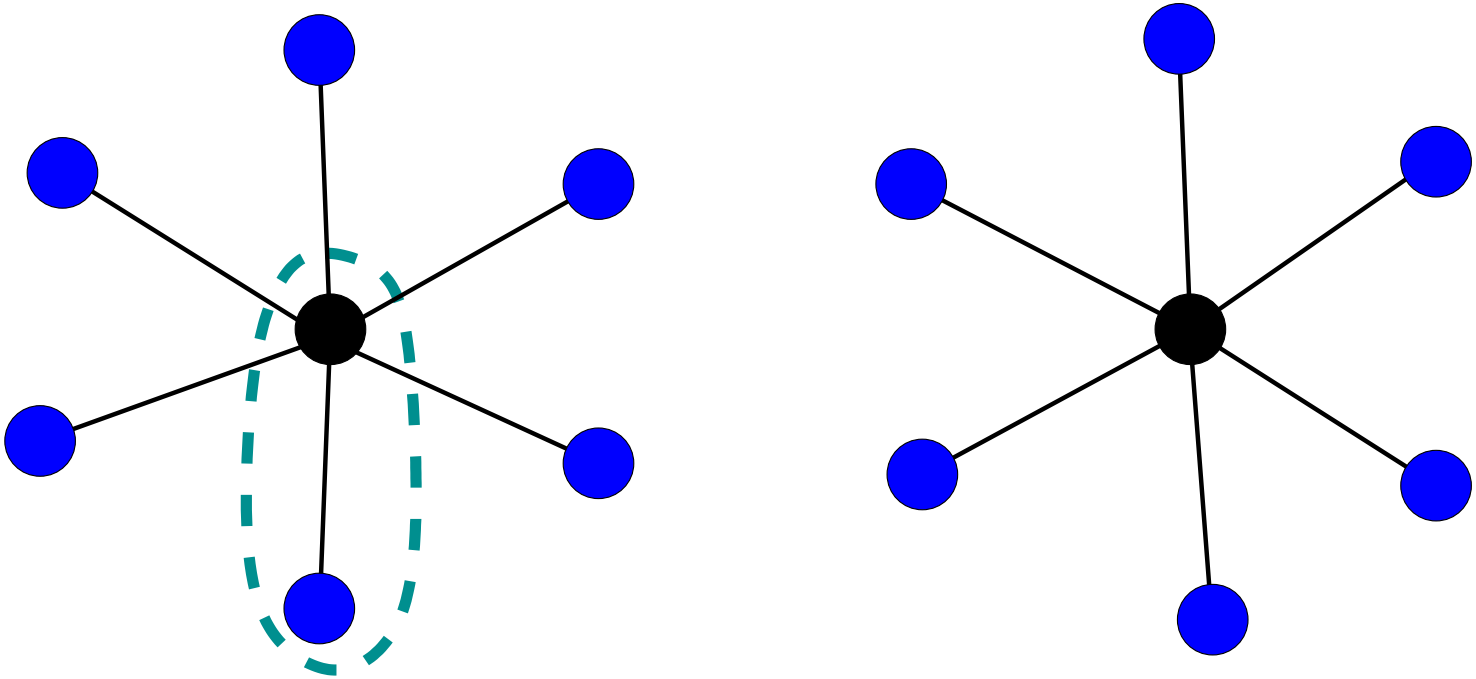
G_f is a vertex sparsifier



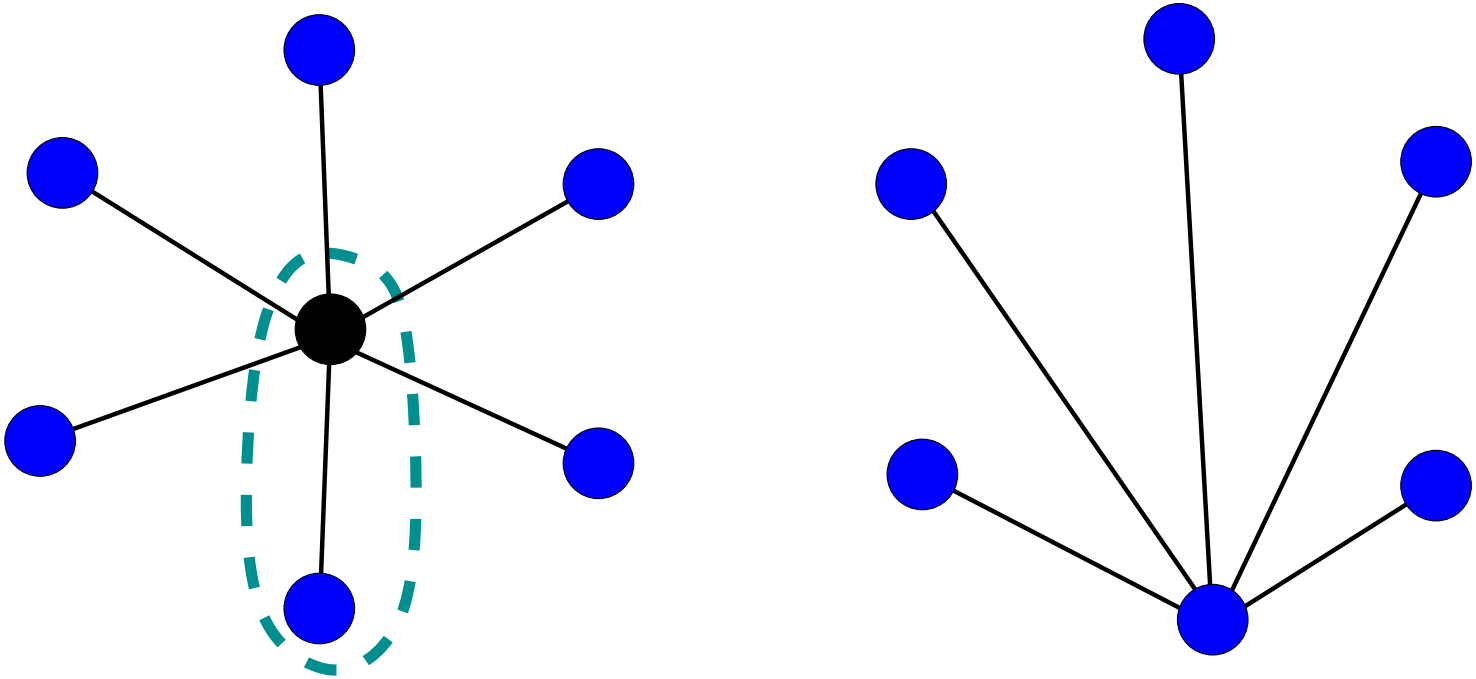
An Example



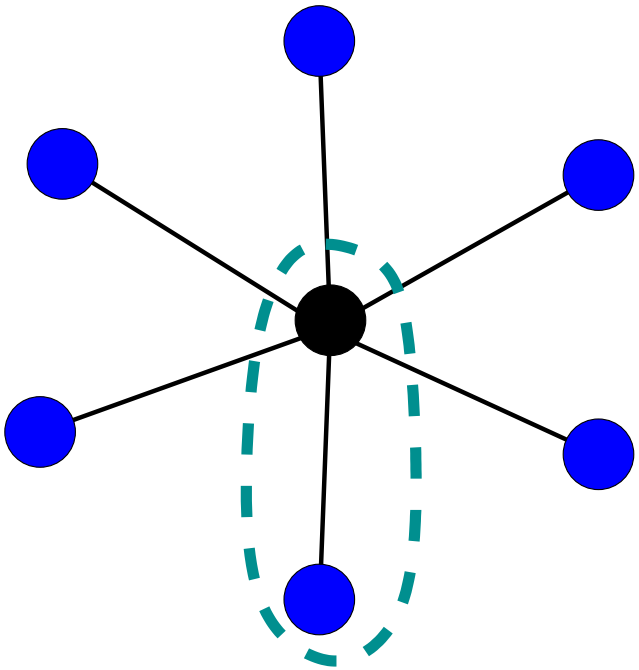
An Example



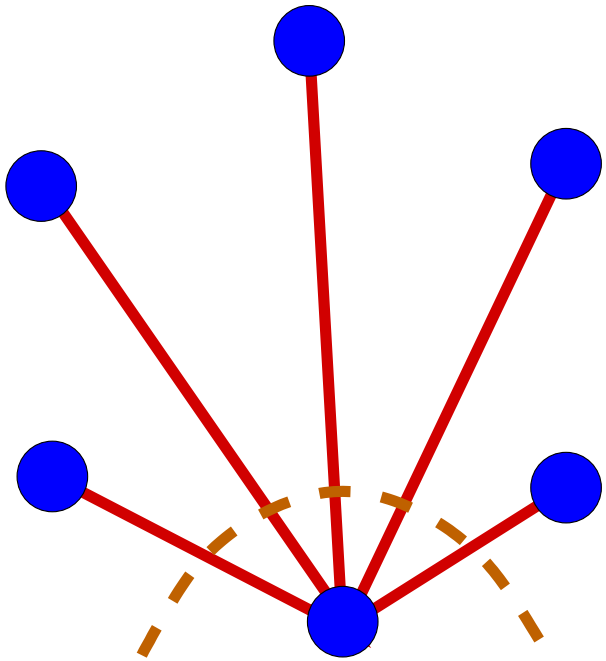
An Example



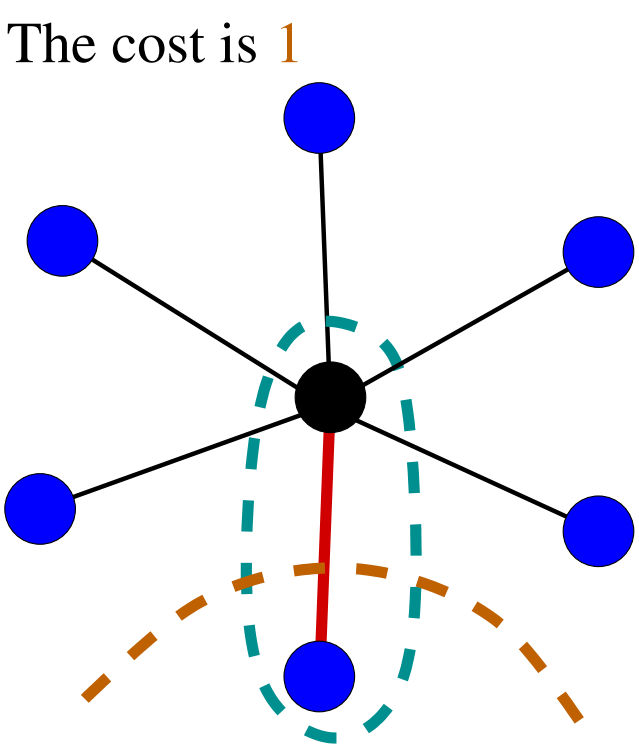
An Example

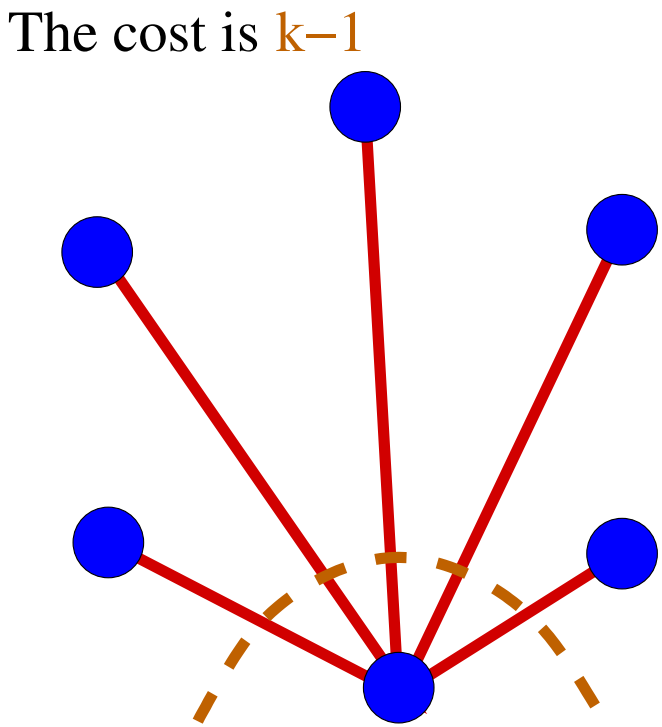


The cost is k–1

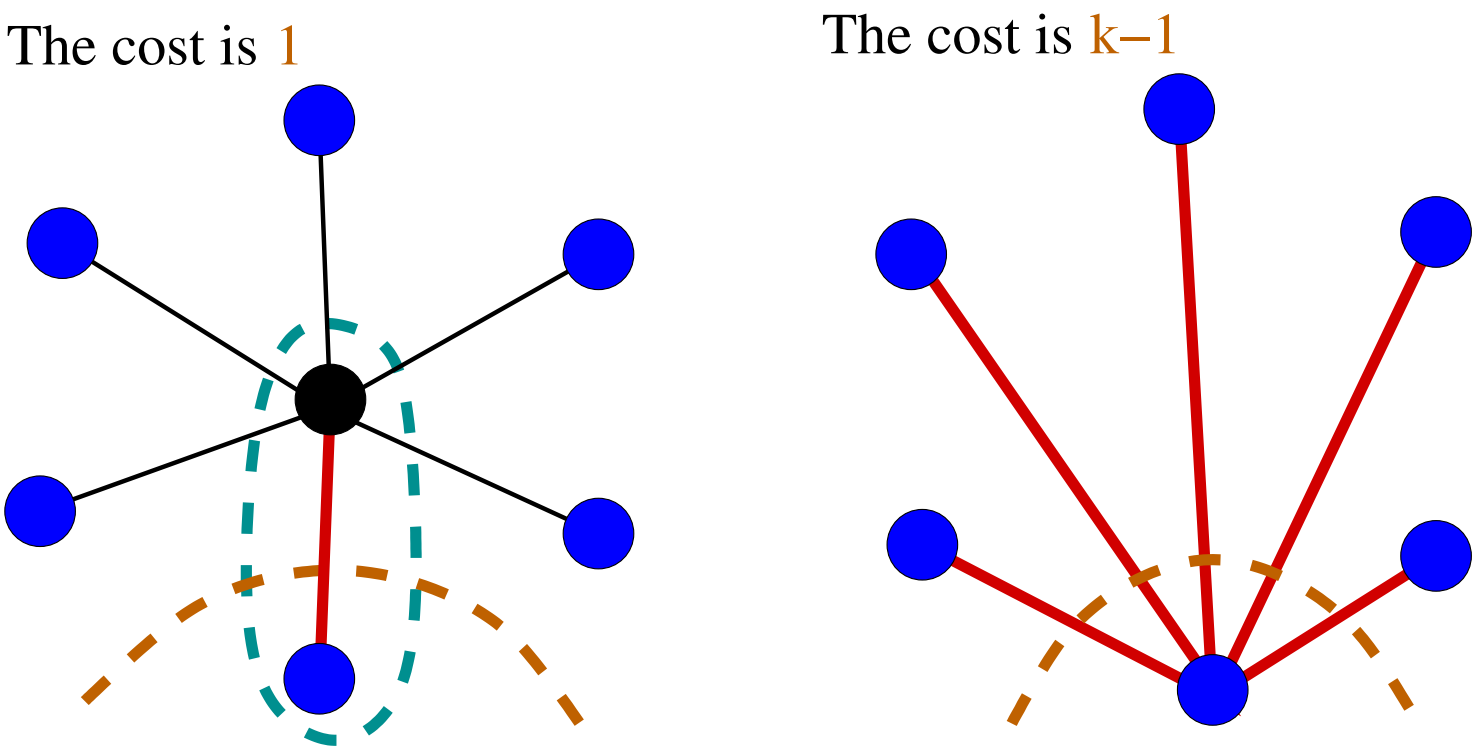


An Example

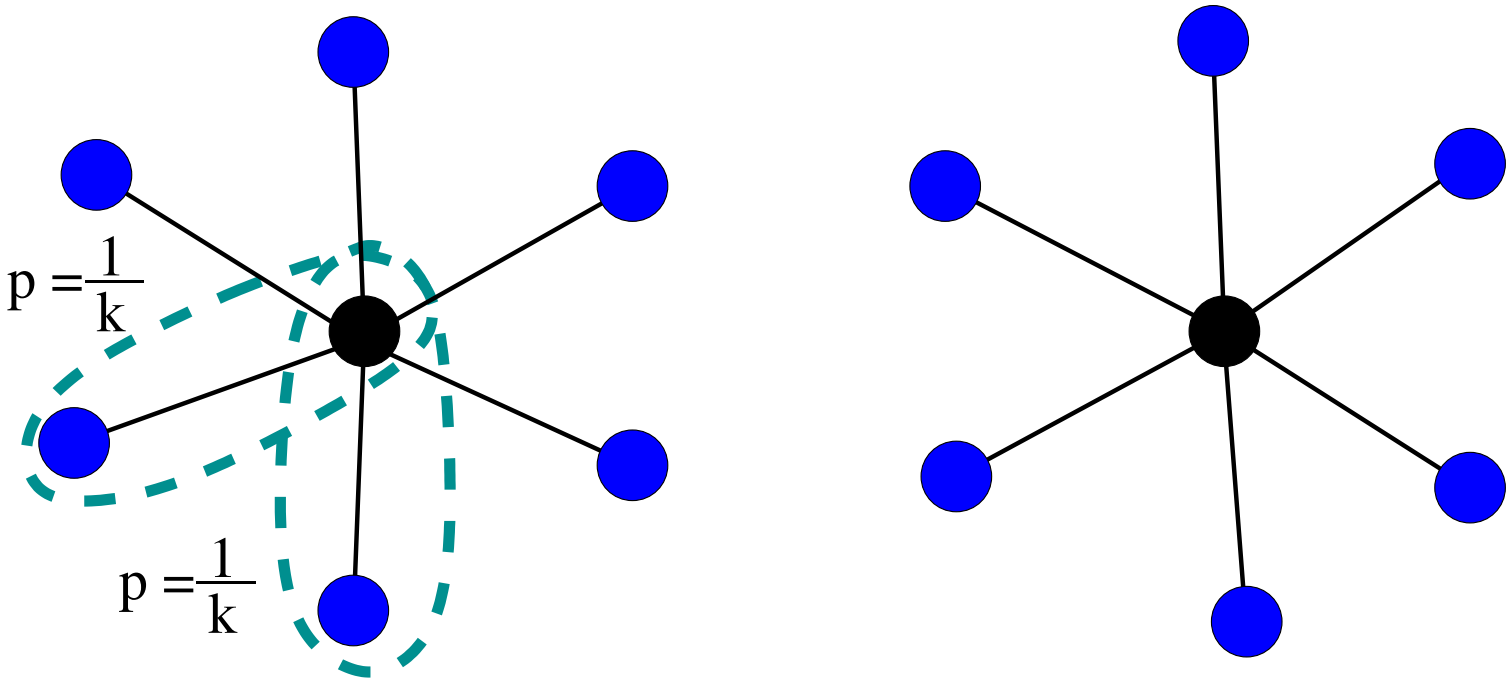




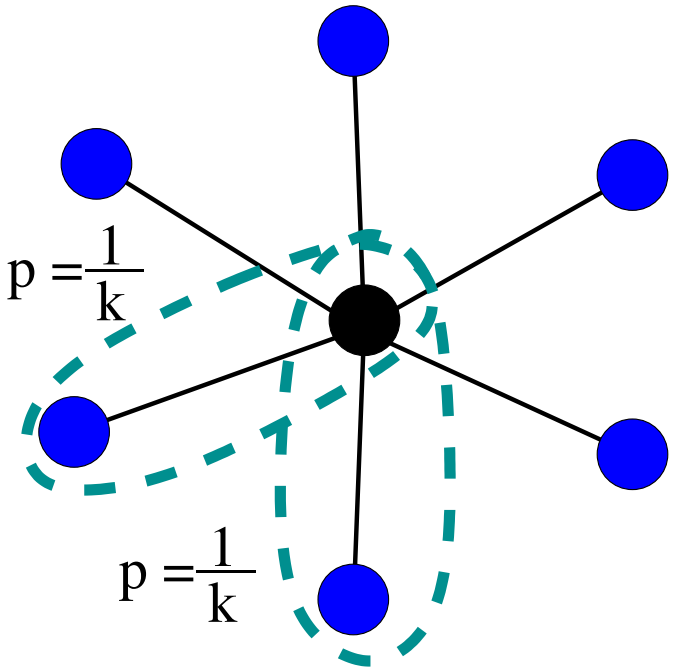
An Example

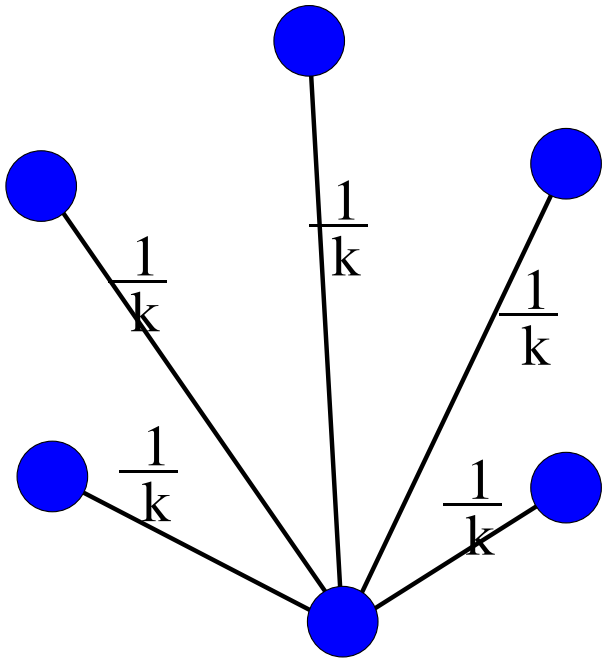

The cost is 1
The cost is k−1
The Quality of the Sparsifier is k−1

An Example: A Second Attempt

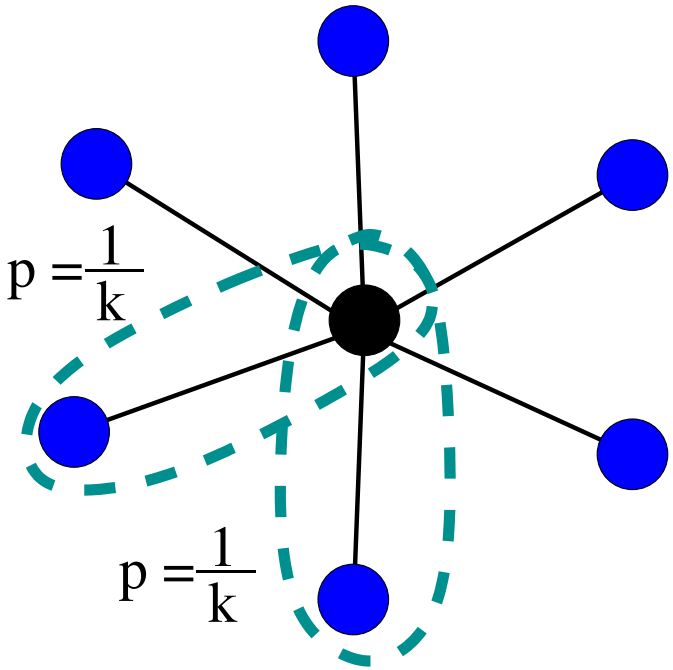


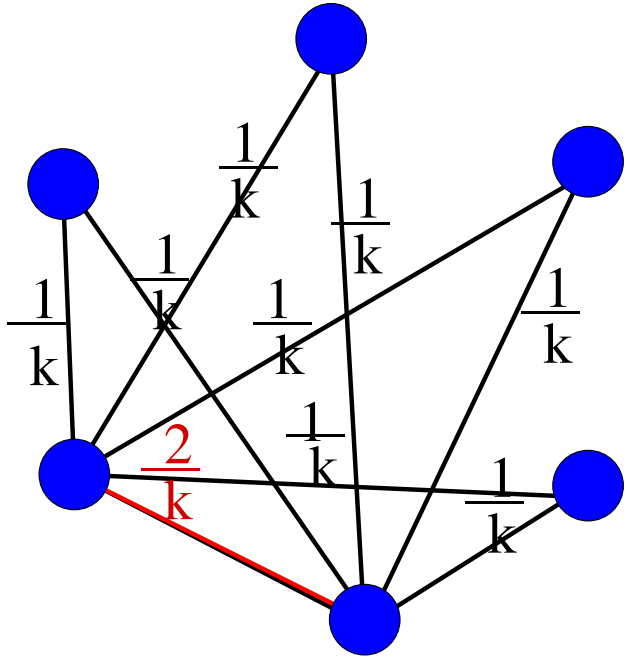
An Example: A Second Attempt



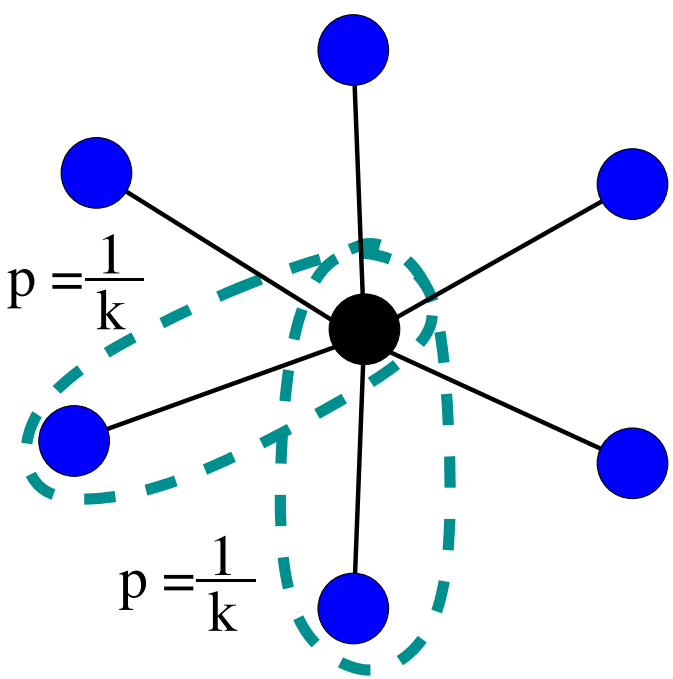


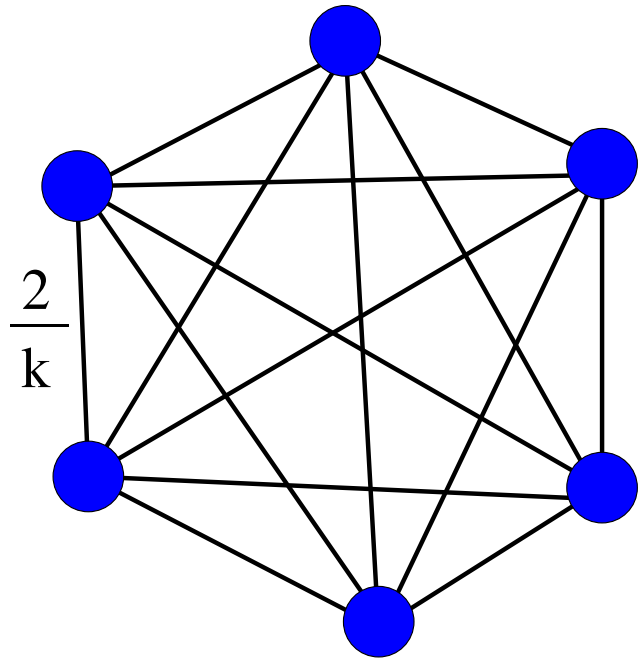
An Example: A Second Attempt



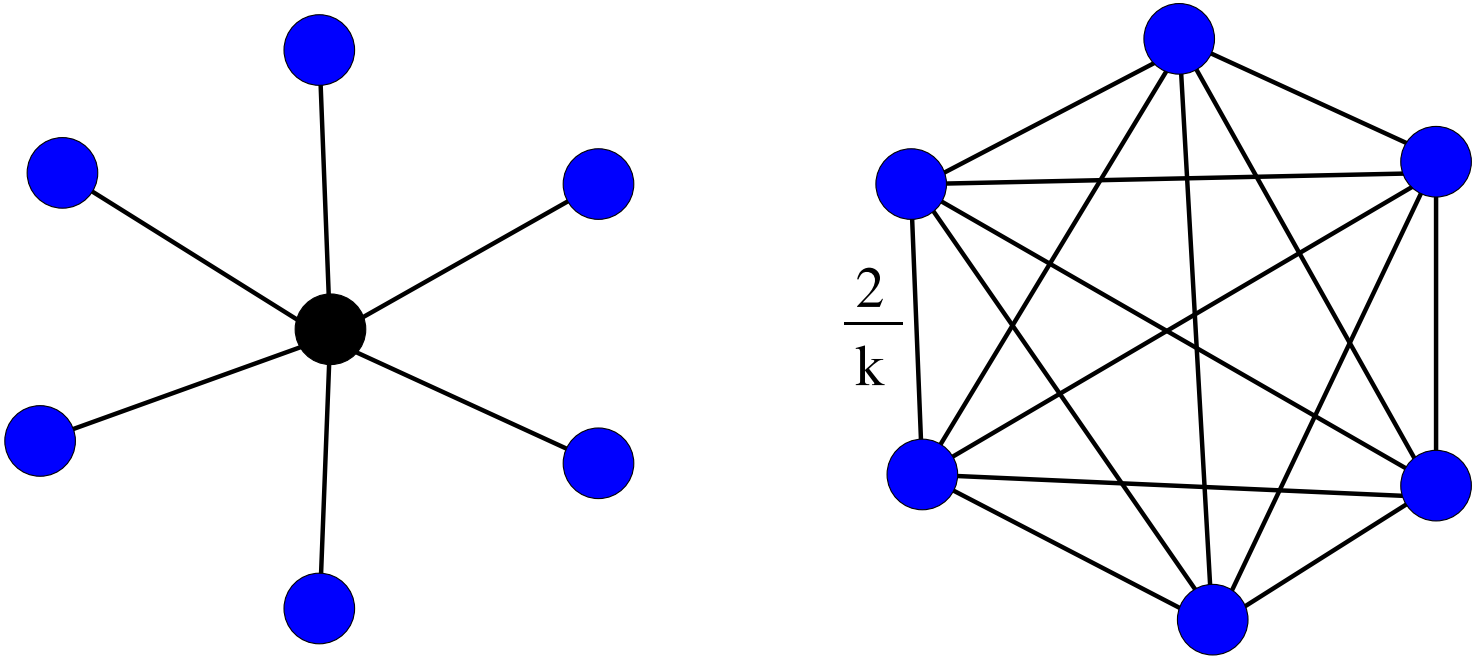


An Example: A Second Attempt



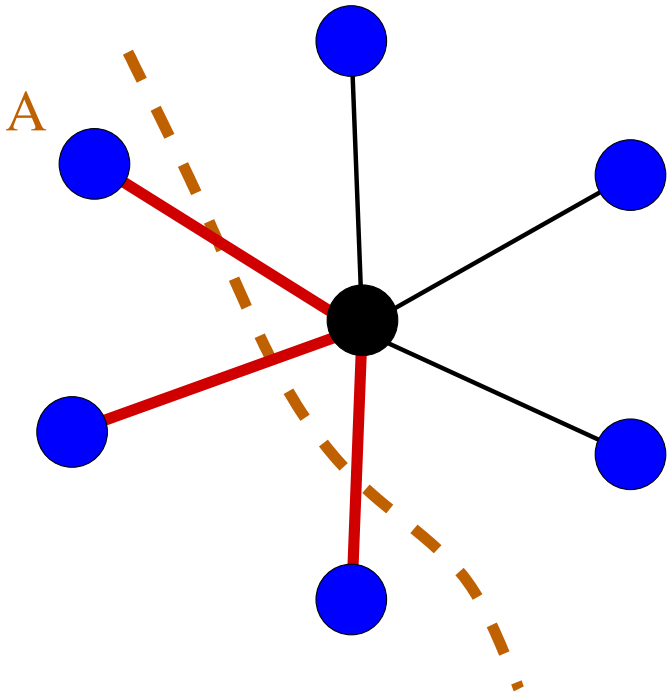


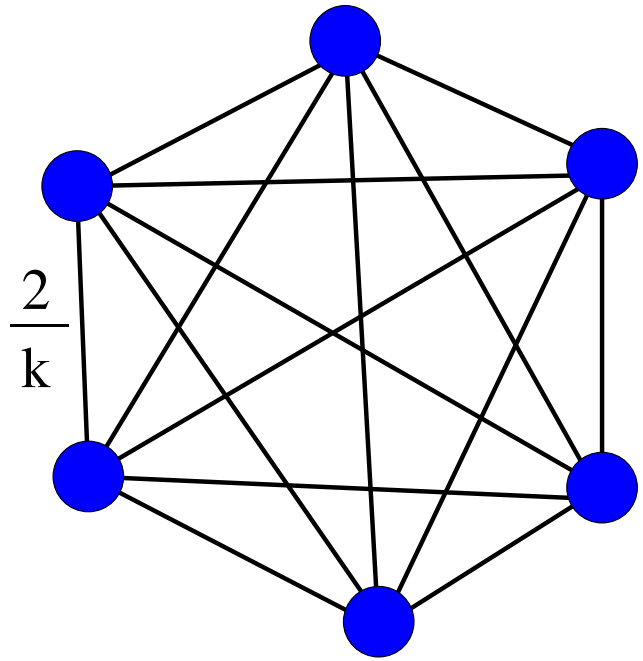
An Example: A Second Attempt



An Example: A Second Attempt

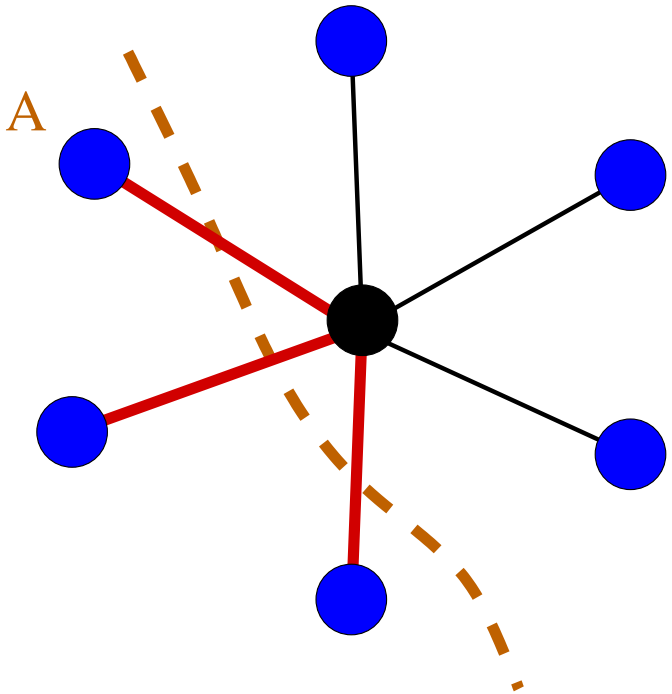
The cost is $\min(|A|, |K-A|)$



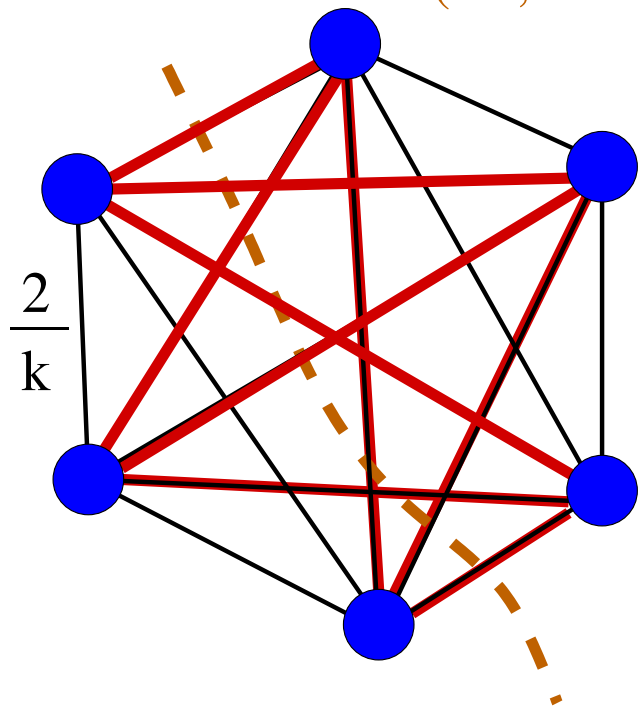


An Example: A Second Attempt

The cost is $\min(|A|, |K-A|)$

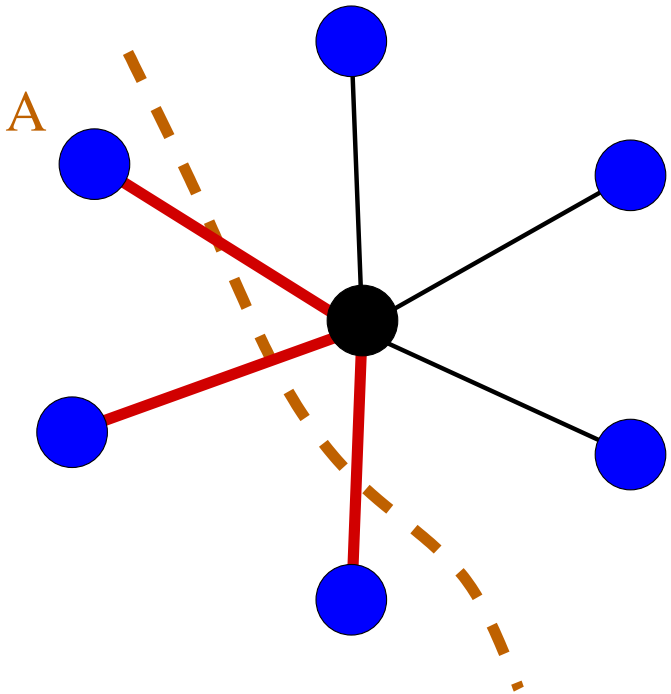


The cost is at most $2\min(|A|, |K-A|)$

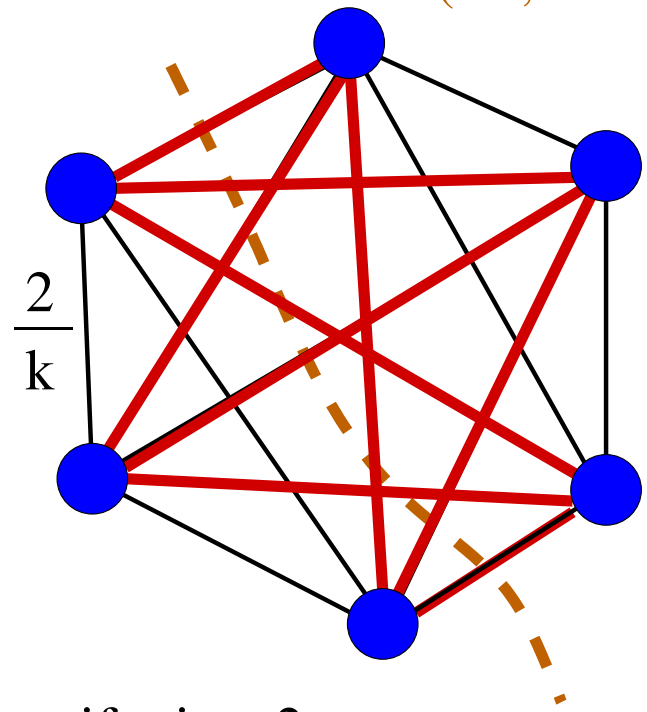


An Example: A Second Attempt

The cost is $\min(|A|, |K-A|)$



The cost is at most $2\min(|A|, |K-A|)$



The Quality of the Sparsifer is < 2

Approximate Vertex Sparsifiers via 0-Extensions

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a distribution γ on 0-extensions such that $G' = \sum_f \gamma(f) G_f$ is a $O(\frac{\log k}{\log \log k})$-quality Vertex Sparsifier for G, K.

Approximate Vertex Sparsifiers via 0-Extensions

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a distribution γ on 0-extensions such that $G' = \sum_f \gamma(f) G_f$ is a $O(\frac{\log k}{\log \log k})$-quality Vertex Sparsifier for G, K.

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a polynomial (in n and k) time algorithm to construct G' that is a poly$(\log k)$-quality Vertex Sparsifier for G, K.

Approximate Vertex Sparsifiers via 0-Extensions

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a distribution γ on 0-extensions such that $G' = \sum_f \gamma(f) G_f$ is a $O(\frac{\log k}{\log \log k})$-quality Vertex Sparsifier for G, K.

Theorem

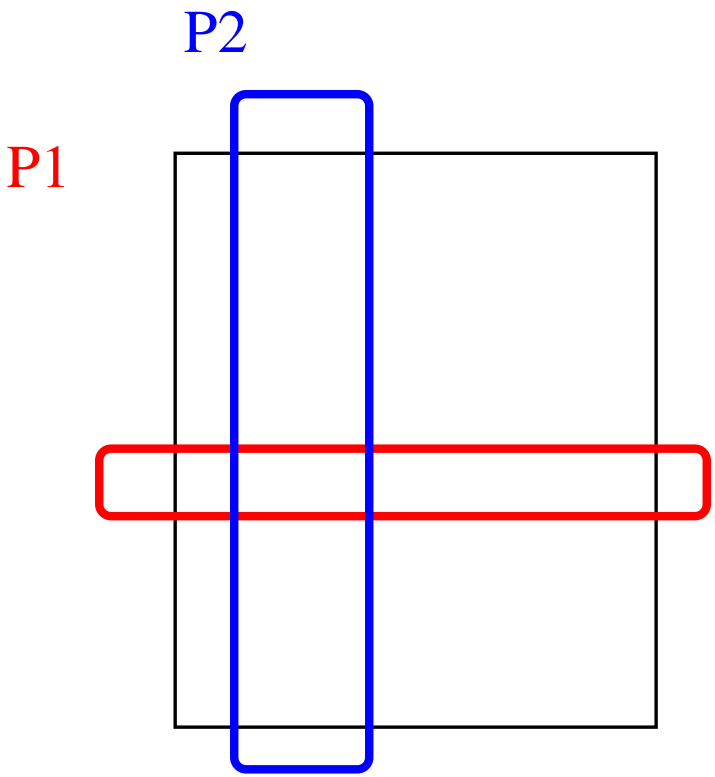
For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a polynomial (in n and k) time algorithm to construct G' that is a poly$(\log k)$-quality Vertex Sparsifier for G, K.

Proof Outline

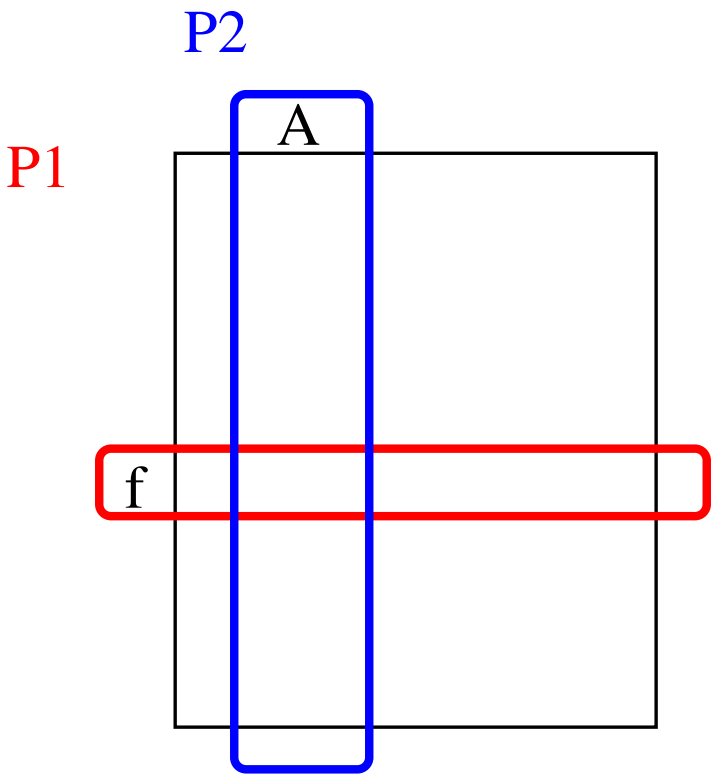
Proof Outline

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

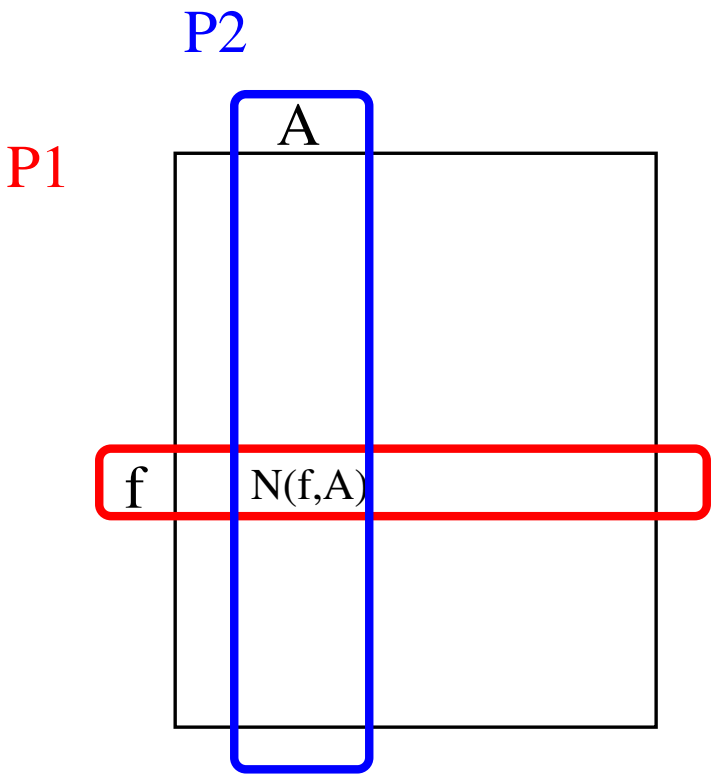
The Extension-Cut Game



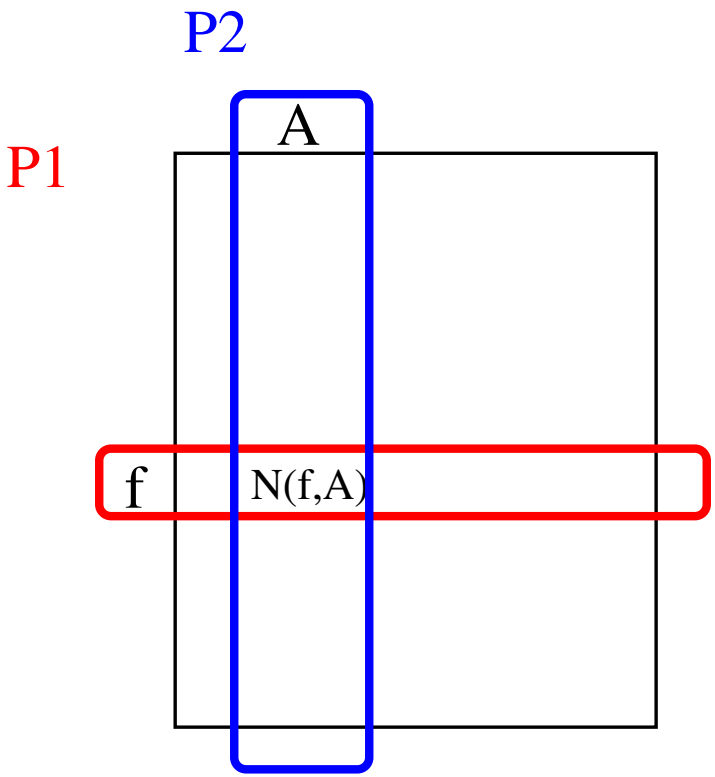
The Extension-Cut Game

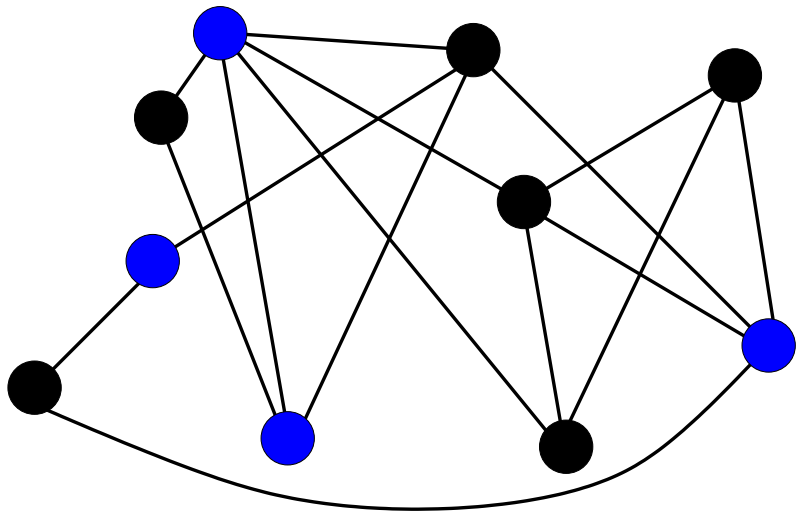


The Extension-Cut Game

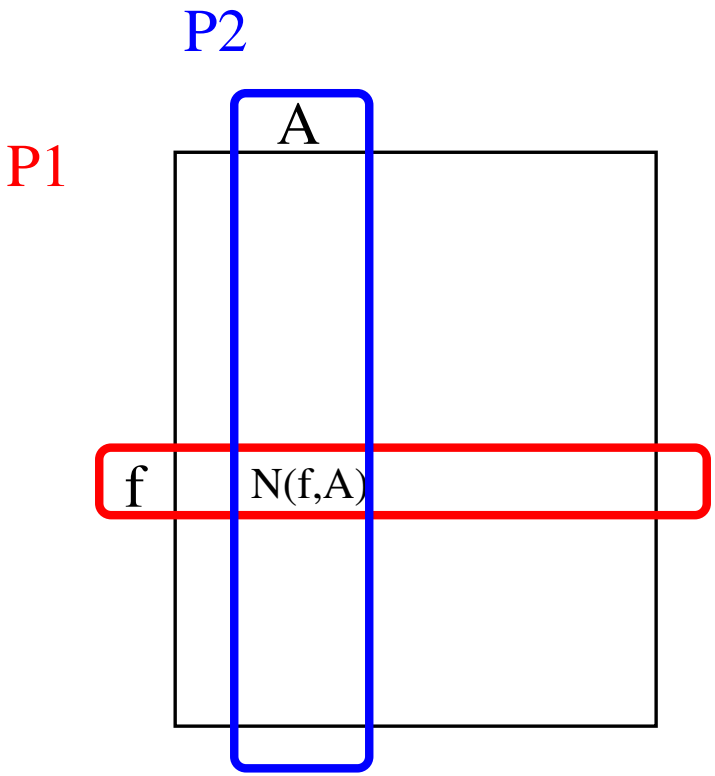


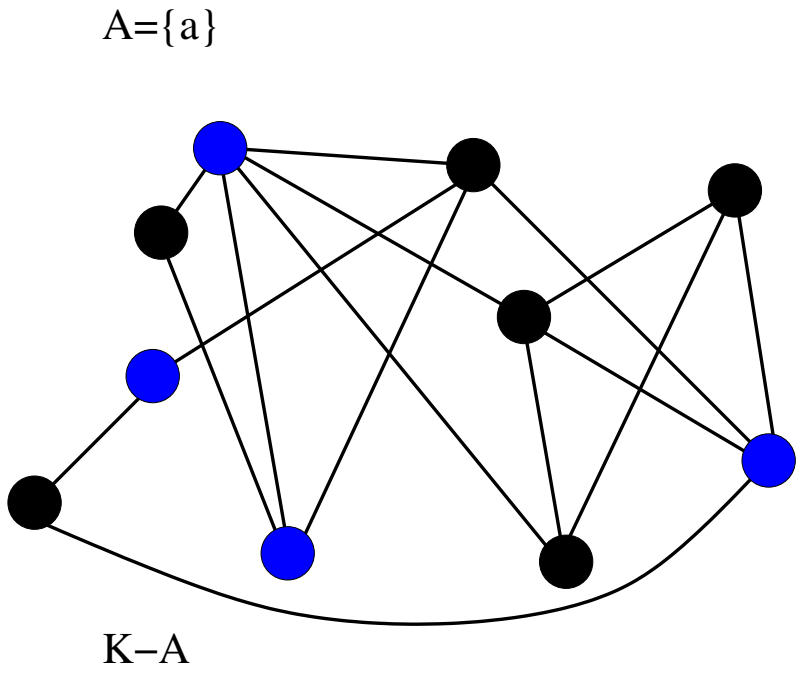
The Extension-Cut Game



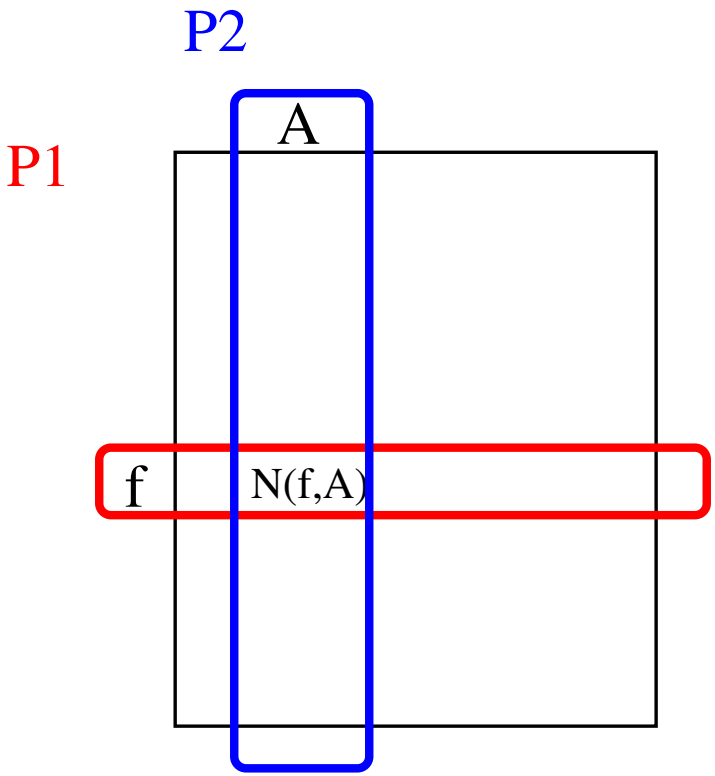


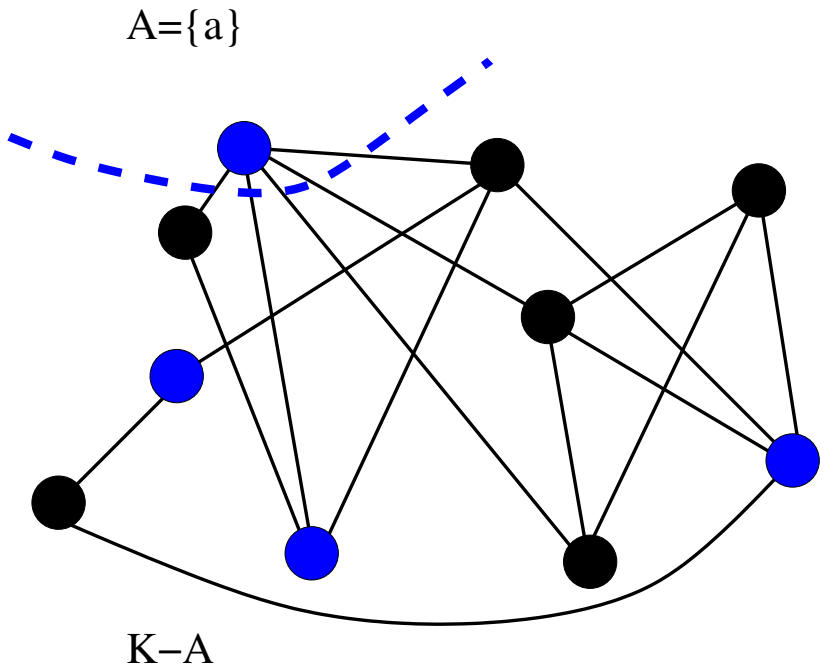
The Extension-Cut Game



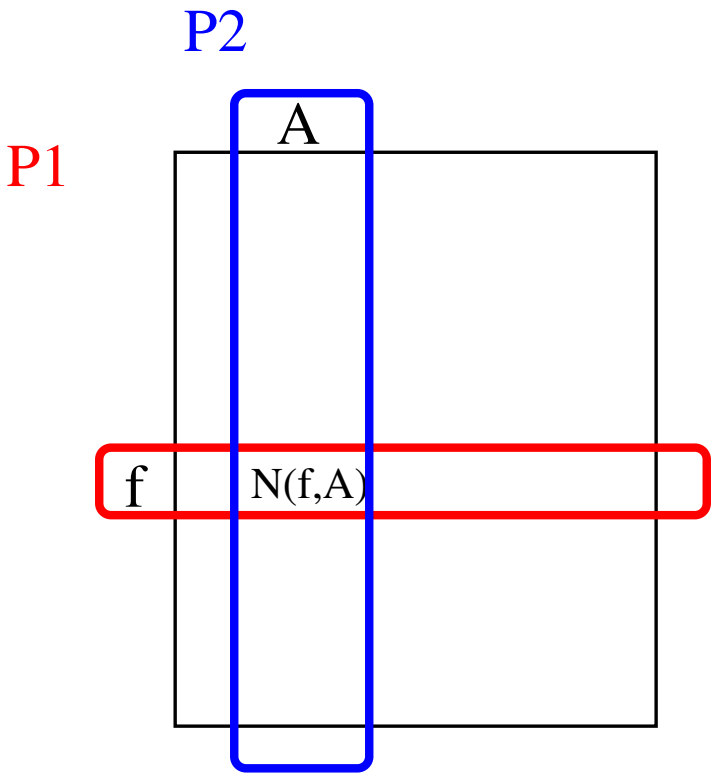


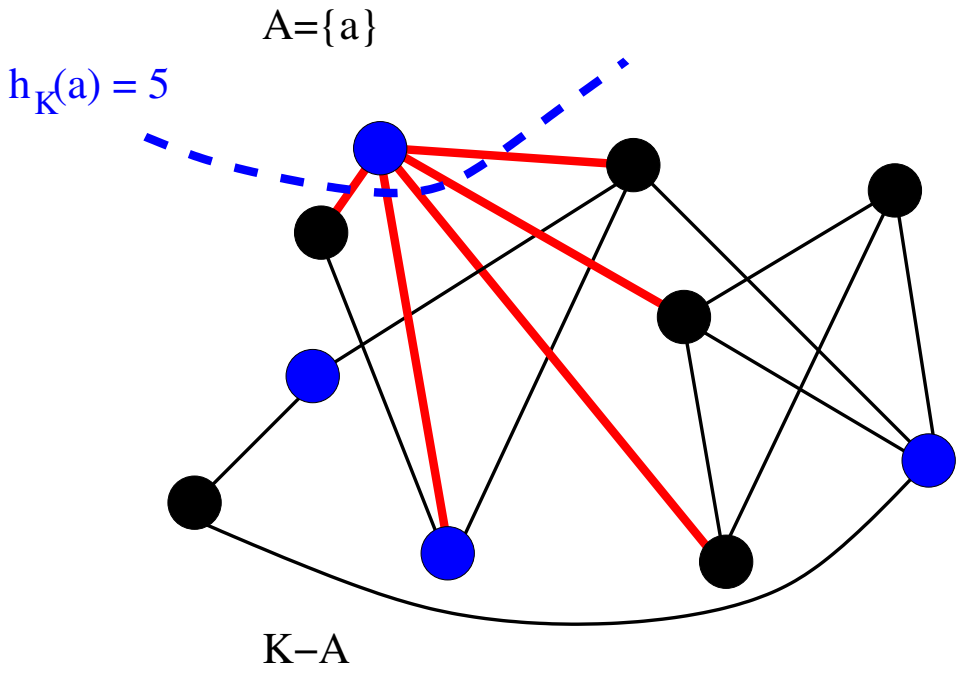
The Extension-Cut Game



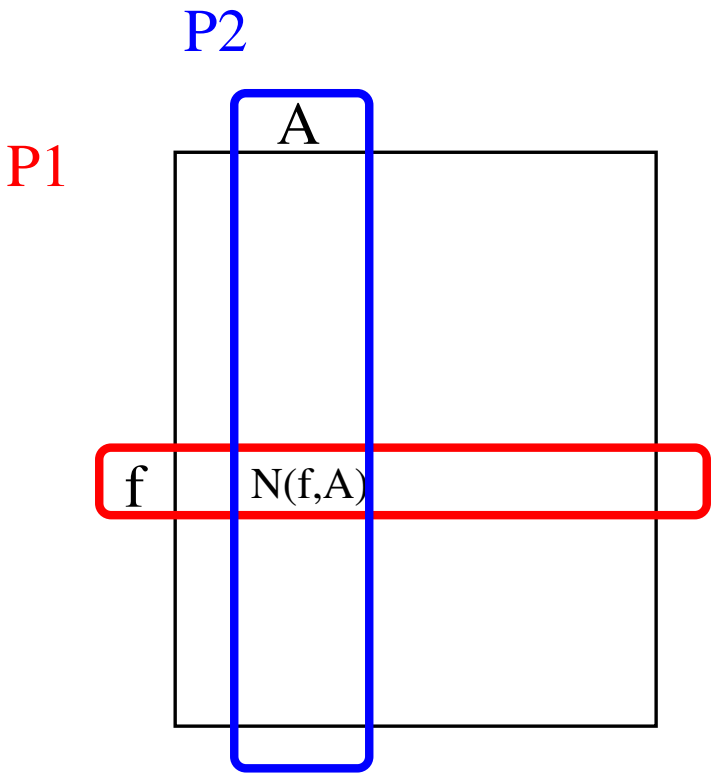


The Extension-Cut Game

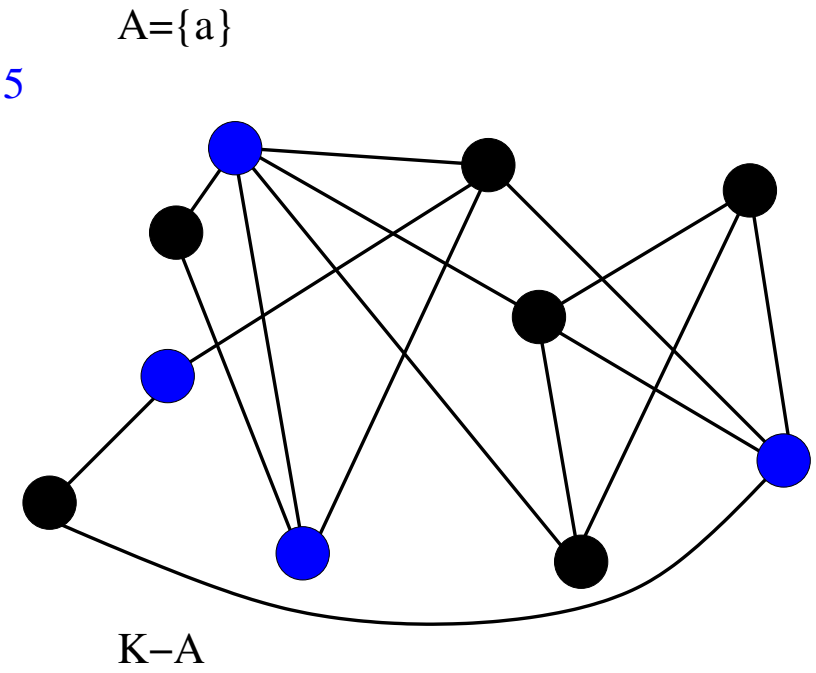




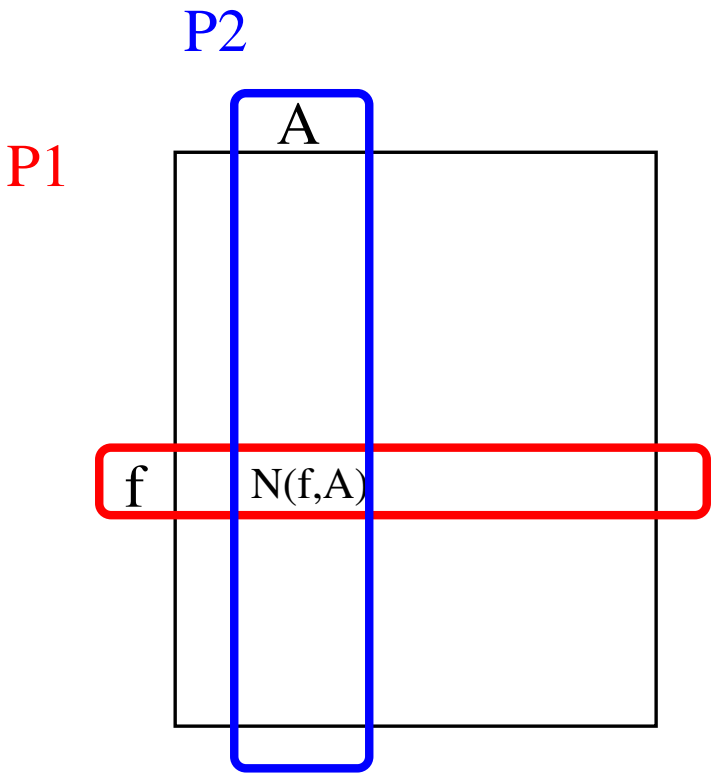
The Extension-Cut Game

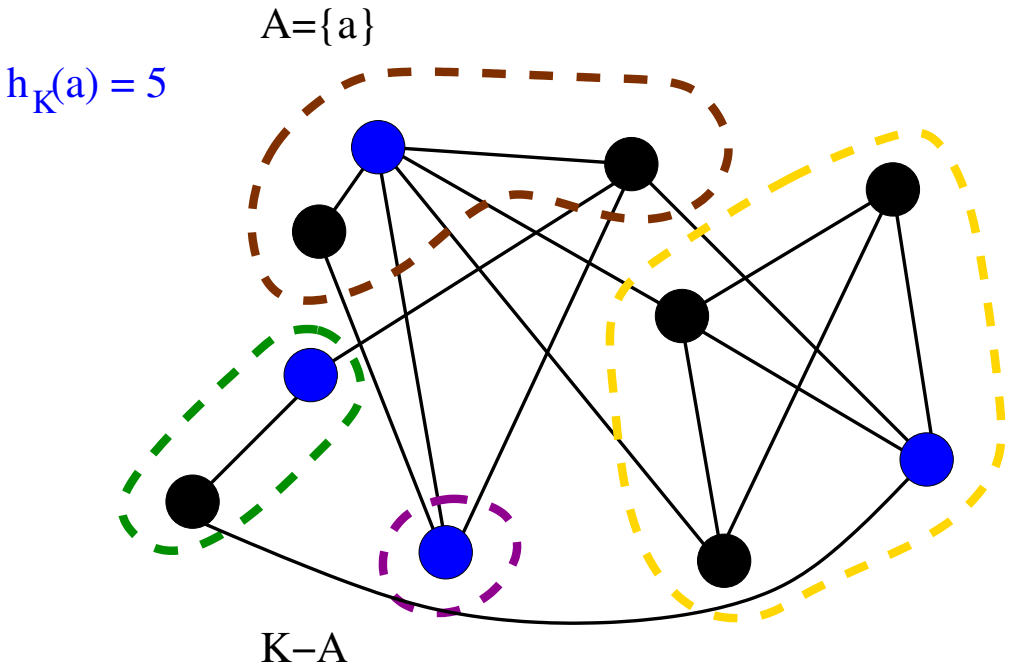


$h_K(a) = 5$

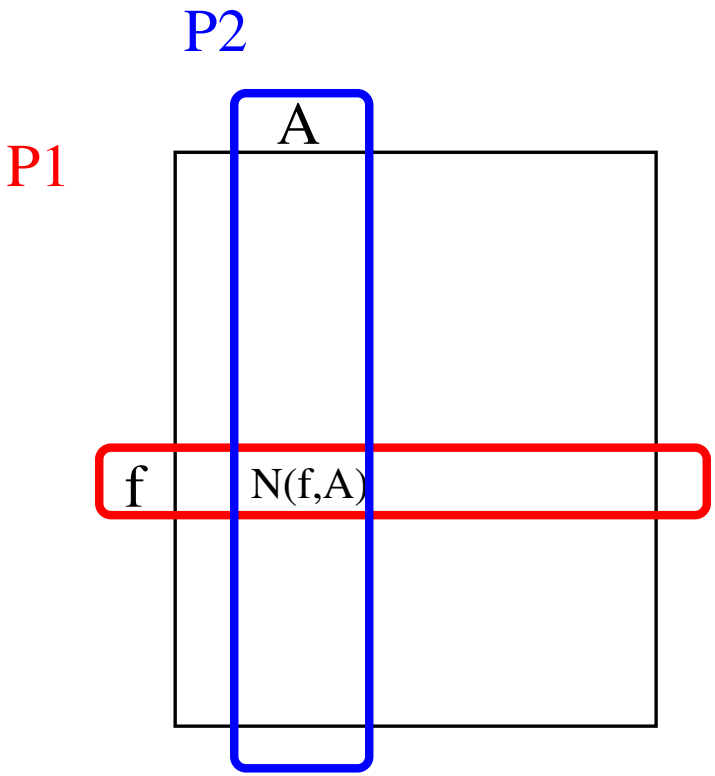


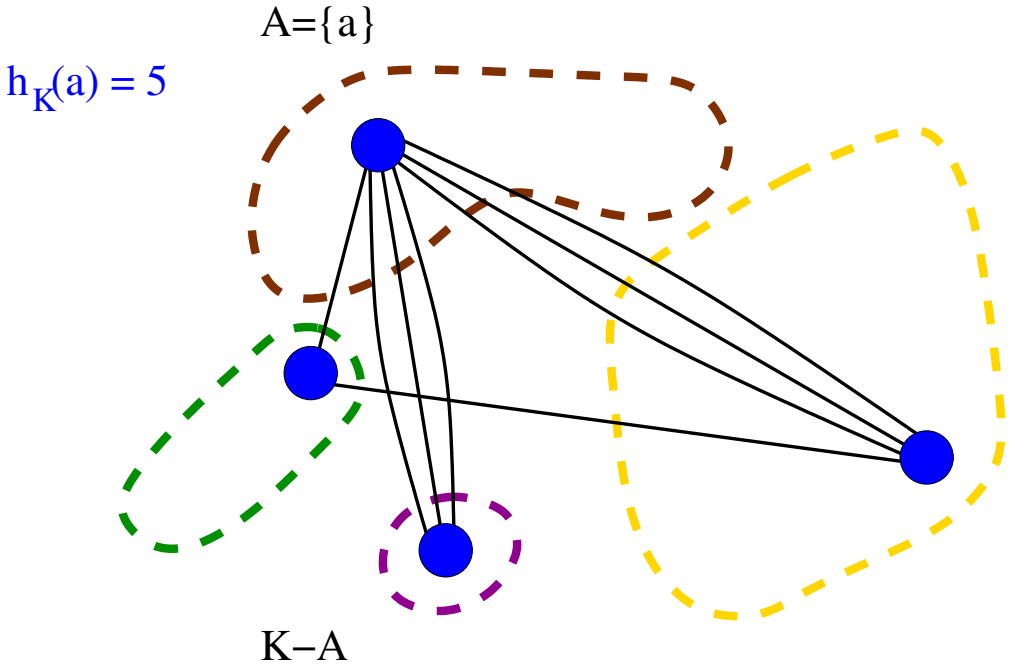
The Extension-Cut Game



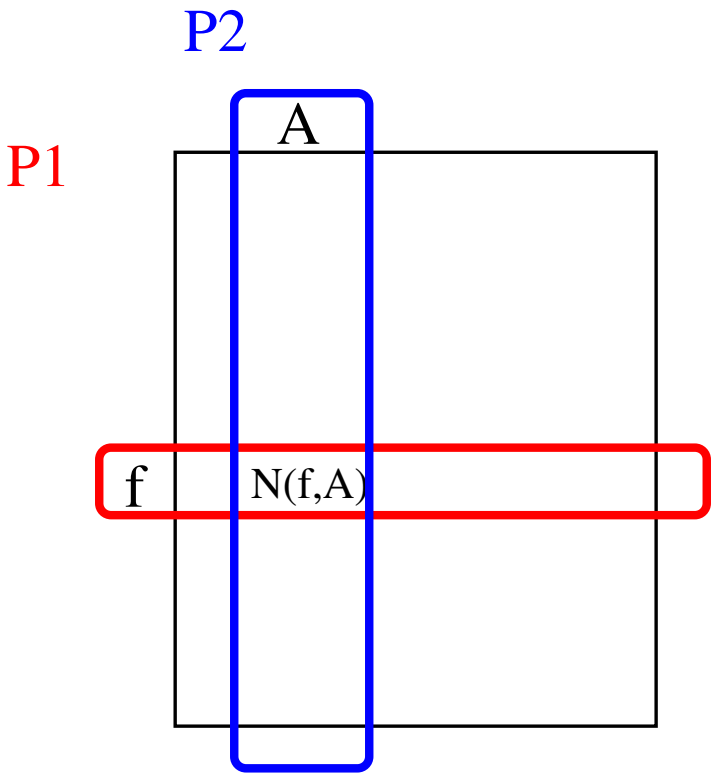


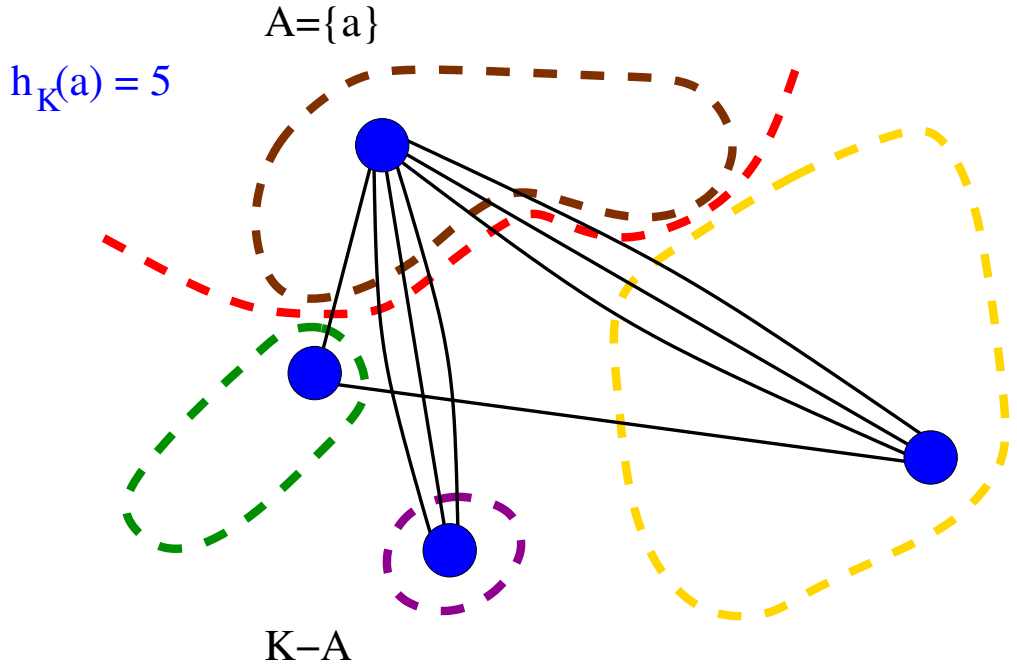
The Extension-Cut Game



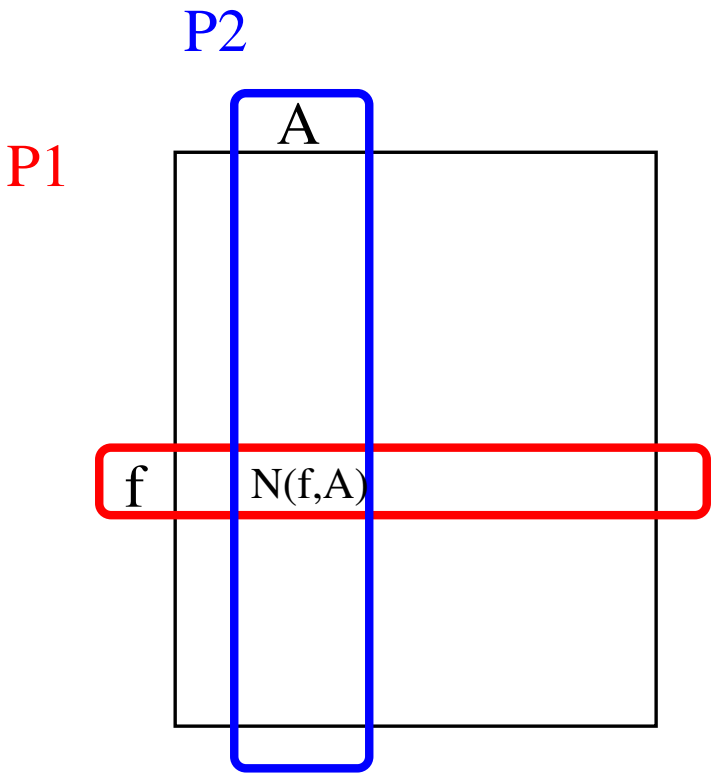


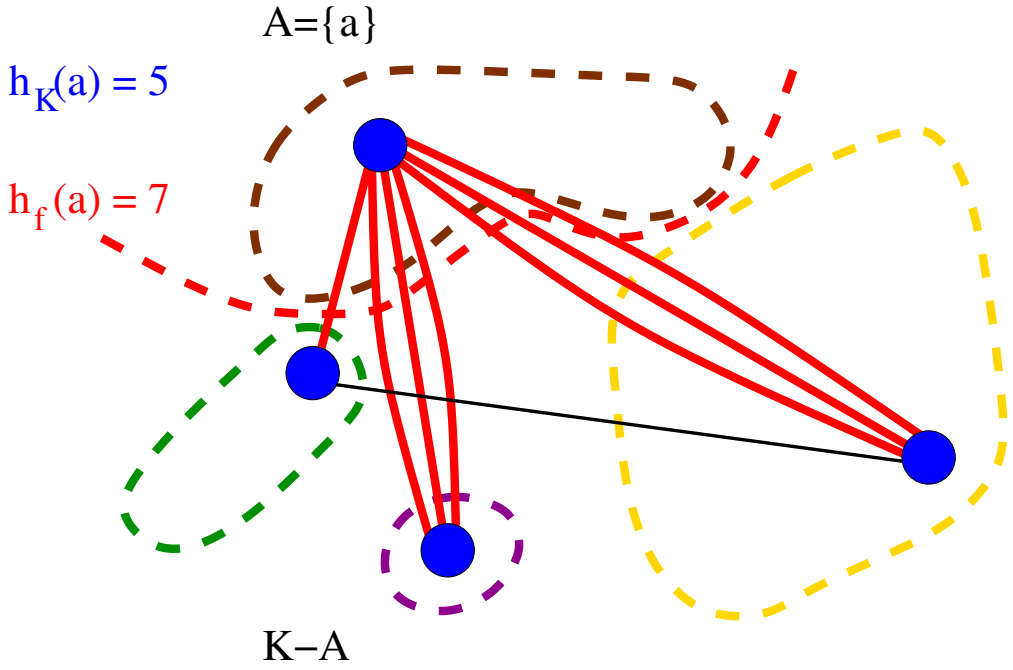
The Extension-Cut Game



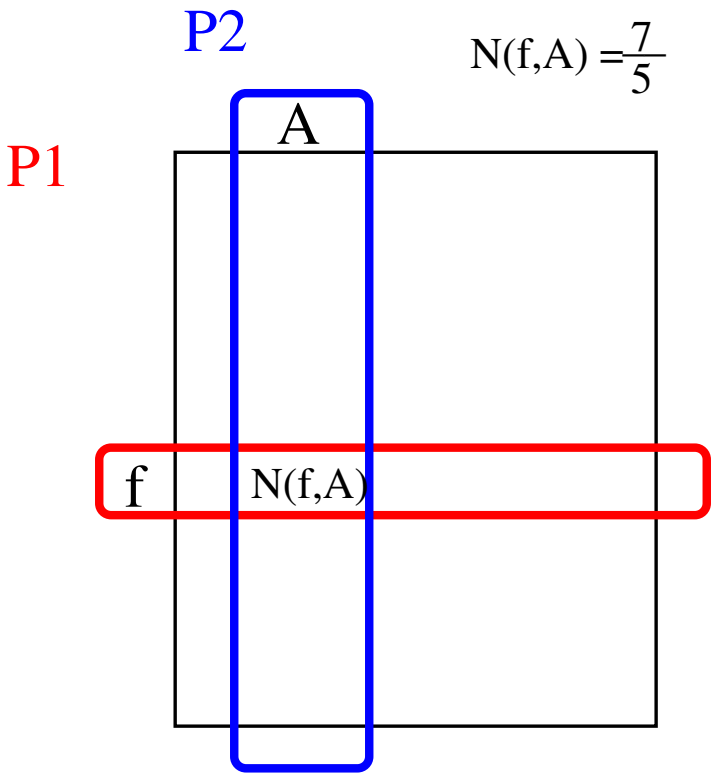


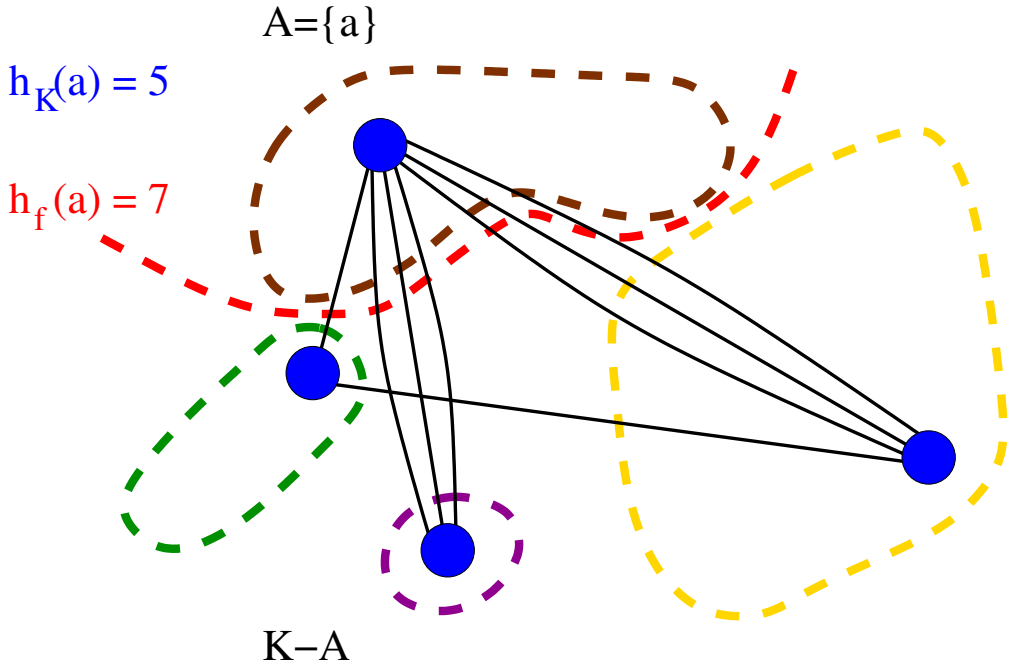
The Extension-Cut Game





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 $A \subset K$:

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

$$E_{f \leftarrow \gamma}[N(f, A)] = \sum_{f \in supp(\gamma)} \gamma(f) \frac{h_f(A)}{h_K(A)} = \frac{E_{f \leftarrow \gamma}[h_f(A)]}{h_K(A)} \leq \nu$$

Definition

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

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

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

$$E_{f \leftarrow \gamma}[N(f, A)] = \sum_{f \in supp(\gamma)} \gamma(f) \frac{h_f(A)}{h_K(A)} = \frac{E_{f \leftarrow \gamma}[h_f(A)]}{h_K(A)} \leq \nu$$

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

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

Proof Outline

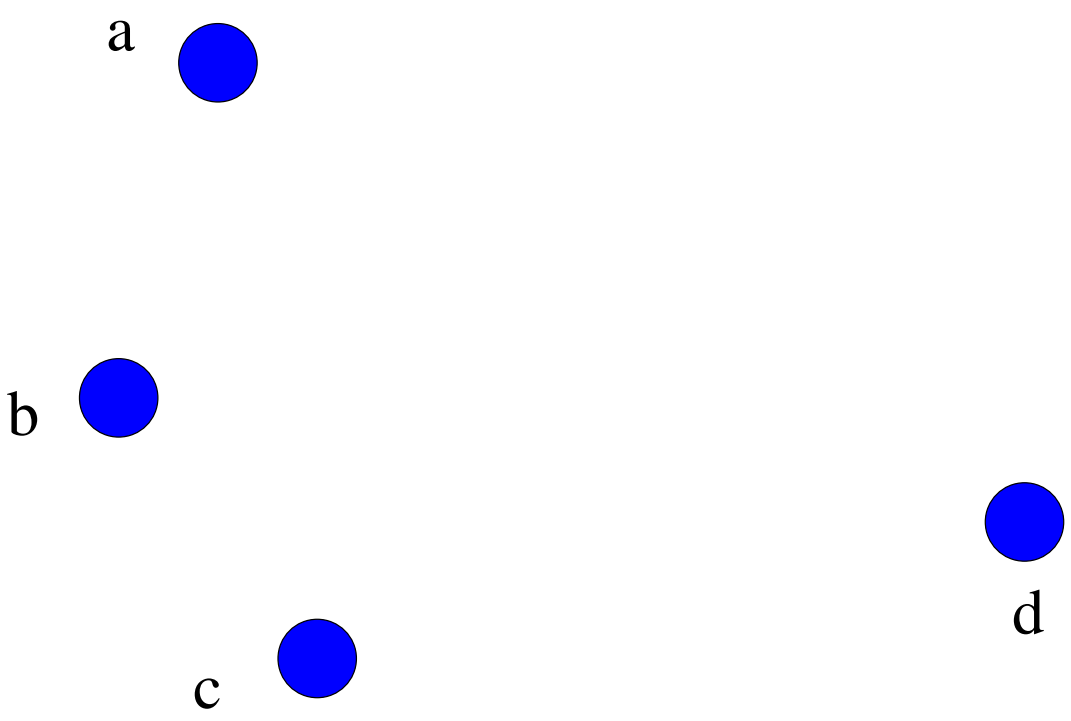
1. Define a **Zero-Sum Game**

Proof Outline

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

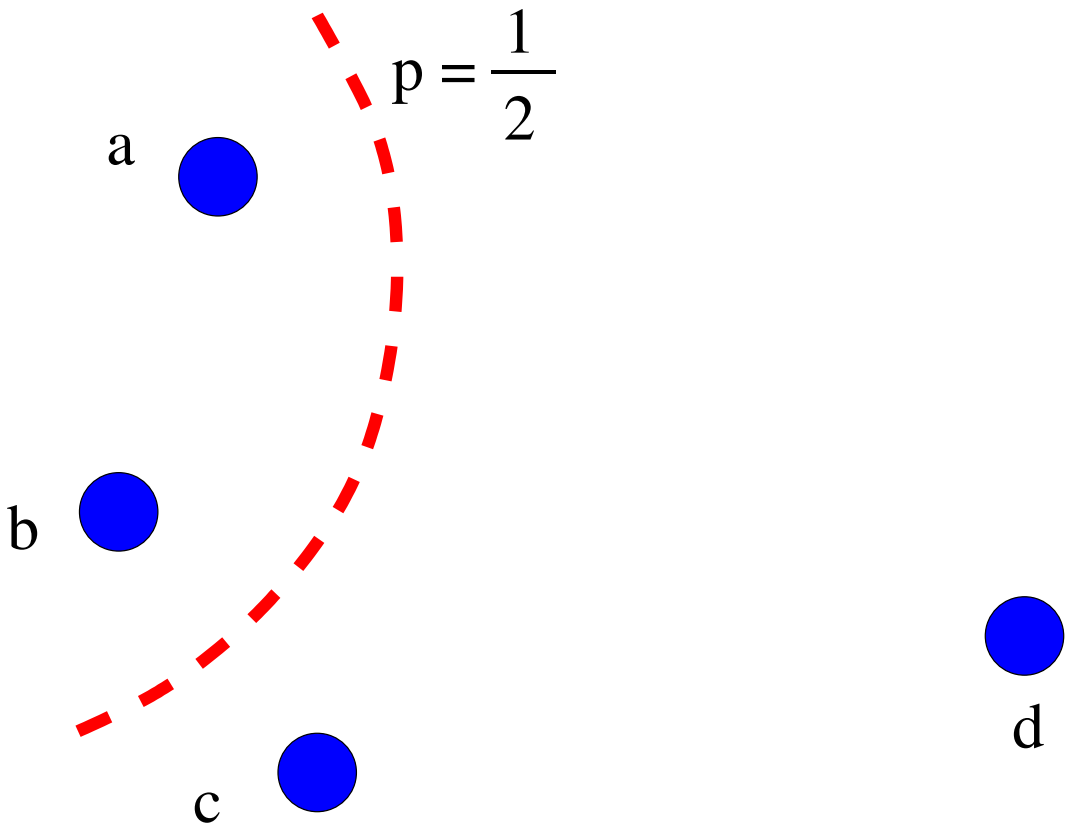
Best Response?

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



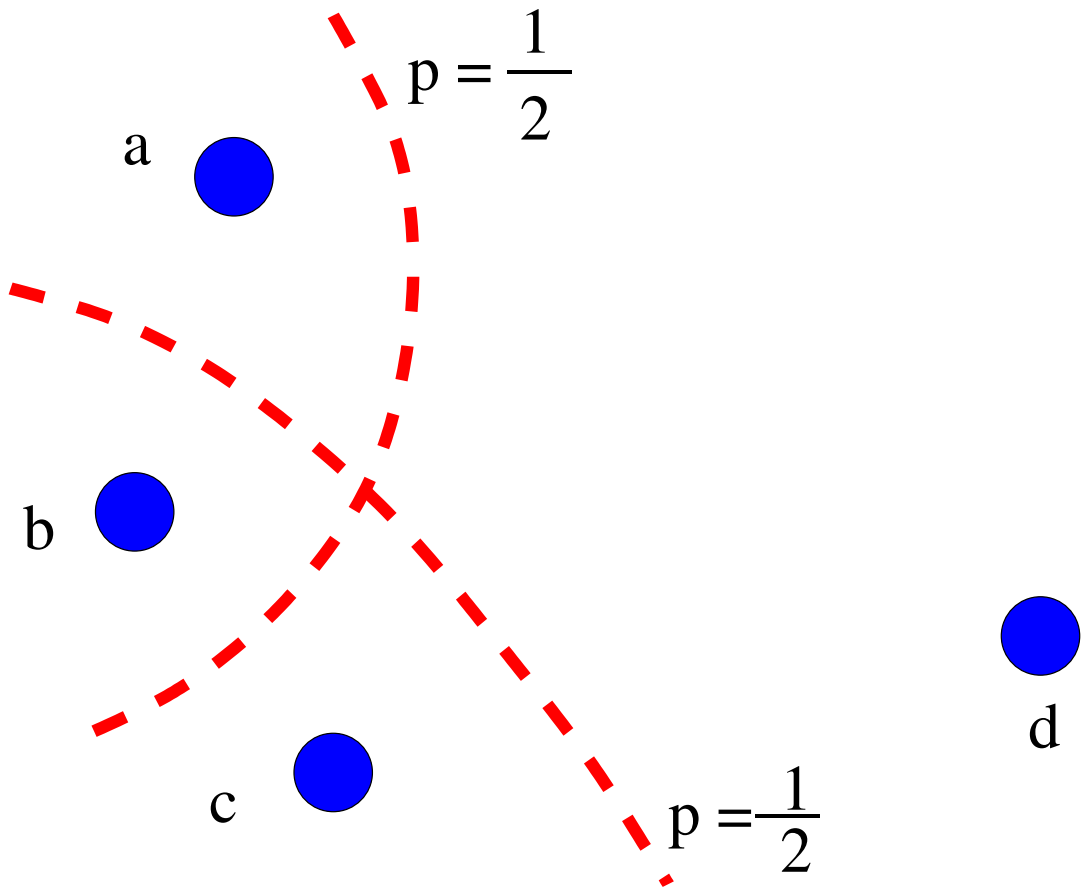
Best Response?

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



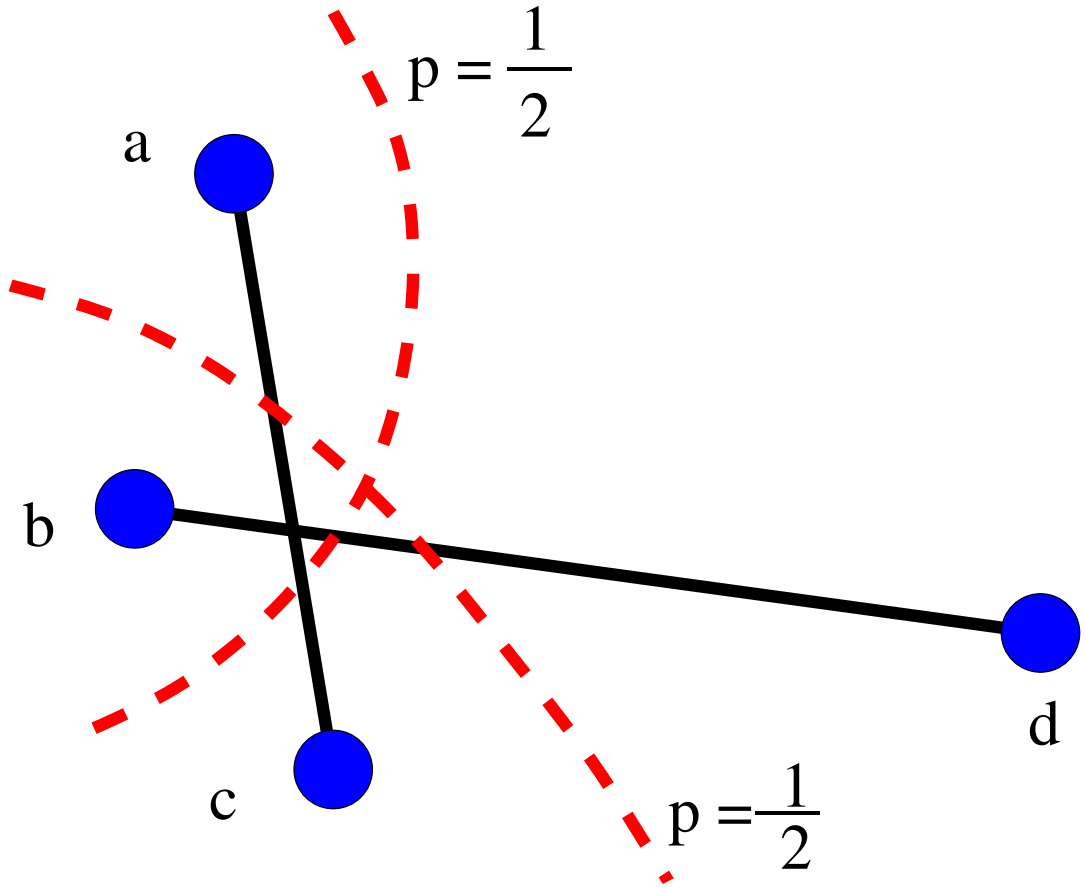
Best Response?

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



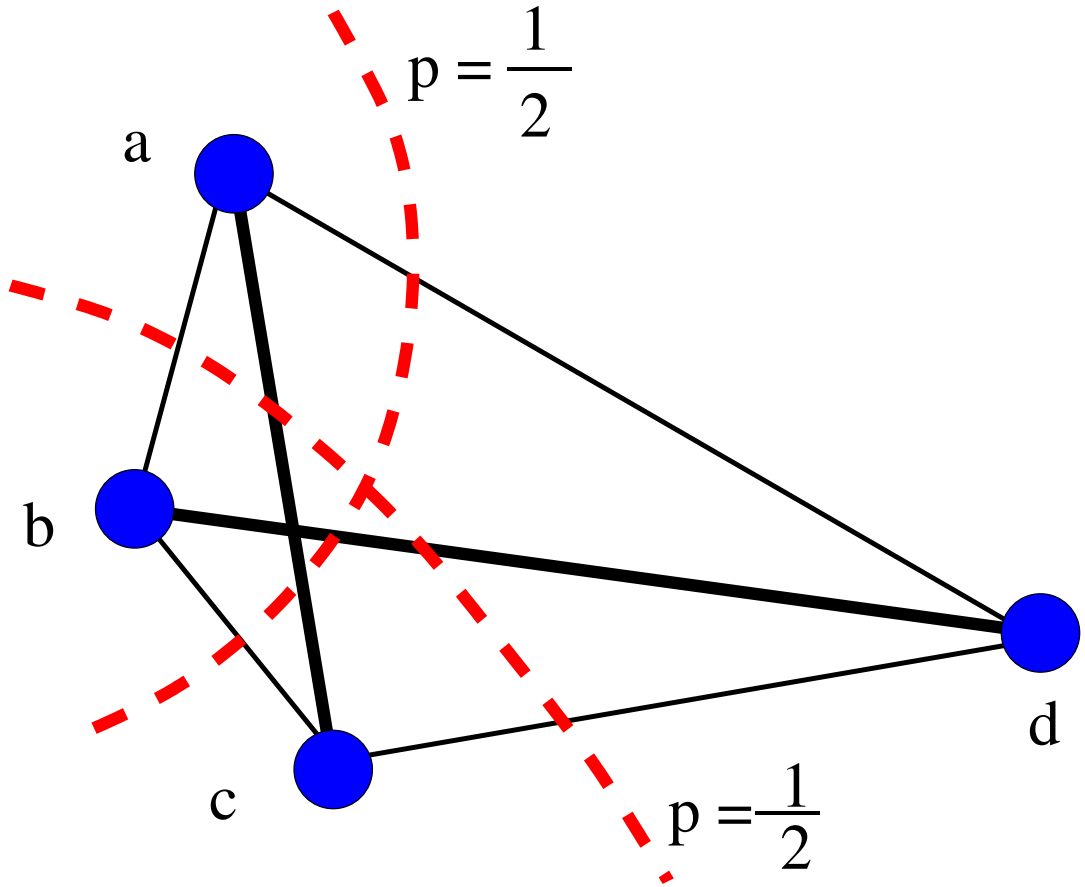
Best Response?

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



Best Response?

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



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**

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]

Approximate Vertex Sparsifiers via 0-Extensions

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a distribution γ on 0-extensions such that $G' = \sum_f \gamma(f) G_f$ is a $O(\frac{\log k}{\log \log k})$-quality Vertex Sparsifier for G, K.

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a polynomial (in n and k) time algorithm to construct G' that is a poly$(\log k)$-quality Vertex Sparsifier for G, K.

Approximate Vertex Sparsifiers via 0-Extensions

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a distribution γ on 0-extensions such that $G' = \sum_f \gamma(f) G_f$ is a $O(\frac{\log k}{\log \log k})$-quality Vertex Sparsifier for G, K.

Theorem

For all graphs $G = (V, E)$ and all sets $K \subset V$, there is a polynomial (in n and k) time algorithm to construct G' that is a poly$(\log k)$-quality Vertex Sparsifier for G, K.

Constructively Finding G'

Unfortunately, this is a whole other talk!

Improvements and Open Questions

Let $P_G \subset \Re^{\binom{k}{2}}$ be the set of all demands that can be routed in G (where demands are restricted to K).

Theorem

There is a graph $G' = (K, E')$ *such that* $P_G \subset P_{G'} \subset O(\frac{\log k}{\log \log k}) P_G$

Improvements and Open Questions

Let $P_G \subset \Re^{\binom{k}{2}}$ be the set of all demands that can be routed in G (where demands are restricted to K).

Theorem

There is a graph $G' = (K, E')$ *such that* $P_G \subset P_{G'} \subset O(\frac{\log k}{\log \log k}) P_G$

This is a strengthening of the results presented here!

Improvements and Open Questions

Let $P_G \subset \Re^{\binom{k}{2}}$ be the set of all demands that can be routed in G (where demands are restricted to K).

Theorem

There is a graph $G' = (K, E')$ such that $P_G \subset P_{G'} \subset O(\frac{\log k}{\log \log k}) P_G$

This is a strengthening of the results presented here!

Open Question

What is the integrality gap of the linear program for 0-extensions?

Improvements and Open Questions

Let $P_G \subset \Re^{\binom{k}{2}}$ be the set of all demands that can be routed in G (where demands are restricted to K).

Theorem

There is a graph $G' = (K, E')$ such that $P_G \subset P_{G'} \subset O(\frac{\log k}{\log \log k}) P_G$

This is a strengthening of the results presented here!

Open Question

What is the integrality gap of the linear program for 0-extensions?

Is it $\sqrt{\log k}$, $\frac{\log k}{\log \log k}$, ...?

Improvements and Open Questions

Theorem (Leighton, M)

There is a graph G and a set K so that for any graph $G' = (K, E')$ which is a better flow network - i.e. $P_G \subset P_{G'}$, $P_{G'} \notin \Omega(\log \log k) P_G$

Improvements and Open Questions

Theorem (Leighton, M)

There is a graph G and a set K so that for any graph $G' = (K, E')$ which is a better flow network - i.e. $P_G \subset P_{G'}$, $P_{G'} \notin \Omega(\log \log k) P_G$

Open Question

What if we are only interested in cuts, and not all flows? Is there an $O(1)$ upper bound?

Improvements and Open Questions

Theorem (Leighton, M)

There is a graph G and a set K so that for any graph $G' = (K, E')$ which is a better flow network - i.e. $P_G \subset P_{G'}$, $P_{G'} \notin \Omega(\log \log k) P_G$

Open Question

What if we are only interested in cuts, and not all flows? Is there an $O(1)$ upper bound?

Equivalently, is there an $O(1)$ upper bound for the linear program for 0-extensions when Δ is an ℓ_1 semi-metric?

Thanks!