The Number of Minimum k-cuts: Beating the Karger-Stein Bound

Jason Li

Joint work with Anupam Gupta (CMU), Euiwoong Lee (NYU)

1/23/19

CMU Theory Lunch

Introduction

minimum k-cut: delete min weight edges to cut graph into $\geq k$ connect components

Setting: exact algorithm, k constant

Q: How fast to compute min k-cut?

Q: How **many** min k-cuts are there?

Prior Work

Goldschmidt–Hochbaum 1994: $O(n^{(1/2-o(1))k^2})$ time deterministic

Karger–Stein 1994: $\tilde{O}(n^{2(k-1)})$ time randomized

Thorup 2008: $\tilde{O}(mn^{2k-2})$ time deterministic

Chekuri et al. 2018: $\tilde{O}(mn^{2k-3})$ time deterministic

This work: $O_k(n^{(1.981+o(1))k})$ time randomized

All of these algorithms can <u>enumerate</u> all min k-cuts

$\Longrightarrow$ get corresponding extremal bound

Prior Work

Goldschmidt–Hochbaum 1994: $O(n^{(1/2-o(1))k^2})$ time deterministic

Karger–Stein 1994: $\tilde{O}(n^{2(k-1)})$ time randomized

Thorup 2008: $\tilde{O}(mn^{2k-2})$ time deterministic

Chekuri et al. 2018: $\tilde{O}(mn^{2k-3})$ time deterministic

This work: $O_k(n^{(1.981+o(1))k})$ time randomized

All of these algorithms can enumerate all min k-cuts

$\Longrightarrow$ get corresponding extremal bound

Same authors, 2018: compute one min k-cut in $O_k(n^{(2\omega/3+o(1))k})$ time deterministic, integer weights $\le n^{O(1)}$

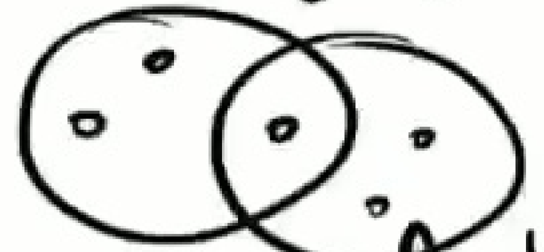
Lower bound: as hard as min weight k-clique: $\Omega(n^{(1-o(1))k})$

Our Approach

- Combines Karger-Stein's randomized contraction with Thorup's tree packing algorithm
- Makes new connection to extremal set theory in context of graph cut algorithms

Our Approach

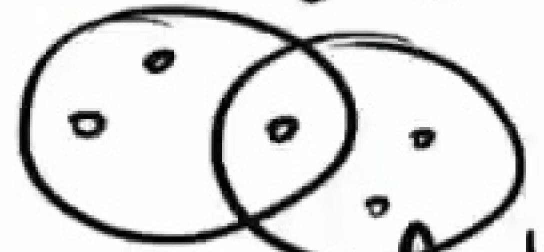
- Combines Karger-Stein's randomized contraction with Thorup's tree packing algorithm
- Makes new connection to extremal set theory in context of graph cut algorithms

[Folklore] If $\mathcal{A}$ is a collection of distinct sets on $[n]$ (a set system), and $|\mathcal{A}| \geq 2n$, then $\exists A, B \in \mathcal{A}$ that cross: 

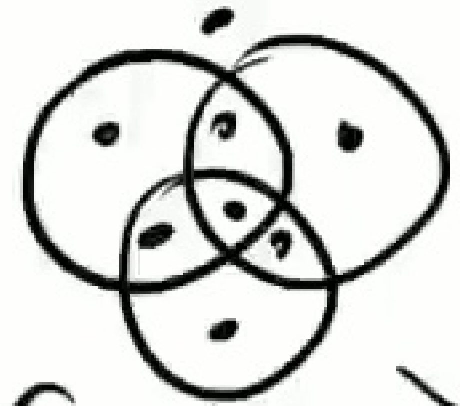
"If $|\mathcal{A}| \geq 2n$, then $\mathcal{A}$ has dual VC dimension ≥ 2"

Our Approach

- Combines Karger-Stein's randomized contraction with Thorup's tree packing algorithm
- Makes new connection to extremal set theory in context of graph cut algorithms

[Folklore] If $\mathcal{A}$ is a collection of distinct sets on $[n]$ (a set system), and $|\mathcal{A}| \geq 2n$, then $\exists A, B \in \mathcal{A}$ that cross: 

"If $|\mathcal{A}| \geq 2n$, then $\mathcal{A}$ has dual VC dimension ≥ 2"

[This work] If $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\exists A, B, C \in \mathcal{A}$: 

"If $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\mathcal{A}$ has dual VC dim ≥ 3"

Karger-Stein Algorithm

Karger-Stein Algorithm

While G has $> 2k$ vertices left:

- Select edge at random,
  proportional to its weight

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight

  (edge e selected with prob $\dfrac{w(e)}{\sum_{e'} w(e')}$)

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph: output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[$failure when r vertices left$]$? $(r = n, n-1, \ldots, 2k+1)$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph: output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[\text{failure when } r \text{ vertices left}]$? $(r = n, n-1, \dots, 2k+1)$

Obs: $(\text{min k-cut}) \leq (\text{sum of } k-1 \text{ smallest weighted degrees})$

$\leq (k-1)(\text{avg of } k-1 \text{ smallest degs})$

$\leq (k-1)(\text{avg degree})$

$= (k-1) \cdot \frac{(\text{sum of degs})}{r} = (k-1) \frac{2 \sum_e w_e}{r}$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges

For each of $(2k)^k$ k-cuts on remaining graph: output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[\text{failure when } r \text{ vertices left}]$? $(r = n, n-1, \dots, 2k+1)$

Obs: $(\text{min k-cut}) \le (\text{sum of } k-1 \text{ smallest weighted degrees})$

$\le (k-1)(\text{avg of } k-1 \text{ smallest degs})$

$\le (k-1)(\text{avg degree})$

$= (k-1) \cdot \frac{(\text{sum of degs})}{r} = (k-1)\frac{2\sum_e w_e}{r}$

$\Rightarrow \Pr[\text{failure}] = \frac{OPT}{\sum_e w_e}$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

Thm: Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[\text{failure when } r \text{ vertices left}]$? $(r = n, n-1, \ldots, 2k+1)$

Obs: $(\text{min k-cut}) \le (\text{sum of } k-1 \text{ smallest weighted degrees})$

$\le (k-1)(\text{avg of } k-1 \text{ smallest degs})$

$\le (k-1)(\text{avg degree})$

$= (k-1) \cdot \frac{(\text{sum of degs})}{r} = (k-1)\frac{2\cancel{\sum_e w_e}}{r}$

$\Rightarrow \Pr[\text{failure}] = \frac{OPT}{\cancel{\sum_e w_e}} \le \frac{2(k-1)}{r}.$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

$$\Pr[\text{success}] = \left(1-\frac{2(k-1)}{n}\right)\left(1-\frac{2(k-1)}{n-1}\right)\cdots\left(1-\frac{2(k-1)}{2k+1}\right)$$
$$= \frac{n-2k+1}{n}\cdot\frac{n-2k}{n-1}\cdot\frac{n-2k-1}{n-2}\cdot\ldots\cdot\frac{n-4k+2}{n-2k+1}\cdot\ldots\ldots$$

Thm: Let $\mathcal{S}=\{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[$failure when r vertices left$]$? $(r = n, n-1, \ldots, 2k+1)$

Obs: (min k-cut) $\leq$ (sum of k-1 smallest weighted degrees)

$\leq (k-1)$(avg of k-1 smallest degs)

$\leq (k-1)$(avg degree)

$= (k-1)\cdot\frac{(\text{sum of degs})}{r} = (k-1)\frac{2\cancel{\sum_e w_e}}{r}$

$$\Rightarrow \Pr[\text{failure}] = \frac{OPT}{\cancel{\sum_e w_e}} \leq \frac{2(k-1)}{r}.$$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\frac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

$$\Pr[\text{success}] = \left(1-\frac{2(k-1)}{n}\right)\left(1-\frac{2(k-1)}{n-1}\right)\cdots\left(1-\frac{2(k-1)}{2k+1}\right)$$

$$= \frac{\cancel{n-2k+1}}{n}\cdot\frac{\cancel{n-2k}}{n-1}\cdot\frac{\cancel{n-2k-1}}{n-2}\cdot\ldots\cdot\frac{n-4k+2}{\cancel{n-2k+1}}\cdot\ldots\ldots$$

2(k-1) terms not cancelled

Thm: Let $\mathcal{S}=\{S_1,S_2,\ldots,S_k\}$ be a minimum k-cut. K-S returns $\mathcal{S}$ with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[\mathcal{S}]$ on some iteration.

$\Pr[\text{failure when } r \text{ vertices left}]$? $(r = n, n-1, \ldots, 2k+1)$

Obs: (min k-cut) $\le$ (sum of k-1 smallest weighted degrees)

$\le (k-1)$(avg of k-1 smallest degs)

$\le (k-1)$(avg degree)

$= (k-1)\cdot\frac{(\text{sum of degs})}{r} = (k-1)\frac{2\cancel{\sum_e w_e}}{r}$

$$\Rightarrow \Pr[\text{failure}] = \frac{OPT}{\cancel{\sum_e w_e}} \le \frac{2(k-1)}{r}.$$

Karger-Stein Algorithm

While G has $>2k$ vertices left:

- Select edge at random, proportional to its weight (edge e selected with prob $\dfrac{w(e)}{\sum_{e'} w(e')}$)
- Contract this edge
- Merge parallel edges.

For each of $(2k)^k$ k-cuts on remaining graph:
output the k-cut formed by "uncontracting" each component

$$\Pr[\text{success}] = \left(1-\frac{2(k-1)}{n}\right)\left(1-\frac{2(k-1)}{n-1}\right)\cdots\left(1-\frac{2(k-1)}{2k+1}\right)$$

$$= \frac{\cancel{n-2k+1}}{n}\cdot\frac{\cancel{n-2k}}{n-1}\cdot\frac{\cancel{n-2k-1}}{n-2}\cdot\ldots\cdot\frac{n-4k+2}{\cancel{n-2k+1}}\cdot\ldots\ldots$$

2(k-1) terms not cancelled

$$\geq \frac{1}{n^{2(k-1)}}.$$

Thm: Let $S = \{S_1, S_2, \ldots, S_k\}$ be a minimum k-cut. K-S returns S with prob $\Omega\left(\frac{1}{n^{2k}}\right)$.

Pf: Failure: contracted an edge in $E[S]$ on some iteration.

Pr[failure when r vertices left]? $(r = n, n-1, \ldots, 2k+1)$

Obs: (min k-cut) $\leq$ (sum of k-1 smallest weighted degrees)

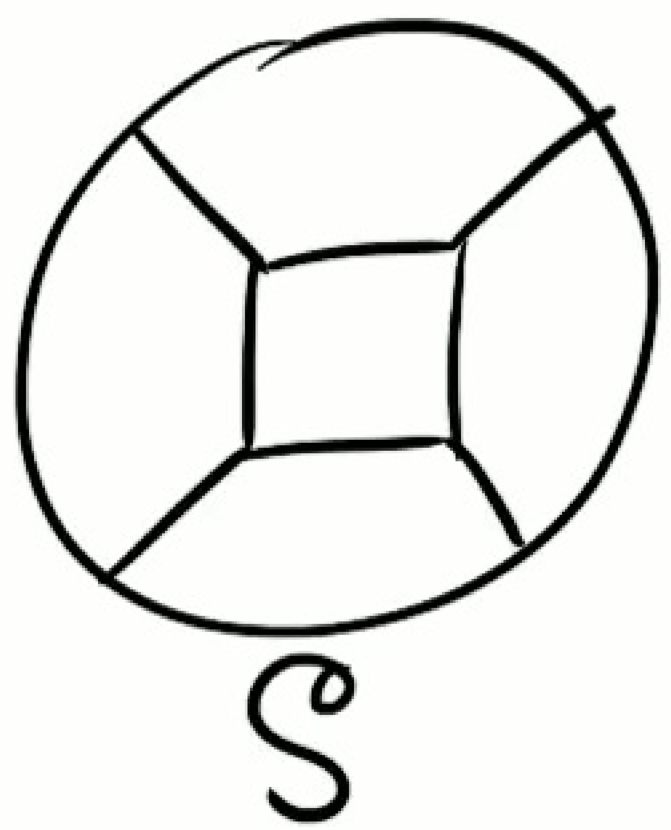
$\leq (k-1)$(avg of k-1 smallest degs)

$\leq (k-1)$(avg degree)

$= (k-1)\cdot\dfrac{(\text{sum of degs})}{r} = (k-1)\dfrac{2\cancel{\sum_e w_e}}{r}$

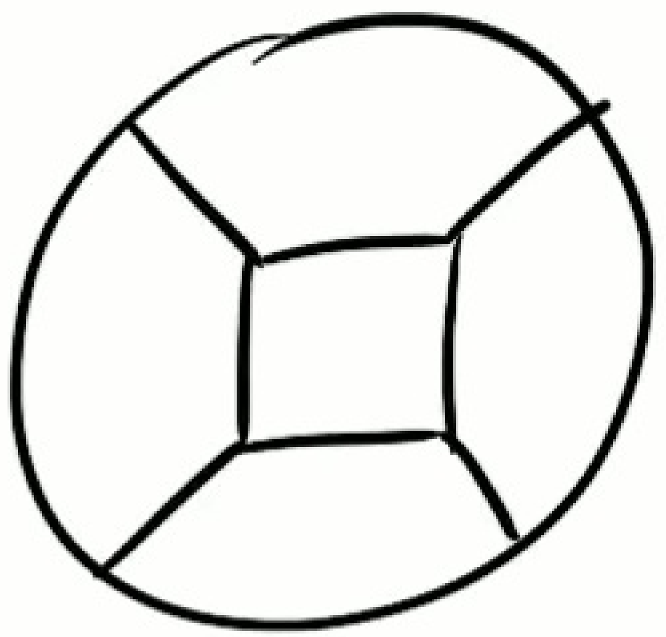
$$\Rightarrow \Pr[\text{failure}] = \frac{OPT}{\cancel{\sum_e w_e}} \leq \frac{2(k-1)}{r}.$$

Branch and Bound

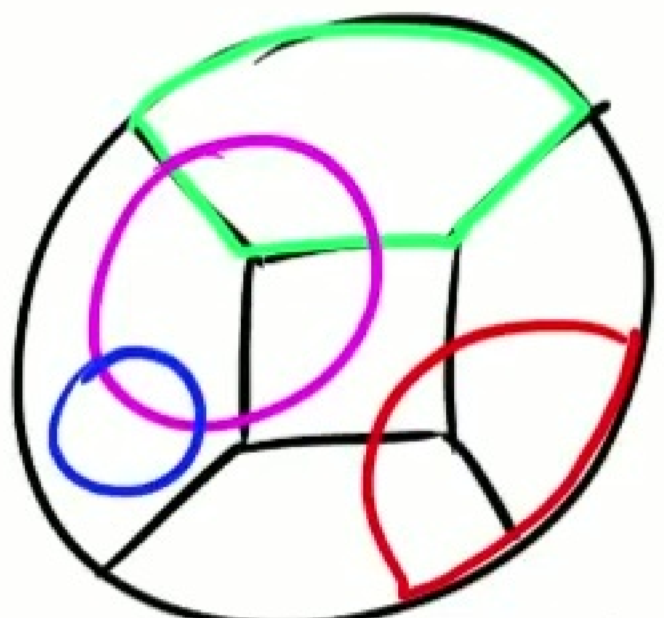


:

Branch and Bound

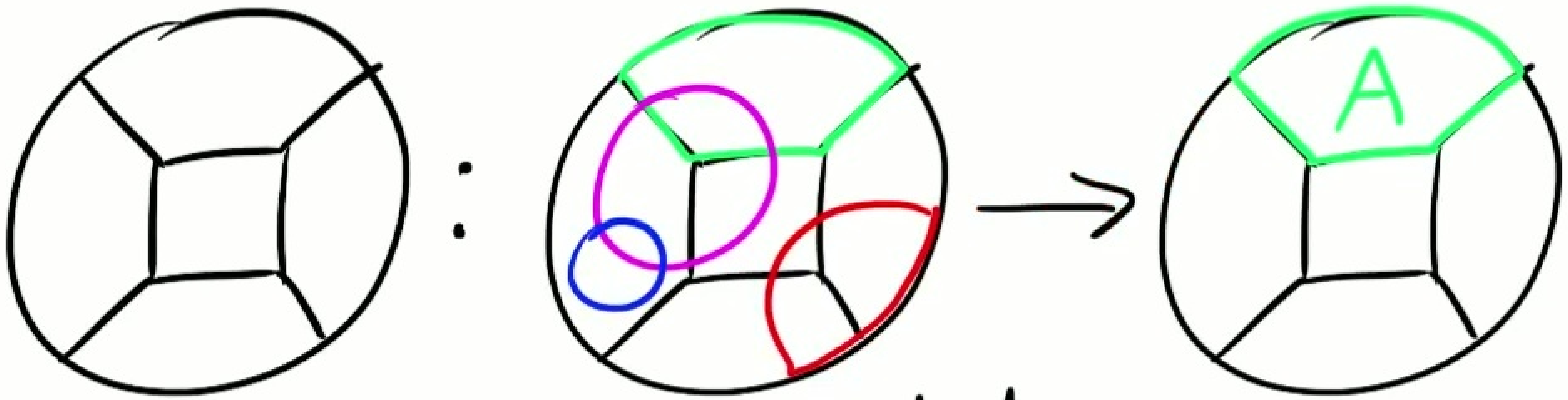


S



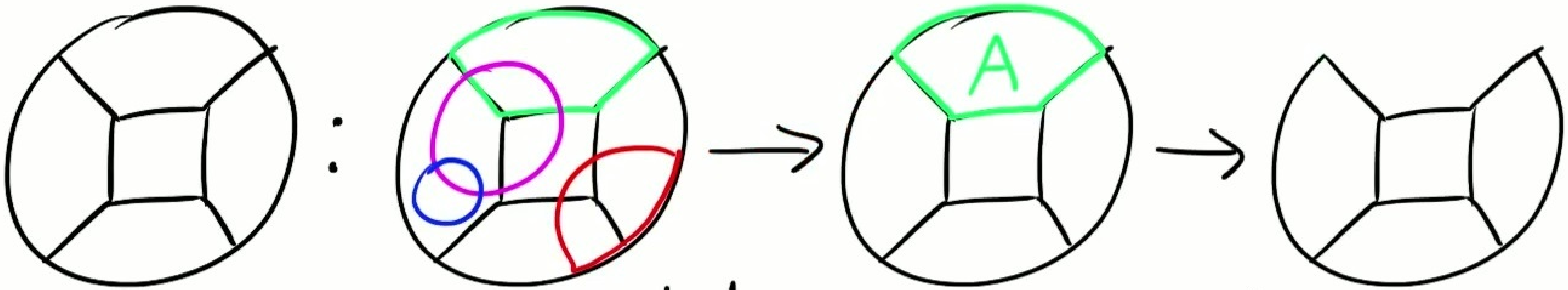
Guess a set $\mathcal{A}$ of subsets of V

Branch and Bound



S : Guess a set $\mathcal{A}$ of subsets of V $\rightarrow$ For each set $A \in \mathcal{A}$:

Branch and Bound



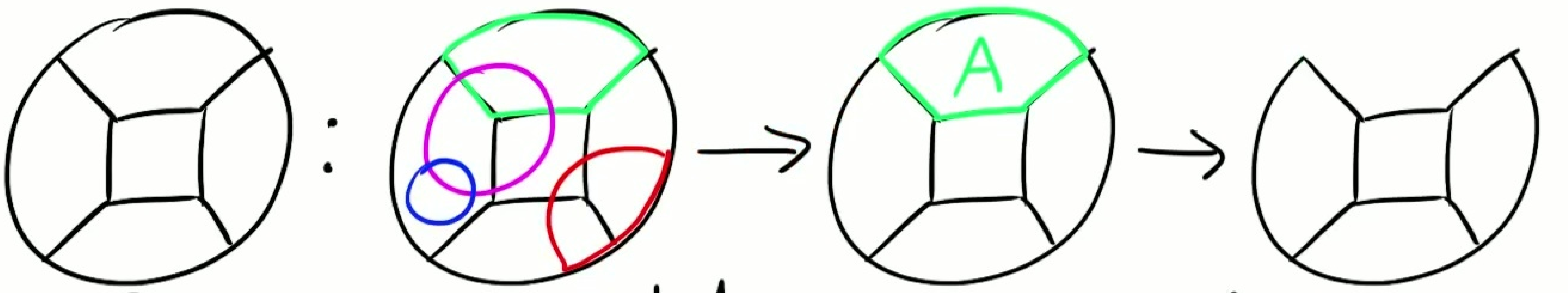
S :

Guess a set $\mathcal{A}$ of subsets of V

For each set $A \in \mathcal{A}$:

A is one component; recurse by calling $(k-1)$-cut on $G-A$

Branch and Bound



S

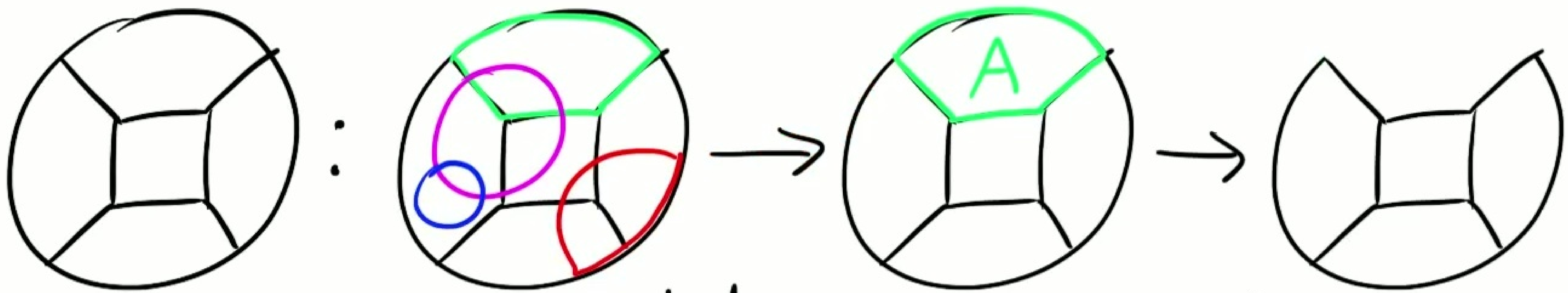
Guess a set $\mathcal{A}$ of subsets of V

For each set $A \in \mathcal{A}$:

A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of S

Branch and Bound



S :

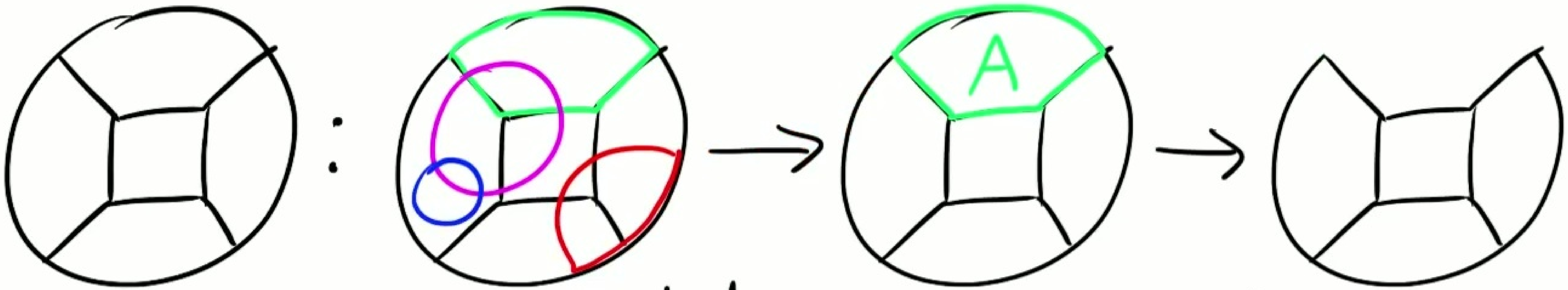
Guess a set $\mathcal{A}$ of subsets of V

For each set $A \in \mathcal{A}$:

A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of S
- Branching overhead: $|\mathcal{A}|$

Branch and Bound



S

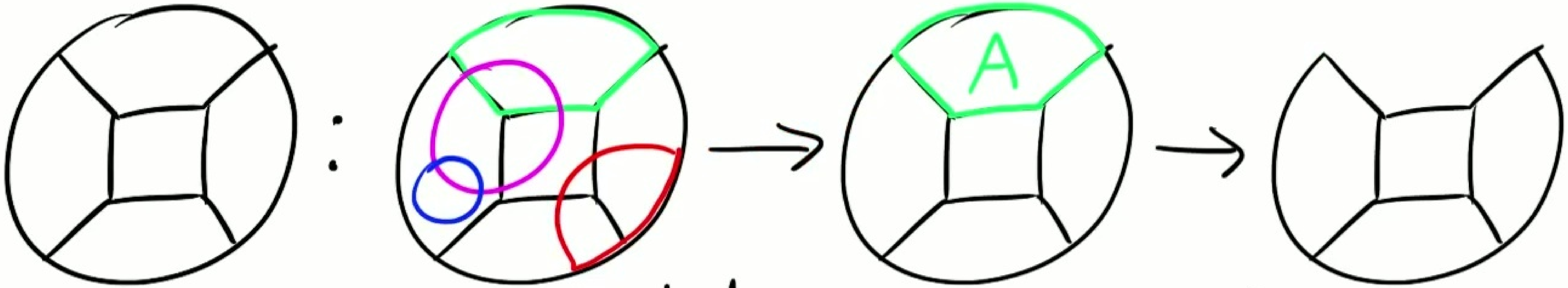
Guess a set $\mathcal{A}$ of subsets of V

For each set $A \in \mathcal{A}$:

A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of S
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.

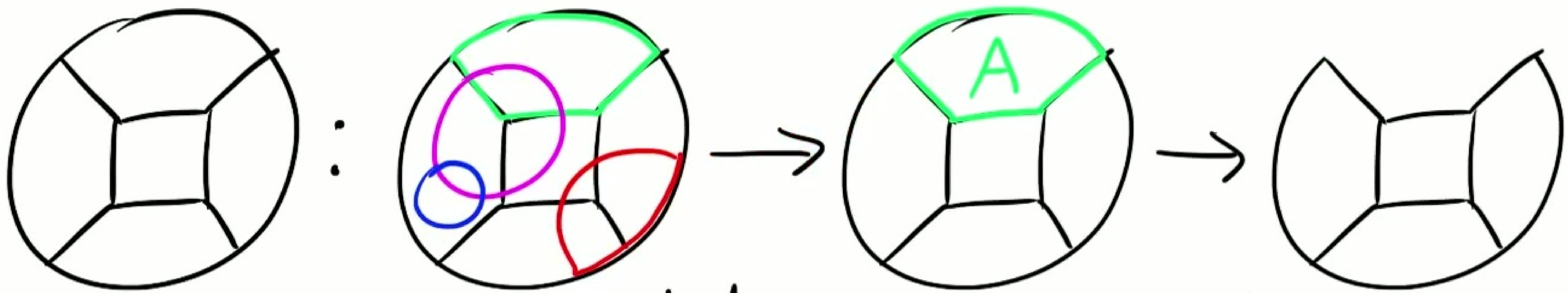
Branch and Bound



S : Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G - A$

- Works if $\exists A \in \mathcal{A}$ component of S
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n, k-1)$

Branch and Bound

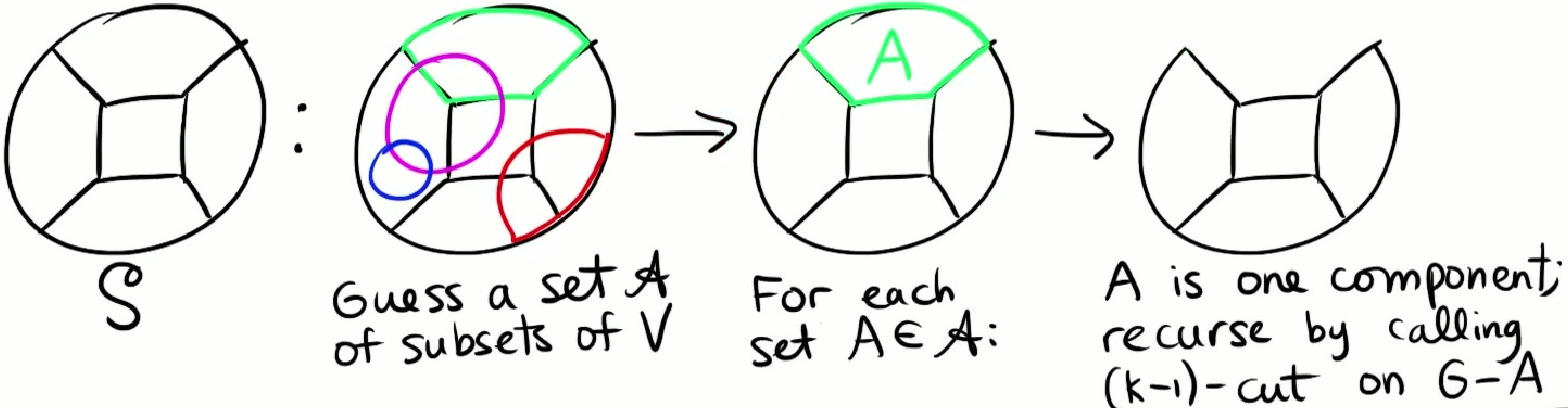


S : Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of S
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n, k-1)$

Illustrative Example

Branch and Bound

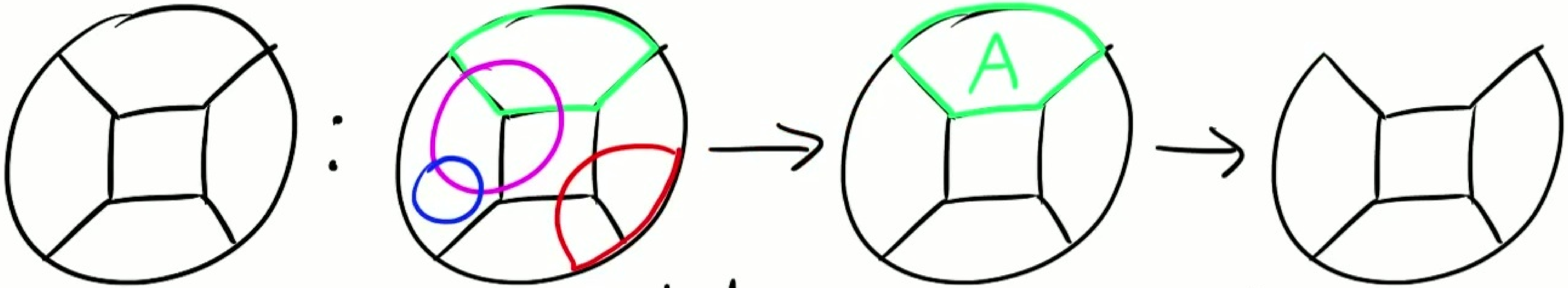


- Works if $\exists A \in \mathcal{A}$ component of S
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n, k-1)$

Illustrative Example

Let $\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} \mathrm{OPT}\}$

Branch and Bound



$\mathcal{S}$: Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of $\mathcal{S}$
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n,k-1)$

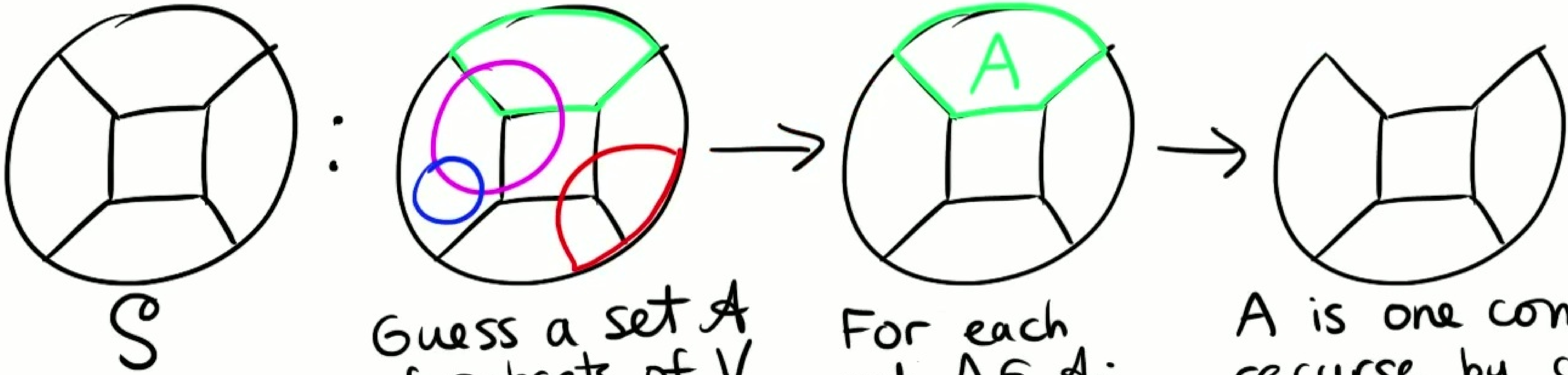
Illustrative Example

"much smaller than average"

Let $\mathcal{A}' := \left\{ A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT} \right\}$

Note: average $S \in \mathcal{S}$ is $\frac{2}{k}\mathrm{OPT}$, since $\sum_{S \in \mathcal{S}} w(\partial_G S) = 2\,\mathrm{OPT}$.

Branch and Bound



$\mathcal{S}$: Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G - A$

- Works if $\exists A \in \mathcal{A}$ component of $\mathcal{S}$
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n, k-1)$

Illustrative Example

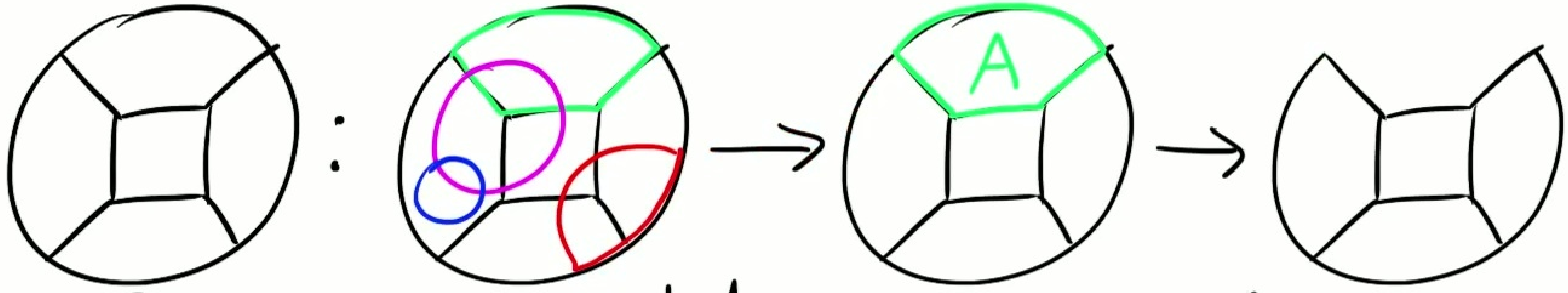
"much smaller than average"

Let $\mathcal{A}' := \left\{ A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT} \right\}$

Note: average $S \in \mathcal{S}$ is $\frac{2}{k}\mathrm{OPT}$, since $\sum_{S \in \mathcal{S}} w(\partial_G S) = 2\,\mathrm{OPT}$.

So, only works if G contains a component "much smaller than avg"

Branch and Bound



$\mathcal{S}$: Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of $\mathcal{S}$
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n, k-1)$

Illustrative Example

"much smaller than average"

Let $\mathcal{A}' := \left\{ A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT \right\}$

Note: average $S \in \mathcal{S}$ is $\frac{2}{k} OPT$, since

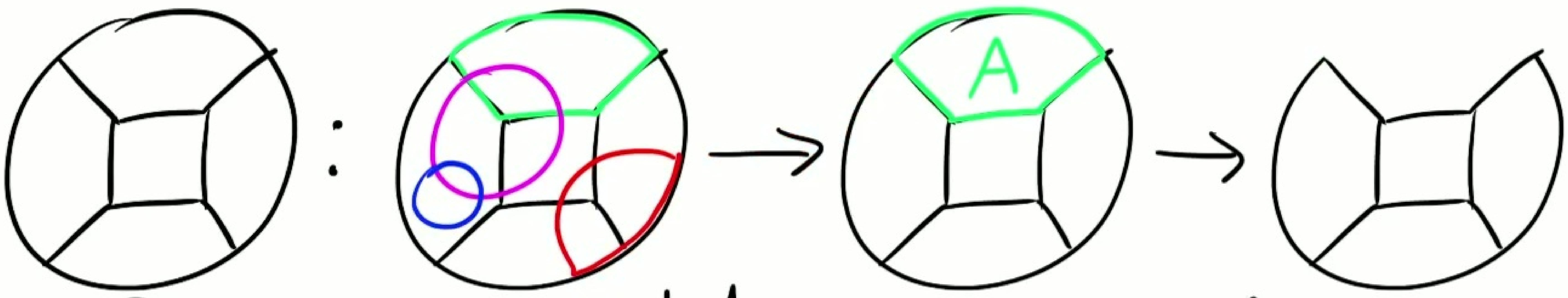
$$\sum_{S \in \mathcal{S}} w(\partial_G S) = 2\,OPT.$$

So, only works if G contains a component "much smaller than avg"

Claim (bound):

$|\mathcal{A}'| = O_k(n)$.

Branch and Bound



$\mathcal{S}$

Guess a set $\mathcal{A}$ of subsets of V

For each set $A \in \mathcal{A}$:

A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of $\mathcal{S}$
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A}) + |\mathcal{A}| \cdot T(n,k-1)$

Illustrative Example

"much smaller than average"

Let $\mathcal{A}' := \left\{ A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT} \right\}$

Note: average $S \in \mathcal{S}$ is $\frac{2}{k}\mathrm{OPT}$, since

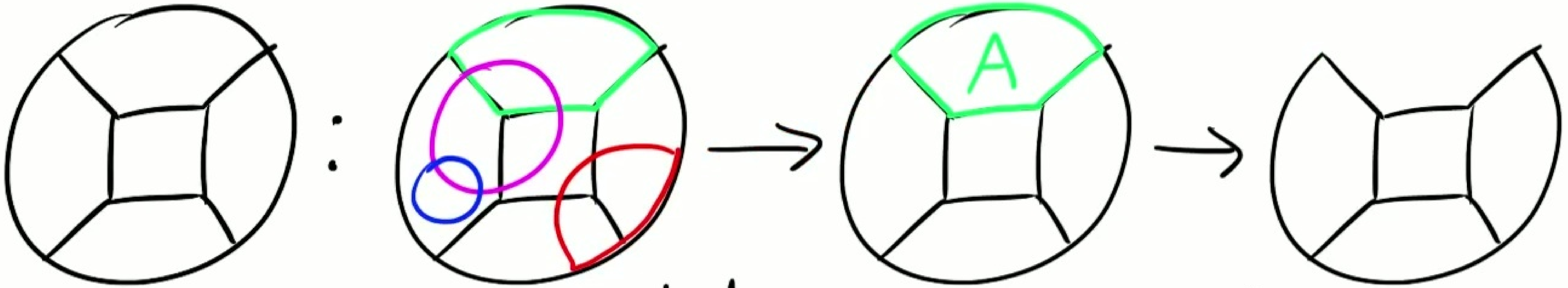
$$\sum_{S \in \mathcal{S}} w(\partial_G S) = 2\,\mathrm{OPT}.$$

So, only works if G contains a component "much smaller than avg"

Claim (bound): $|\mathcal{A}'| = O_k(n)$.

Claim (time): $\mathcal{A}'$ can be computed in $\mathrm{poly}(n)$.

Branch and Bound



$\mathcal{S}$: Guess a set $\mathcal{A}$ of subsets of V → For each set $A \in \mathcal{A}$: → A is one component; recurse by calling $(k-1)$-cut on $G-A$

- Works if $\exists A \in \mathcal{A}$ component of $\mathcal{S}$
- Branching overhead: $|\mathcal{A}|$
- Want: small $|\mathcal{A}|$, and computable efficiently.
- $T(n,k) = (\text{time to compute } \mathcal{A})$ poly(n)

  $+ |\mathcal{A}| \cdot T(n, k-1)$ n

  $\Rightarrow O_k(n^{k+O(1)})$!

Illustrative Example

Let $\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$ "much smaller than average"

Note: average $S \in \mathcal{S}$ is $\frac{2}{k} OPT$, since $\sum_{S \in \mathcal{S}} w(\partial_G S) = 2 OPT$.

So, only works if G contains a component "much smaller than avg"

Claim (bound): $|\mathcal{A}'| = O_k(n)$.

Claim (time): $\mathcal{A}'$ can be computed in poly(n).

Extremal Bound

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution $< OPT$, contradiction.

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Extremal Bound

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} \mathrm{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution $<$ OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that <u>cross</u>.

Extremal Bound

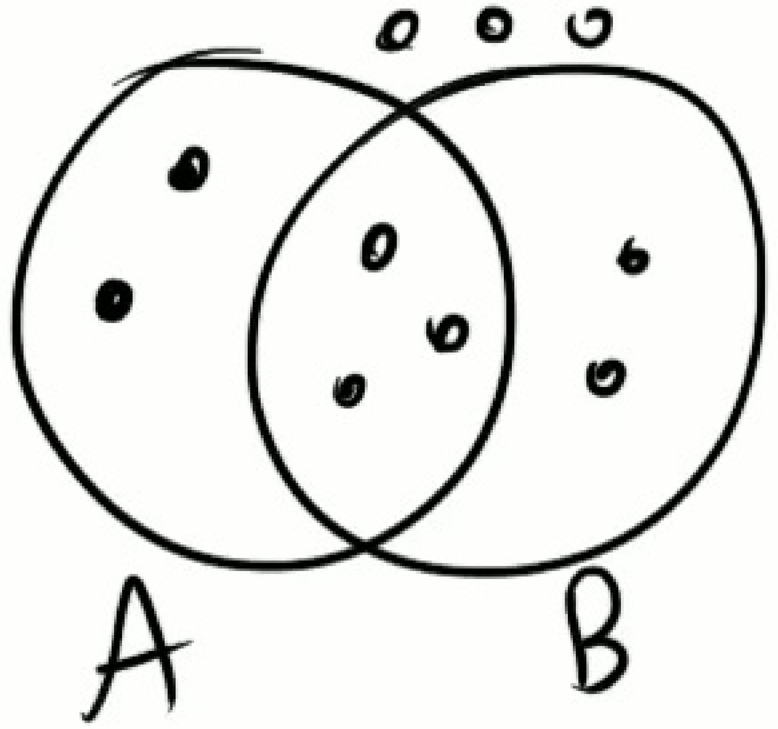
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

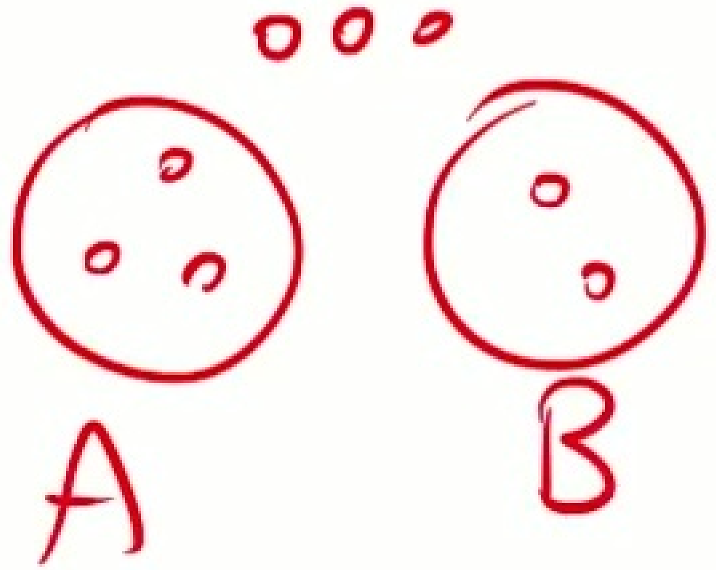
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution $<$ OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that cross.

cross:



don't cross:



Extremal Bound

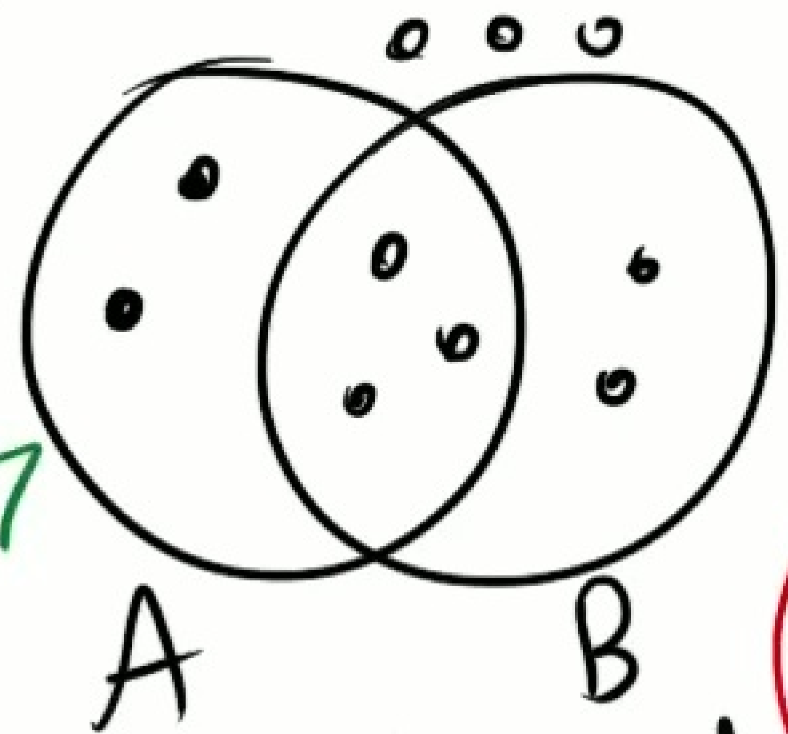
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

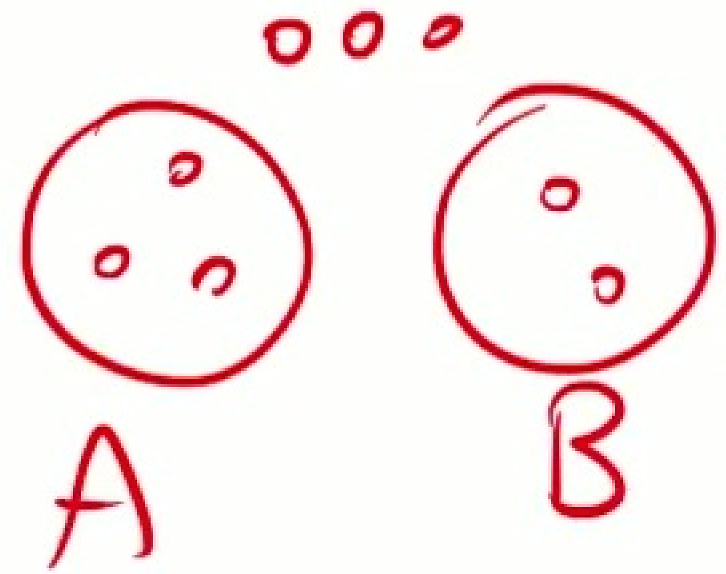
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution < OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that <u>cross</u>.

cross:



don't cross:



- Suppose we "cut out" A and B. (cut the edges $\partial A \cup \partial B$.)
  We get a <u>4-cut</u> (<u>not a 3-cut</u>) since A and B cross.

Extremal Bound

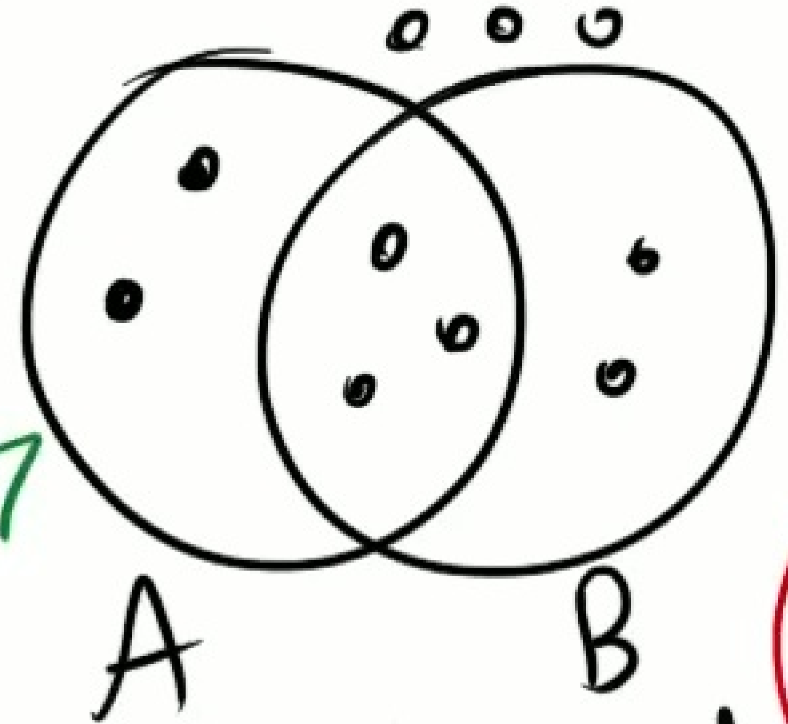
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k}\text{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

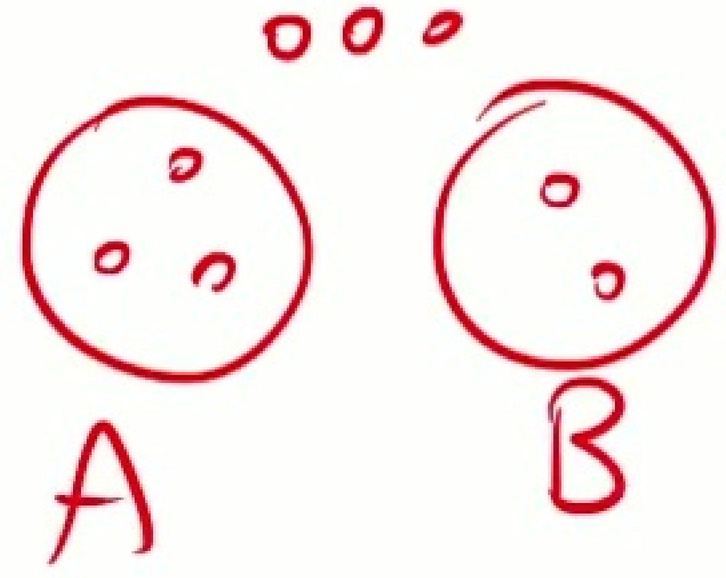
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution < OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that cross.

cross:



don't cross:



- Suppose we "cut out" A and B. (cut the edges $\partial A \cup \partial B$.)
  We get a 4-cut (not a 3-cut) since A and B cross.
- Think of it as +3 additional components for price of $\le 2 \cdot \frac{1.49}{k}\text{OPT}$

$\Longrightarrow \frac{1-\varepsilon}{k}\text{OPT}$ price per +1 component.

Extremal Bound

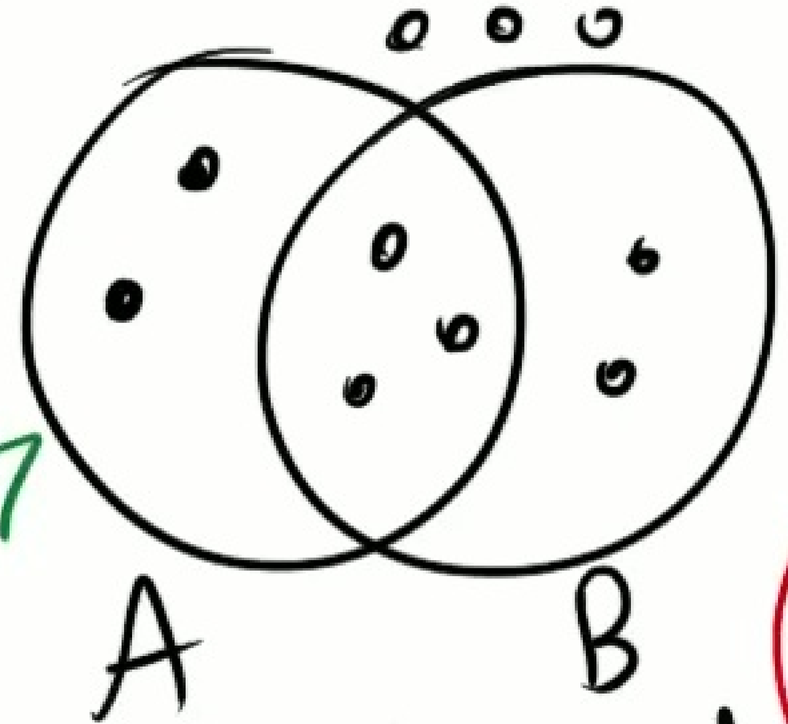
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

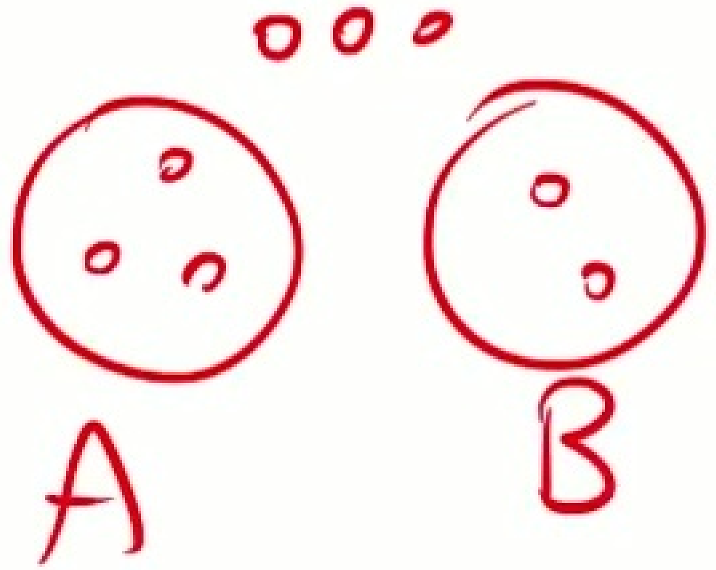
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution $<$ OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that <u>cross</u>.

cross:



don't cross:



- Suppose we "cut out" A and B. (cut the edges $\partial A \cup \partial B$.)
  We get a <u>4-cut</u> <u>(not a 3-cut)</u> since A and B cross.
- Think of it as +3 additional components for price of $\le 2 \cdot \frac{1.49}{k} \mathrm{OPT}$
  $\Longrightarrow \frac{1-\varepsilon}{k} \mathrm{OPT}$ price per +1 component.
- Repeat until k comps?

Extremal Bound

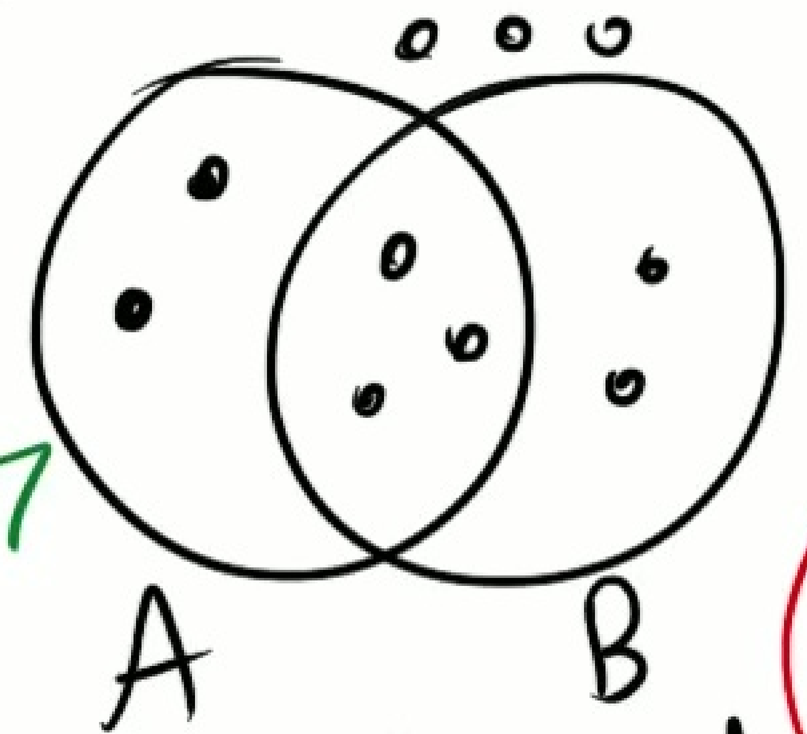
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

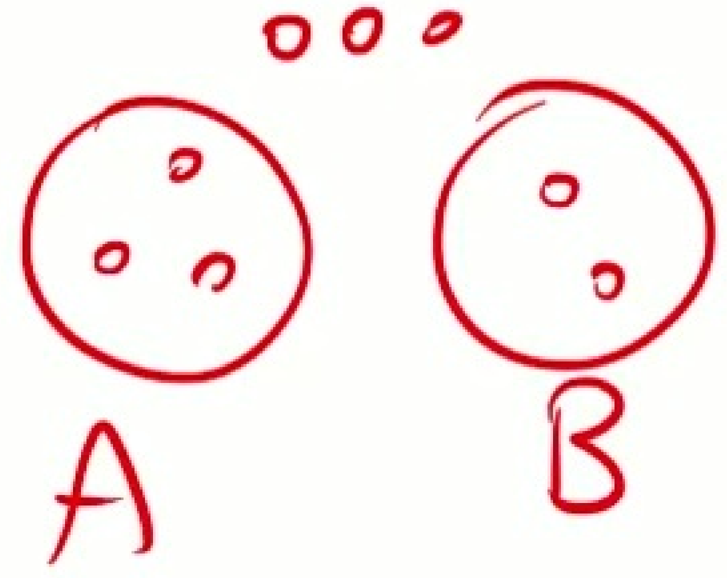
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution < OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that <u>cross</u>.

cross:

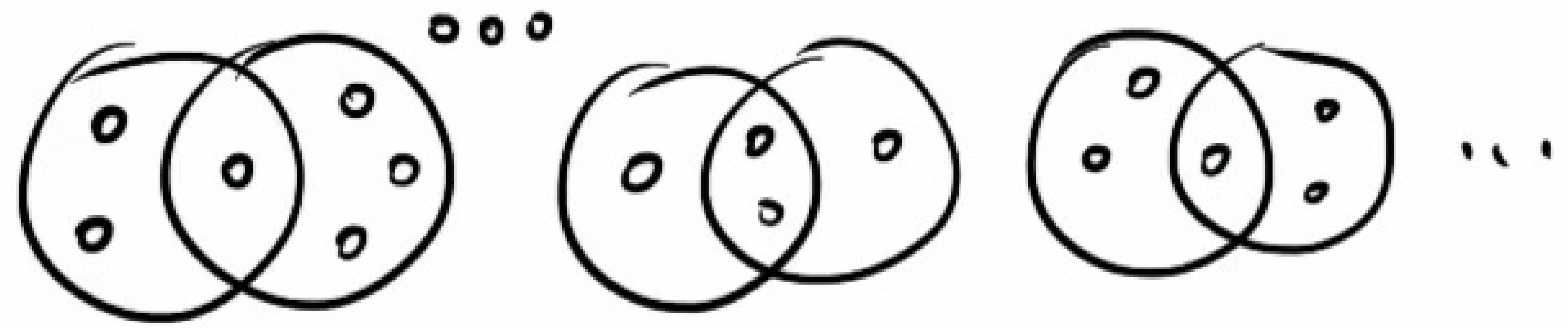


don't cross:



- Suppose we "cut out" A and B. (cut the edges $\partial A \cup \partial B$.)
  We get a <u>4-cut</u> <u>(not a 3-cut)</u> since A and B cross.
- Think of it as +3 additional components for price of $\le 2 \cdot \frac{1.49}{k} \mathrm{OPT}$

  $\Longrightarrow \frac{1-\varepsilon}{k} \mathrm{OPT}$ price per +1 component.
- Repeat until k comps?



Extremal Bound

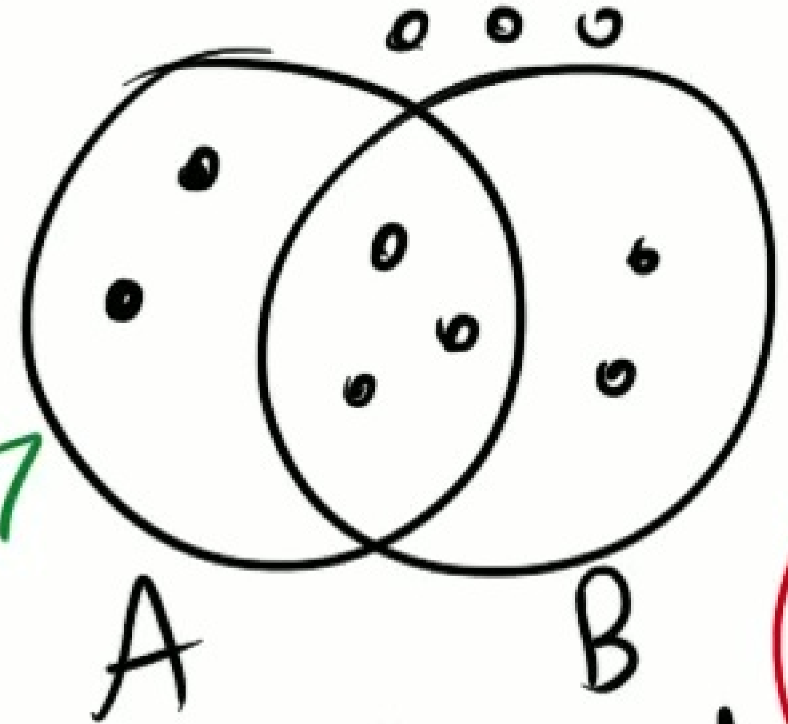
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$\rightarrow |\mathcal{A}'| = O_k(n)$

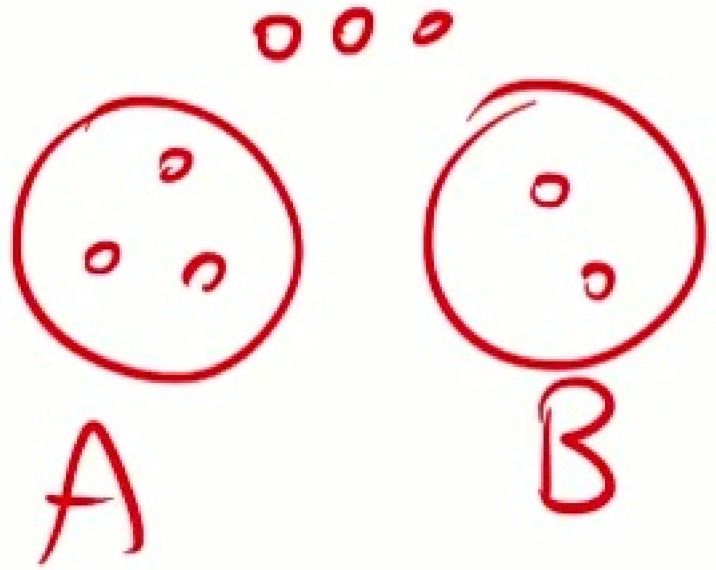
$\mathcal{A}'$ computed in polytime

- Strategy: if $|\mathcal{A}'| \gg n$, i.e. too many cheap cuts, then can construct a k-cut solution $<$ OPT, contradiction.
- Fact: if a set system $|\mathcal{A}| > 2n$, then there are two sets $A, B \in \mathcal{A}$ that <u>cross</u>.

cross:



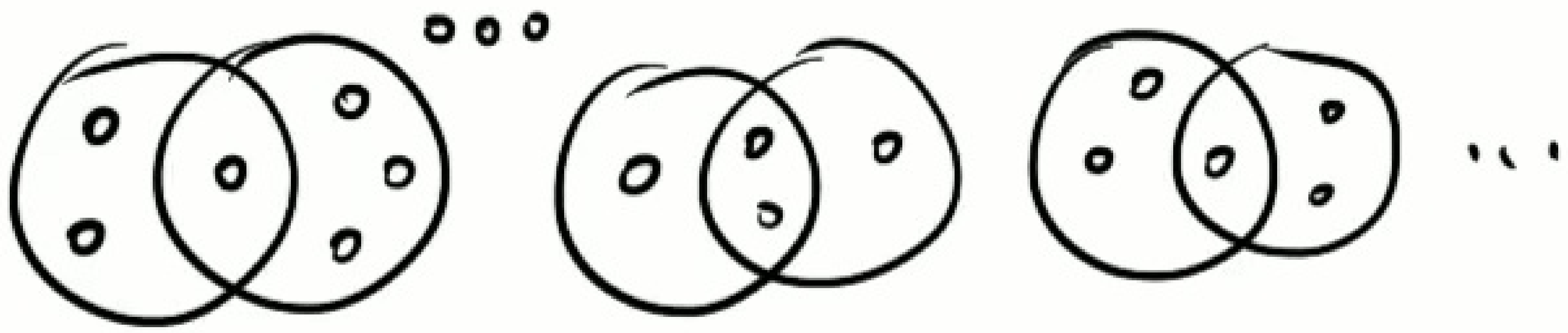
don't cross:



- Suppose we "cut out" A and B. (cut the edges $\partial A \cup \partial B$.) We get a 4-cut (not a 3-cut) since A and B cross.
- Think of it as +3 additional components for price of $\le 2 \cdot \frac{1.49}{k} OPT$

$\Longrightarrow \frac{1-\varepsilon}{k} OPT$ price per +1 component.

- Repeat until k comps?



pay $\lceil \frac{k}{3} \rceil \cdot 2 \cdot \frac{1.49}{k} OPT < OPT$

for k large enough, contradiction.

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

- Recall: assume for contradiction that $|\mathcal{A}'| \geq \Omega(2^k n)$.

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

- Recall: assume for contradiction that $|\mathcal{A}'| \geq \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

- Recall: assume for contradiction that $|\mathcal{A}'| \geq \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\leq 2 \cdot \frac{1-\varepsilon}{k} OPT$ for $+2$ component, or

  ② $\leq 3 \cdot \frac{1-\varepsilon}{k} OPT$ for $+3$ components

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

- Recall: assume for contradiction that $|\mathcal{A}'| \geq \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\leq 2 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +2 component, or

  ② $\leq 3 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\,\mathrm{OPT}$ (k large enough)

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

- Recall: assume for contradiction that $|\mathcal{A}'| \geq \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\leq 2 \cdot \frac{1-\varepsilon}{k} OPT$ for $+2$ component, or

  ② $\leq 3 \cdot \frac{1-\varepsilon}{k} OPT$ for $+3$ components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before

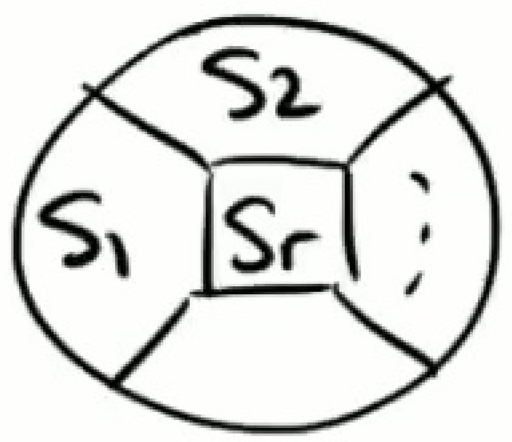
Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} OPT$ for +2 component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} OPT$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



Formal Proof: $|\mathcal{A}'| = O(2^k n)$

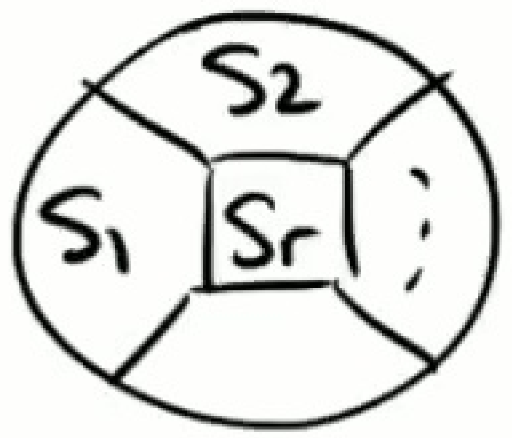
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k}\text{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k}\text{OPT}$ for +2 component, or

  ② $\le 3 \cdot \frac{1-\varepsilon}{k}\text{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\,\text{OPT}$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



- If $\exists\, A \in \mathcal{A}'$ that crosses at least two S_i,

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

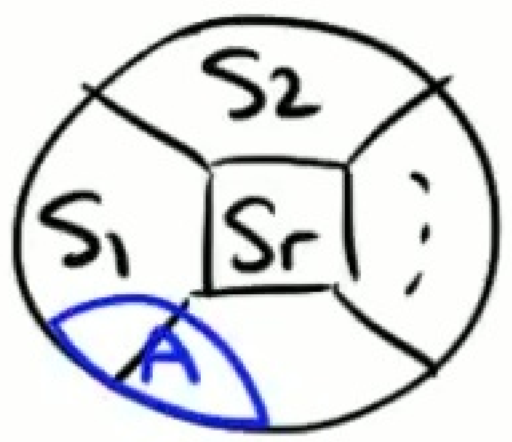
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} OPT$ for $+2$ component, or

  ② $\le 3 \cdot \frac{1-\varepsilon}{k} OPT$ for $+3$ components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i,

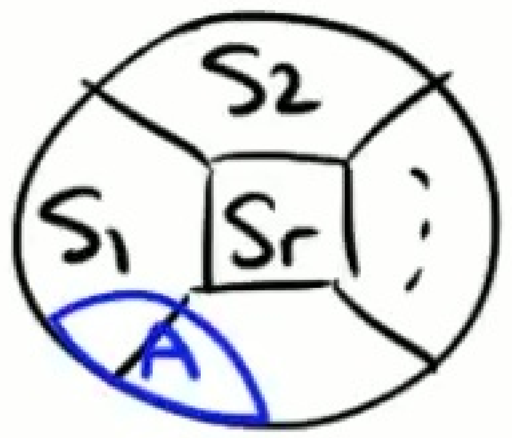
Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} OPT$ for +2 component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} OPT$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



- If $\exists\, A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A

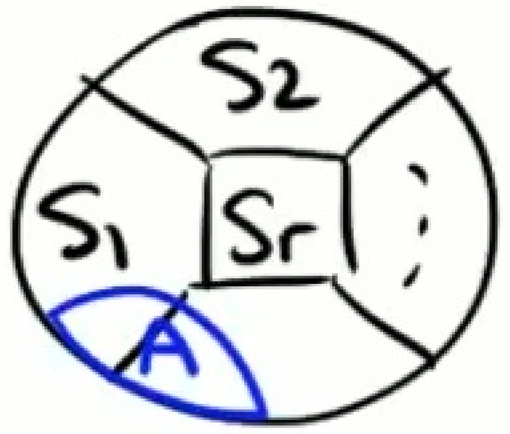
Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +2 component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\,\mathrm{OPT}$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \ldots, S_r\}$:



- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k} \mathrm{OPT}$ for +2 components, satisfies ①.

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

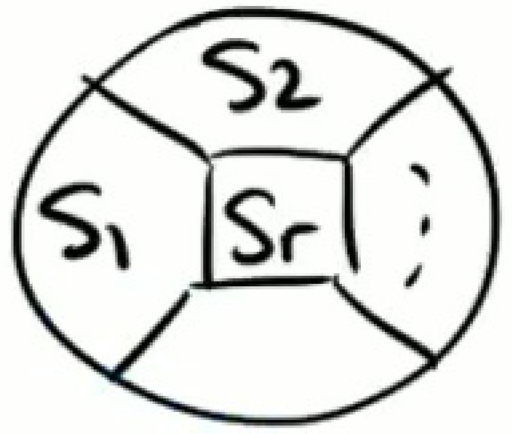
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k}\mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k}\mathrm{OPT}$ for +2 component, or

  ② $\le 3 \cdot \frac{1-\varepsilon}{k}\mathrm{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\mathrm{OPT}$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \ldots, S_r\}$:



- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k}\mathrm{OPT}$ for +2 components, satisfies ①.

Else, for each $A \in \mathcal{A}'$:

either crosses no S_i

(i.e. $A \cap S_i = \emptyset$ or S_i $\forall S_i$)

or crosses one S_i

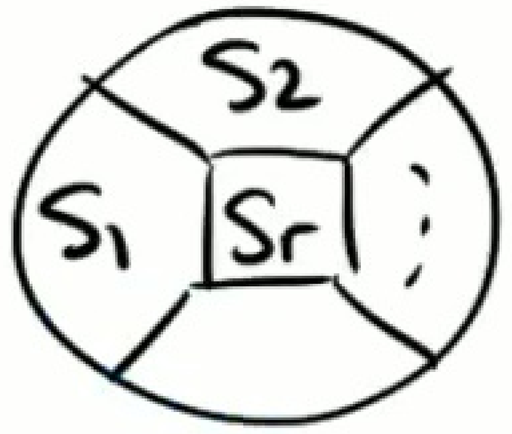
Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +2 component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\mathrm{OPT}$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



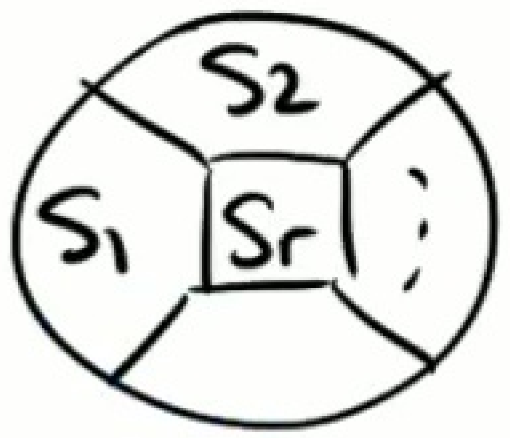
- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k} \mathrm{OPT}$ for +2 components, satisfies ①.

Else, for each $A \in \mathcal{A}'$:
either crosses no S_i — at most 2^r such possibilities, ignore
(i.e. $A \cap S_i = \emptyset$ or S_i $\forall S_i$)
or crosses one S_i

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$
$|\mathcal{A}'| = O_k(n)$
$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} OPT$ for $+2$ component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} OPT$ for $+3$ components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k} OPT$ for $+2$ components, satisfies ①.

Else, for each $A \in \mathcal{A}'$:
either crosses no S_i (i.e. $A \cap S_i = \emptyset$ or S_i $\forall S_i$) → at most 2^r such possibilities, ignore
or crosses one S_i → group each $A \in \mathcal{A}'$ based on which S_i it intersects

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

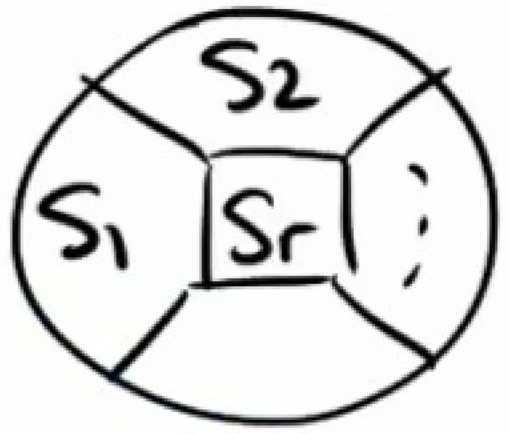
$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} OPT$ for +2 component, or

  ② $\le 3 \cdot \frac{1-\varepsilon}{k} OPT$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon) OPT$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:



- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k} OPT$ for +2 components, satisfies ①.

Else, for each $A \in \mathcal{A}'$:

either crosses no S_i — at most 2^r such possibilities, ignore

(i.e. $A \cap S_i = \emptyset$ or S_i $\forall S_i$)

or crosses one S_i — group each $A \in \mathcal{A}'$ based on which S_i it intersects

Since $|\mathcal{A}'| \ge \Omega(2^k n)$, $\exists$ group with $\ge 2n$ sets

Formal Proof: $|\mathcal{A}'| = O(2^k n)$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

- Recall: assume for contradiction that $|\mathcal{A}'| \ge \Omega(2^k n)$.
- Build k-cut iteratively, beginning with $\{V\}$
- Each iteration: pay ① $\le 2 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +2 component, or
  ② $\le 3 \cdot \frac{1-\varepsilon}{k} \mathrm{OPT}$ for +3 components
- End up with k-cut value $\lesssim (1-\varepsilon)\mathrm{OPT}$ (k large enough)
- First iteration: can do ② as shown before
- Current cut $\{S_1, S_2, \dots, S_r\}$:

- If $\exists A \in \mathcal{A}'$ that crosses at least two S_i, then greedily cut edges in ∂A
- Pay $\frac{1.49}{k}\mathrm{OPT}$ for +2 components, satisfies ①.

Else, for each $A \in \mathcal{A}'$:
either crosses no S_i → at most 2^r such possibilities, ignore
(i.e. $A \cap S_i = \emptyset$ or S_i $\forall S_i$)
or crosses one S_i → group each $A \in \mathcal{A}'$ based on which S_i it intersects

Since $|\mathcal{A}'| \ge \Omega(2^k n)$, $\exists$ group with $\ge 2n$ sets

Find a crossing pair in that group

Get +3 components for $2 \cdot \frac{1.49}{k}\mathrm{OPT}$, satisfying ②.

Computing $\mathcal{A}'$

- Idea: modified Karger-Stein algorithm.

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

Computing $\mathcal{A}'$

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $> k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.
- Repeat poly(n) times.

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.
- Repeat poly(n) times.

Similar analysis: analyze prob failure when r vertices left $(r = n, n-1, \dots, k+1)$.

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.
- Repeat poly(n) times.

Similar analysis: analyze prob failure when r vertices left $(r = n, n-1, \ldots, k+1)$.

From before: $(\text{min } k\text{-cut}) \le (k-1) \cdot \frac{2 \sum_e w_e}{r}$.

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.
- Repeat poly(n) times.

Similar analysis: analyze prob failure when r vertices left ($r = n, n-1, \ldots, k+1$).

From before: $(\min k\text{-cut}) \leq (k-1) \cdot \frac{2 \sum_e w_e}{r}$.

$$\Pr[\text{failure}] = \frac{w(\partial_G A)}{\sum_e w_e} \leq \frac{\frac{1.49}{k} OPT}{\sum_e w_e}$$

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow \mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:
  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.
- Repeat poly(n) times.

Similar analysis: analyze prob failure when r vertices left $(r = n, n-1, \ldots, k+1)$.

From before: $(\text{min } k\text{-cut}) \le \cancel{(k-1)} \cdot \frac{2 \cancel{\sum_e w_e}}{r}$.

$$\Pr[\text{failure}] = \frac{w(\partial_G A)}{\sum_e w_e} \le \frac{\frac{1.49}{\cancel{k}} OPT}{\cancel{\sum_e w_e}} \le \frac{2.98}{r}$$

Computing $\mathcal{A}'$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\rightarrow$ $\mathcal{A}'$ computed in polytime

- Idea: modified Karger-Stein algorithm.

  While G has $>k$ vertices left:

  - Contract random edge (proportional to weight)
  - Merge parallel edges

  For each of the 2^k subsets, output the set formed by uncontracting it.

- Repeat poly(n) times.

Similar analysis: analyze prob failure when r vertices left $(r = n, n-1, \ldots, k+1)$.

From before: $(\text{min } k\text{-cut}) \le \cancel{(k-1)} \cdot \frac{2 \cancel{\sum_e w_e}}{r}$.

$$\Pr[\text{failure}] = \frac{w(\partial_G A)}{\sum_e w_e} \le \frac{\frac{1.49}{\cancel{k}} \mathrm{OPT}}{\cancel{\sum_e w_e}} \le \frac{2.98}{r}$$

$$\Longrightarrow \Pr[\text{success}] = \left(1 - \frac{2.98}{n}\right)\left(1 - \frac{2.98}{n-1}\right) \cdots \gtrsim \frac{1}{n^{2.98}}$$

- Repeat $\Theta(n^{2.98} \log n)$ times. Output the $\Theta(2^k n)$ sets with smallest boundaries found.

What if $\mathcal{S}$ "balanced"?

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k} OPT \;\; \forall i$

What if $\mathcal{S}$ "balanced"?

Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT} \;\; \forall i$

- Could try taking $\mathcal{A} = \{A \subseteq V : w(\partial_G A) \le \frac{2}{k}\mathrm{OPT}\}$

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k}\mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

What if $\mathcal{S}$ "balanced"?

Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT} \ \forall i$

- Could try taking $\mathcal{A} = \{A \subseteq V : w(\partial_G A) \leq \frac{2}{k}\mathrm{OPT}\}$
- Problem: can have $|\mathcal{A}| \geq \Omega(n^2)$.

$\implies$ recursion is $T(n,k) \geq \Omega(n^2) \cdot T(n, k-1)$

solves to n^{2k}, no good!

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \leq \frac{1.49}{k}\mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

What if $\mathcal{S}$ "balanced"?

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k} OPT \ \forall i$

- Could try taking $\mathcal{A} = \{A \subseteq V : w(\partial_G A) \le \frac{2}{k} OPT\}$
- Problem: can have $|\mathcal{A}| \ge \Omega(n^2)$.

  $\Rightarrow$ recursion is $T(n,k) \ge \Omega(n^2) \cdot T(n, k-1)$

  solves to n^{2k}, no good!
- Message: cannot restart from scratch each time. Keep global measure of progress.

What if $\mathcal{S}$ "balanced"?

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} OPT\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k} OPT \ \forall i$

- Could try taking $\mathcal{A} = \{A \subseteq V : w(\partial_G A) \le \frac{2}{k} OPT\}$
- Problem: can have $|\mathcal{A}| \ge \Omega(n^2)$.

  $\implies$ recursion is $T(n,k) \ge \Omega(n^2) \cdot T(n,k-1)$

  solves to n^{2k}, no good!
- Message: cannot restart from scratch each time. Keep global measure of progress.
- Every time we branch on a large $\mathcal{A}$ (e.g. $|\mathcal{A}| \ge \Omega(n^2)$), make sure we make enough progress.

What if $\mathcal{S}$ "balanced"?

$\mathcal{A}' := \{A \subseteq V : w(\partial_G A) \le \frac{1.49}{k} \mathrm{OPT}\}$

$|\mathcal{A}'| = O_k(n)$

$\mathcal{A}'$ computed in polytime

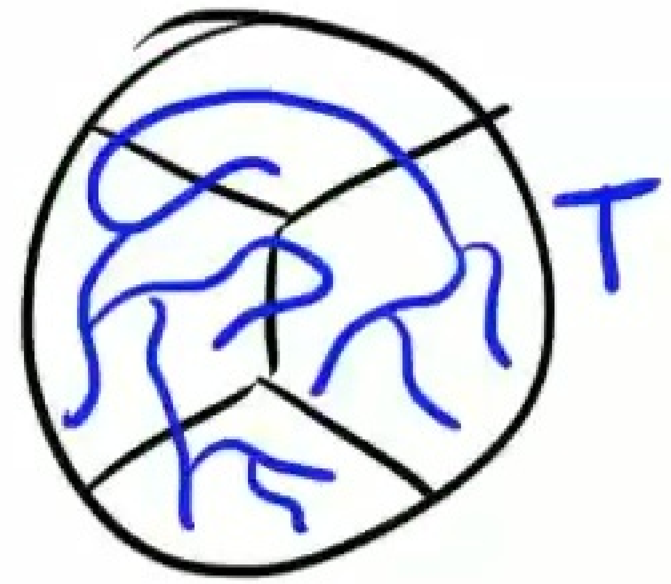
Example: $\mathcal{S}$ is "perfectly balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT} \ \forall i$

- Could try taking $\mathcal{A} = \{A \subseteq V : w(\partial_G A) \le \frac{2}{k}\mathrm{OPT}\}$
- Problem: can have $|\mathcal{A}| \ge \Omega(n^2)$.
  $\Longrightarrow$ recursion is $T(n,k) \ge \Omega(n^2) \cdot T(n, k-1)$
  solves to n^{2k}, no good!
- Message: cannot restart from scratch each time. Keep global measure of progress.
- Every time we branch on a large $\mathcal{A}$ (e.g. $|\mathcal{A}| \ge \Omega(n^2)$), make sure we make enough progress.
- Progress is based on tree packing [Thorup 2008].

Thorup's Tree Packing

Thm [Thorup '08] Can compute in poly(n) time a set $\mathcal{T}$ of poly(n) spanning trees of G, s.t. for any min k-cut S, there exists $T \in \mathcal{T}$ that crosses S $\leq 2k-2$ times.

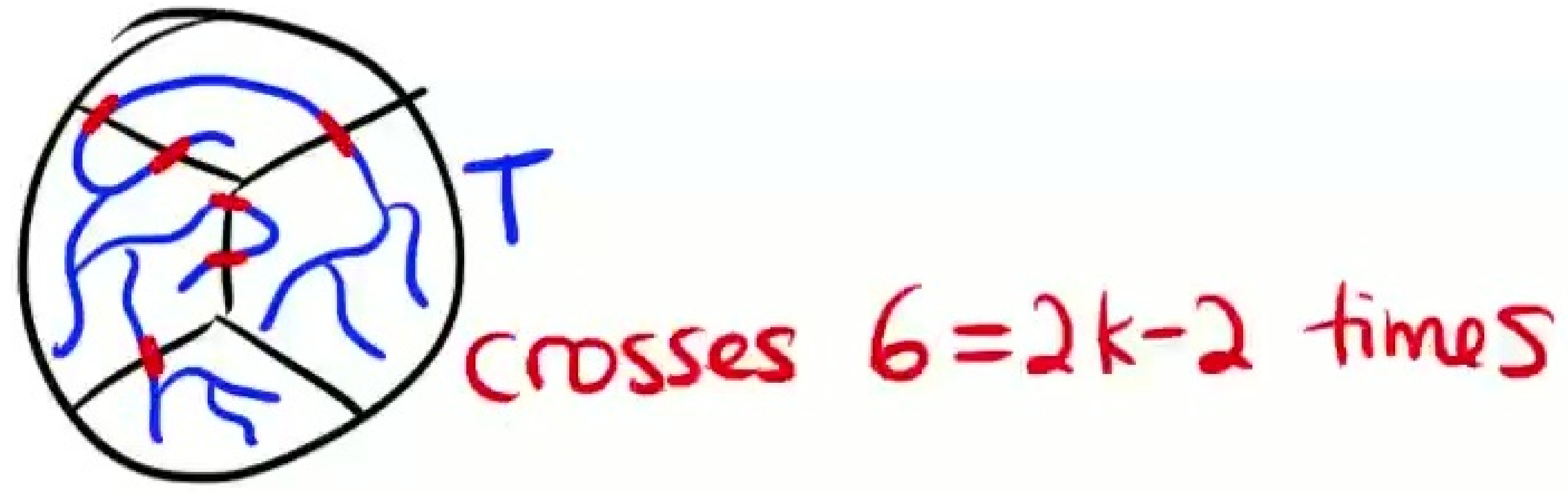
$k=4$:



Thorup's Tree Packing

Thm [Thorup '08] Can compute in poly(n) time a set $\mathcal{T}$ of poly(n) spanning trees of G, s.t. for any min k-cut S, there exists $T \in \mathcal{T}$ that crosses S $\leq 2k-2$ times.

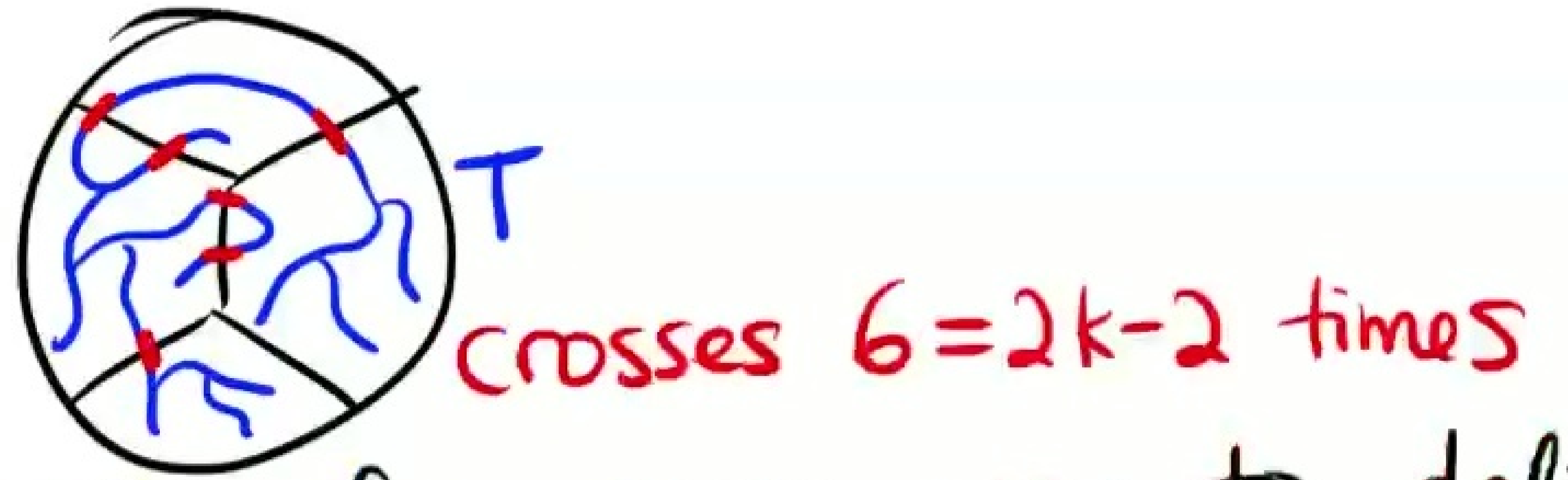
$k=4$:



Thorup's Tree Packing

Thm [Thorup '08] Can compute in poly(n) time a set $\mathcal{T}$ of poly(n) spanning trees of G, s.t. for any min k-cut S, there exists $T \in \mathcal{T}$ that crosses S $\leq 2k-2$ times.

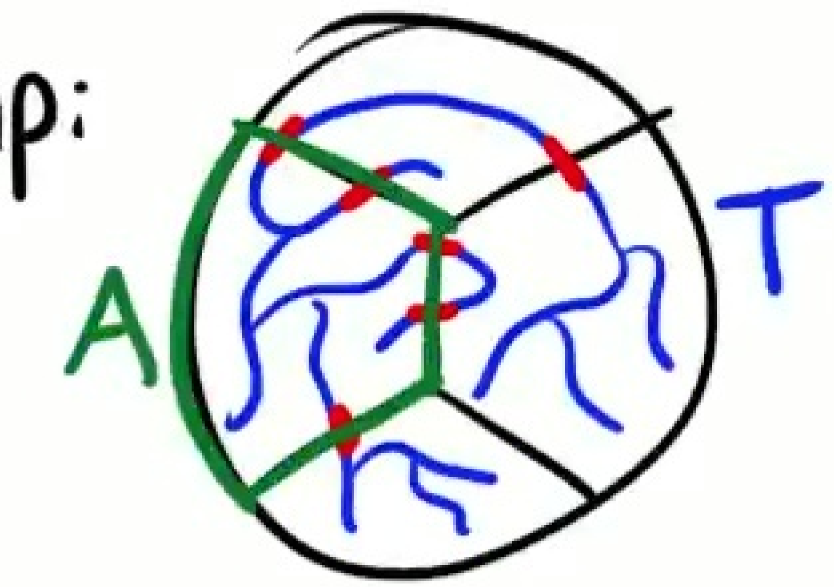
$k=4$:



Thorup's min k-cut algo: for each $T \in \mathcal{T}$, for every way to delete $\leq 2k-2$ edges of T and merge the components into k components, compute the min k-cut value.

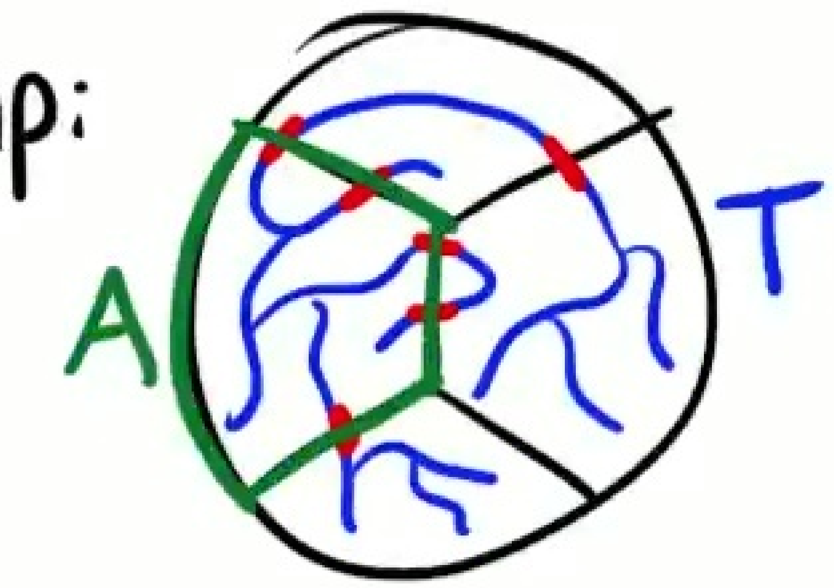
Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:



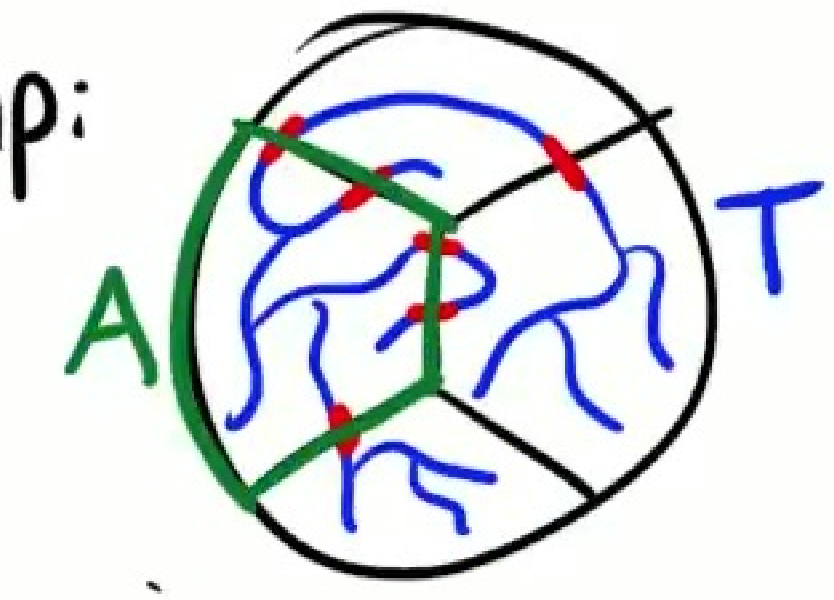
Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:
- If we cut out A, we also delete many edges of T.



Tree as a Measure of Progress

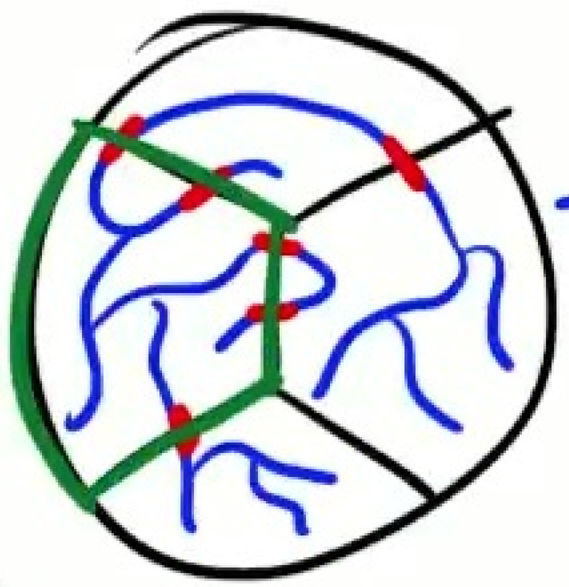
- Suppose $\mathcal{A}$ contains this comp:



- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)

Tree as a Measure of Progress

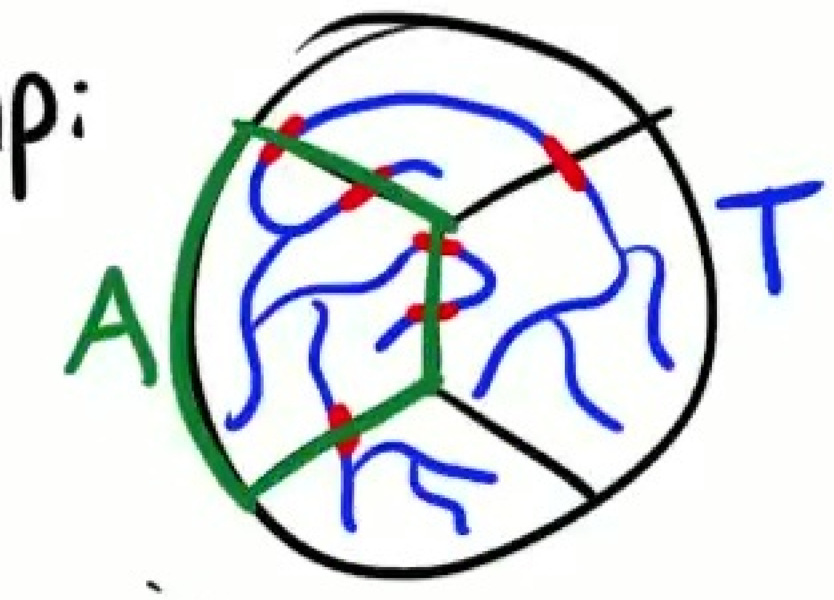
- Suppose $\mathcal{A}$ contains this comp:



- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time

Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:

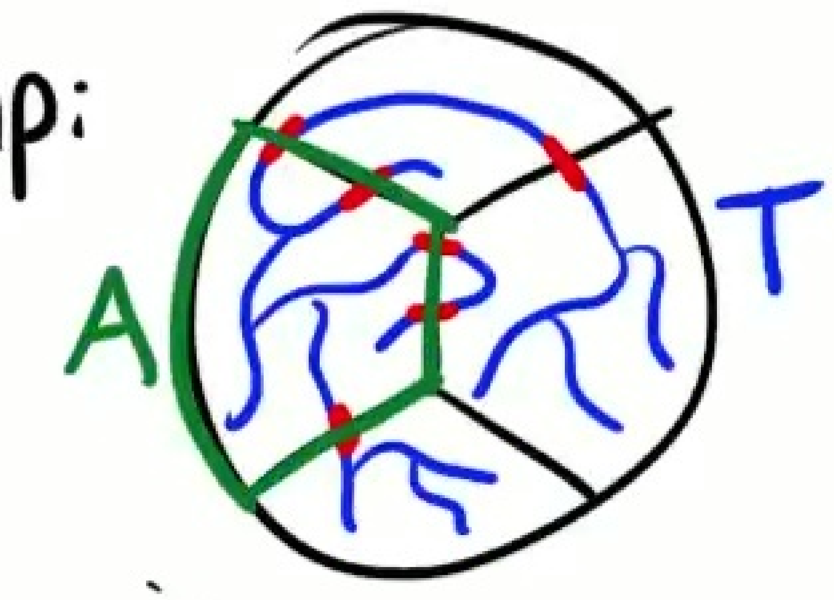


- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time

$\Longrightarrow$ can "beat brute force"

Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:



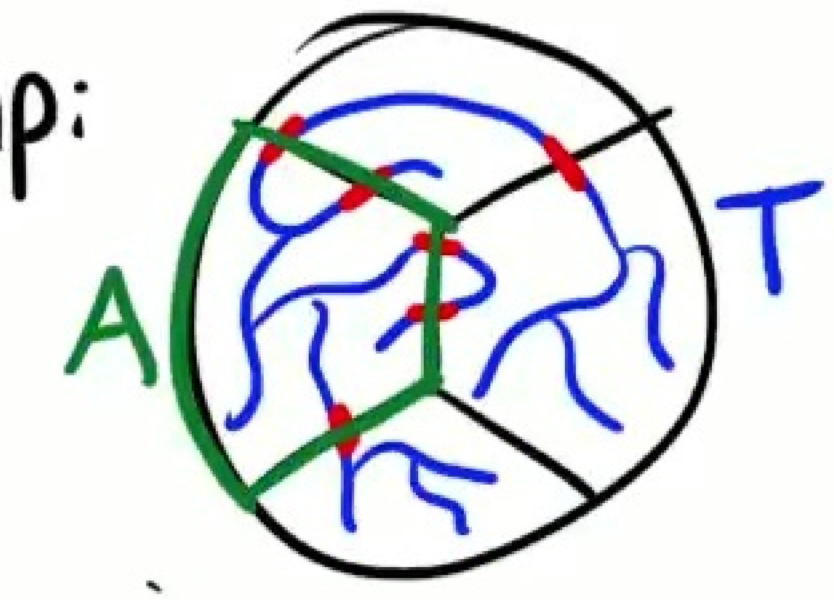
- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time

$\Longrightarrow$ can "beat brute force"

Measure of progress: how many edges of T have we deleted?

Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:

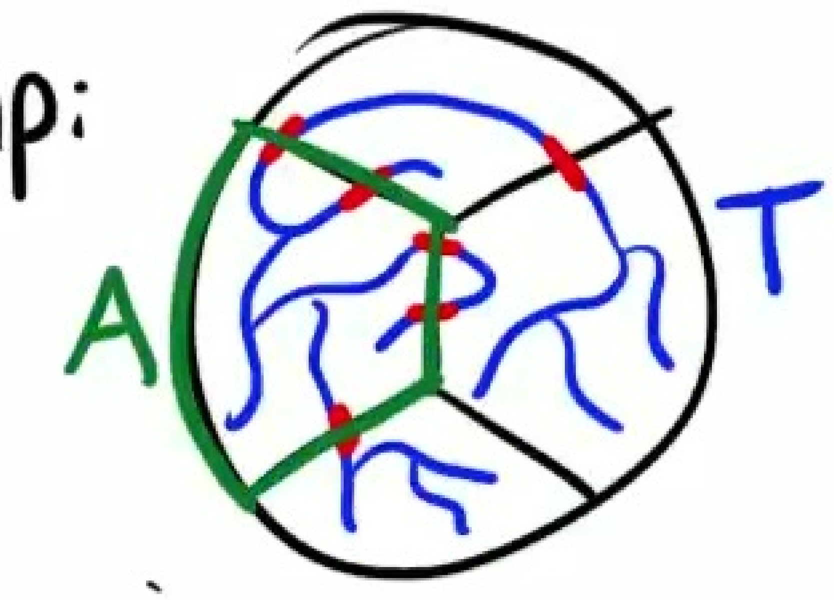


- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time
  $\Longrightarrow$ can "beat brute force"

Measure of progress: how many edges of T have we deleted?
If deleted s so far, only need to guess remaining $(2k-2)-s$ edges (n^{2k-2-s} time).

Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:



- If we cut out A, we also delete many edges of T.
- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time

$\Longrightarrow$ can "beat brute force"

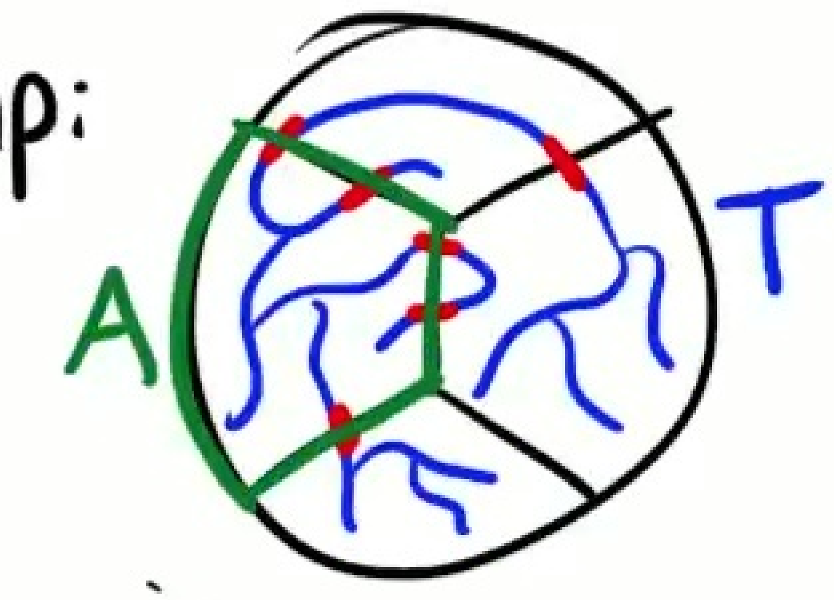
Measure of progress: how many edges of T have we deleted?

If deleted s so far, only need to guess remaining $(2k-2)-s$ edges (n^{2k-2-s} time).

If $|\mathcal{A}| = n^{\beta}$ and $\exists A \in \mathcal{A}$ cutting $> \beta$ edges of T, then we make progress.

Tree as a Measure of Progress

- Suppose $\mathcal{A}$ contains this comp:
- If we cut out A, we also delete many edges of T.



- Guessing $A \in \mathcal{A}$ is "worth" guessing 5 edges of T to delete (in Thorup's brute-force algo)
- As long as $|\mathcal{A}| \ll n^5$, we can guess 5 edges in $\ll n^5$ time

$\Longrightarrow$ can "beat brute force"

Measure of progress: how many edges of T have we deleted?
If deleted s so far, only need to guess remaining $(2k-2)-s$ edges (n^{2k-2-s} time).

If $|\mathcal{A}| = n^\beta$ and $\exists A \in \mathcal{A}$ cutting $> \beta$ edges of T, then we make progress.

E.g. $|\mathcal{A}'| = O_k(n)$, so if $\exists S_i \in \mathcal{A}'$ cutting ≥ 2 edges of T, then branch on $\mathcal{A}$.

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k} OPT$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists S_i \in \mathcal{A}$ cutting ≥ 4 edges of T.

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists\, S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists\, S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

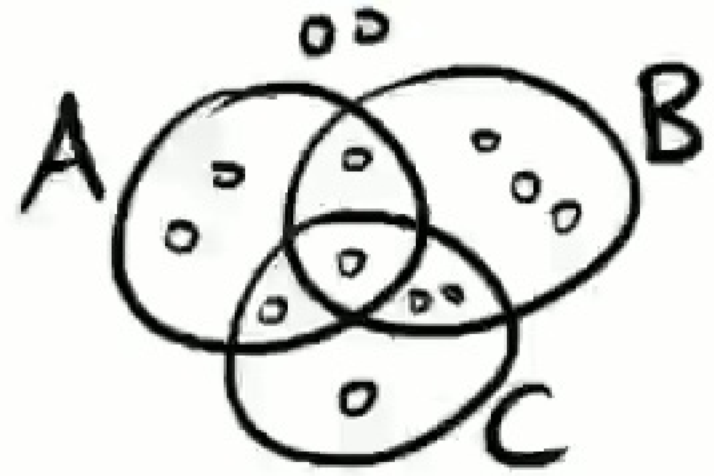
Thm: Let $\mathcal{A} := \{A \subseteq V : w(\partial_G S_i) \leq \frac{2.3}{k}\mathrm{OPT}\}$. Then, $|\mathcal{A}| \leq n^{3.75}$

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists\, S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

Thm: Let $\mathcal{A} := \{ A \subseteq V : w(\partial_G S_i) \leq \frac{2.3}{k}\mathrm{OPT} \}$. Then, $|\mathcal{A}| \leq n^{3.75}$

Pf: Use extremal result: if $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\exists$ sets A, B, C:

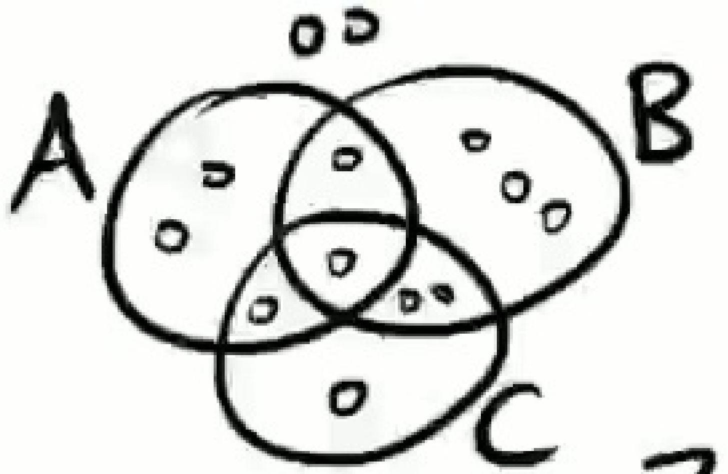


Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

Thm: Let $\mathcal{A} := \{ A \subseteq V : w(\partial_G S_i) \leq \frac{2.3}{k}\mathrm{OPT} \}$. Then, $|\mathcal{A}| \leq n^{3.75}$

Pf: Use extremal result: if $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\exists$ sets A, B, C:



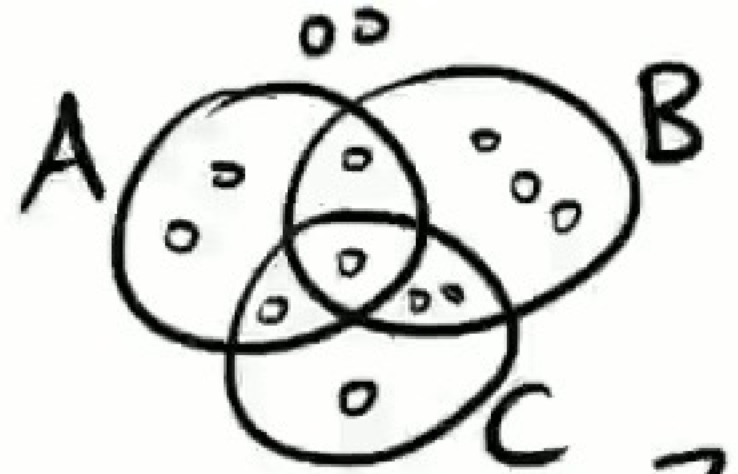
Cut out A, B, C: pay $3 \cdot \frac{2.3}{k}\mathrm{OPT} = \frac{7-\varepsilon}{k}\mathrm{OPT}$ for $+7$ components.

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\mathrm{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists\, S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

Thm: Let $\mathcal{A} := \{A \subseteq V : w(\partial_G S_i) \leq \frac{2.3}{k}\mathrm{OPT}\}$. Then, $|\mathcal{A}| \leq n^{3.75}$

Pf: Use extremal result: if $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\exists$ sets A, B, C:



Cut out A, B, C: pay $3 \cdot \frac{2.3}{k}\mathrm{OPT} = \frac{7-\varepsilon}{k}\mathrm{OPT}$ for $+7$ components.

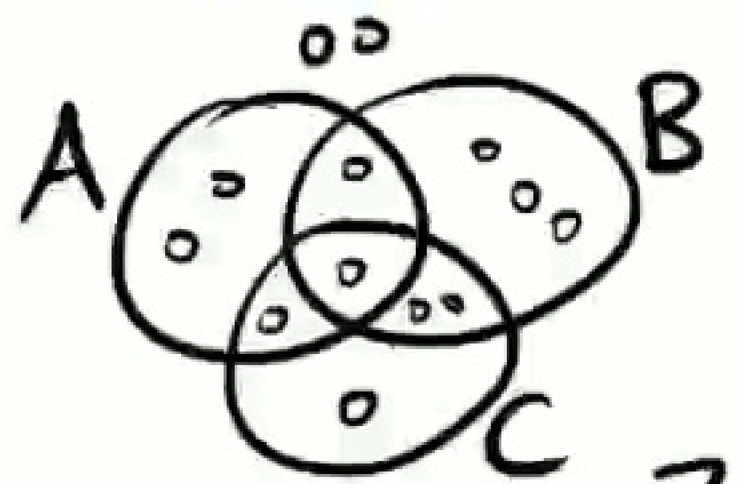
Construct cut iteratively, cost $< \mathrm{OPT}$, contradiction.

Balanced Example

- Suppose G is "balanced": $w(\partial_G S_i) = \frac{2}{k}\text{OPT}$ for all S_i.
- To make progress, find $\mathcal{A}$, $|\mathcal{A}| = O_k(n^{4-\varepsilon})$, s.t. $\exists\, S_i \in \mathcal{A}$ cutting ≥ 4 edges of T. (One exists because avg $|\partial_T S_i|$ is $\frac{4k-4}{k} \approx 4$)

Thm: Let $\mathcal{A} := \{A \subseteq V : w(\partial_G S_i) \leq \frac{2.3}{k}\text{OPT}\}$. Then, $|\mathcal{A}| \leq n^{3.75}$

Pf: Use extremal result: if $|\mathcal{A}| \geq \Omega(n^{3.75})$, then $\exists$ sets A, B, C:



Cut out A, B, C: pay $3 \cdot \frac{2.3}{k}\text{OPT} = \frac{7-\varepsilon}{k}\text{OPT}$ for $+7$ components.

Construct cut iteratively, cost $<$ OPT, contradiction.

To find $\mathcal{A}$: run modified Karger-Stein again, output smallest $\Theta(n^{3.75})$ cuts.

Algorithm Outline

Algorithm Outline

Let

$$\mathcal{A}^2 = \left\{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} \mathrm{OPT} \text{ and } |\partial_T A| = 2\right\} \quad \left(|\mathcal{A}^2| = O_k(n)\right)$$

$$\mathcal{A}^4 = \left\{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} \mathrm{OPT} \text{ and } |\partial_T A| = 4\right\} \quad \left(|\mathcal{A}^4| = O_k(n^{3.75})\right)$$

Algorithm Outline

Let

$$\mathcal{A}^2 = \{A : \underbrace{w(\partial_G A) \le \tfrac{1.5-\varepsilon}{k} OPT}_{A'} \text{ and } |\partial_T A| = 2\} \quad (|\mathcal{A}^2| = O_k(n))$$

$$\mathcal{A}^4 = \{A : w(\partial_G A) \le \tfrac{2.33-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 4\} \quad (|\mathcal{A}^4| = O_k(n^{3.75}))$$

If $\exists\, S_i \in \mathcal{A}^2$ or $\mathcal{A}^4$, then can branch and make progress.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$

$\mathcal{A}^2 = \{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 2\}$ $\quad (|\mathcal{A}^2| = O_k(n))$

$\mathcal{A}^3 = \{A : w(\partial_G A) \le \frac{2-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 3\}$ $\quad (|\mathcal{A}^3| = O_k(n^{2.75}))$

$\mathcal{A}^4 = \{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 4\}$ $\quad (|\mathcal{A}^4| = O_k(n^{3.75}))$

If $\exists S_i \in \mathcal{A}^2$ or $\mathcal{A}^4$, then can branch and make progress.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$

$\mathcal{A}^2 = \{A : w(\partial_G A) \leq \frac{1.5-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 2\} \quad (|\mathcal{A}^2| = O_k(n))$

$\mathcal{A}^3 = \{A : w(\partial_G A) \leq \frac{2-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 3\} \quad (|\mathcal{A}^3| = O_k(n^{2.75}))$

$\mathcal{A}^4 = \{A : w(\partial_G A) \leq \frac{2.33-\varepsilon}{k} OPT \text{ and } |\partial_T A| = 4\} \quad (|\mathcal{A}^4| = O_k(n^{3.75}))$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \leq 4$ have comparatively large $w(\partial_G S_i)$.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$ recall: $\sum w(\partial_G A) = 2\,OPT$, $\sum |\partial_T A| = 4k-4$

$\mathcal{A}^2 = \{A : w(\partial_G A) \leq \frac{1.5-\varepsilon}{k} OPT$ and $|\partial_T A| = 2\}$ $\quad (|\mathcal{A}^2| = O_k(n))$ $\quad \gg 1$

$\mathcal{A}^3 = \{A : w(\partial_G A) \leq \frac{2-\varepsilon}{k} OPT$ and $|\partial_T A| = 3\}$ $\quad (|\mathcal{A}^3| = O_k(n^{2.75}))$ $\quad \gg 1.5$

$\mathcal{A}^4 = \{A : w(\partial_G A) \leq \frac{2.33-\varepsilon}{k} OPT$ and $|\partial_T A| = 4\}$ $\quad (|\mathcal{A}^4| = O_k(n^{3.75}))$ $\quad \gg 2$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \leq 4$ have comparatively large $w(\partial_G S_i)$.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$ recall: $\sum w(\partial_G A) = 2\,OPT$, $\sum |\partial_T A| = 4k-4$

$\mathcal{A}^2 = \{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} OPT$ and $|\partial_T A| = 2\}$ $\quad$ ($\gg 1$) $\quad (|\mathcal{A}^2| = O_k(n))$

$\mathcal{A}^3 = \{A : w(\partial_G A) \le \frac{2-\varepsilon}{k} OPT$ and $|\partial_T A| = 3\}$ $\quad$ ($\gg 1.5$) $\quad (|\mathcal{A}^3| = O_k(n^{2.75}))$

$\mathcal{A}^4 = \{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} OPT$ and $|\partial_T A| = 4\}$ $\quad$ ($\gg 2$) $\quad (|\mathcal{A}^4| = O_k(n^{3.75}))$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \le 4$ have comparatively large $w(\partial_G S_i)$.

But since average $w(\partial_T S_i) = \frac{4k-4}{k} \le 4$, this is "most" S_i.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$ recall: $\sum w(\partial_G A) = 2\,OPT$, $\sum |\partial_T A| = 4k-4$

$\mathcal{A}^2 = \{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} OPT$ and $|\partial_T A| = 2\}$ $\quad (|\mathcal{A}^2| = O_k(n))$ $\quad \gg 1$

$\mathcal{A}^3 = \{A : w(\partial_G A) \le \frac{2-\varepsilon}{k} OPT$ and $|\partial_T A| = 3\}$ $\quad (|\mathcal{A}^3| = O_k(n^{2.75}))$ $\quad \gg 1.5$

$\mathcal{A}^4 = \{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} OPT$ and $|\partial_T A| = 4\}$ $\quad (|\mathcal{A}^4| = O_k(n^{3.75}))$ $\quad \gg 2$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \le 4$ have comparatively large $w(\partial_G S_i)$.

But since average $w(\partial_T S_i) = \frac{4k-4}{k} \le 4$, this is "most" S_i.

So there must be some S_i : $|\partial_T S_i| \ge 5$, with comparatively small $w(\partial_G S_i)$.

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$ recall: $\sum w(\partial_G A) = 2\,OPT$, $\sum |\partial_T A| = 4k-4$

$\mathcal{A}^2 = \{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} OPT$ and $|\partial_T A| = 2\}$ $\quad (|\mathcal{A}^2| = O_k(n))$ $\quad \gg 1$

$\mathcal{A}^3 = \{A : w(\partial_G A) \le \frac{2-\varepsilon}{k} OPT$ and $|\partial_T A| = 3\}$ $\quad (|\mathcal{A}^3| = O_k(n^{2.75}))$ $\quad \gg 1.5$

$\mathcal{A}^4 = \{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} OPT$ and $|\partial_T A| = 4\}$ $\quad (|\mathcal{A}^4| = O_k(n^{3.75}))$ $\quad \gg 2$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \le 4$ have comparatively large $w(\partial_G S_i)$.

But since average $w(\partial_T S_i) = \frac{4k-4}{k} \le 4$, this is "most" S_i.

So there must be some S_i : $|\partial_T S_i| \ge 5$, with comparatively small $w(\partial_G S_i)$.

$(\beta \ge 5)$ $\mathcal{A}^\beta = \{A : w(\partial_G A) \le \frac{(1-\varepsilon)\beta/2}{k} OPT$ and $|\partial_T A| = \beta\}$ $\quad (|\mathcal{A}^\beta| = O_k(n^{(1-\varepsilon)\beta}))$

default bound from modified K-S

Algorithm Outline

Let $\mathcal{A}^0 = \{A : w(\partial_T A) = 0\}$

$\mathcal{A}^1 = \{A : w(\partial_T A) = 1\}$ recall: $\sum w(\partial_G A) = 2\,OPT$, $\sum |\partial_T A| = 4k-4$

$\mathcal{A}^2 = \{A : w(\partial_G A) \le \frac{1.5-\varepsilon}{k} OPT$ and $|\partial_T A| = 2\}$ $\quad$ $(|\mathcal{A}^2| = O_k(n))$ $\quad$ $\gg 1$

$\mathcal{A}^3 = \{A : w(\partial_G A) \le \frac{2-\varepsilon}{k} OPT$ and $|\partial_T A| = 3\}$ $\quad$ $(|\mathcal{A}^3| = O_k(n^{2.75}))$ $\quad$ $\gg 1.5$

$\mathcal{A}^4 = \{A : w(\partial_G A) \le \frac{2.33-\varepsilon}{k} OPT$ and $|\partial_T A| = 4\}$ $\quad$ $(|\mathcal{A}^4| = O_k(n^{3.75}))$ $\quad$ $\gg 2$

Suppose no S_i in any of these; i.e. all S_i with $|\partial_T S_i| \le 4$ have comparatively large $w(\partial_G S_i)$.

But since average $w(\partial_T S_i) = \frac{4k-4}{k} \le 4$, this is "most" S_i.

So there must be some S_i: $|\partial_T S_i| \ge 5$, with comparatively small $w(\partial_G S_i)$.

$(\beta \ge 5)$ $\mathcal{A}^\beta = \{A : w(\partial_G A) \le \frac{(1-\varepsilon)\beta/2}{k} OPT$ and $|\partial_T A| = \beta\}$ $\quad$ $(|\mathcal{A}^\beta| = O_k(n^{(1-\varepsilon)\beta}))$

default bound from modified K-S

Claim: $\exists S_i$ in some $\mathcal{A}^j$. So, branch on all $\mathcal{A}^j$.

Recursive Algorithm

Recursive Algorithm

- Replace T with forest F

Recursive Algorithm

- Replace T with forest F
- Previous arguments have slack; as long as we still need to delete $(1-\varepsilon)\cdot 2k$ edges, can still make progress

Recursive Algorithm

- Replace T with forest F
- Previous arguments have slack; as long as we still need to delete $(1-\varepsilon)\cdot 2k$ edges, can still make progress
- Once we've made enough progress, brute-force remaining edges (like Thorup's algo)

Open Questions

Open Questions

Cleaner analysis? Better constant than 1.981

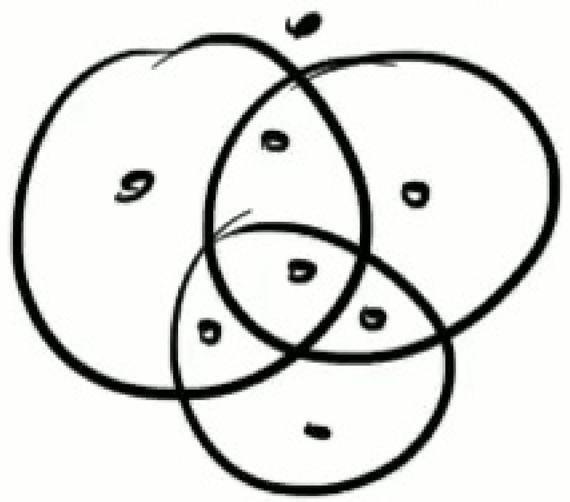
- New ideas likely needed to break 1.5

Open Questions

Cleaner analysis? Better constant than 1.981

- New ideas likely needed to break 1.5

Conjecture: If $|\mathcal{A}| \geq \Theta(n^3)$, then $\exists A, B, C$:



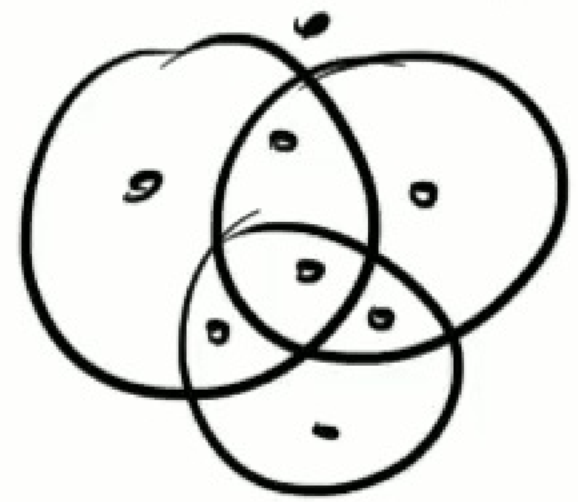
(easy lower bound of $\Omega(n^3)$)

Open Questions

Cleaner analysis? Better constant than 1.981

- New ideas likely needed to break 1.5

Conjecture: If $|\mathcal{A}| \geq \Theta(n^3)$, then $\exists A, B, C$:



(easy lower bound of $\Omega(n^3)$)

- Would imply slightly better bound