$(2-\epsilon)$-approximate minimum k-cut in FPT time

Jason Li

Joint work with Anupam Gupta, Euiwoong Lee

Carnegie Mellon University

January 10, 2018

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2 - \epsilon)$-approx is NP-hard assuming SSEH

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2 - \epsilon)$-approx is NP-hard assuming SSEH

Constant k

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2 - \epsilon)$-approx is NP-hard assuming SSEH

Constant k

- Karger's random edge sampling: exact $O(n^{2k})$ time

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2 - \epsilon)$-approx is NP-hard assuming SSEH

Constant k

- Karger's random edge sampling: exact $O(n^{2k})$ time

FPT in k (running time $f(k)n^{O(1)}$)

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2-\epsilon)$-approx is NP-hard assuming SSEH

Constant k

- Karger's random edge sampling: exact $O(n^{2k})$ time

FPT in k (running time $f(k)n^{O(1)}$)

- Exact is W[1]-hard parameterized by k

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2-\epsilon)$-approx is NP-hard assuming SSEH

Constant k

- Karger's random edge sampling: exact $O(n^{2k})$ time

FPT in k (running time $f(k)n^{O(1)}$)

- Exact is W[1]-hard parameterized by k
- Approx parameterized by k?

Min k-cut problem

- Given graph G, remove min weight set of edges that splits G into $\geq k$ connected components

General k

- [SV95] Greedy 2-approx
- [Man17] $(2 - \epsilon)$-approx is NP-hard assuming SSEH

Constant k

- Karger's random edge sampling: exact $O(n^{2k})$ time

FPT in k (running time $f(k)n^{O(1)}$)

- Exact is W[1]-hard parameterized by k
- Approx parameterized by k?
- **This talk:** 1.9997**-approx in** $2^{O(k^6)} \cdot \tilde{O}(n^4)$ **time.**

High level idea

- Follow the greedy 2-approx algorithm of [SV95]

High level idea

- Follow the greedy 2-approx algorithm of [SV95]
- If graph does not have a specific structure, then a simple modification of [SV95] already guarantees $(2-\epsilon)$-approx

High level idea

- Follow the greedy 2-approx algorithm of [SV95]
- If graph does not have a specific structure, then a simple modification of [SV95] already guarantees $(2 - \epsilon)$-approx
- Separate $(2 - \epsilon)$-approx algorithm that exploits this structure

2-approx [SV95]

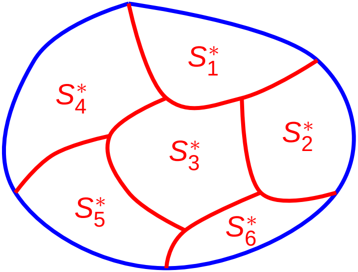
- For $k - 1$ iterations, greedily take the min global cut

2-approx [SV95]

- For $k - 1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1

2-approx [SV95]

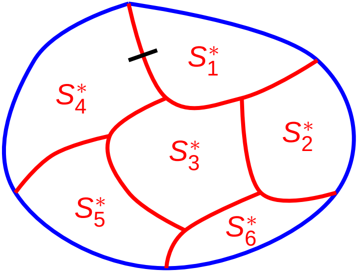
- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

2-approx [SV95]

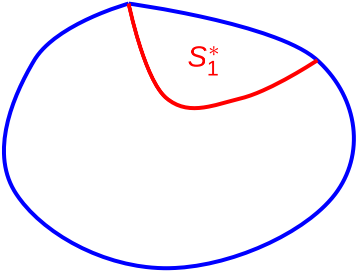
- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$
$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

2-approx [SV95]

- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:

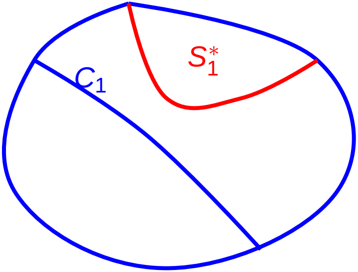


$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

2-approx [SV95]

- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

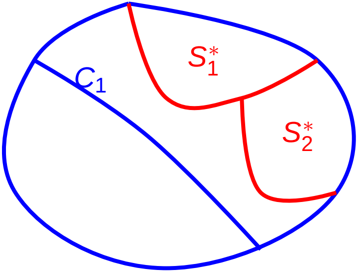
$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

∂S_1^* is possible cut

$$\implies |C_1| \leq |\partial S_1^*|$$

2-approx [SV95]

- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

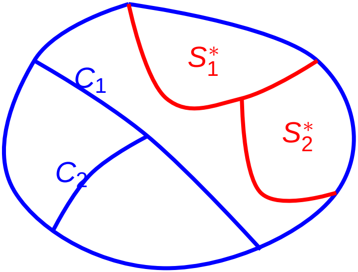
$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

∂S_1^* is possible cut

$$\implies |C_1| \leq |\partial S_1^*|$$

2-approx [SV95]

- For $k - 1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

∂S_1^* is possible cut

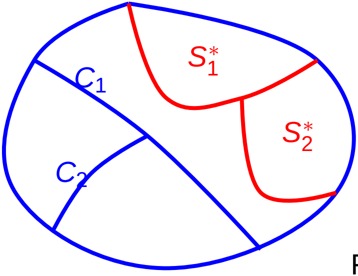
$$\implies |C_1| \leq |\partial S_1^*|$$

Either ∂S_1^* or ∂S_2^* is possible cut

$$\implies |C_2| \leq |\partial S_2^*|$$

2-approx [SV95]

- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \le |\partial S_2^*| \le \cdots \le |\partial S_k^*|$$

$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

∂S_1^* is possible cut

$$\implies |C_1| \le |\partial S_1^*|$$

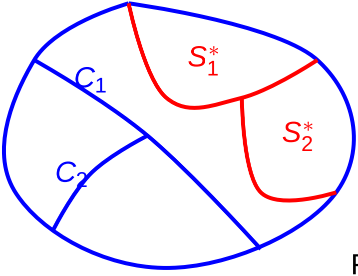
Either ∂S_1^* or ∂S_2^* is possible cut

$$\implies |C_2| \le |\partial S_2^*|$$

For all $i \in [k-1]$: $|C_i| \le |\partial S_i^*|$

2-approx [SV95]

- For $k-1$ iterations, greedily take the min global cut
- Min cut that increases # connected components by 1
- Consider OPT:



$$|\partial S_1^*| \leq |\partial S_2^*| \leq \cdots \leq |\partial S_k^*|$$

$$\sum_{i=1}^{k} |\partial S_i^*| = 2 \cdot OPT$$

∂S_1^* is possible cut

$$\implies |C_1| \leq |\partial S_1^*|$$

Either ∂S_1^* or ∂S_2^* is possible cut

$$\implies |C_2| \leq |\partial S_2^*|$$

For all $i \in [k-1]$: $|C_i| \leq |\partial S_i^*|$

$$ALG = \sum_{i=1}^{k-1} |C_i| \leq \sum_{i=1}^{k-1} |\partial S_i^*| \leq 2 \cdot OPT$$

Branching

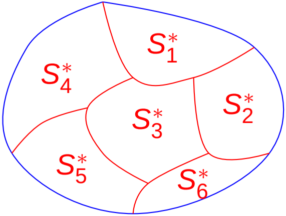
- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.

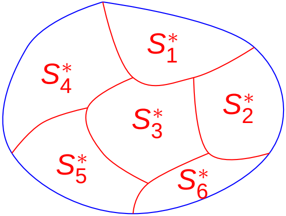
Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.



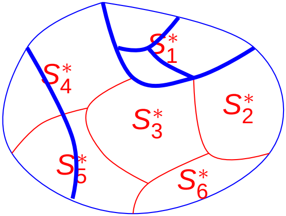
-

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

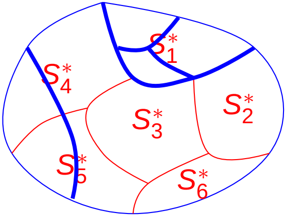
- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

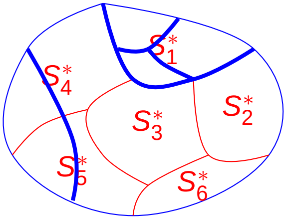
- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!

Branching

- Recall that $|C_i| \le |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

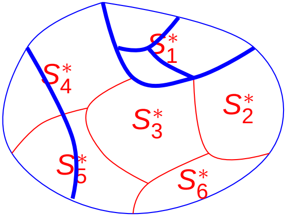
- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!
- Consider the algorithm's components so far. The union of some components is exactly S_1^*.

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

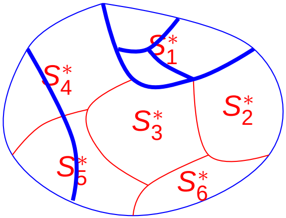
- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!
- Consider the algorithm's components so far. The union of some components is exactly S_1^*.
- Idea: guess all subsets of components

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!
- Consider the algorithm's components so far. The union of some components is exactly S_1^*.
- Idea: guess all subsets of components
- Branching factor: 2^k, branching depth: $k \implies 2^{k^2}$ time

Branching

- Recall that $|C_i| \leq |\partial S_i^*|$ for all i. What about $|C_i|$ vs. $|\partial S_1^*|$?
- Suppose, at some iteration, $|C_i| > |\partial S_1^*|$.
- 

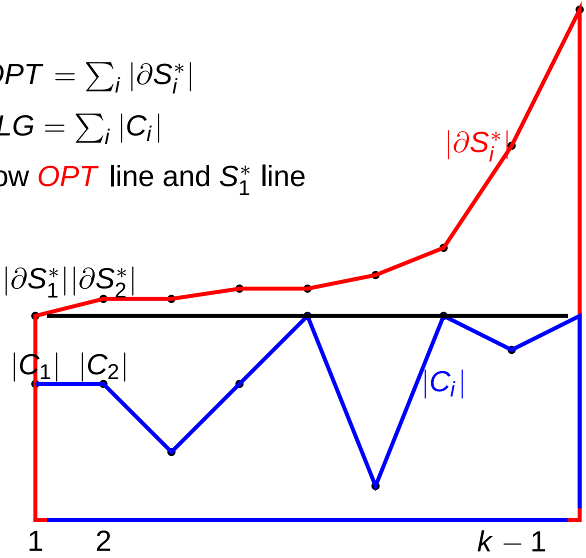
- $|\partial S_1^*|$ must be completely cut. Otherwise, it's a valid cut!
- Consider the algorithm's components so far. The union of some components is exactly S_1^*.
- Idea: guess all subsets of components
- Branching factor: 2^k, branching depth: $k \implies 2^{k^2}$ time
- Henceforth, assume $|C_i| \leq |\partial S_1^*|$ for all $i \in [k-1]$.

Case $|C_i| \leq |\partial S_1^*|$

$$2 \cdot OPT = \sum_i |\partial S_i^*|$$

$$ALG = \sum_i |C_i|$$

ALG line below *OPT* line and S_1^* line



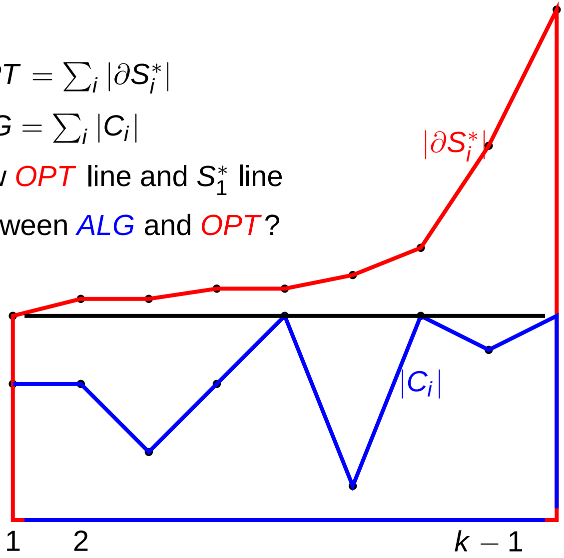
Case $|C_i| \leq |\partial S_1^*|$

$$2 \cdot OPT = \sum_i |\partial S_i^*|$$

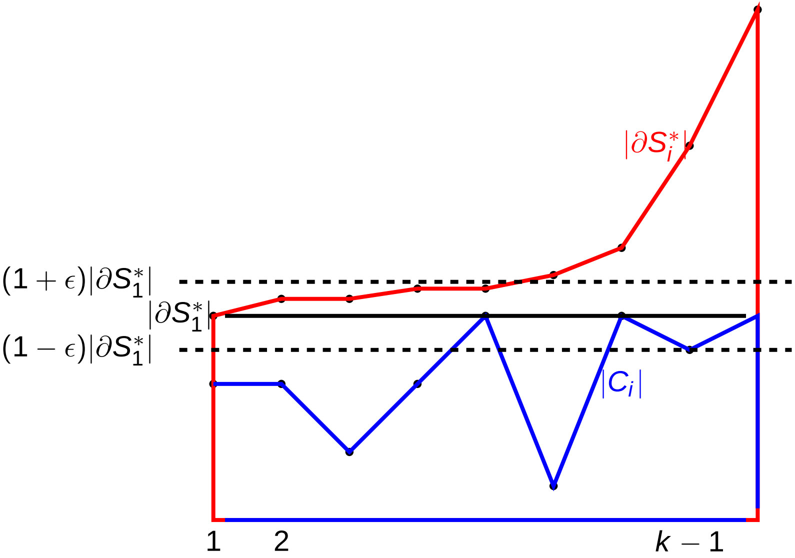
$$ALG = \sum_i |C_i|$$

ALG line below *OPT* line and S_1^* line

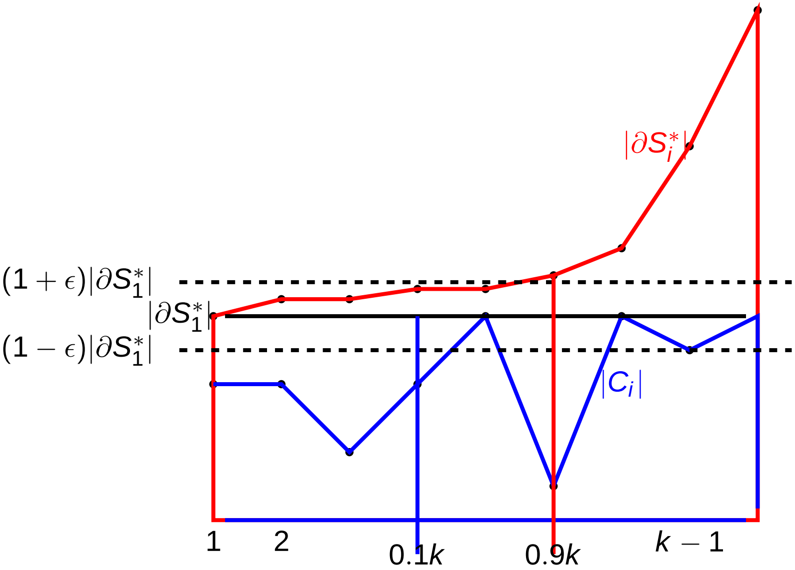
Find a gap between *ALG* and *OPT*?



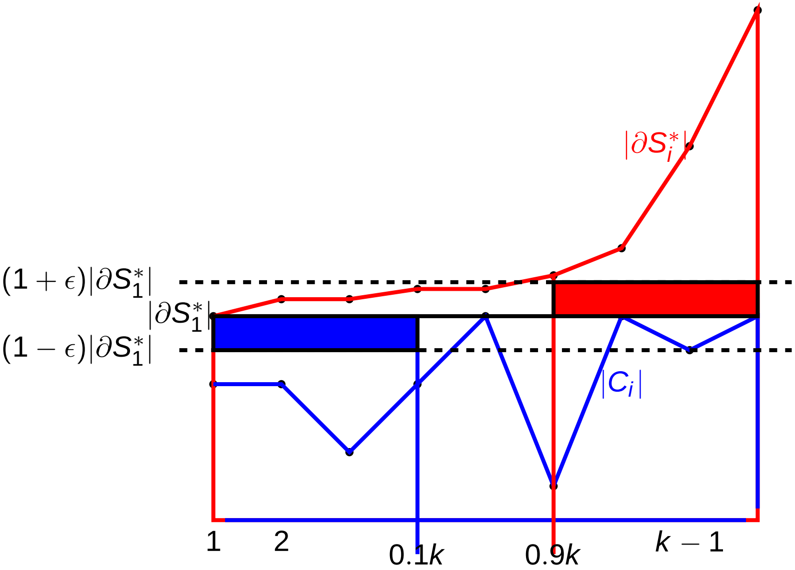
Case $|C_i| \le |\partial S_1^*|$



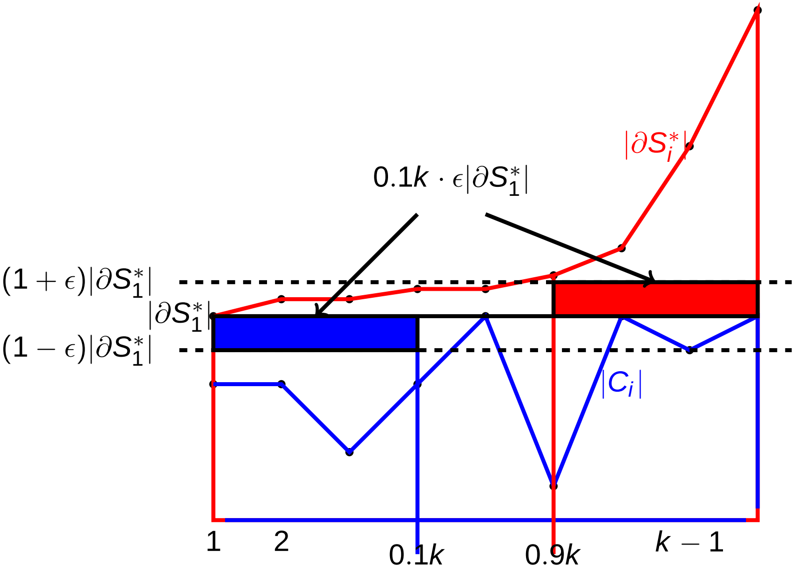
Case $|C_i| \leq |\partial S_1^*|$



Case $|C_i| \le |\partial S_1^*|$

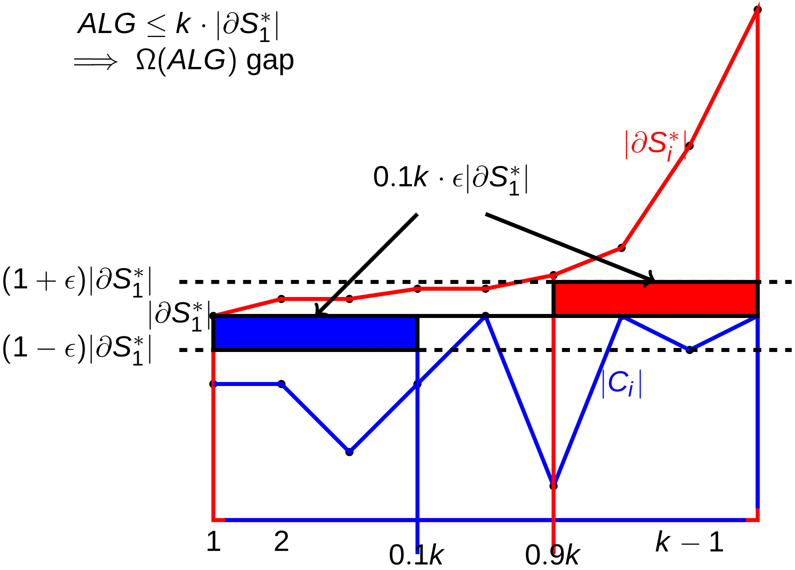


Case $|C_i| \leq |\partial S_1^*|$



Case $|C_i| \leq |\partial S_1^*|$

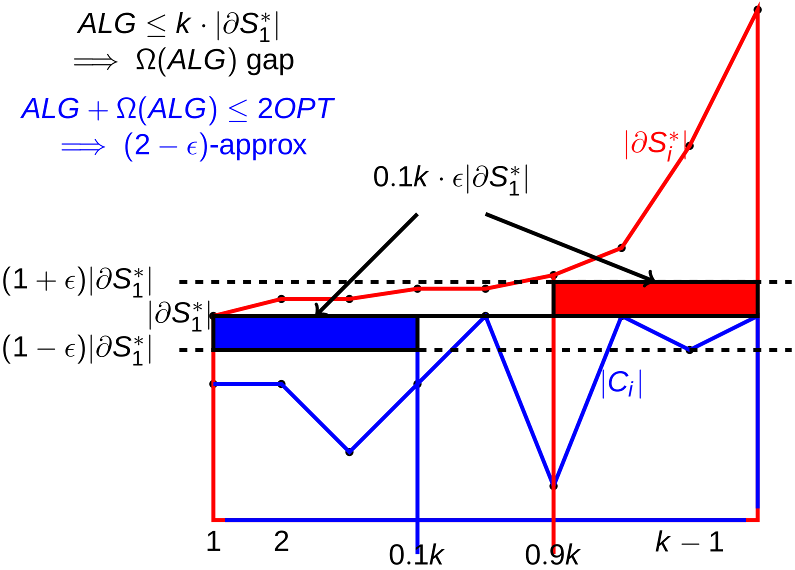
$ALG \leq k \cdot |\partial S_1^*|$
$\implies \Omega(ALG)$ gap



Case $|C_i| \leq |\partial S_1^*|$

$ALG \leq k \cdot |\partial S_1^*|$
$\implies \Omega(ALG)$ gap

$ALG + \Omega(ALG) \leq 2OPT$
$\implies (2-\epsilon)$-approx

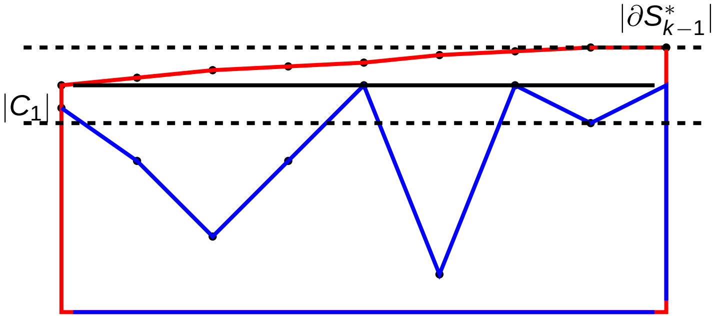


Hard case for greedy

- No such gap

Hard case for greedy

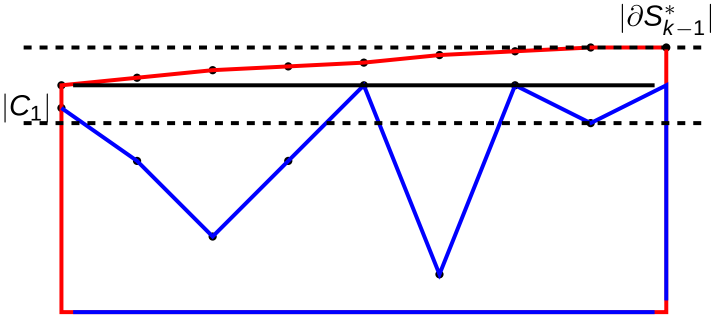
- No such gap



-

Hard case for greedy

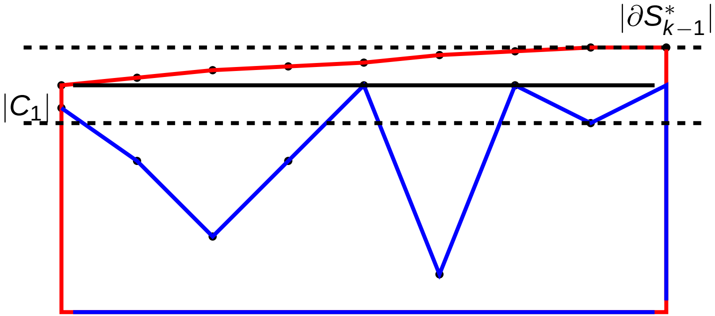
- No such gap



-
- $\text{mincut}(G) = |C_1| \approx |\partial S^*_1| \approx |\partial S^*_i|$ for all i

Hard case for greedy

- No such gap



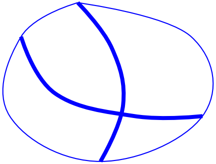
-
- $\text{mincut}(G) = |C_1| \approx |\partial S^*_1| \approx |\partial S^*_i|$ for all i
- We have $|\partial S^*_i| \approx \text{mincut}(G)$ for all $i \leq k-1$.

Crossing cuts

- Hard case: all ∂S_i^* are near-mincuts ($i \leq k - 1$)

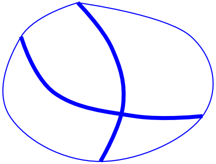
Crossing cuts

- Hard case: all ∂S_i^* are near-mincuts ($i \leq k - 1$)
- Consider the set of near-mincuts of G. Suppose two near-mincuts cross.



Crossing cuts

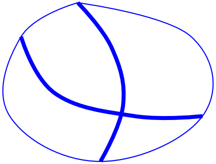
- Hard case: all ∂S_i^* are near-mincuts ($i \leq k-1$)
- Consider the set of near-mincuts of G. Suppose two near-mincuts cross.



- min-4-cut$(G) \leq 2(1+O(\epsilon)) \cdot$ mincut(G)

Crossing cuts

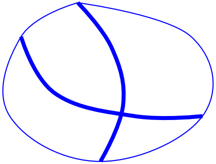
- Hard case: all ∂S_i^* are near-mincuts ($i \leq k-1$)
- Consider the set of near-mincuts of G. Suppose two near-mincuts cross.



- min-4-cut(G) $\leq 2(1+O(\epsilon)) \cdot$ mincut(G)
- Greedily take best min-4-cut. Pay $(2+O(\epsilon)) \cdot$ mincut(G) for 3 additional components $\implies (\frac{2}{3}+O(\epsilon)) \cdot$ mincut(G) cost per additional component.

Crossing cuts

- Hard case: all ∂S_i^* are near-mincuts ($i \leq k-1$)
- Consider the set of near-mincuts of G. Suppose two near-mincuts cross.



- min-4-cut$(G) \leq 2(1+O(\epsilon)) \cdot$ mincut(G)
- Greedily take best min-4-cut. Pay $(2+O(\epsilon)) \cdot$ mincut(G) for 3 additional components $\implies (\frac{2}{3}+O(\epsilon)) \cdot$ mincut(G) cost per additional component.
- If we can repeat this $\Omega(k)$ times, we save $\Omega(OPT)$ $\implies (2-\epsilon)$-approx.

Hard case

- Hard case:

Hard case

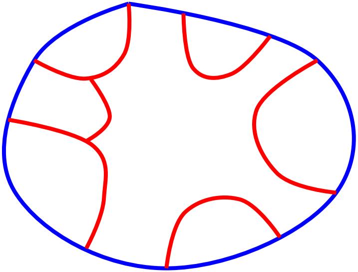
- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k - 1$)

Hard case

- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k - 1$)
  - No two near-mincuts cross

Hard case

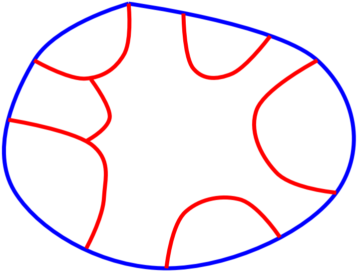
- Hard case:
    - All ∂S_i^* are near-mincuts ($i \leq k-1$)
    - No two near-mincuts cross



-

Hard case

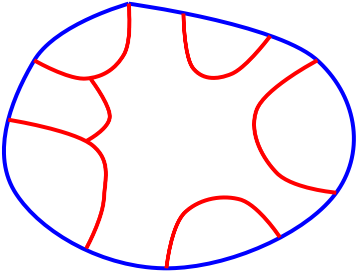
- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k-1$)
  - No two near-mincuts cross



-
- Only $n^{O(1+\epsilon)}$ many $(1+\epsilon)$-near-mincuts

Hard case

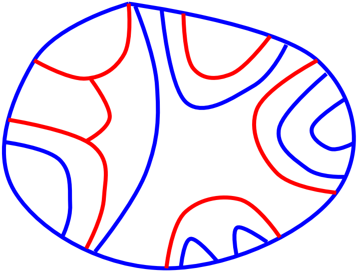
- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k - 1$)
  - No two near-mincuts cross



-
- Only $n^{O(1+\epsilon)}$ many $(1+\epsilon)$-near-mincuts
- Don't cross $\implies$ forms laminar family

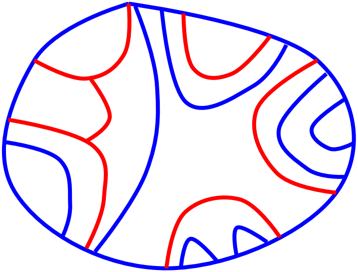
Hard case

- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k - 1$)
  - No two near-mincuts cross



-
- Only $n^{O(1+\epsilon)}$ many $(1+\epsilon)$-near-mincuts
- Don't cross $\implies$ forms laminar family

Hard case

- Hard case:
  - All ∂S_i^* are near-mincuts ($i \leq k-1$)
  - No two near-mincuts cross
- 
- Only $n^{O(1+\epsilon)}$ many $(1+\epsilon)$-near-mincuts
- Don't cross $\implies$ forms laminar family
- Separate FPT $(2-\epsilon)$-approx algorithm.

Open questions

- Improved approximation factor, and/or FPT APX hardness result

Open questions

- Improved approximation factor, and/or FPT APX hardness result
- FPT approximation scheme for min k-cut?