Extensions and Limits to Vertex Sparsification

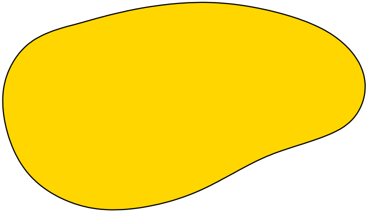
Ankur Moitra, MIT

joint work with Tom Leighton

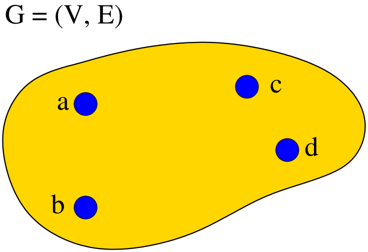
June 5, 2010

Flow Sparsification

$G = (V, E)$

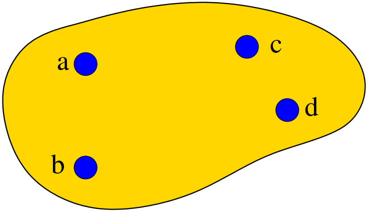


Flow Sparsification



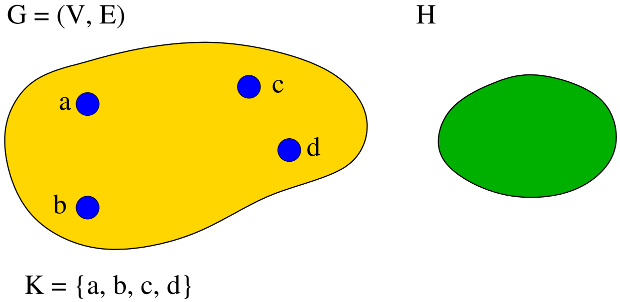
Flow Sparsification

$G = (V, E)$



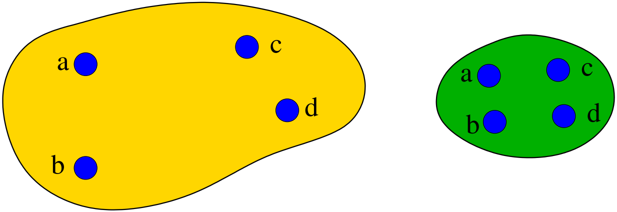
$K = \{a, b, c, d\}$

Flow Sparsification



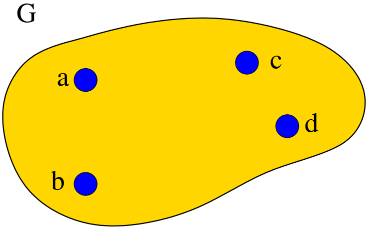
Flow Sparsification

$G = (V, E)$ $\quad$ $H = (K, E_H)$

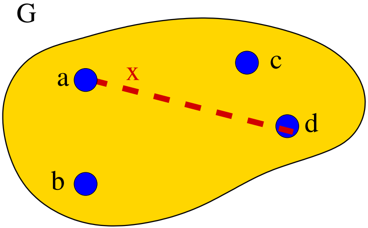


$K = \{a, b, c, d\}$

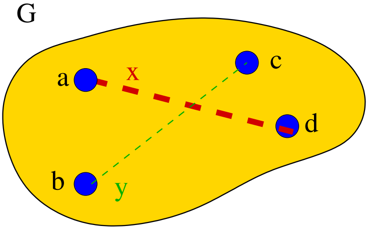
Multicommodity Flow and Congestion



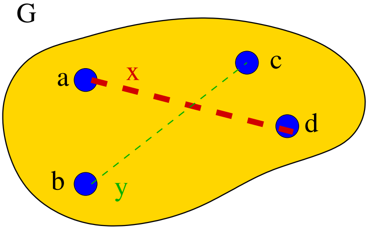
Multicommodity Flow and Congestion



Multicommodity Flow and Congestion

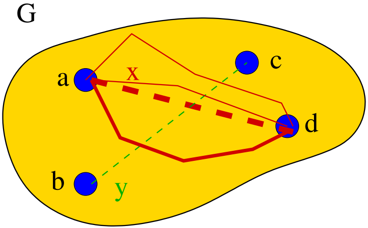


Multicommodity Flow and Congestion



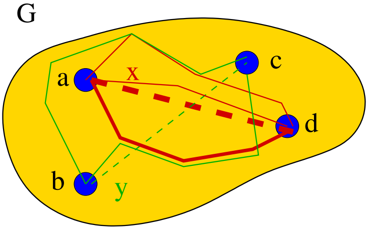
$$\vec{f}=\begin{bmatrix}0\\0\\{\color{red}x}\\{\color{green}y}\\0\\0\end{bmatrix}\begin{matrix}a\longleftrightarrow b\\a\longleftrightarrow c\\a\longleftrightarrow d\\b\longleftrightarrow c\\b\longleftrightarrow d\\c\longleftrightarrow d\end{matrix}$$

Multicommodity Flow and Congestion



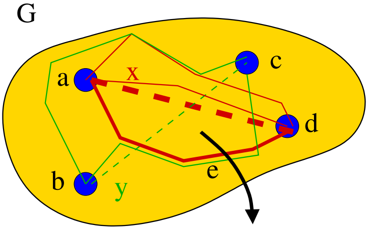
$$\vec{f} = \begin{array}{cc} \begin{bmatrix} 0 \\ 0 \\ x \\ y \\ 0 \\ 0 \end{bmatrix} & \begin{array}{c} a \longleftrightarrow b \\ a \longleftrightarrow c \\ a \longleftrightarrow d \\ b \longleftrightarrow c \\ b \longleftrightarrow d \\ c \longleftrightarrow d \end{array} \end{array}$$

Multicommodity Flow and Congestion



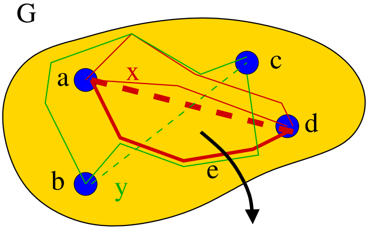
$$\vec{f} = \begin{array}{c} \begin{bmatrix} 0 \\ 0 \\ x \\ y \\ 0 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} a \longleftrightarrow b \\ a \longleftrightarrow c \\ a \longleftrightarrow d \\ b \longleftrightarrow c \\ b \longleftrightarrow d \\ c \longleftrightarrow d \end{array}$$

Multicommodity Flow and Congestion



$$\vec{f} = \begin{array}{cc} \begin{bmatrix} 0 \\ 0 \\ x \\ y \\ 0 \\ 0 \end{bmatrix} & \begin{array}{c} a \longleftrightarrow b \\ a \longleftrightarrow c \\ a \longleftrightarrow d \\ b \longleftrightarrow c \\ b \longleftrightarrow d \\ c \longleftrightarrow d \end{array} \end{array}$$

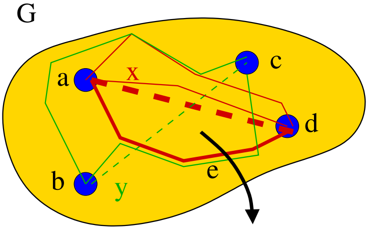
Multicommodity Flow and Congestion



$$\vec{f} = \begin{array}{cc} \begin{bmatrix} 0 \\ 0 \\ \color{red}{x} \\ \color{green}{y} \\ 0 \\ 0 \end{bmatrix} & \begin{array}{c} a \longleftrightarrow b \\ a \longleftrightarrow c \\ a \longleftrightarrow d \\ b \longleftrightarrow c \\ b \longleftrightarrow d \\ c \longleftrightarrow d \end{array} \end{array}$$

$$\text{cong(routing)} = \max_{e} \frac{\text{flow}(e)}{\text{capacity}(e)}$$

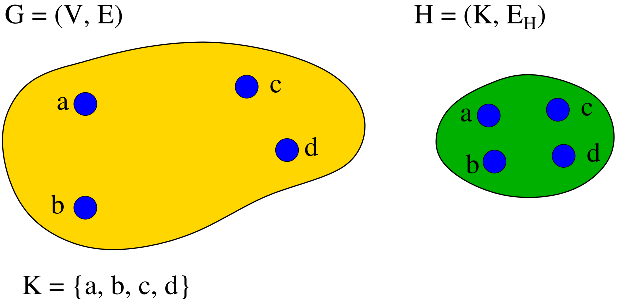
Multicommodity Flow and Congestion



$$\vec{f} = \begin{bmatrix} 0 \\ 0 \\ x \\ y \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} a \leftrightarrow b \\ a \leftrightarrow c \\ a \leftrightarrow d \\ b \leftrightarrow c \\ b \leftrightarrow d \\ c \leftrightarrow d \end{array}$$

$$\text{cong}_G(\vec{f}) = \min_{\text{routings}} \max_{e} \frac{\text{flow}(e)}{\text{capacity}(e)}$$

Flow Sparsification



Definition

A graph $H = (K, E_H)$ <u>on just the terminal set</u> is a **Flow-Sparsifier** if for all demands $\vec{f} \in \Re^{\binom{K}{2}}$

$$cong_H(\vec{f}) \leq cong_G(\vec{f})$$

Definition

A graph $H = (K, E_H)$ on just the terminal set is a **Flow-Sparsifier** if for all demands $\vec{f} \in \Re^{\binom{K}{2}}$

$$cong_H(\vec{f}) \leq cong_G(\vec{f})$$

Definition

A flow-sparsifier H has quality α if additionally for all demands $\vec{f} \in \Re^{\binom{K}{2}}$

$$cong_G(\vec{f}) \leq \alpha cong_H(\vec{f})$$

Definition

A graph $H = (K, E_H)$ on just the terminal set is a **Flow-Sparsifier** if for all demands $\vec{f} \in \Re^{\binom{K}{2}}$

$$cong_H(\vec{f}) \leq cong_G(\vec{f})$$

Definition

A flow-sparsifier H has quality α if additionally for all demands $\vec{f} \in \Re^{\binom{K}{2}}$

$$cong_G(\vec{f}) \leq \alpha cong_H(\vec{f})$$

Quality measures how faithfully H approximates G as a communication network

Our Results I

Theorem

For any undirected, capacitated graph $G = (V, E)$ and any set $K \subset V$ of k terminals, there is an $O(\log k/\log\log k)$-quality flow-sparsifier $H = (K, E_H)$.

Our Results I

Theorem

For any undirected, capacitated graph $G = (V, E)$ and any set $K \subset V$ of k terminals, there is an $O(\log k / \log\log k)$-quality flow-sparsifier $H = (K, E_H)$.

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

Our Results I

Theorem

For any undirected, capacitated graph $G = (V, E)$ and any set $K \subset V$ of k terminals, there is an $O(\log k / \log\log k)$-quality flow-sparsifier $H = (K, E_H)$.

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

Theorem

There is a polynomial (in n and k) time algorithm to compute a $O(\log^2 k / \log\log k)$-quality flow-sparsifier

Previous Results

[M, 09]: There is a graph $H = (K, E_H)$ so that the cut-function of H approximates minimum cuts in G (separating subsets of terminals)

Previous Results

[M, 09]: There is a graph $H = (K, E_H)$ so that the cut-function of H approximates minimum cuts in G (separating subsets of terminals)

We will refer to this as **Cut Sparsification**

Previous Results

[M, 09]: There is a graph $H = (K, E_H)$ so that the cut-function of H approximates minimum cuts in G (separating subsets of terminals)

We will refer to this as **Cut Sparsification**

Flow Sparsification is <u>harder</u> than Cut Sparsification

Our Results I

Theorem

For any undirected, capacitated graph $G = (V, E)$ and any set $K \subset V$ of k terminals, there is an $O(\log k / \log \log k)$-quality flow-sparsifier $H = (K, E_H)$.

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

Theorem

There is a polynomial (in n and k) time algorithm to compute a $O(\log^2 k / \log \log k)$-quality flow-sparsifier

Our Results II

Theorem

There is an infinite family of graphs and sets of terminals for which any flow-sparsifier has quality $\Omega(\log \log k)$

Our Results II

Theorem

There is an infinite family of graphs and sets of terminals for which any flow-sparsifier has quality $\Omega(\log \log k)$

Motivation for vertex sparsification: obtain approximation algorithms with guarantees independent of n

Our Results II

Theorem

There is an infinite family of graphs and sets of terminals for which any flow-sparsifier has quality $\Omega(\log \log k)$

Motivation for vertex sparsification: obtain approximation algorithms with guarantees independent of n

Approximation algorithms that reduce to a k-terminal graph must lose a **super-constant** factor in the approximation guarantee

Outline

Outline

1. Alternate Definition of Quality

Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification

Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification

Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

Computing Quality

Suppose we are given a flow-sparsifier H

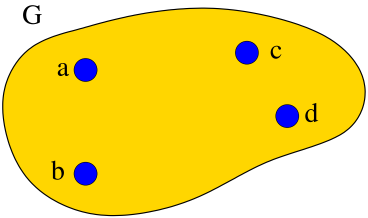
Computing Quality

Suppose we are given a flow-sparsifier H

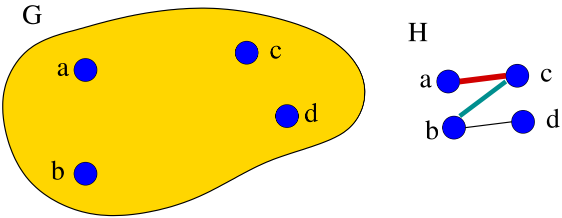
Question

Can we compute the quality of H?

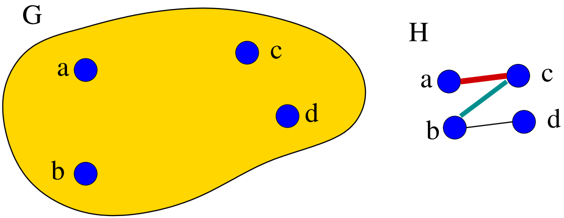
Computing Quality



Computing Quality

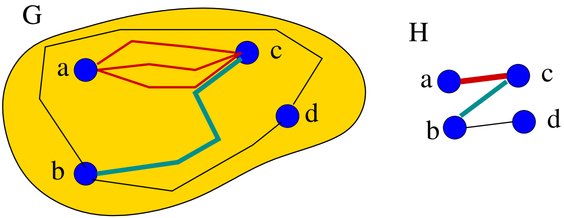


Computing Quality



$$\vec{f}_H = \begin{bmatrix} c_H(a,b) \\ c_H(a,c) \\ c_H(a,d) \\ c_H(b,c) \\ c_H(b,d) \\ c_H(c,d) \end{bmatrix}$$

Computing Quality



$$\vec{f}_H = \begin{bmatrix} c_H(a,b) \\ c_H(a,c) \\ c_H(a,d) \\ c_H(b,c) \\ c_H(b,d) \\ c_H(c,d) \end{bmatrix}$$

Computing Quality

Suppose we are given a flow-sparsifier H

Question

Can we compute the quality of H?

Computing Quality

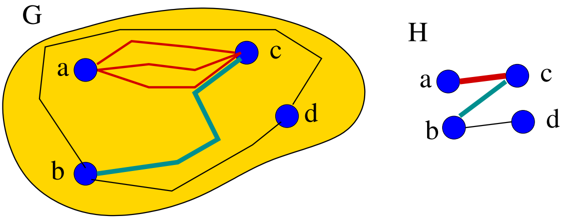
Suppose we are given a flow-sparsifier H

Question

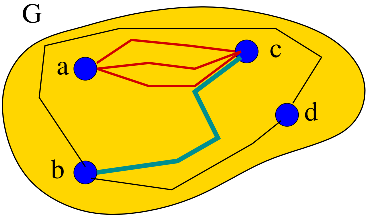
Can we compute the quality of H?

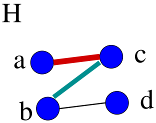
The quality is at least $cong_G(\vec{f}_H)$

Computing Quality



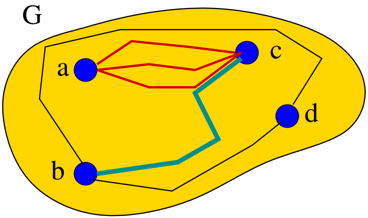
Computing Quality

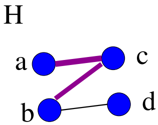




$$\vec{f} = \begin{bmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

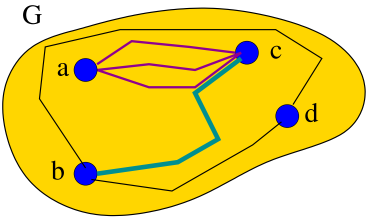
Computing Quality

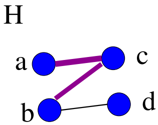




$$\vec{f} = \begin{bmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

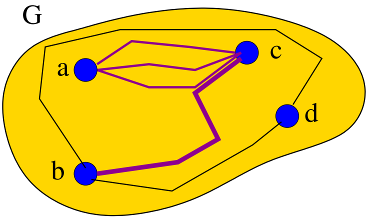
Computing Quality

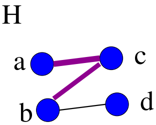




$$\vec{f} = \begin{bmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Computing Quality





$$\vec{f} = \begin{bmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Computing Quality

Suppose we are given a flow-sparsifier H

Question

Can we compute the quality of H?

The quality is at least $cong_G(\vec{f}_H)$

Computing Quality

Suppose we are given a flow-sparsifier H

Question

Can we compute the quality of H?

The quality is ~~at least~~ equal to $cong_G(\vec{f}_H)$

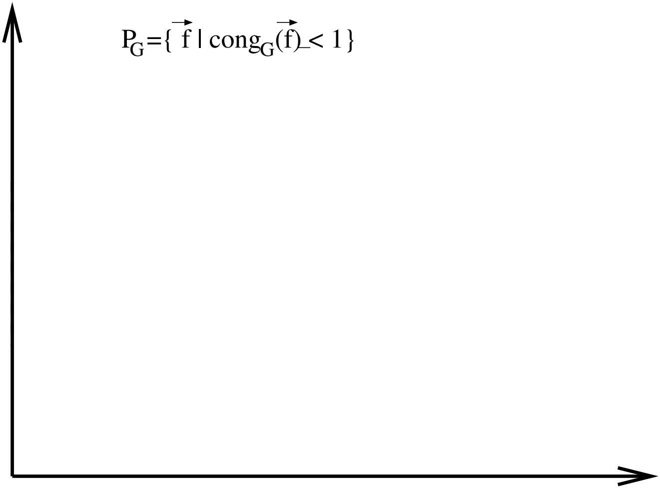
Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

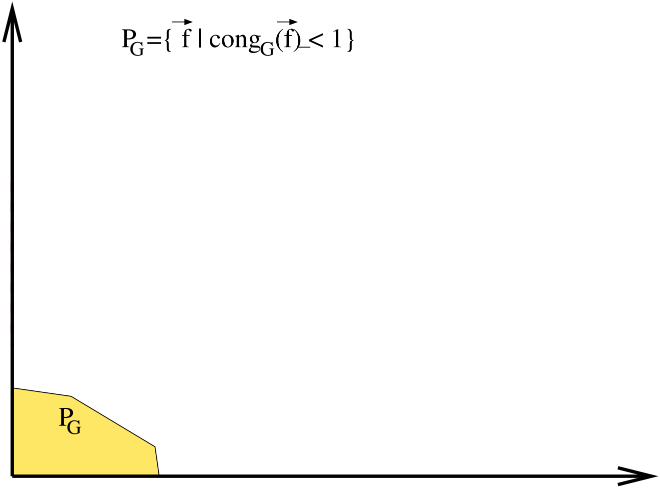
Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

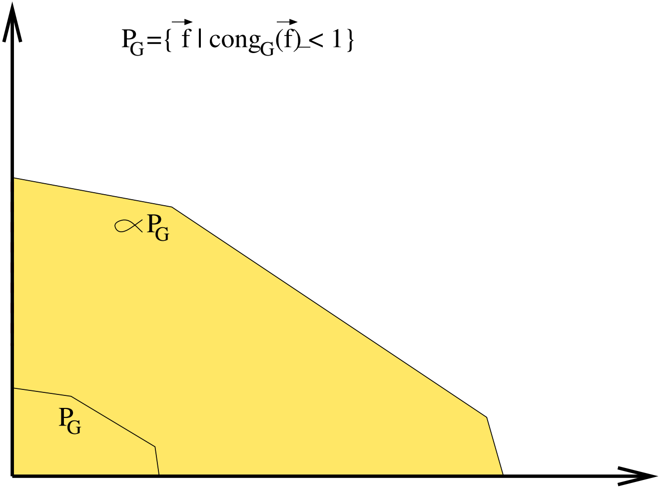
Geometry of Flow Sparsification



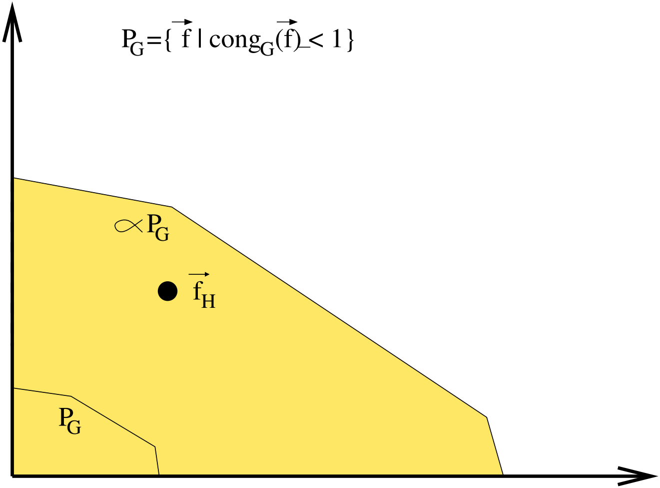
Geometry of Flow Sparsification



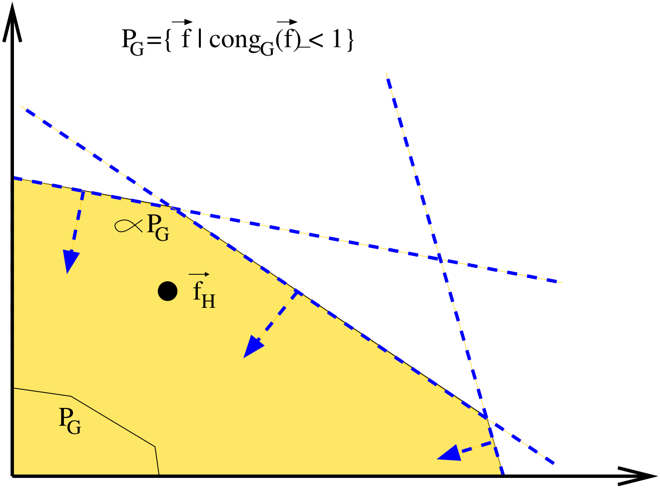
Geometry of Flow Sparsification



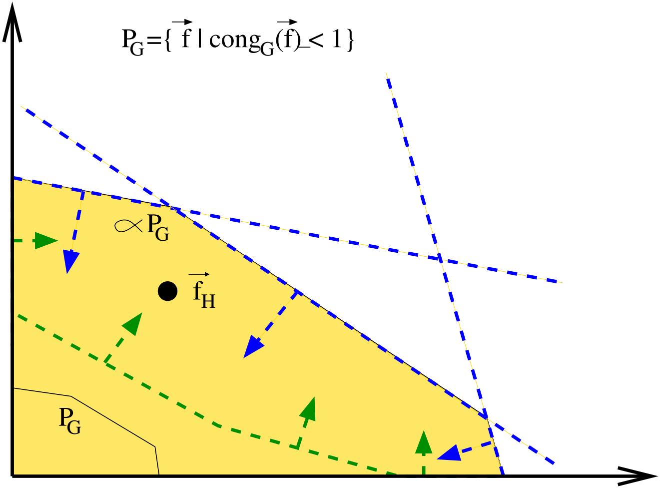
Geometry of Flow Sparsification



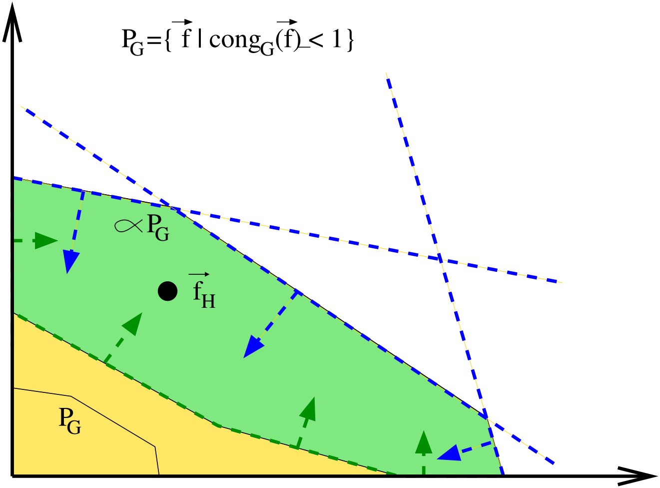
Geometry of Flow Sparsification



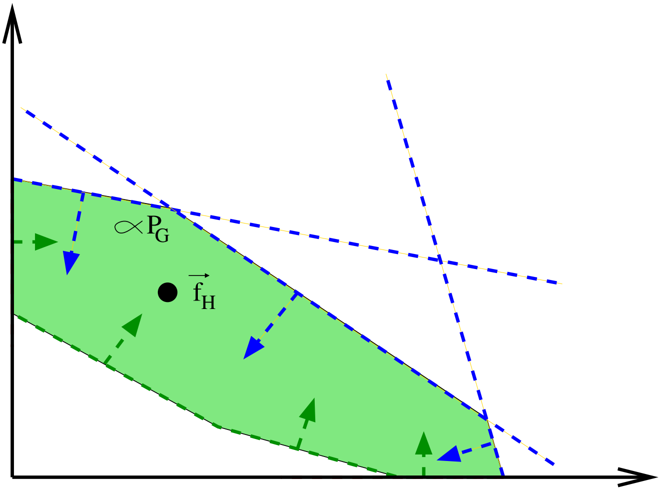
Geometry of Flow Sparsification



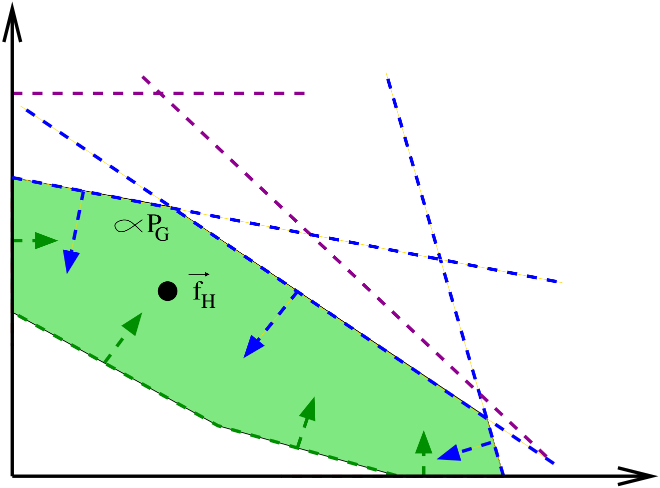
Geometry of Flow Sparsification



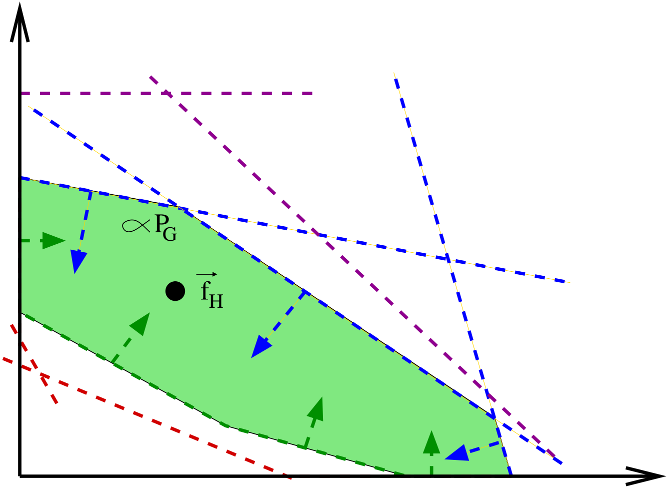
What About Cut Sparsification?



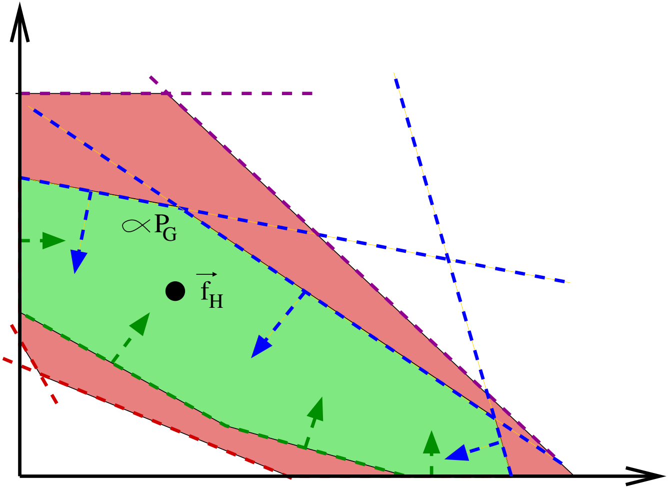
What About Cut Sparsification?



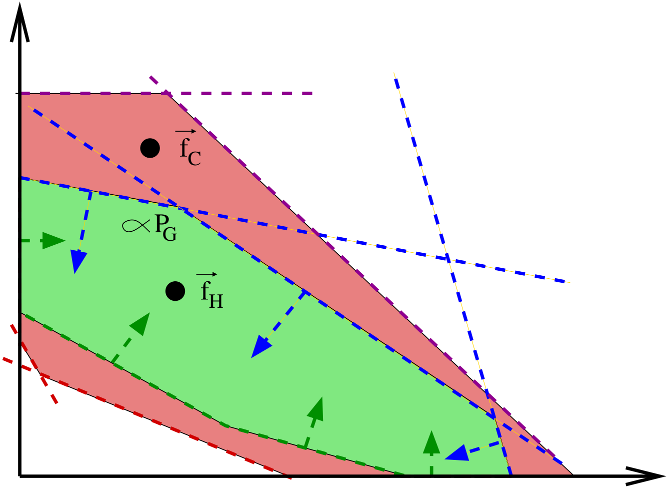
What About Cut Sparsification?



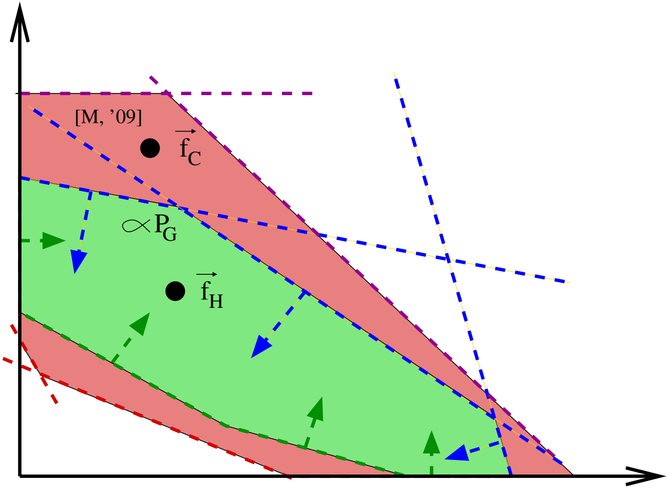
What About Cut Sparsification?



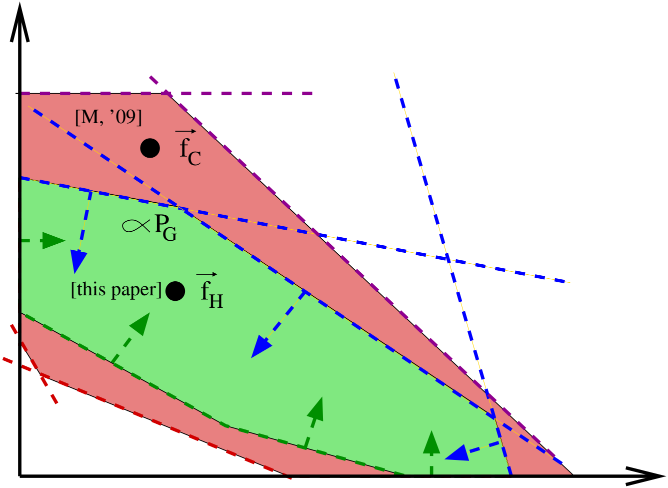
What About Cut Sparsification?



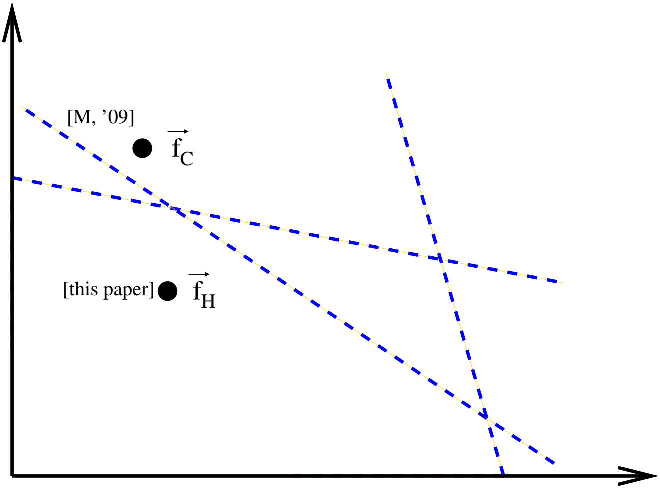
What About Cut Sparsification?



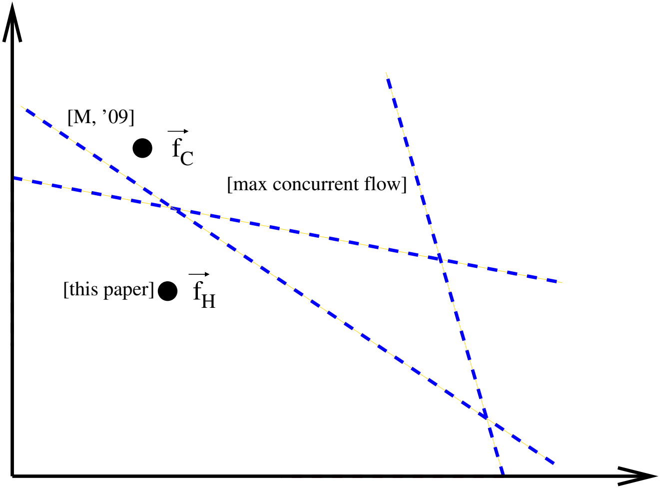
What About Cut Sparsification?



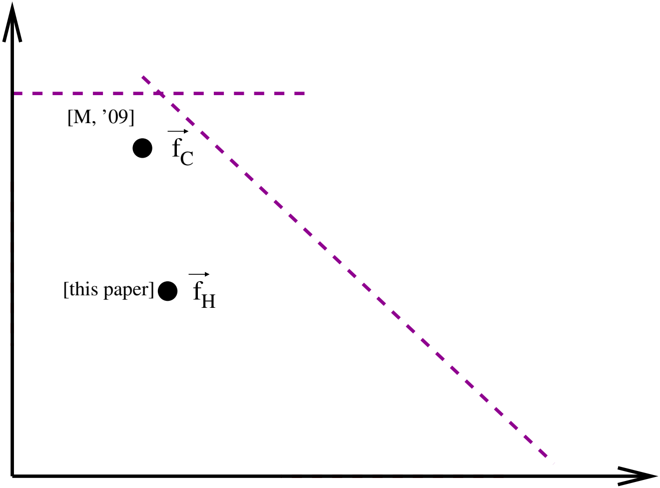
What About Cut Sparsification?



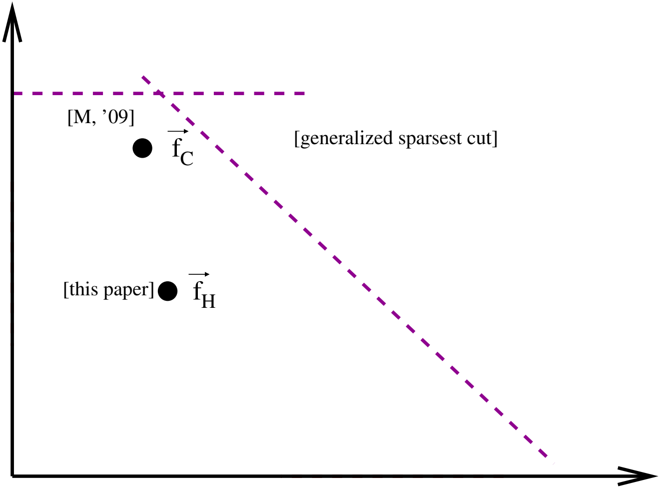
What About Cut Sparsification?



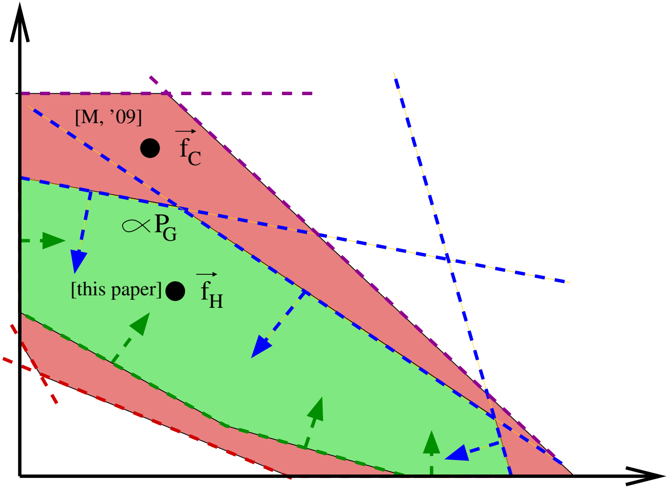
What About Cut Sparsification?



What About Cut Sparsification?



What About Cut Sparsification?



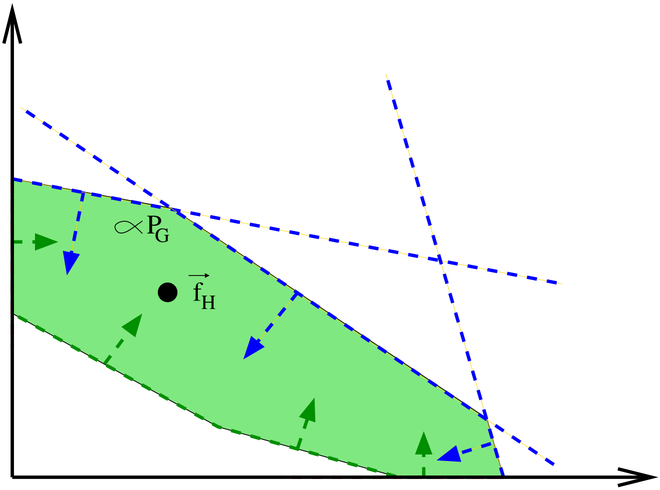
Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

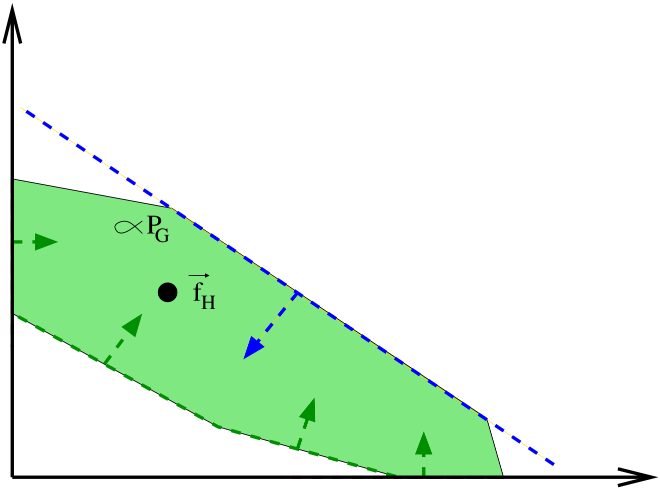
Outline

1. Alternate Definition of Quality
2. Geometric Interpretation of Vertex Sparsification
3. Lower Bound for Flow Sparsification
4. Open Questions

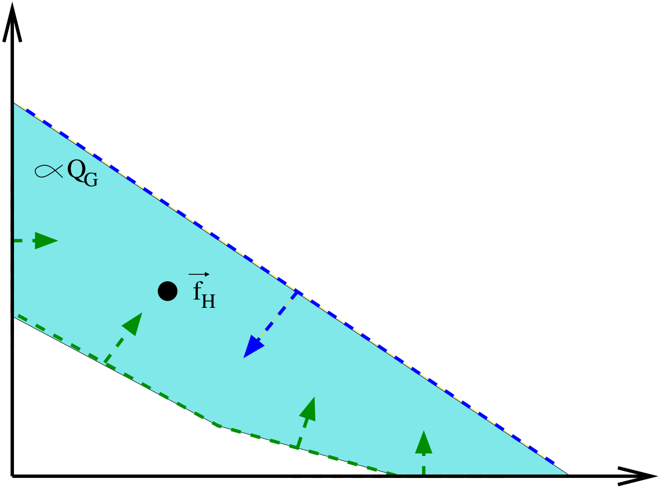
Proving Lower Bounds on Expanders



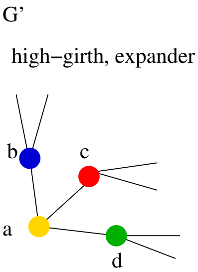
Proving Lower Bounds on Expanders



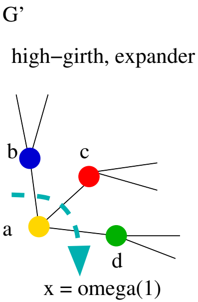
Proving Lower Bounds on Expanders



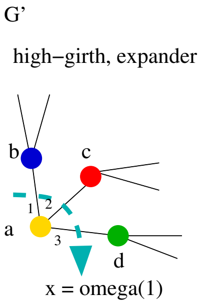
The Graph



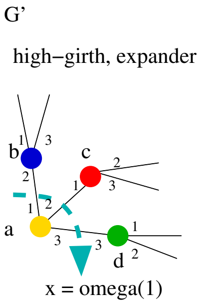
The Graph



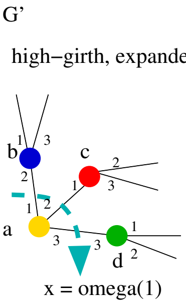
The Graph

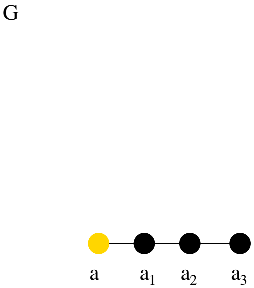


The Graph

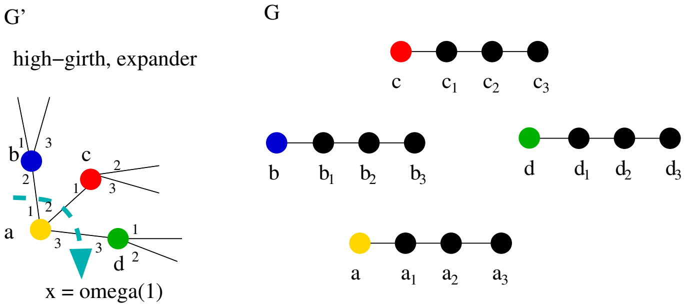


The Graph

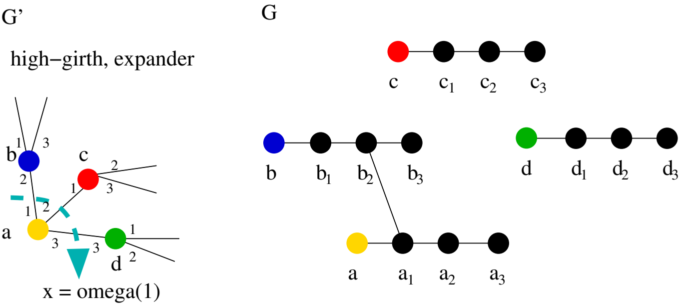




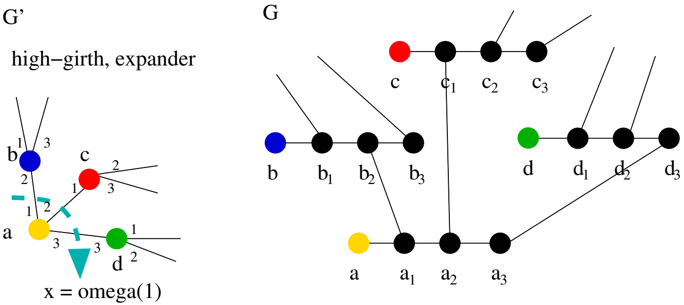
The Graph


G'
high−girth, expander
b
c
a
d
1
3
2
1
2
3
1
2
3
1
2
3
x = omega(1)
G
c
c_1
c_2
c_3
b
b_1
b_2
b_3
d
d_1
d_2
d_3
a
a_1
a_2
a_3

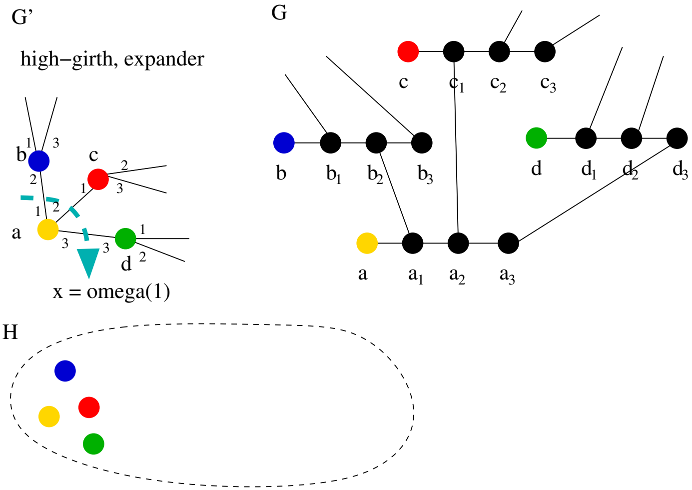
The Graph


G'
high−girth, expander
b
1
3
2
c
2
3
1
1
2
a
3
3
d
1
2
x = omega(1)
G
c
c1
c2
c3
b
b1
b2
b3
d
d1
d2
d3
a
a1
a2
a3

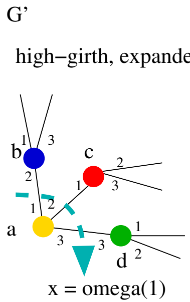
The Graph

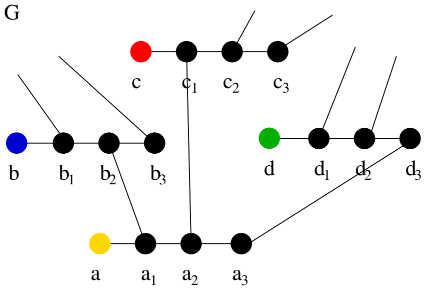

G'
high−girth, expander
b
c
a
d
1
2
3
x = omega(1)
G
c
c1
c2
c3
b
b1
b2
b3
d
d1
d2
d3
a
a1
a2
a3

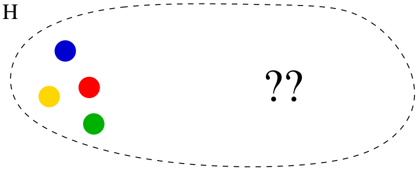
A Thought Experiment



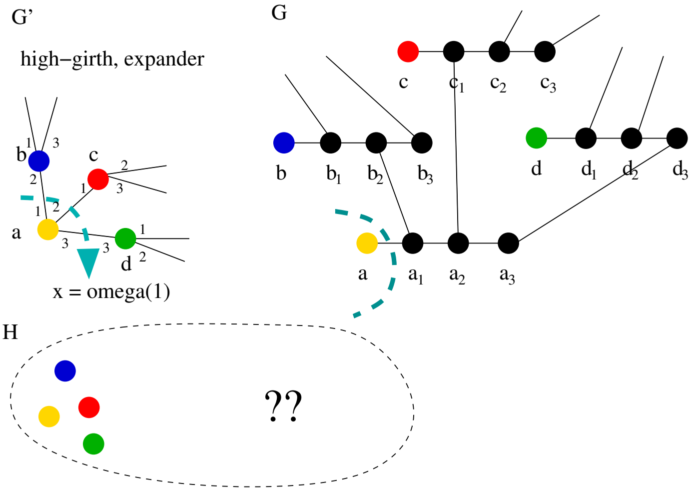
A Thought Experiment



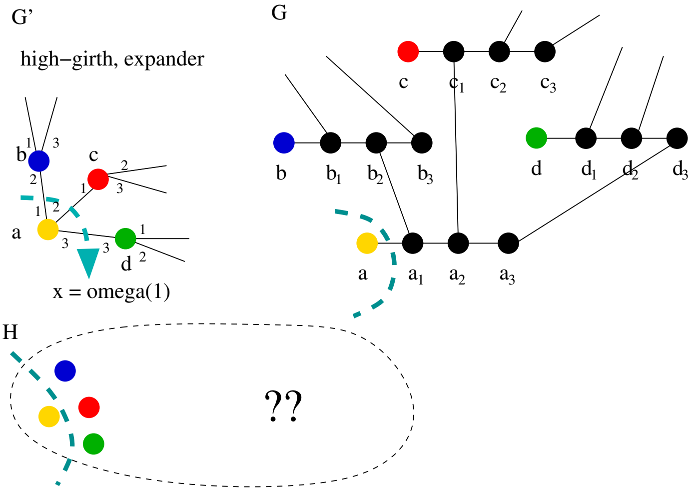




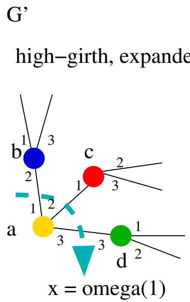
A Thought Experiment

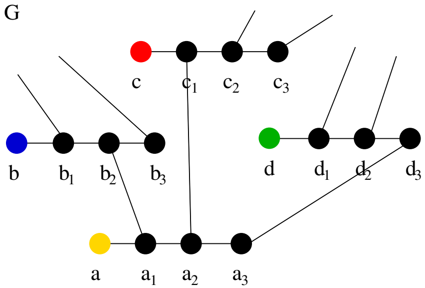


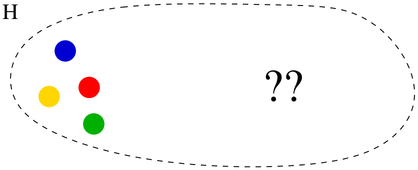
A Thought Experiment



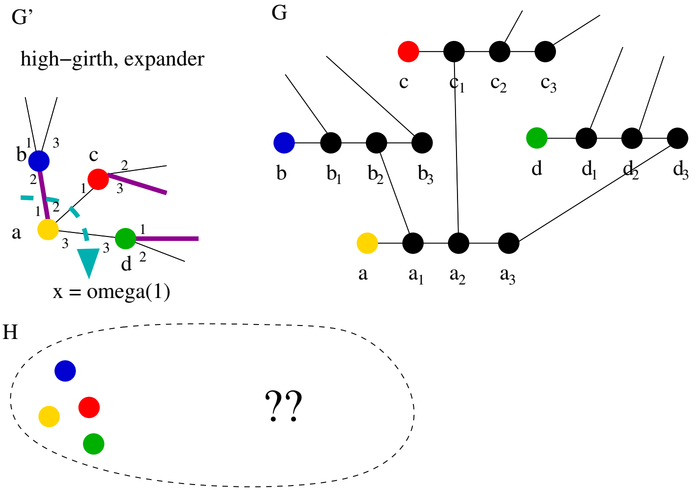
A Thought Experiment



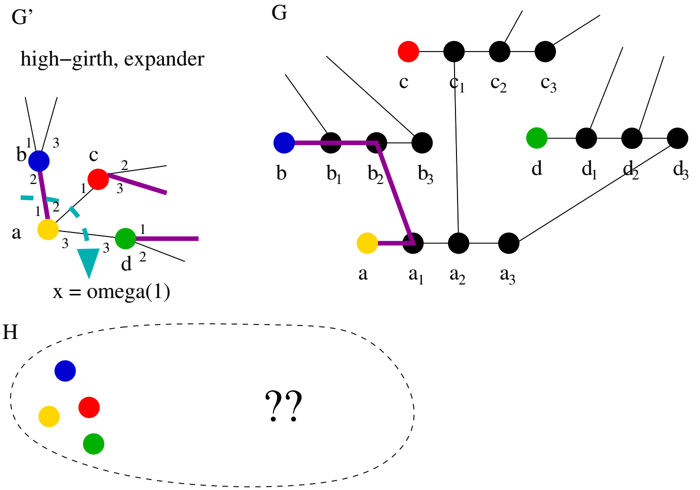




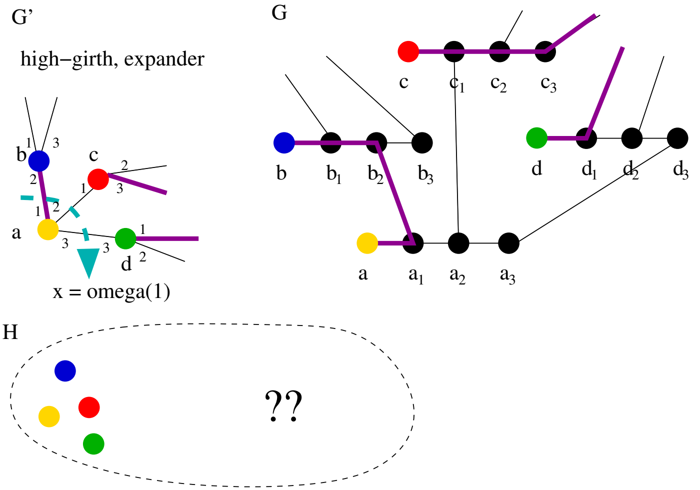
A Thought Experiment



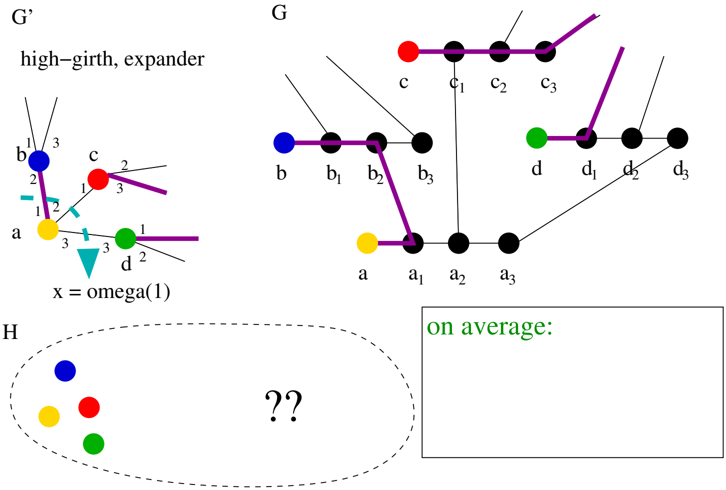
A Thought Experiment



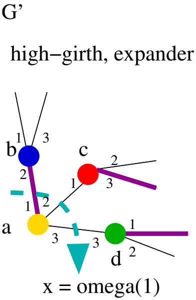
A Thought Experiment

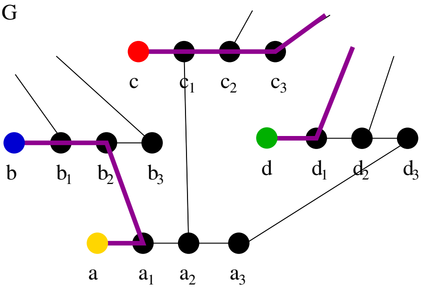


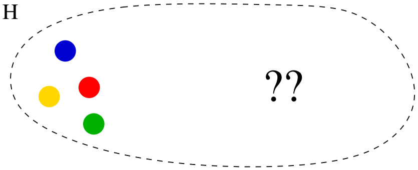
A Thought Experiment



A Thought Experiment



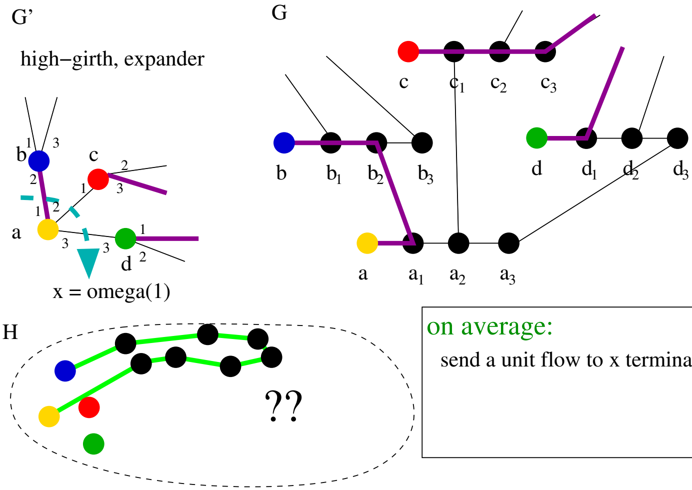




on average:

send a unit flow to x terminals

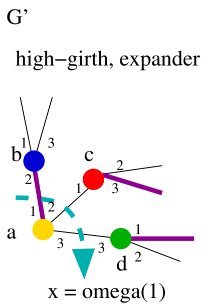
A Thought Experiment

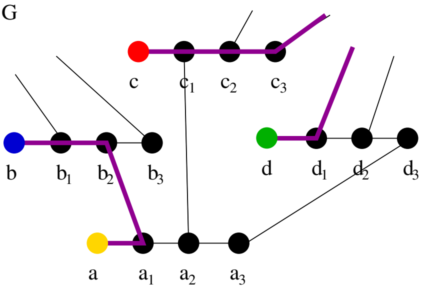


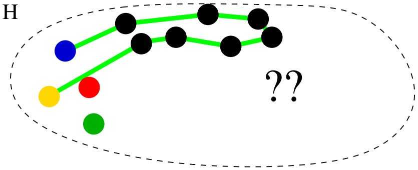
on average:

send a unit flow to x terminals

A Thought Experiment





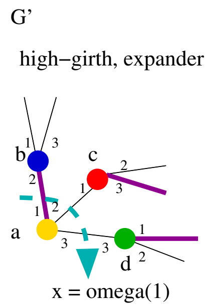


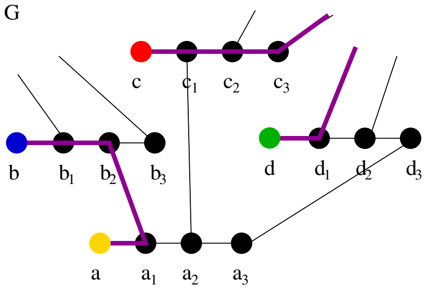
on average:

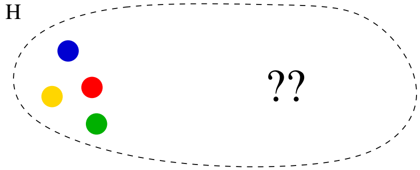
send a unit flow to x terminals

using short paths

A Thought Experiment





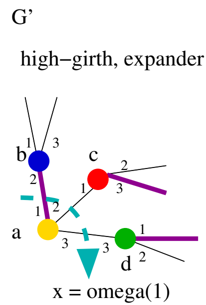


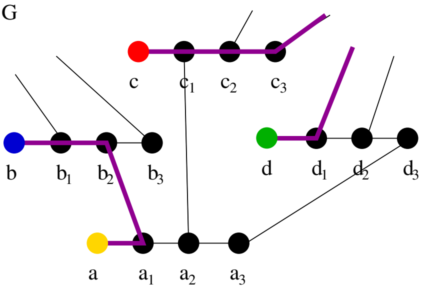
on average:

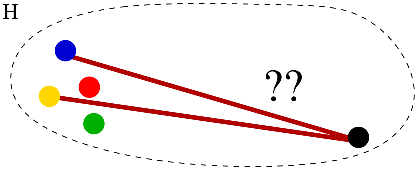
send a unit flow to x terminals

using short paths

A Thought Experiment





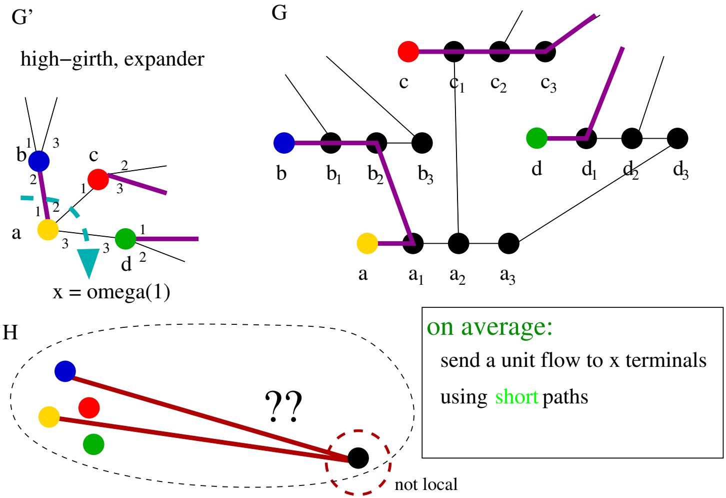


on average:

send a unit flow to x terminals

using short paths

A Thought Experiment

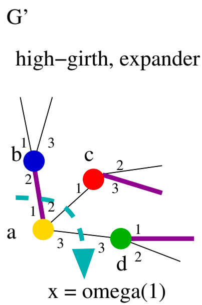


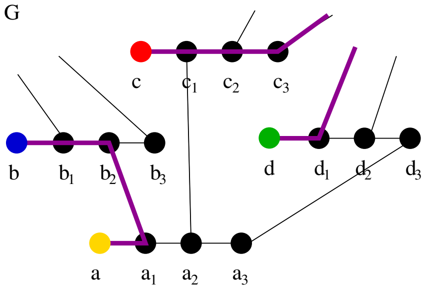
on average:

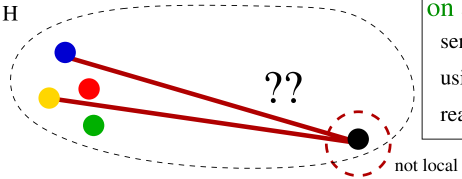
send a unit flow to x terminals

using short paths

A Thought Experiment







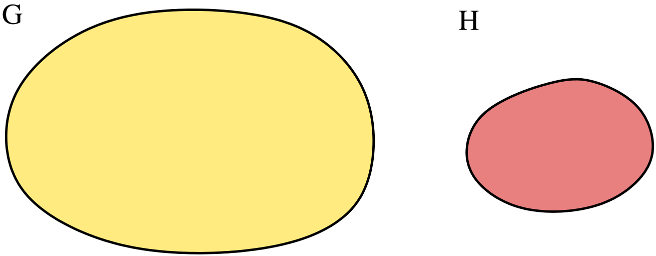
on average:

send a unit flow to x terminals

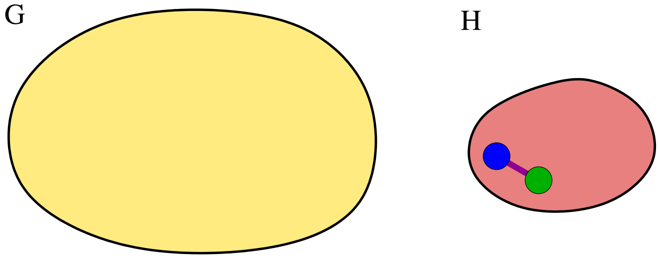
using short paths

reaching only local terminals

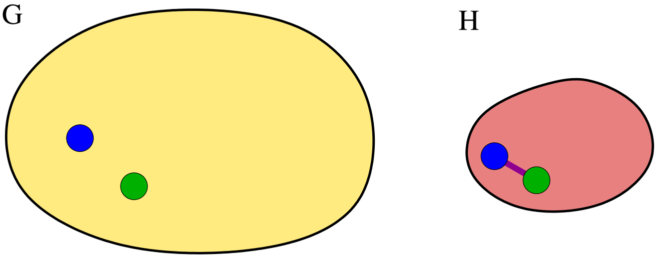
Girth-Routed Edges



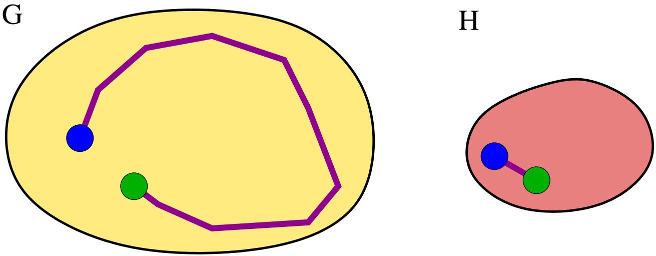
Girth-Routed Edges



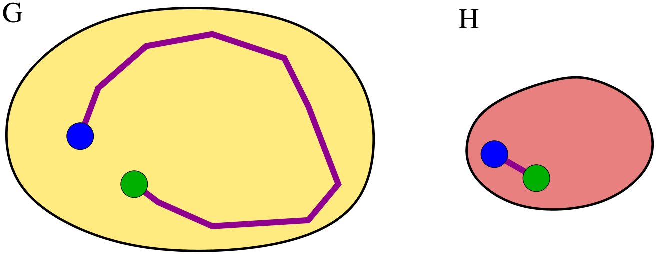
Girth-Routed Edges



Girth-Routed Edges

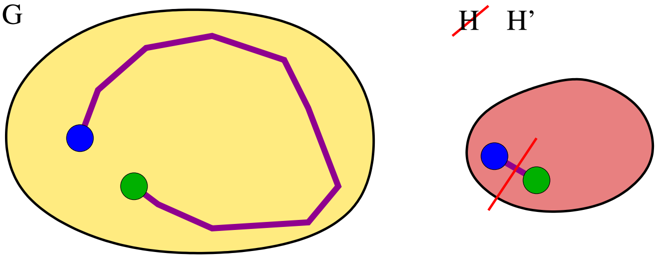


Girth-Routed Edges



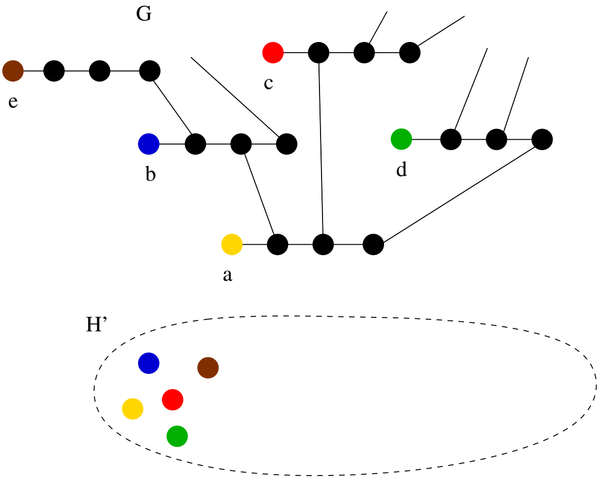
not too many edges can be routed on girth paths

Girth-Routed Edges

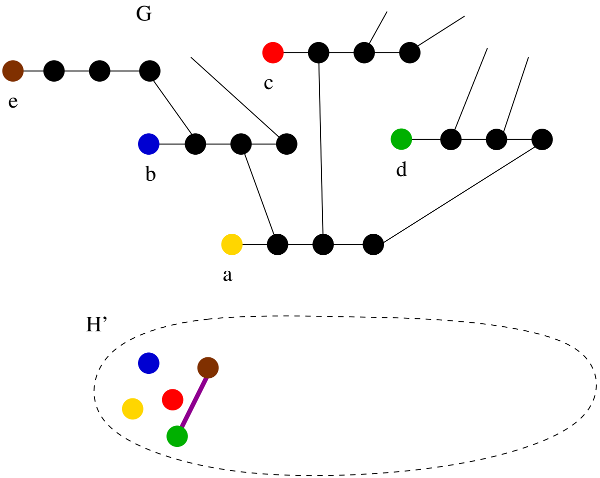


not too many edges can be routed on girth paths

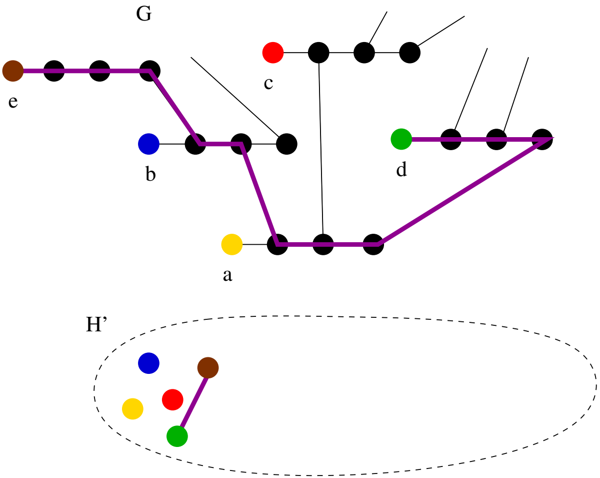
A Reduction to Cut-Width



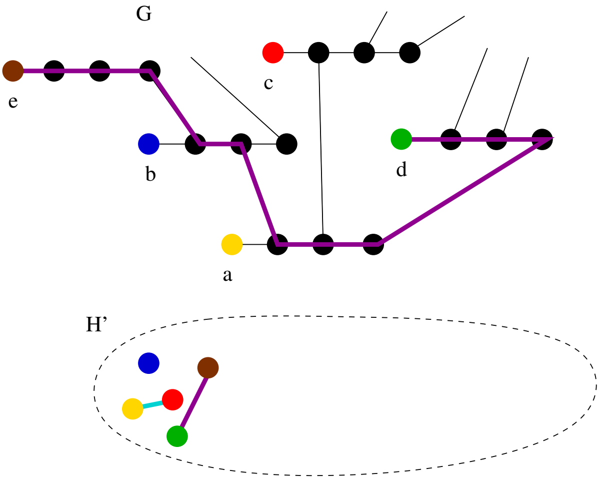
A Reduction to Cut-Width



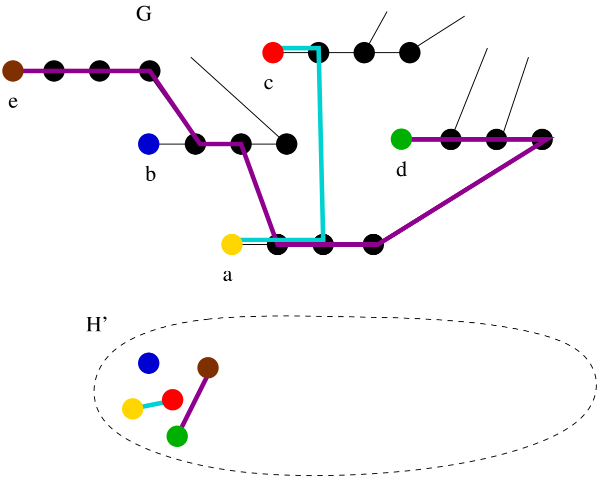
A Reduction to Cut-Width



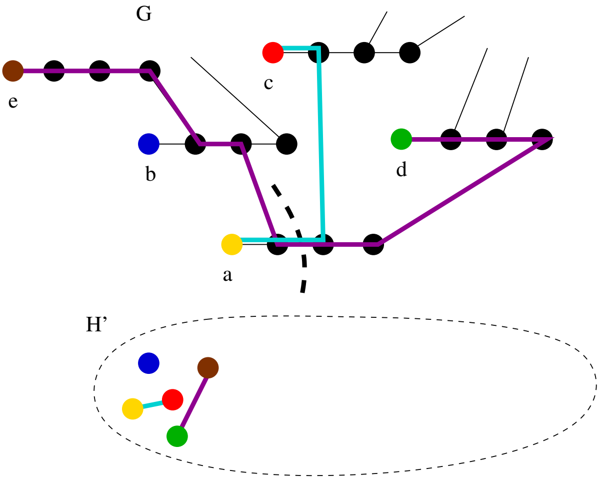
A Reduction to Cut-Width



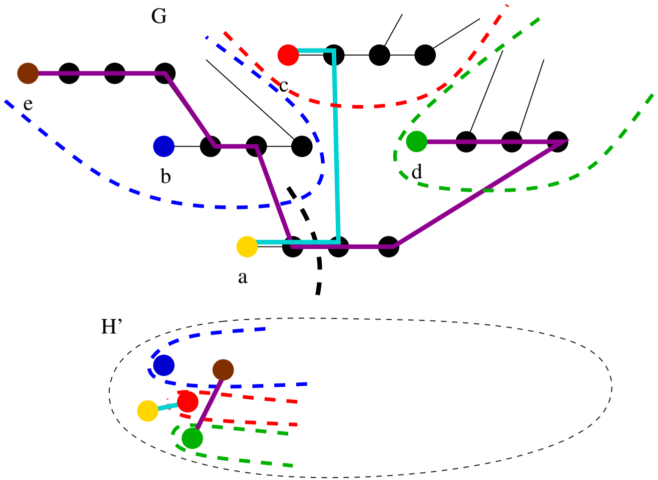
A Reduction to Cut-Width



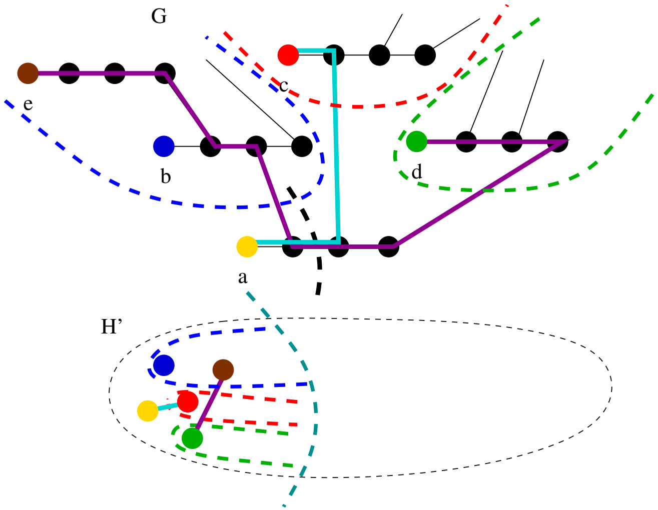
A Reduction to Cut-Width



A Reduction to Cut-Width



A Reduction to Cut-Width



Recent Work

Recent work on both **upper bounds** and **lower bounds**

Recent Work

Recent work on both **upper bounds** and **lower bounds**

1. [Englert, Gupta, Krauthgamer, Raecke, Talgam, Talwar]
2. [Makarychev, Makarychev]
3. [Charikar, Leighton, Li, M]

Recent Work

Recent work on both **upper bounds** and **lower bounds**

1. [Englert, Gupta, Krauthgamer, Raecke, Talgam, Talwar]
2. [Makarychev, Makarychev]
3. [Charikar, Leighton, Li, M]

... super-constant lower bounds for cut sparsification, and **constructive** results that match our **existential** results

Open Questions

Question

Can the approximation for 0*-extension be improved?*

Open Questions

Question

Can the approximation for 0-extension be improved?

Question

What if the semi-metric is ℓ_1?

(immediately implies improvements to cut-sparsification)

Open Questions

Question

Can the approximation for 0-extension be improved?

Question

What if the semi-metric is ℓ_1?

(immediately implies improvements to cut-sparsification)

Question

Is flow-sparsification easier than the 0-extension problem?

Questions?

Thanks!