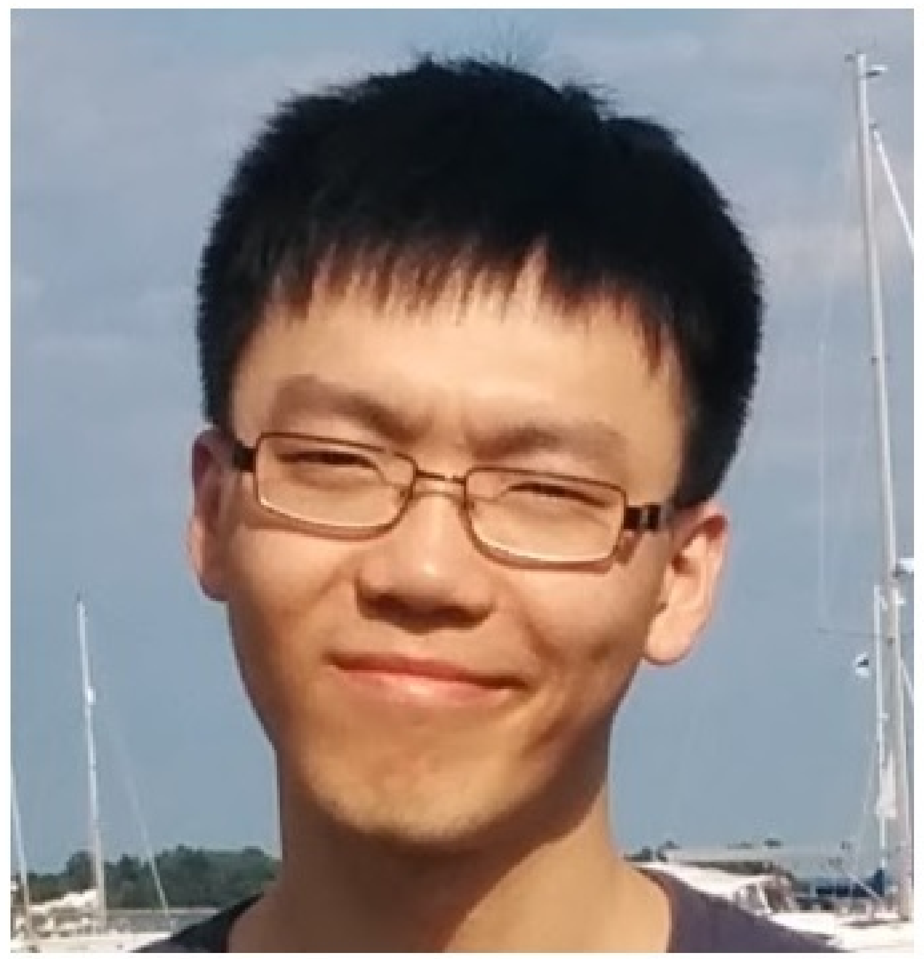
Deterministic Near-Linear Time Minimum Cut for Weighted Graphs

Jason Li (Berkeley→CMU)



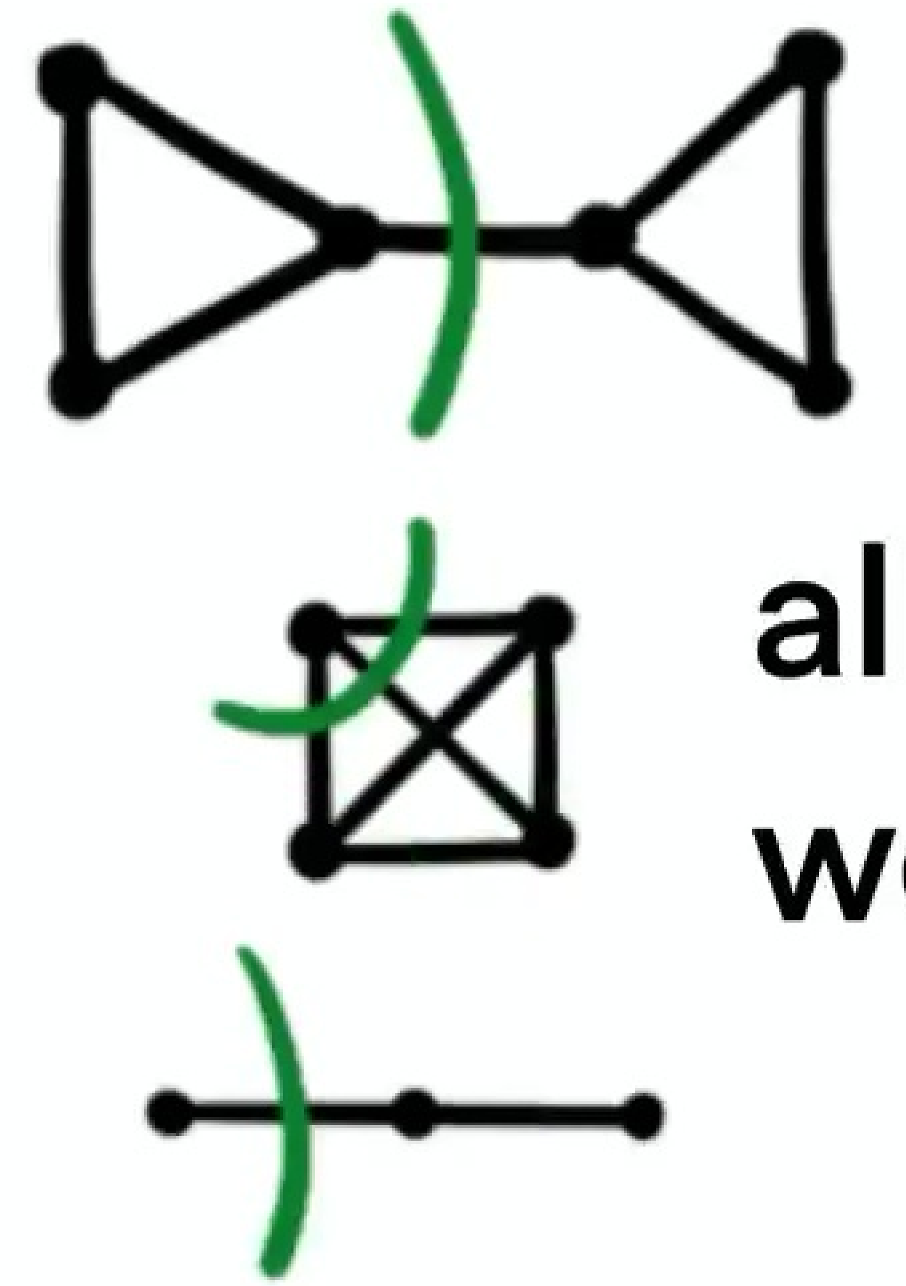
with Monika Henzinger Satish Rao Di Wang

Introduction

- Global mincut problem:
  given a weighted undirected graph,
  delete edges of minimum weight
  to disconnect the graph

Introduction

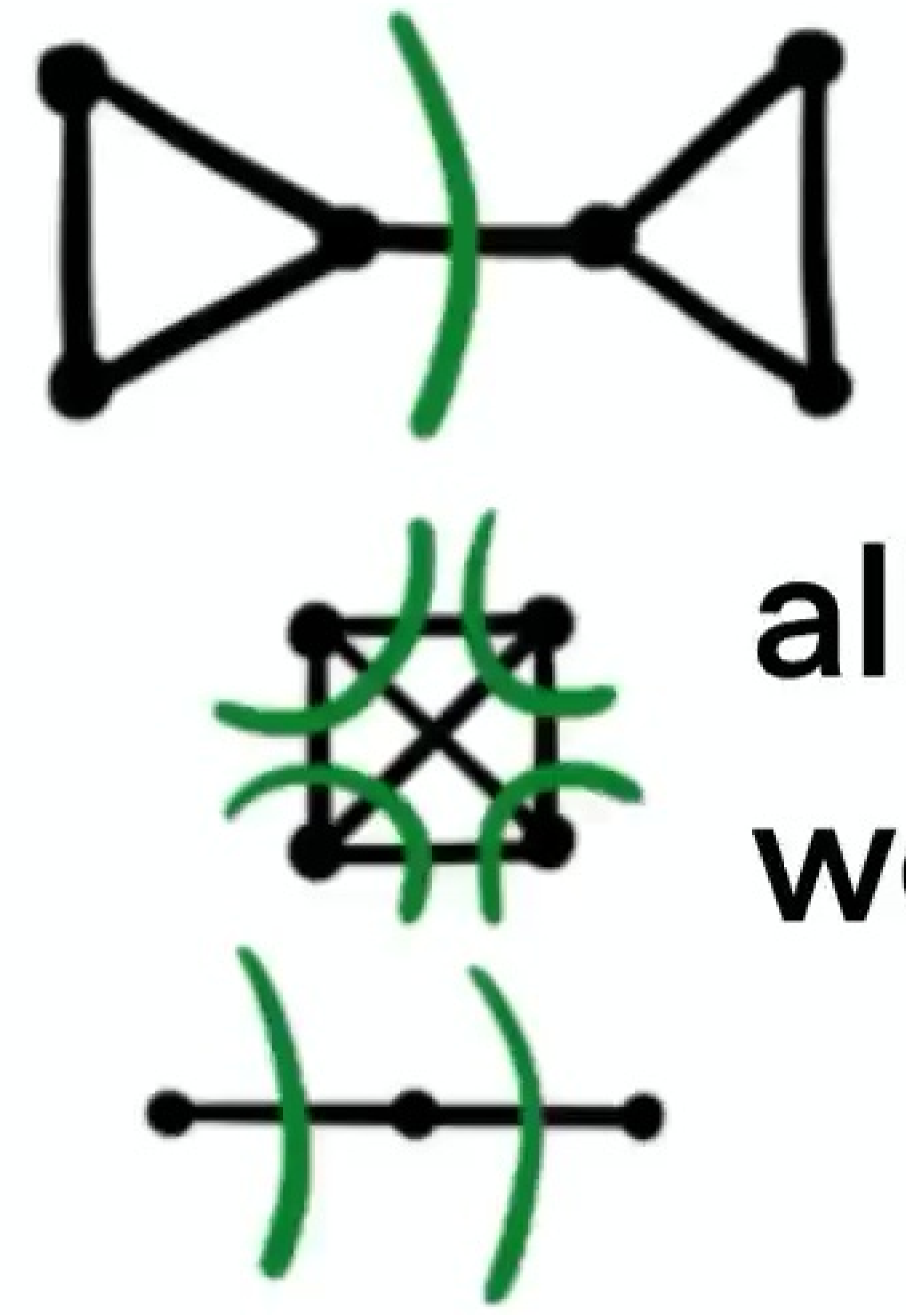
- Global mincut problem:
  given a weighted undirected graph,
  delete edges of minimum weight
  to disconnect the graph



all edges
weight 1

Introduction

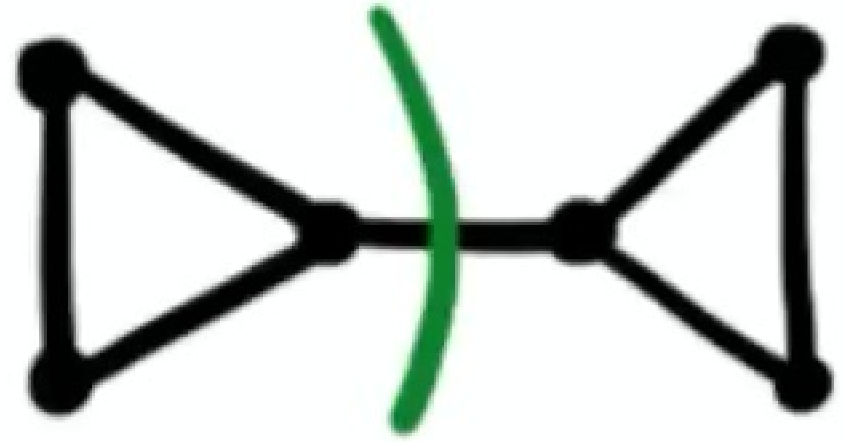
- Global mincut problem:
  given a weighted undirected graph,
  delete edges of minimum weight
  to disconnect the graph

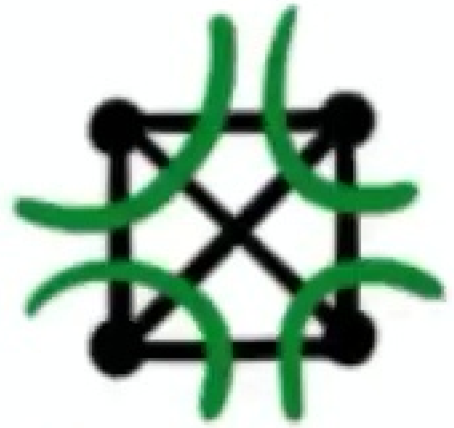


all edges
weight 1

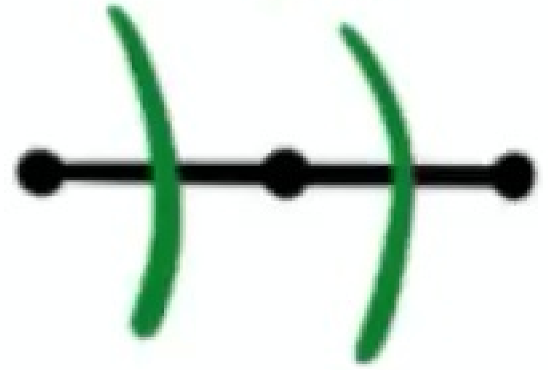
Introduction

- Global mincut problem:
  given a weighted undirected graph,
  delete edges of minimum weight
  to disconnect the graph





all edges
weight 1



- [Karger'96] randomized algorithm in $O(m\log^3 n)$ time
  "Is there a deterministic near-linear time algorithm?"

Introduction

- Best bound remained $\tilde{O}(mn)$ for 20 years until

Introduction

- Best bound remained $\tilde{O}(mn)$ for 20 years until
- [Kawarabayashi-Thorup'15] $\tilde{O}(m)$ for simple graphs

unweighted graphs without parallel edges

Introduction

- Best bound remained $\tilde{O}(mn)$ for 20 years until
- [Kawarabayashi-Thorup'15] $\tilde{O}(m)$ for simple graphs

unweighted graphs without parallel edges

- [Li-Panigrahi'20] max-flow time for weighted graphs

Introduction

- Best bound remained $\tilde{O}(mn)$ for 20 years until
- [Kawarabayashi-Thorup'15] $\tilde{O}(m)$ for simple graphs

unweighted graphs without parallel edges

- [Li-Panigrahi'20] max-flow time for weighted graphs
- [Li'21] $m^{1+o(1)}$ time independent of max-flow

"almost-linear time"

Introduction

- Best bound remained $\tilde{O}(mn)$ for 20 years until
- [Kawarabayashi-Thorup'15] $\tilde{O}(m)$ for simple graphs

  unweighted graphs without parallel edges

- [Li-Panigrahi'20] max-flow time for weighted graphs
- [Li'21] $m^{1+o(1)}$ time independent of max-flow

  "almost-linear time"

- [This work] $\tilde{O}(m)$ time, answering Karger's question

Outline

- Local win/win approach to global mincut

Outline

- Local win/win approach to global mincut
- Our main structural theorem

Outline

- Local win/win approach to global mincut
- Our main structural theorem
- New randomized $(1+\varepsilon)$-approximate global mincut
  (main conceptual contribution)

Outline

- Local win/win approach to global mincut
- Our main structural theorem
- New randomized $(1+\varepsilon)$-approximate global mincut
  (main conceptual contribution)
- Derandomize and obtain exact mincut
  (technical: 40+ pages)

Local Method

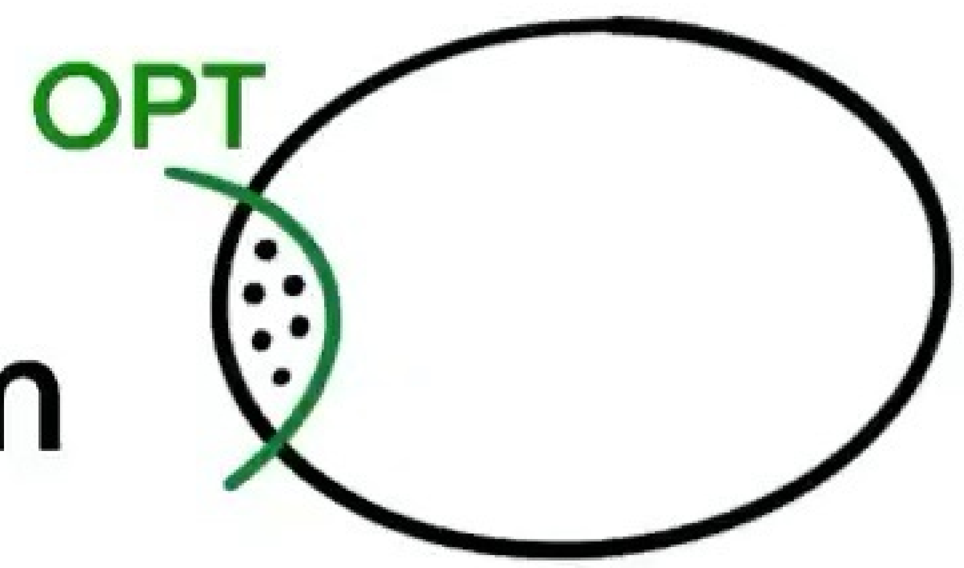
win/win approach:

- if solution is local, then run local algorithm

Local Method

win/win approach:

- if solution is local, then run local algorithm



Local Method

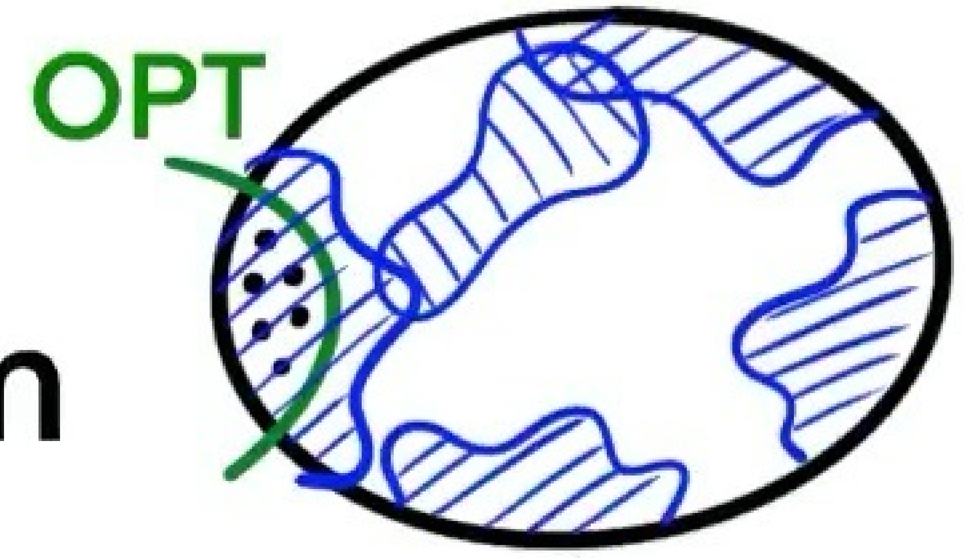
win/win approach:

- if solution is local, then run local algorithm

Local Method

win/win approach:

- if solution is local, then run local algorithm



Local Method

win/win approach:

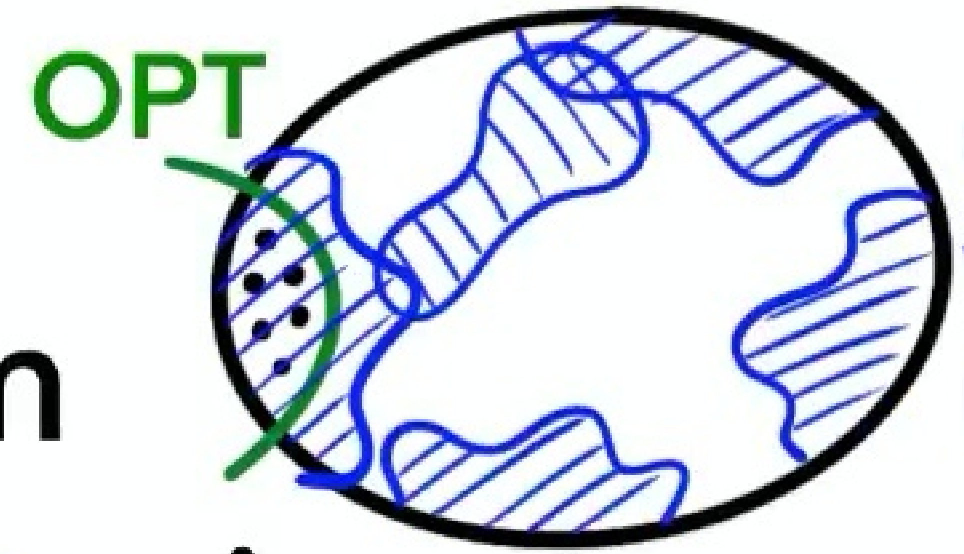
- if solution is local, then run local algorithm

Local Method

win/win approach:

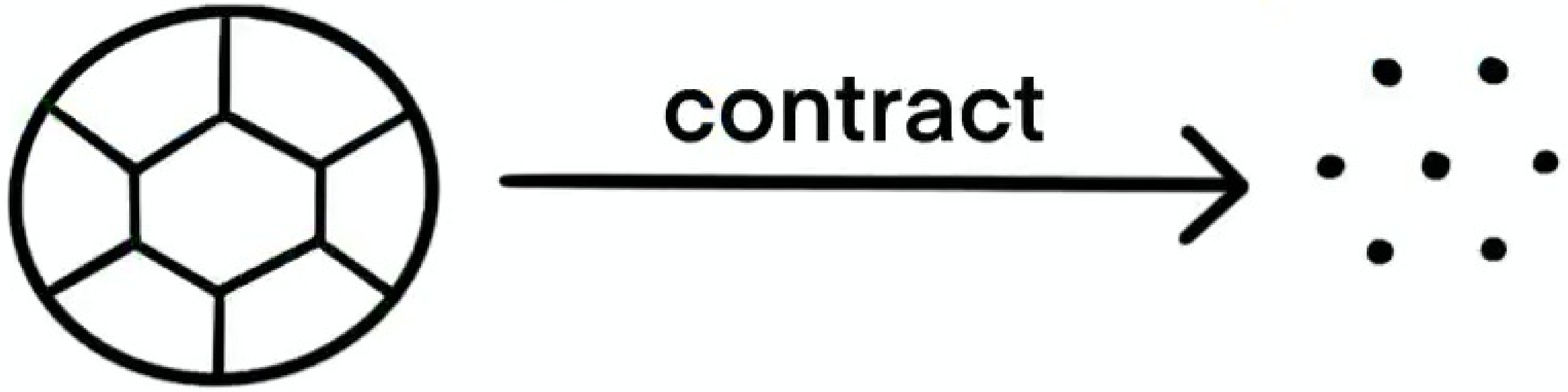
- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut

Local Method



win/win approach:

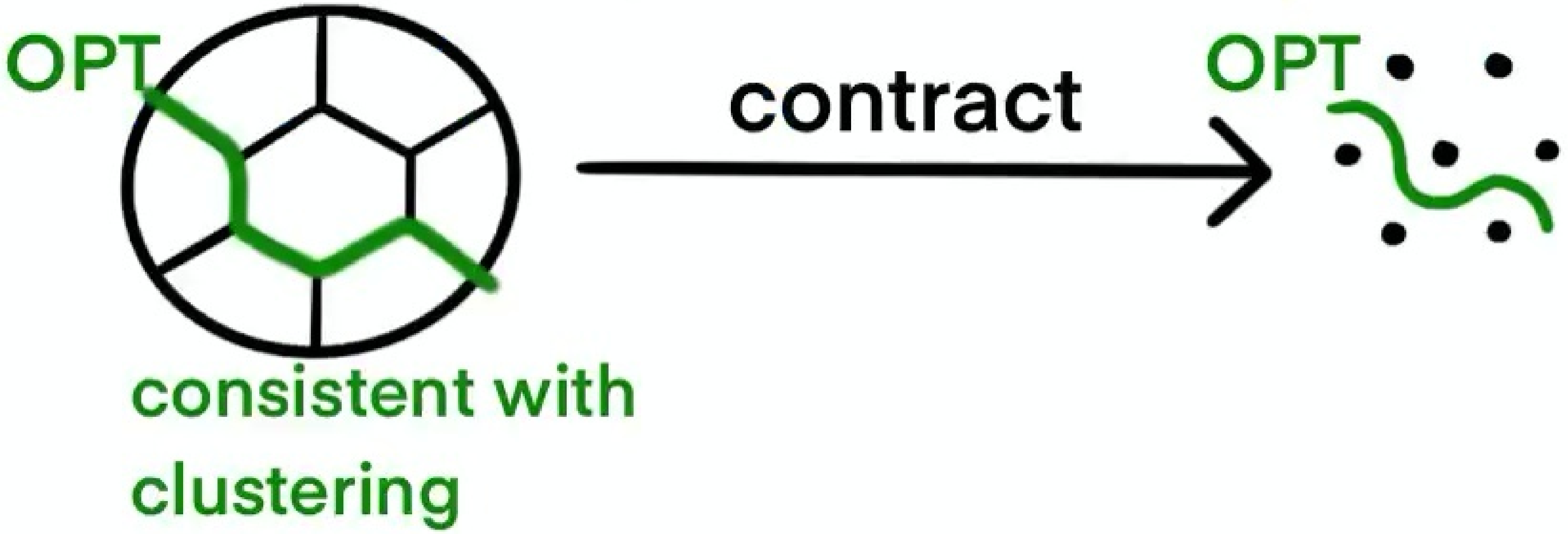
- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut



Local Method

win/win approach:

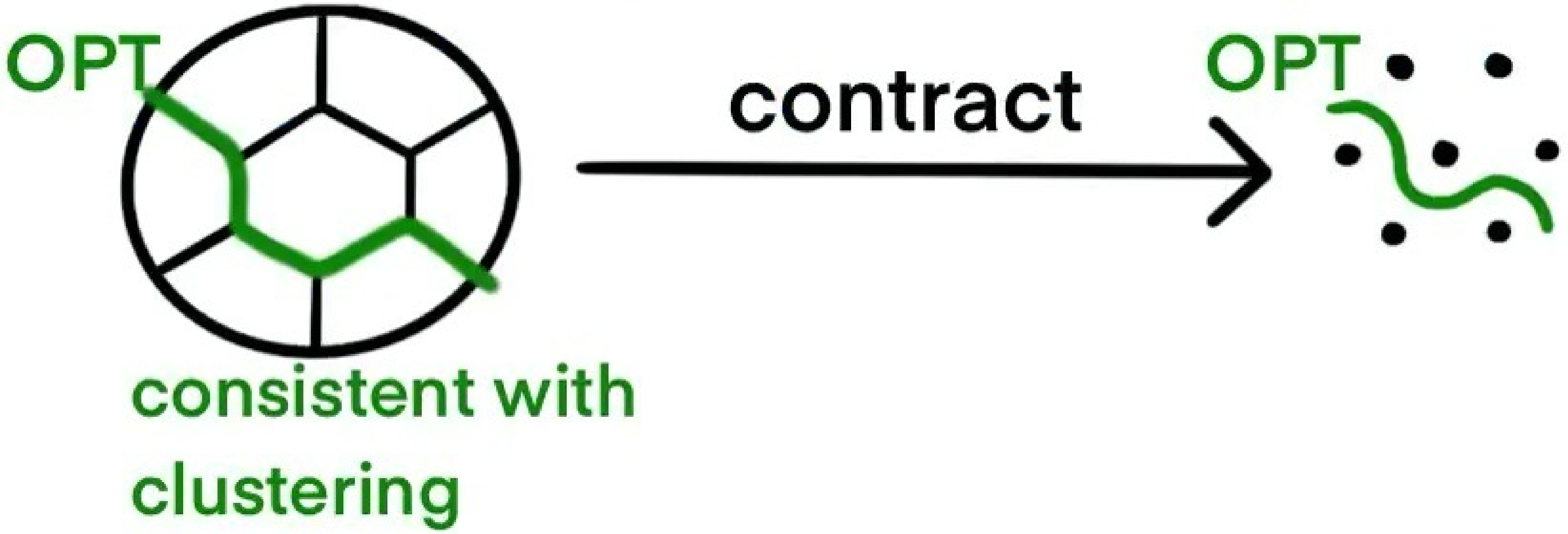
- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut



Local Method

win/win approach:

- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut



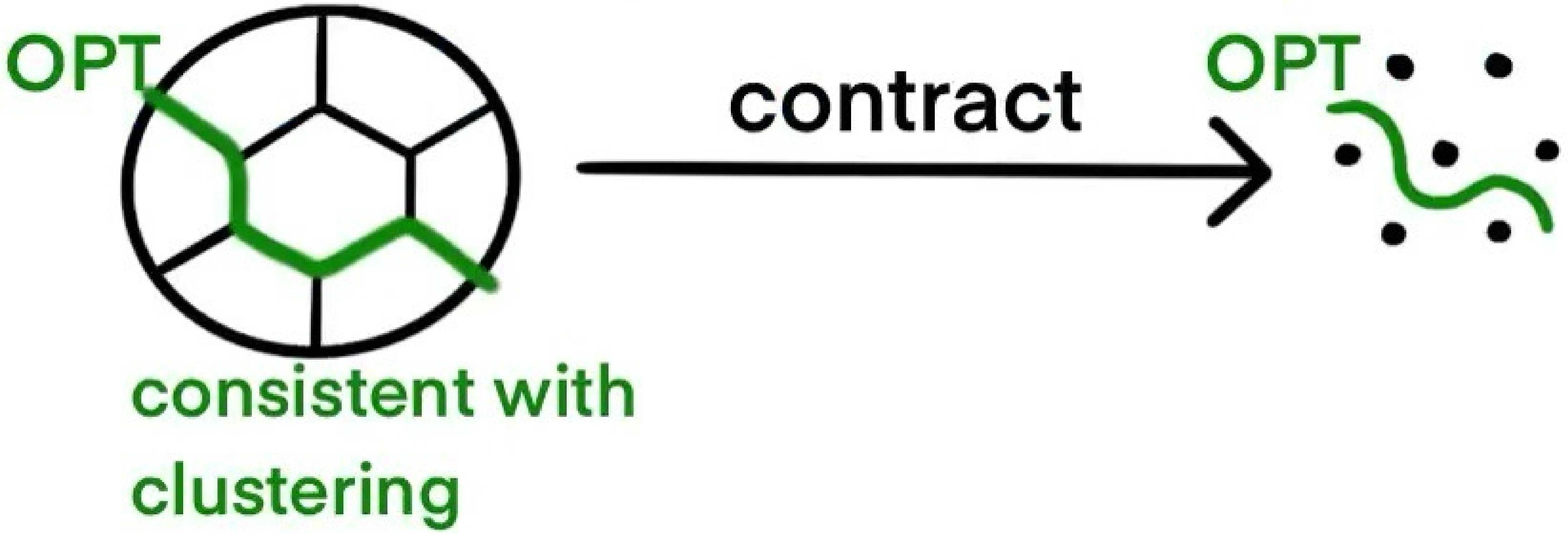
example:

Kawarabayashi-Thorup sparsification for global mincut on simple graphs [KT'15]:

Local Method

win/win approach:

- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut



example:

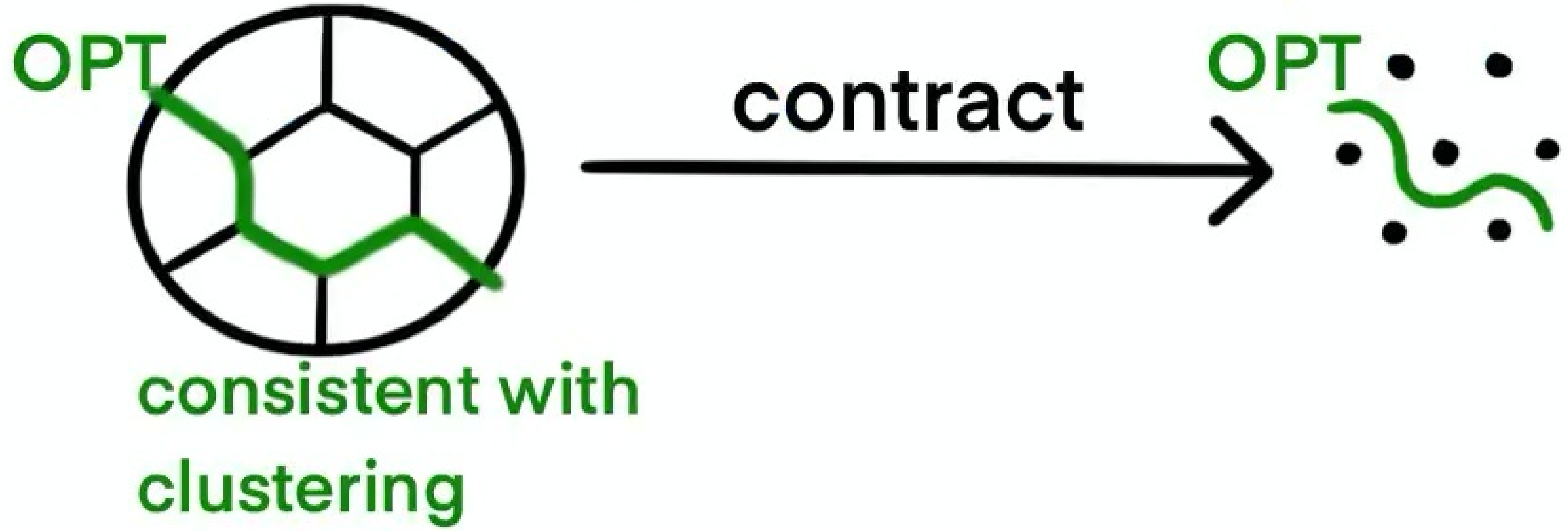
Kawarabayashi-Thorup sparsification for global mincut on simple graphs [KT'15]:

- if mincut is trivial (single vertex on one side): find min degree

Local Method

win/win approach:

- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut



example:

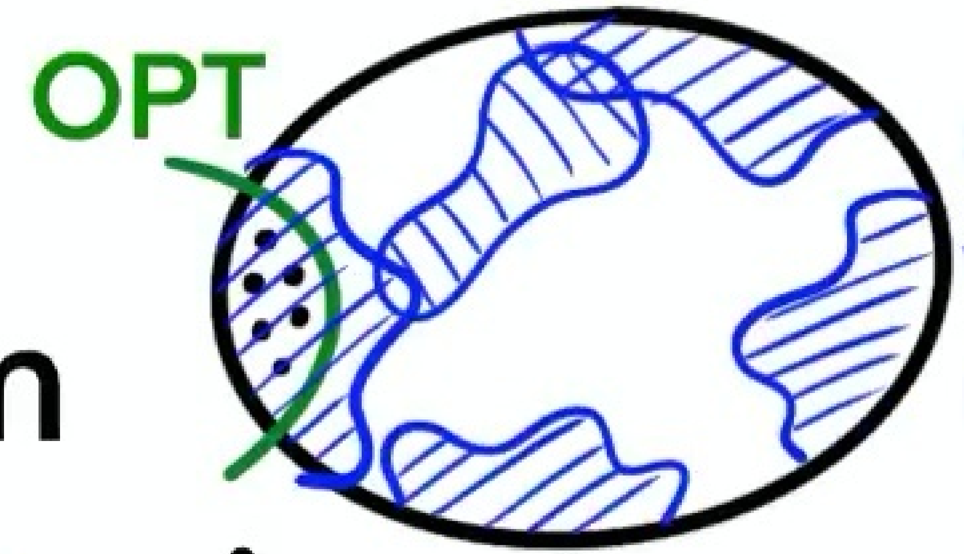
Kawarabayashi-Thorup sparsification for global mincut on simple graphs [KT'15]:

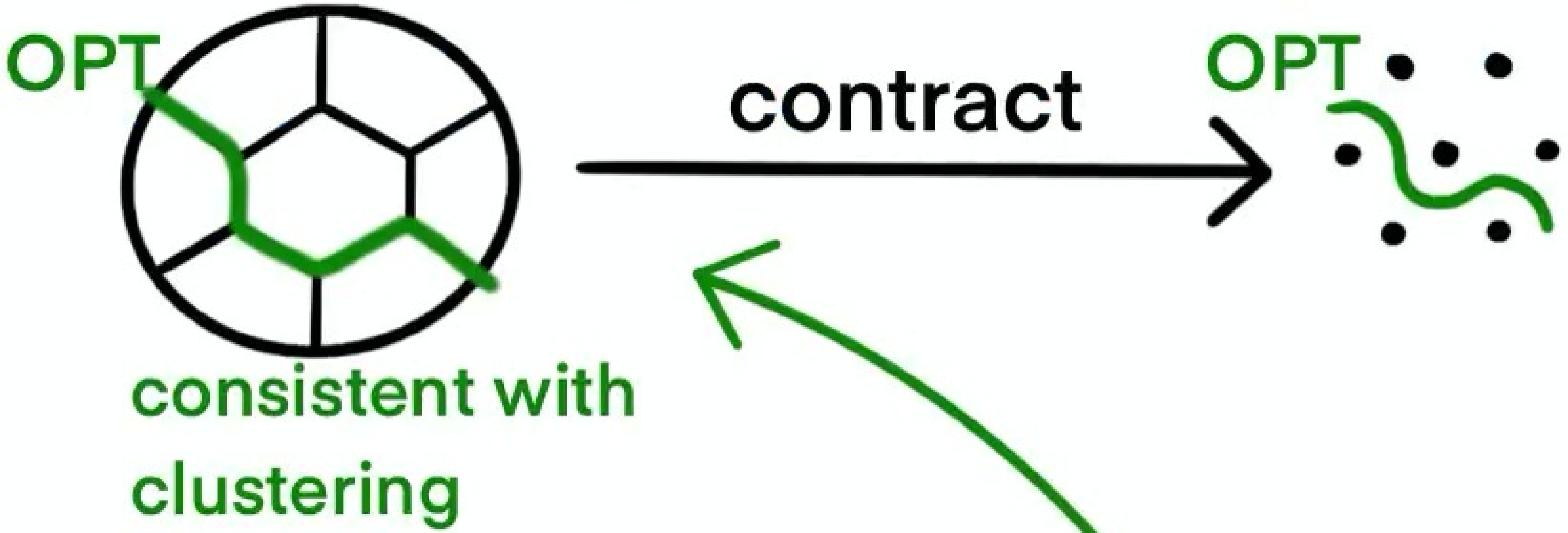
- if mincut is trivial (single vertex on one side): find min degree
- otherwise, cluster and contract to preserve global mincut

Local Method

win/win approach:

- if solution is local, then run local algorithm
- otherwise, reduce graph while preserving mincut





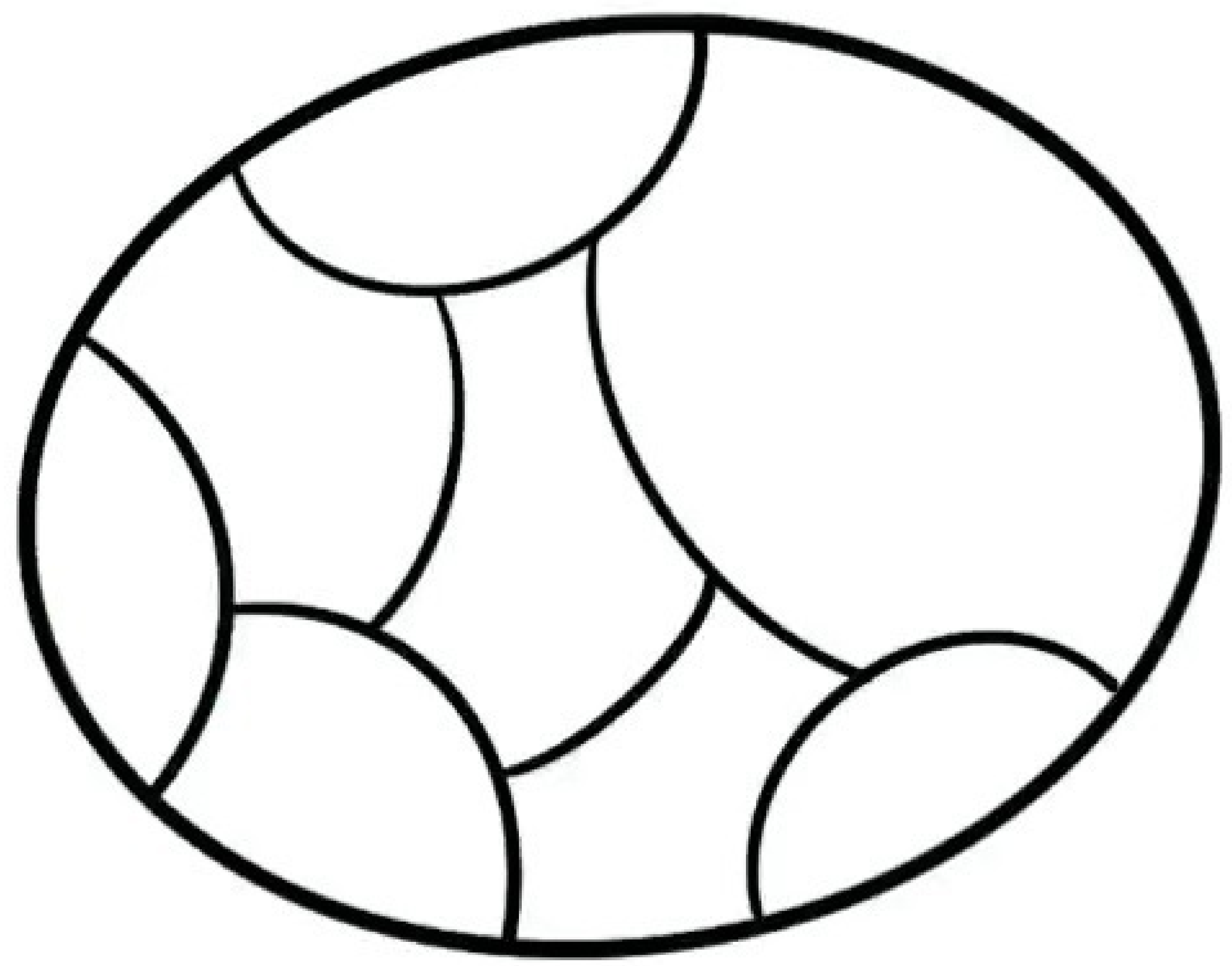
example:

Kawarabayashi-Thorup sparsification for global mincut on simple graphs [KT'15]:

- if mincut is trivial (single vertex on one side): find min degree
- otherwise, cluster and contract to preserve global mincut

Structural Theorem

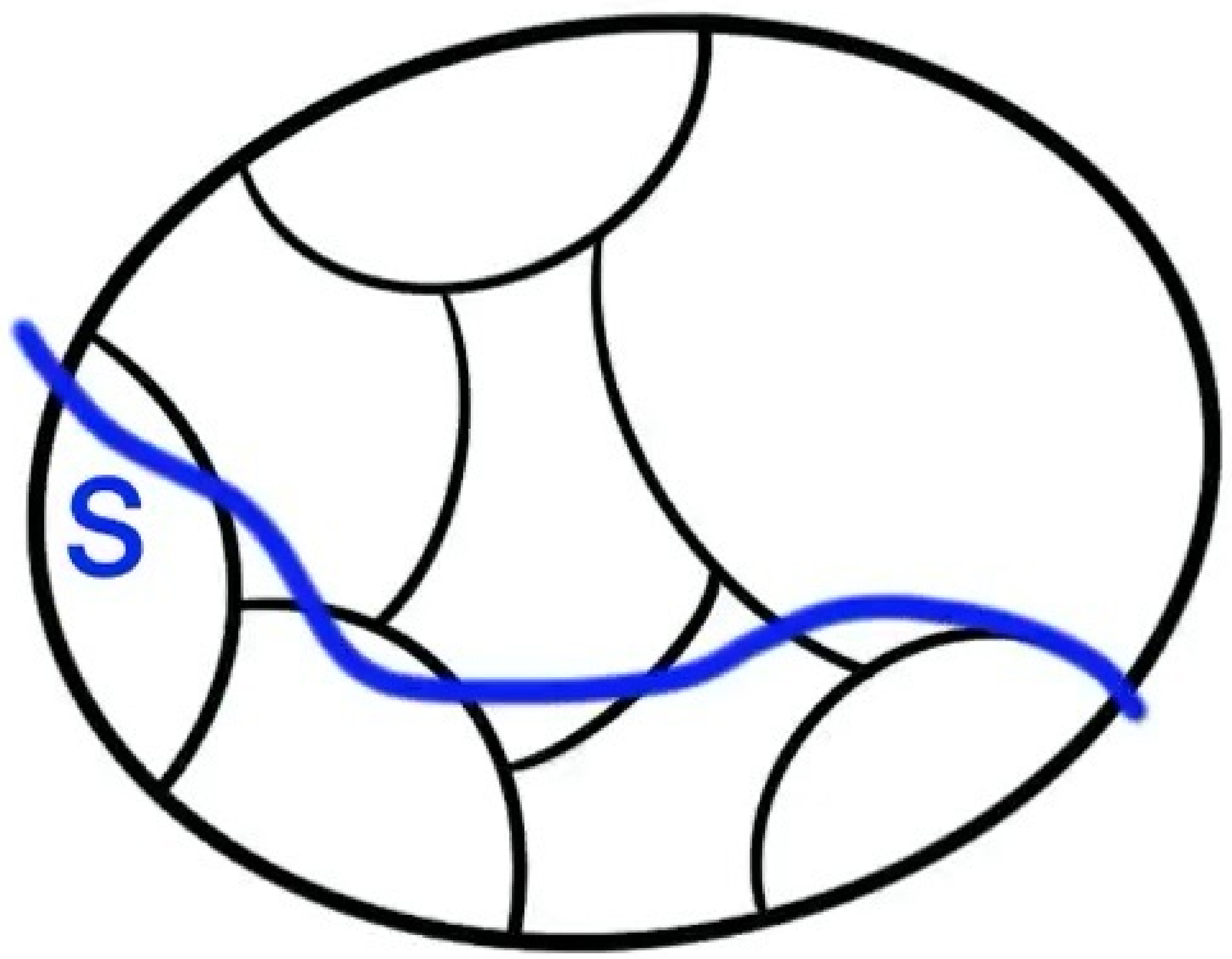
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

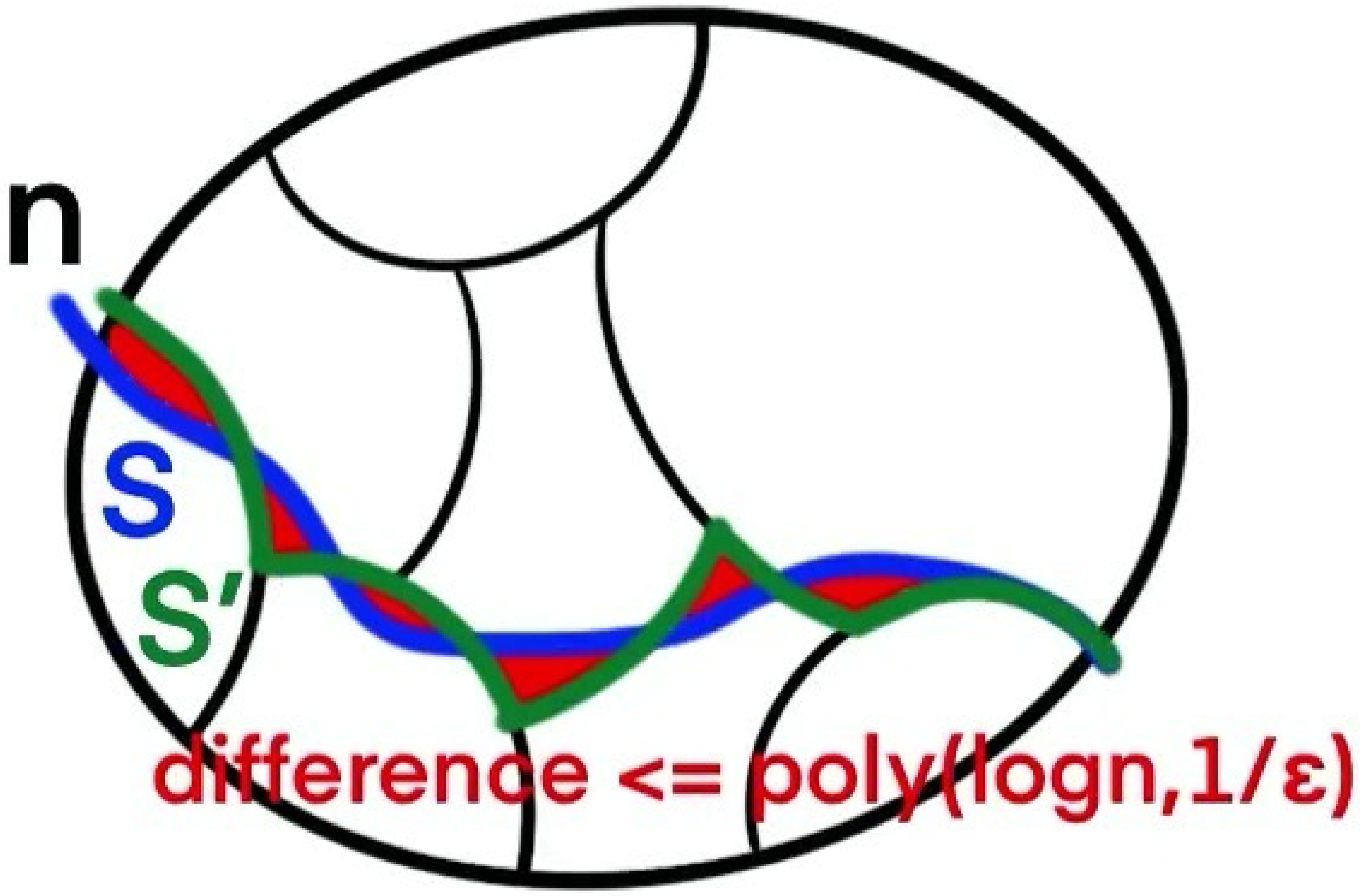
- for any 1.01-approximate mincut side S,



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

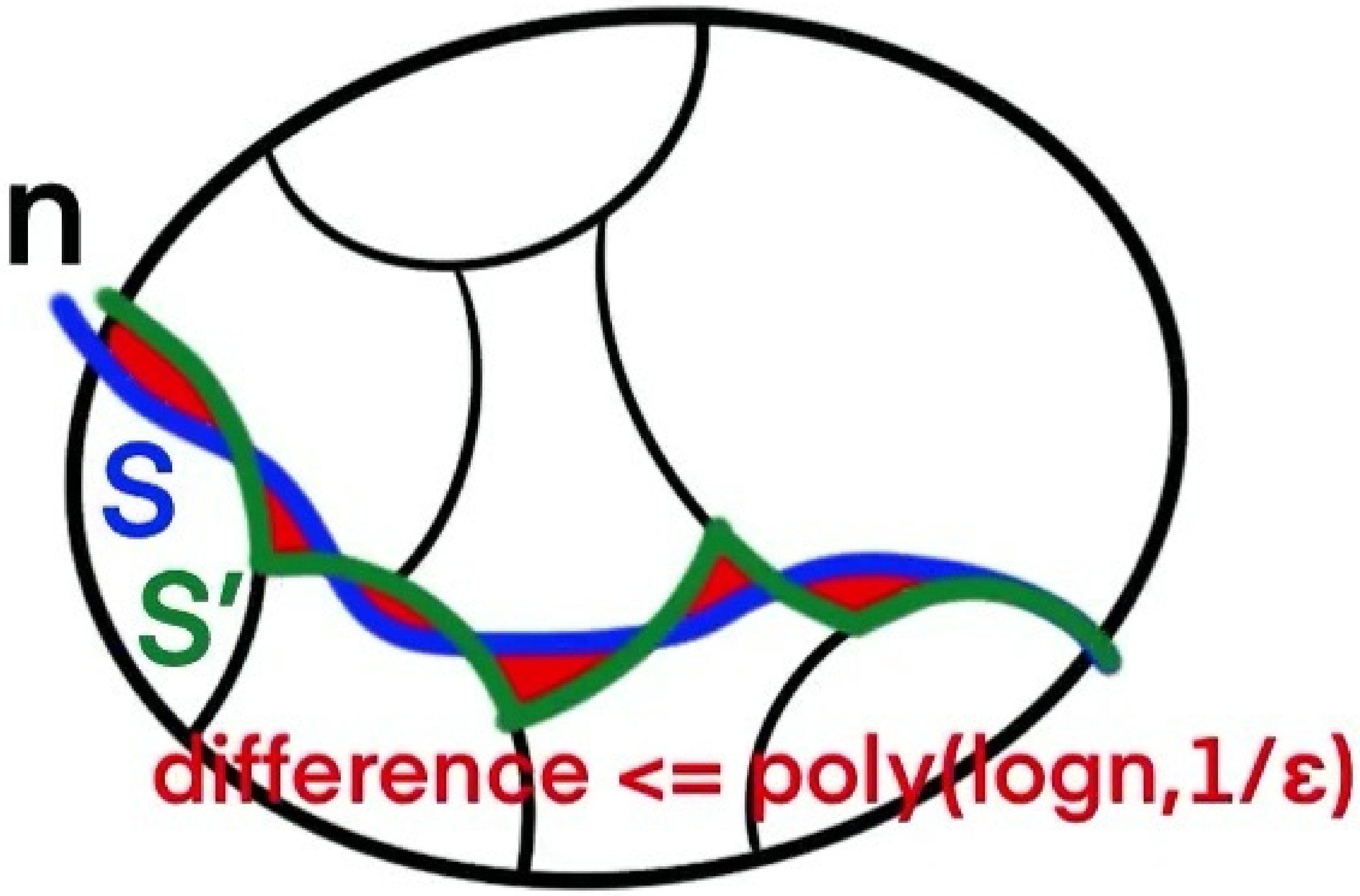
- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

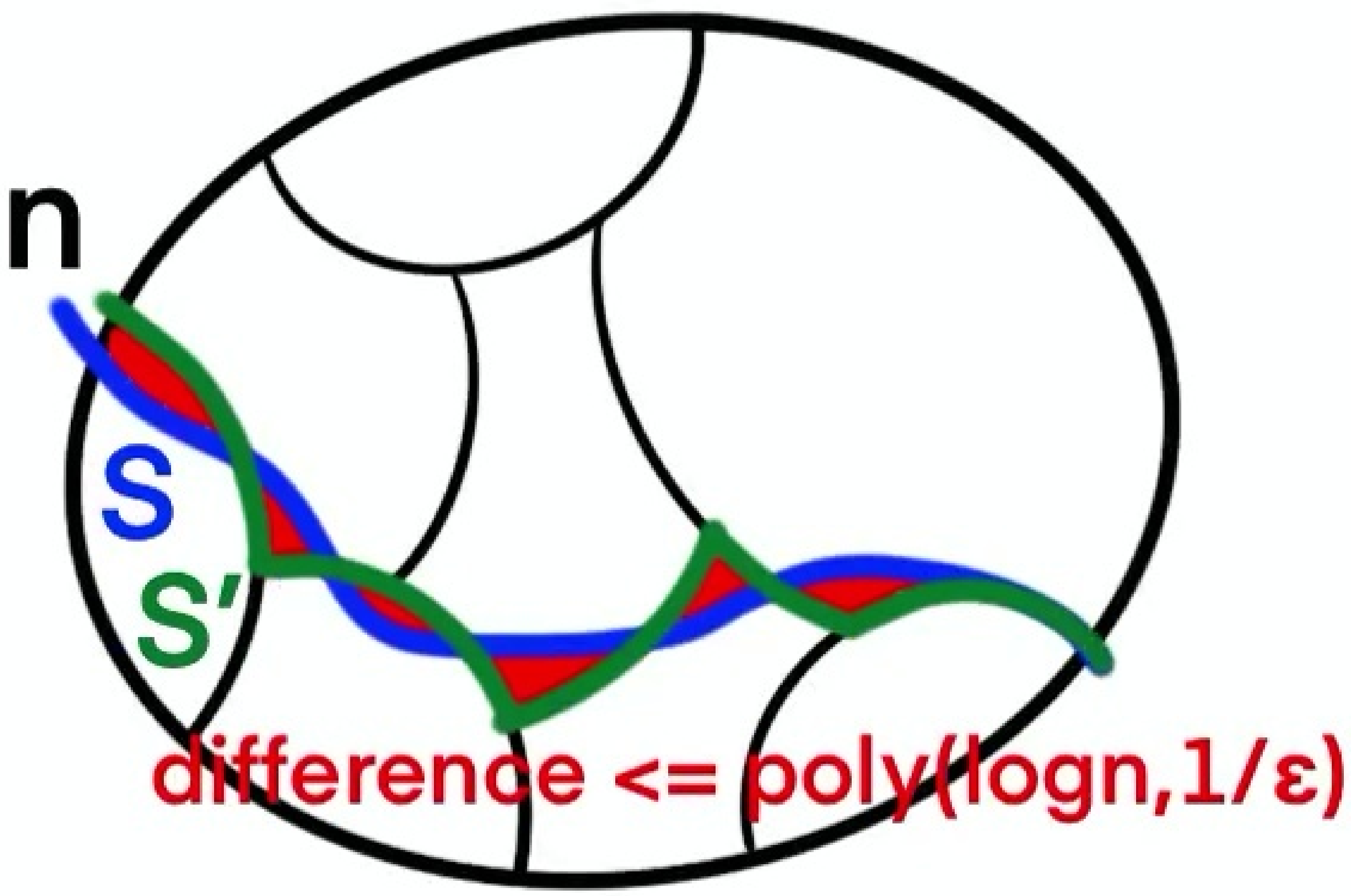
- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

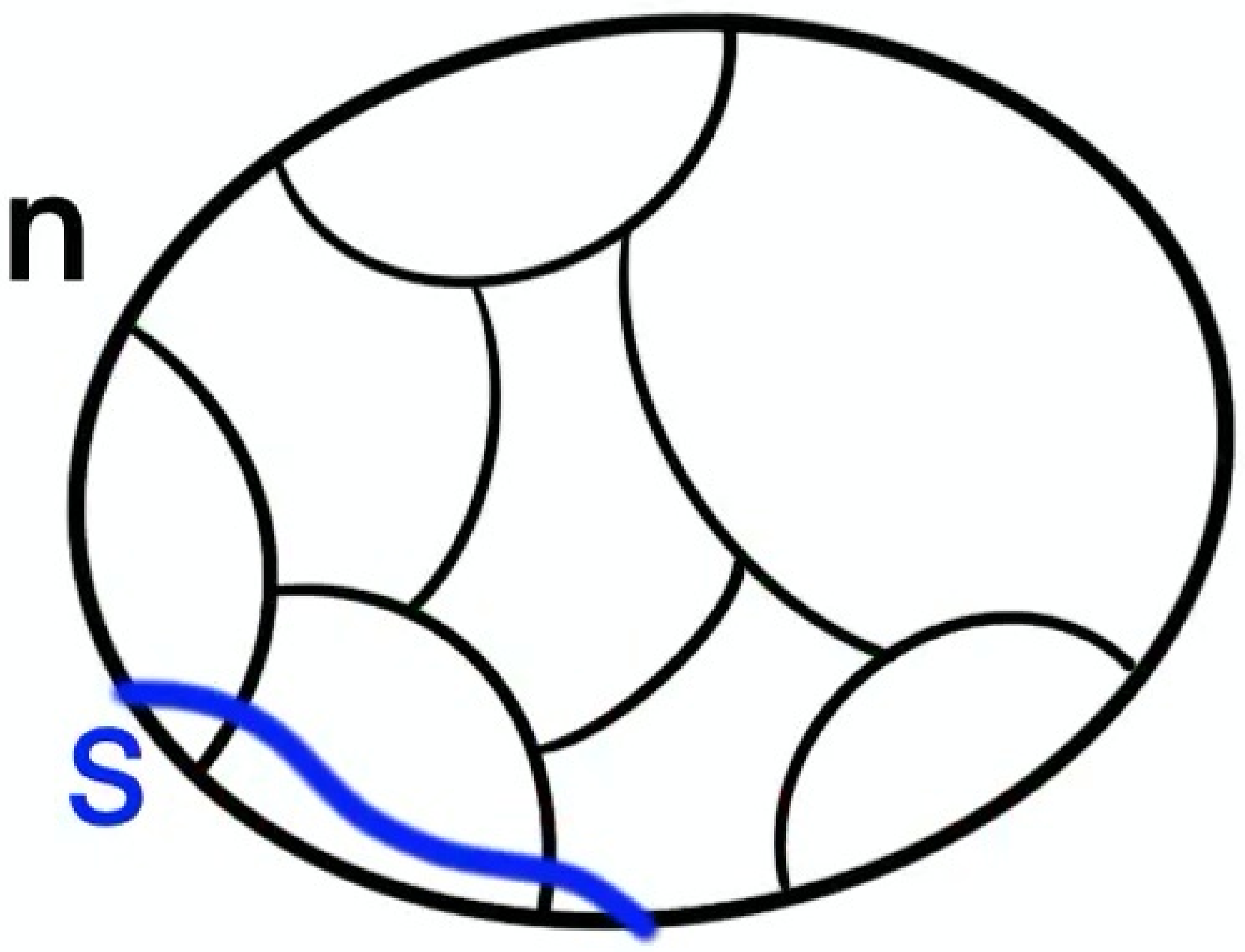
- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

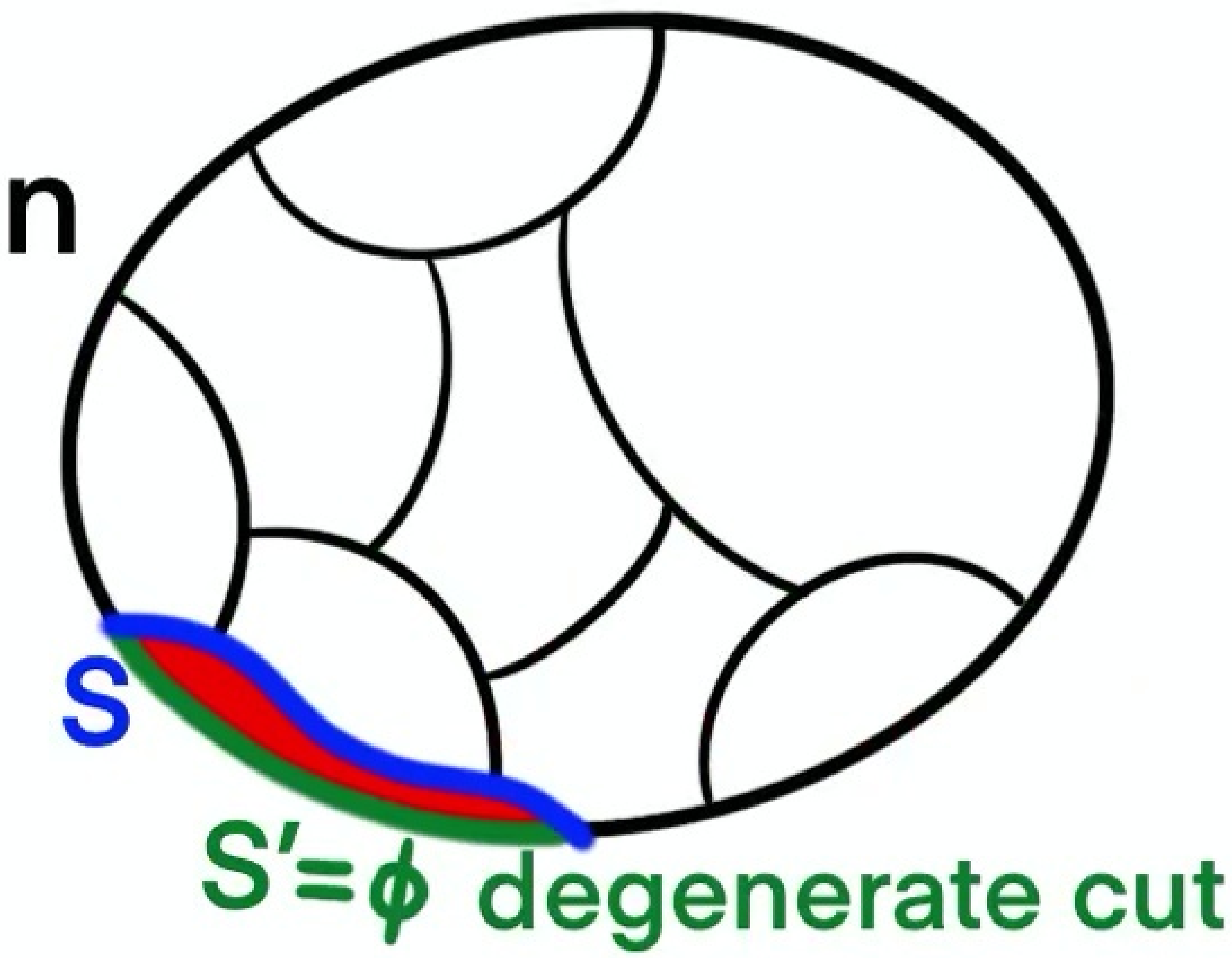
- for any 1.01-approximate mincut side S, can
  add/remove poly(logn,1/ε) vertices → S′
  s.t. (1) cost(S′) <= (1+ε) cost(S), and
  (2) S′ is consistent with clustering



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

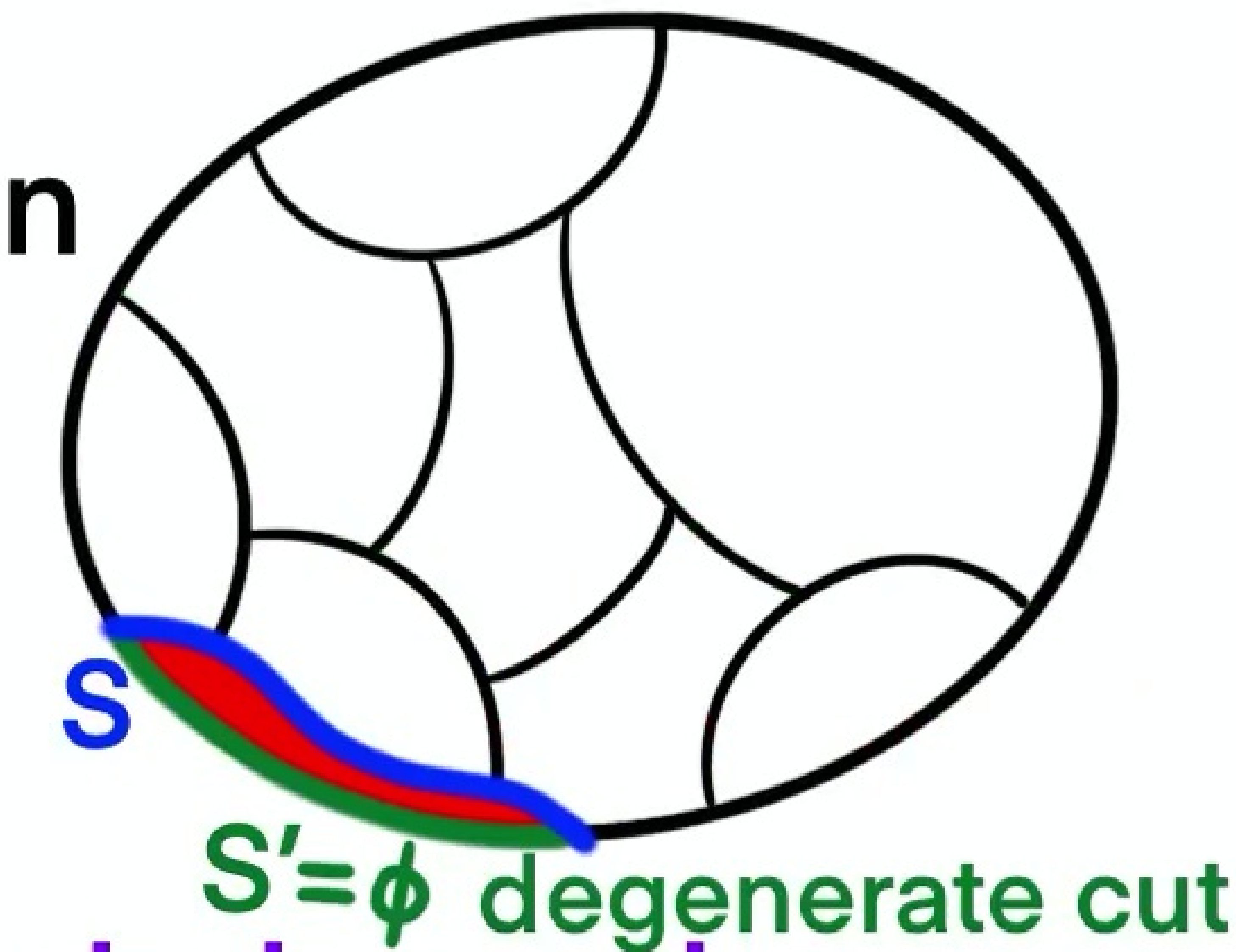
- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering



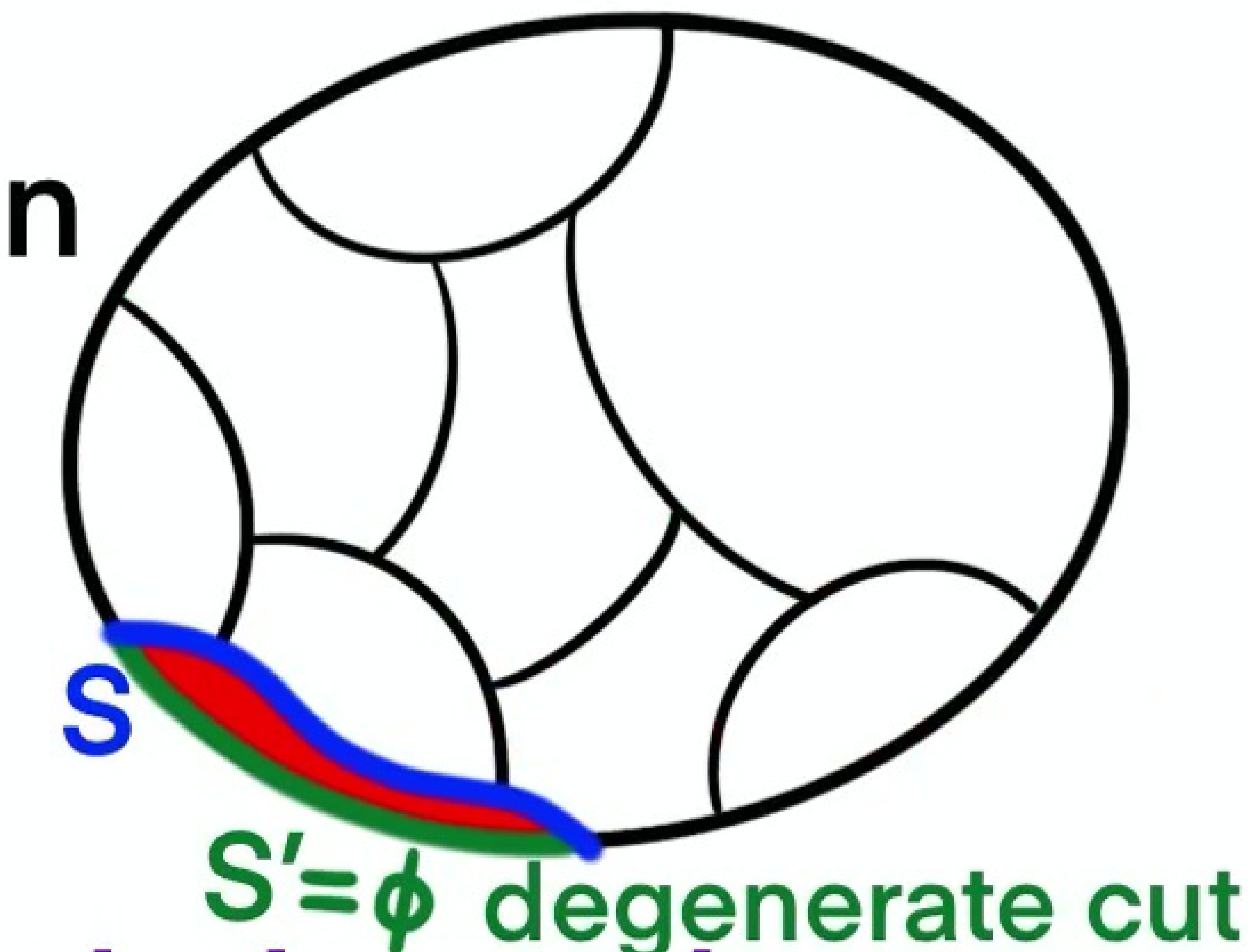
If S′ is degenerate, then S must be unbalanced

Structural Theorem

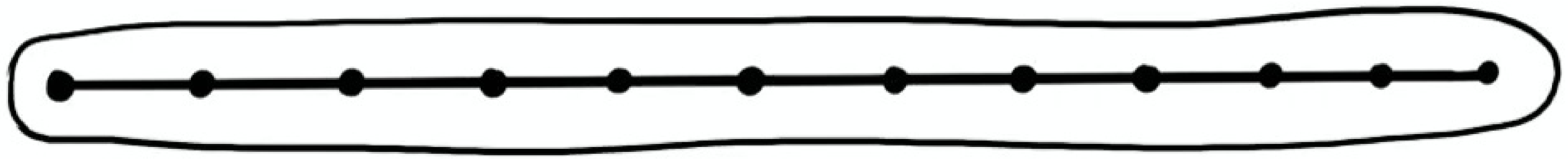
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering

If S′ is degenerate, then S must be unbalanced



Example 1: path graph

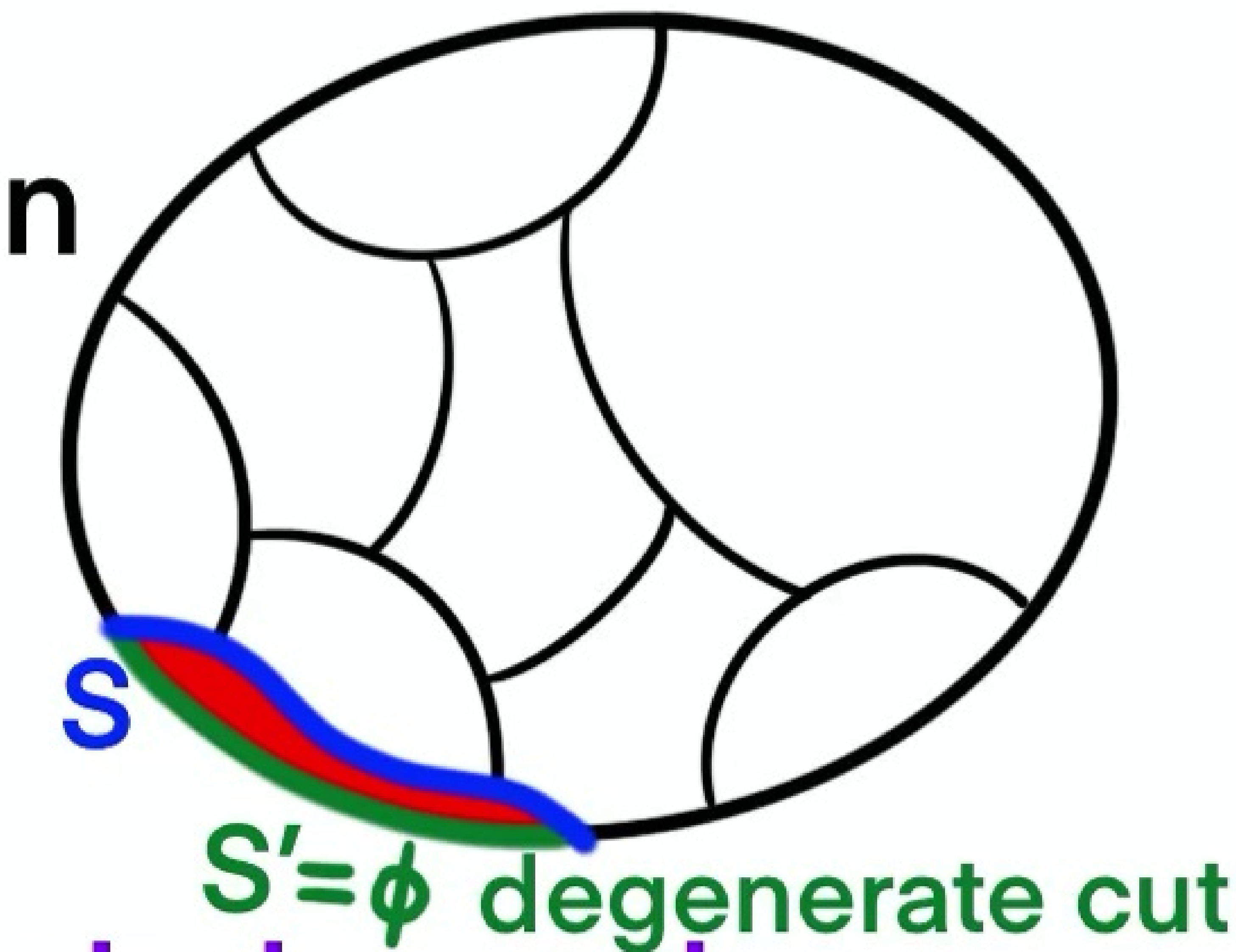


Structural Theorem

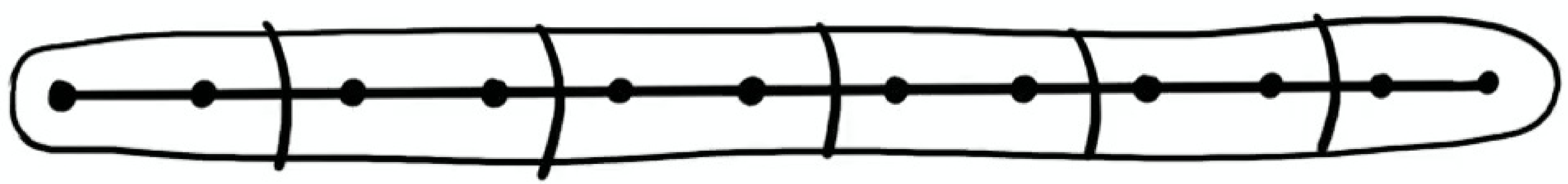
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering

If S′ is degenerate, then S must be unbalanced



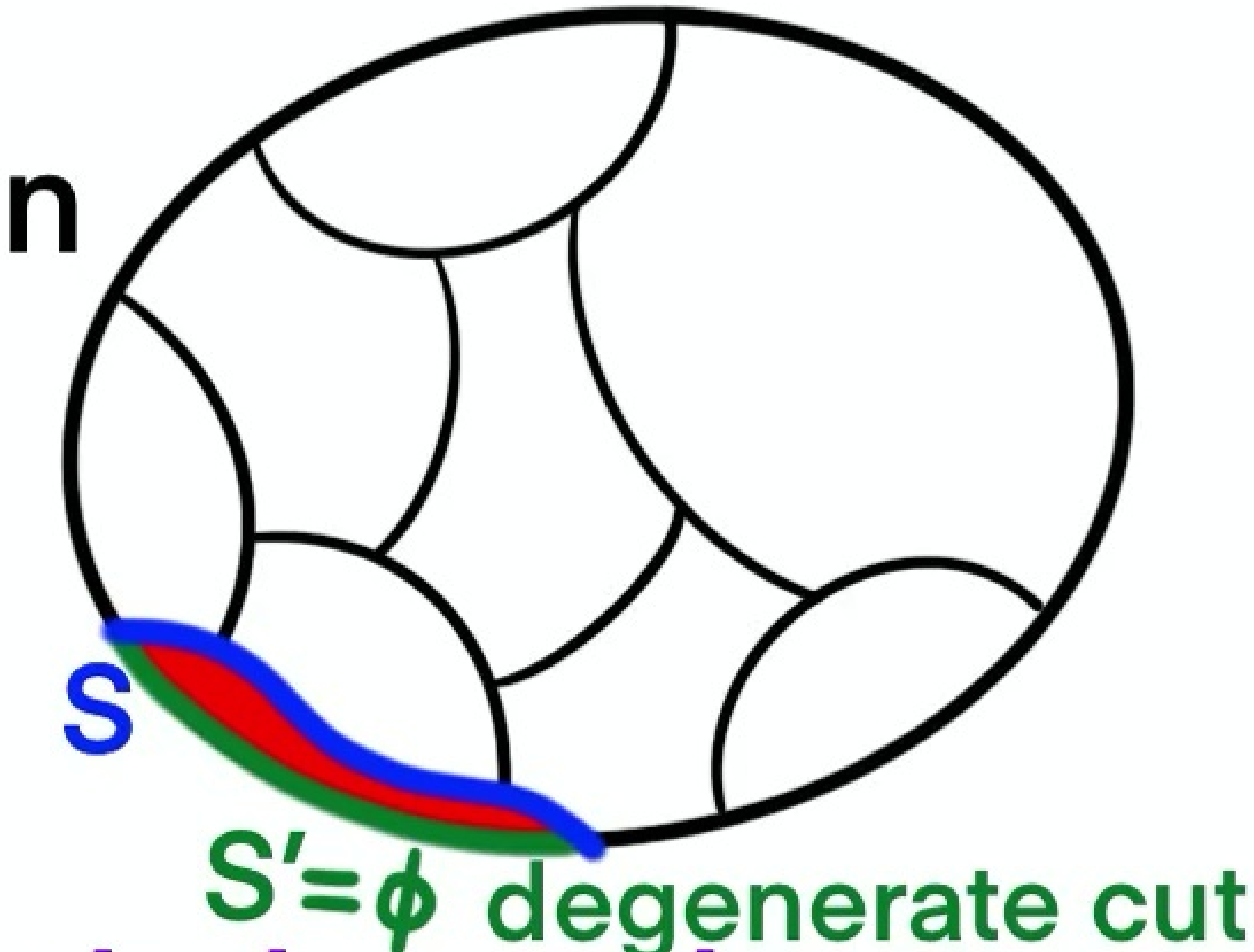
Example 1: path graph



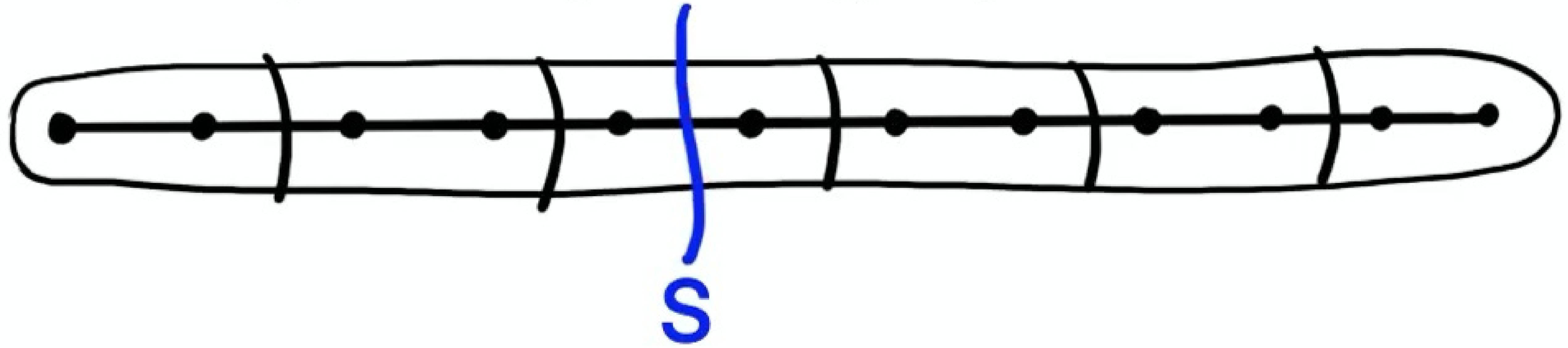
Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and
  (2) S′ is consistent with clustering
  If S′ is degenerate, then S must be unbalanced



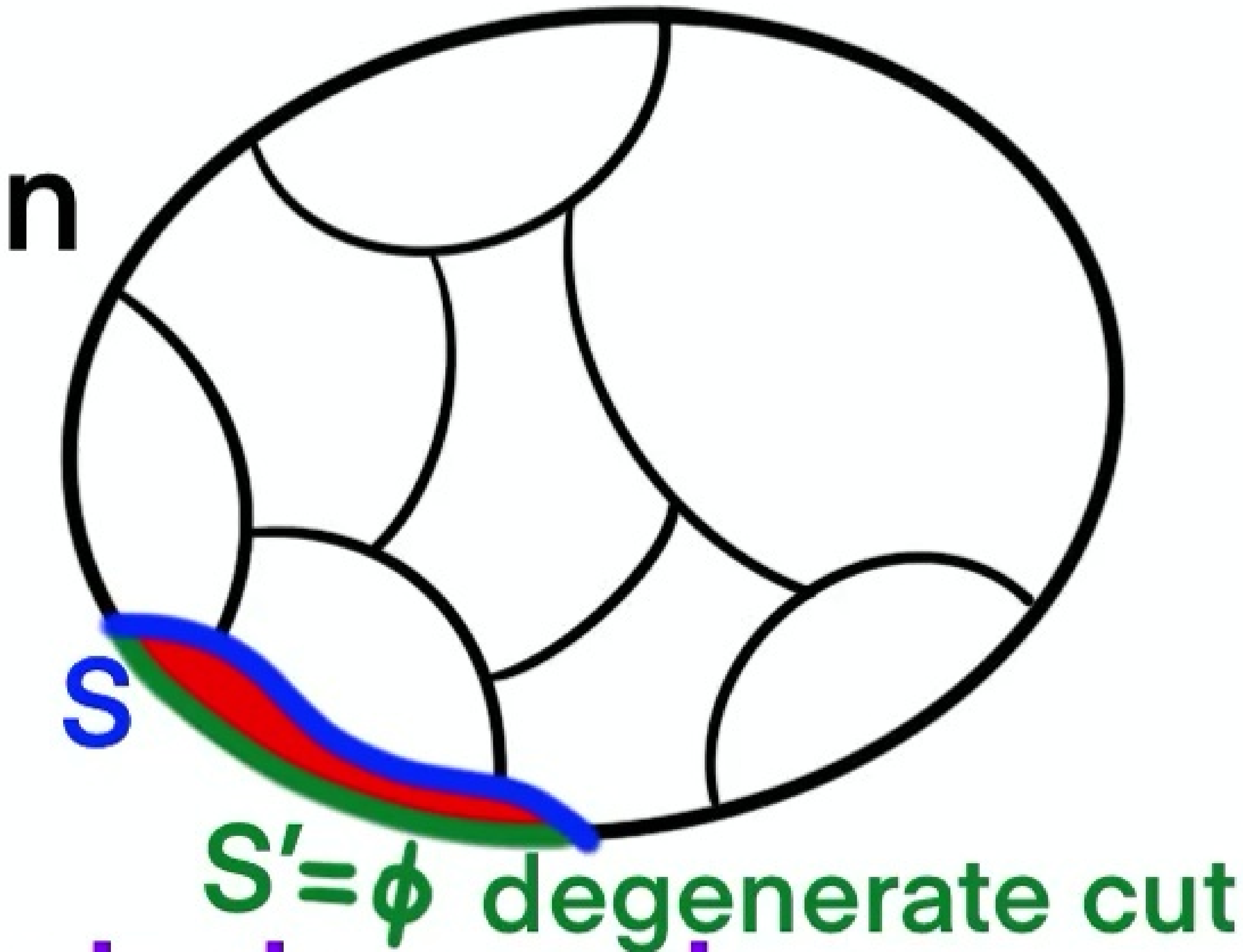
Example 1: path graph



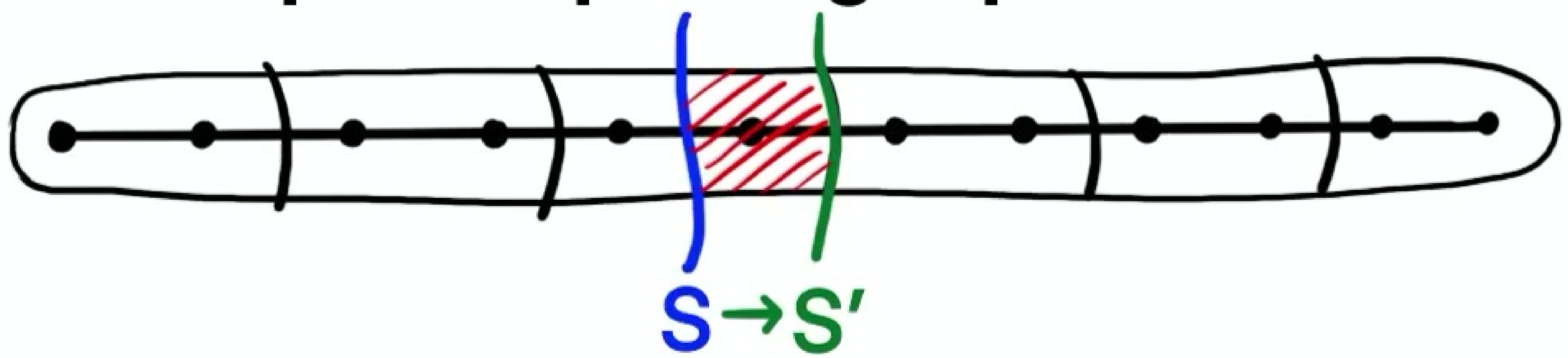
Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can
  add/remove poly(logn,1/ε) vertices → S′
  s.t. (1) cost(S′) <= (1+ε) cost(S), and
  (2) S′ is consistent with clustering
  If S′ is degenerate, then S must be unbalanced



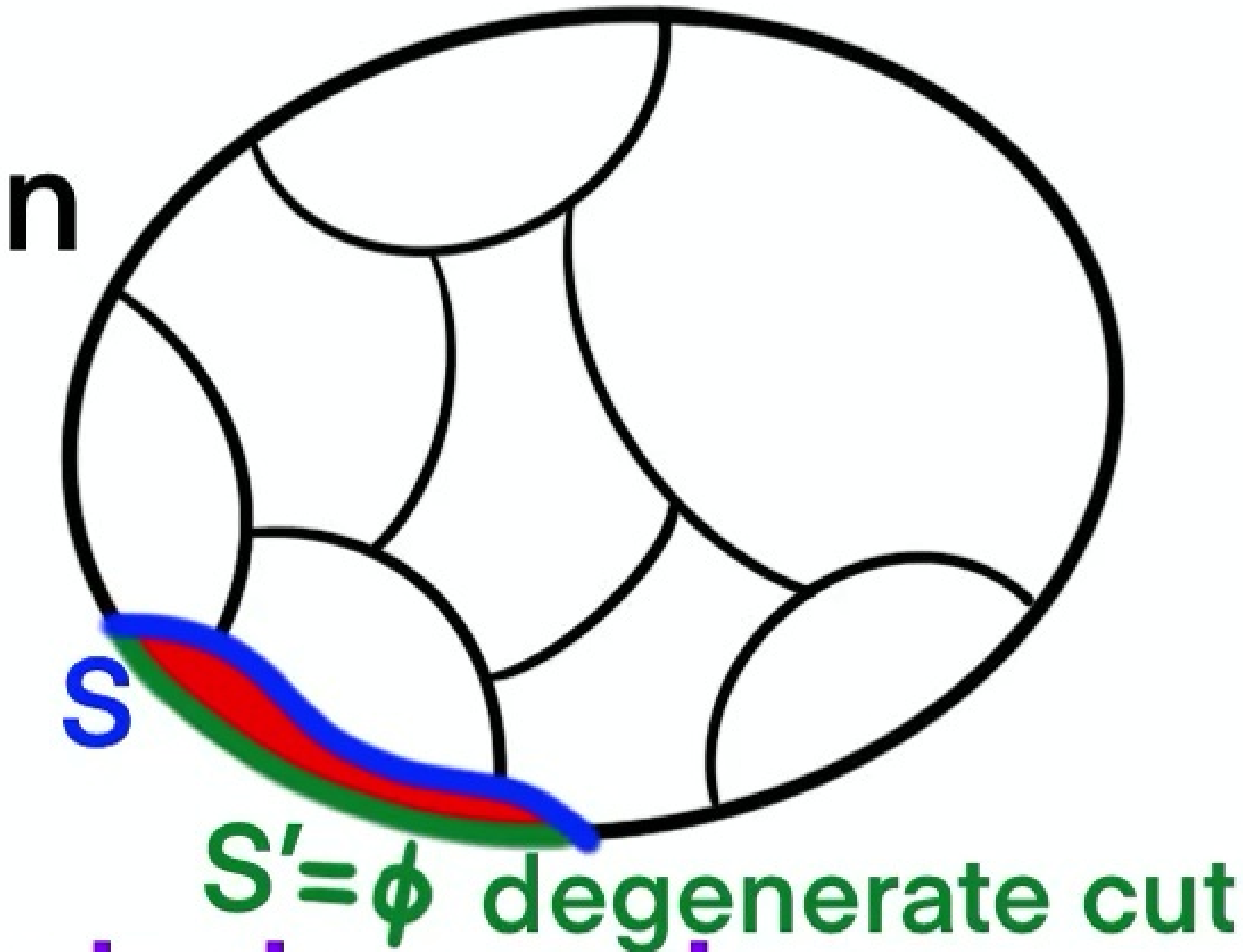
Example 1: path graph



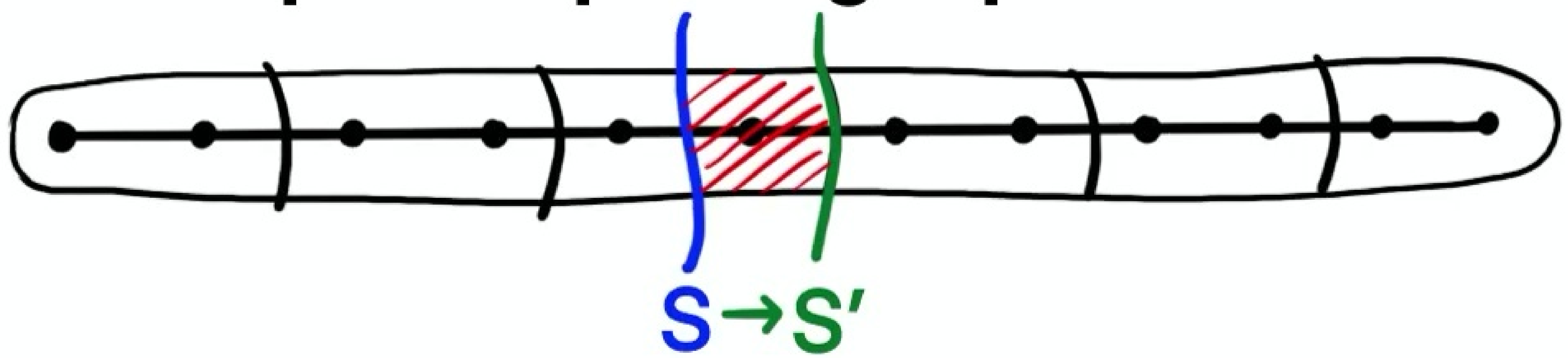
Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

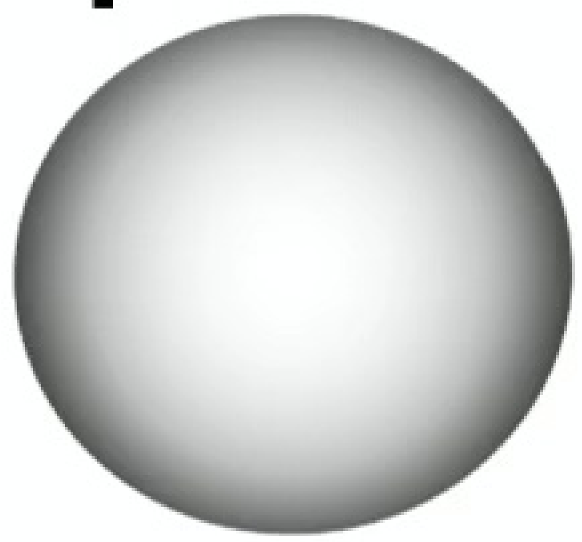
- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering If S′ is degenerate, then S must be unbalanced



Example 1: path graph



Example 2: clique

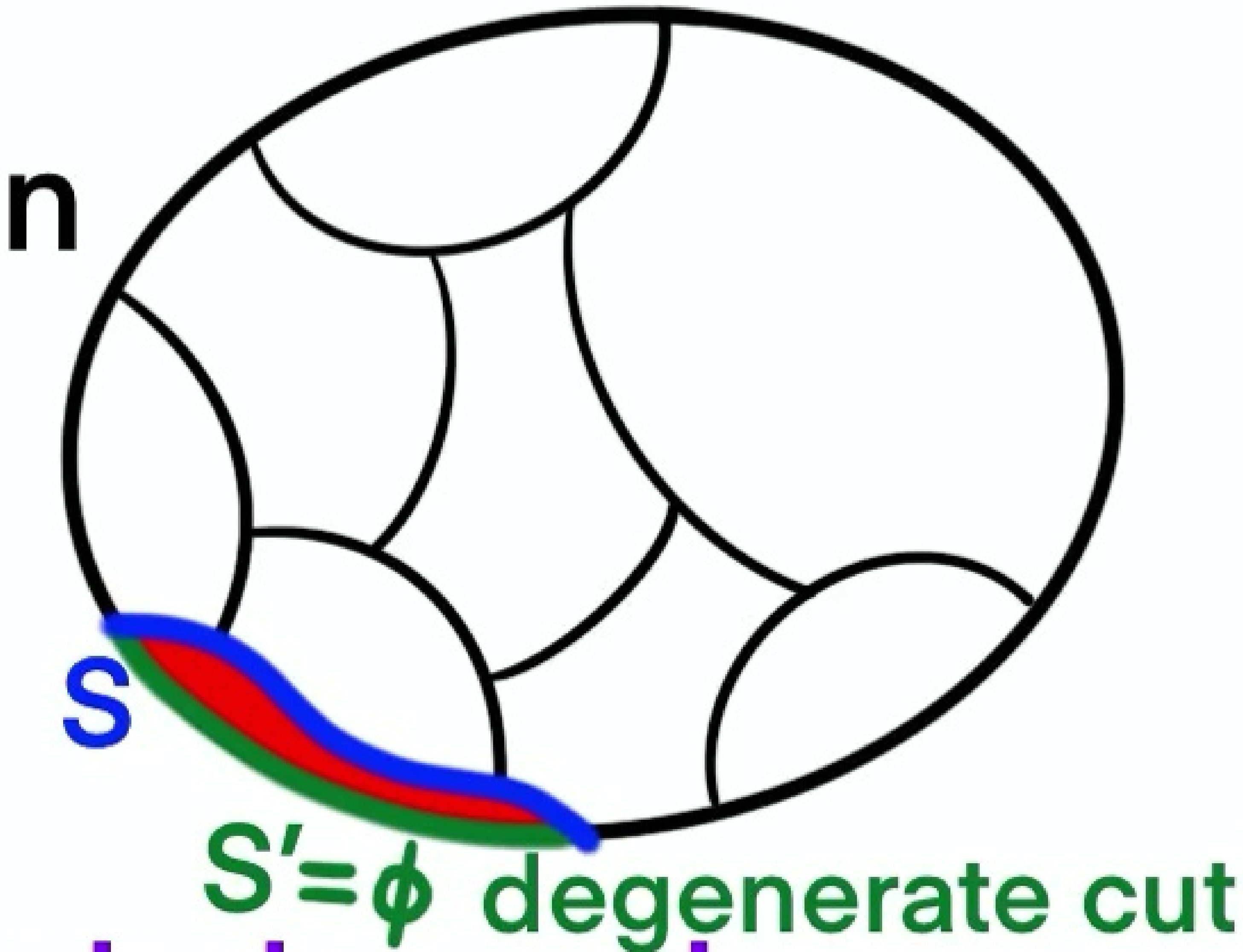


Structural Theorem

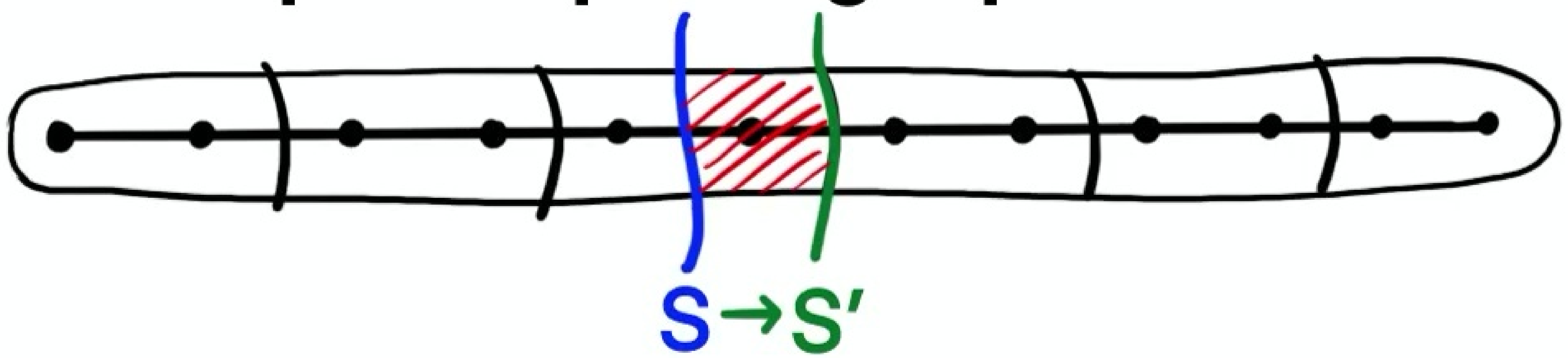
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering

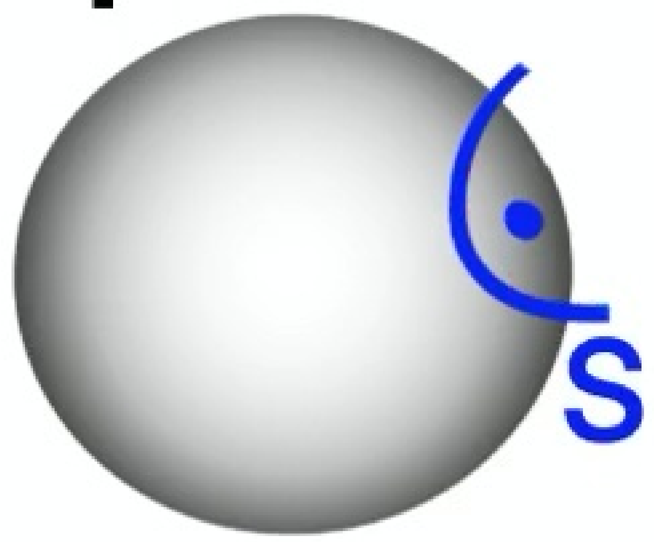
If S′ is degenerate, then S must be unbalanced



Example 1: path graph



Example 2: clique

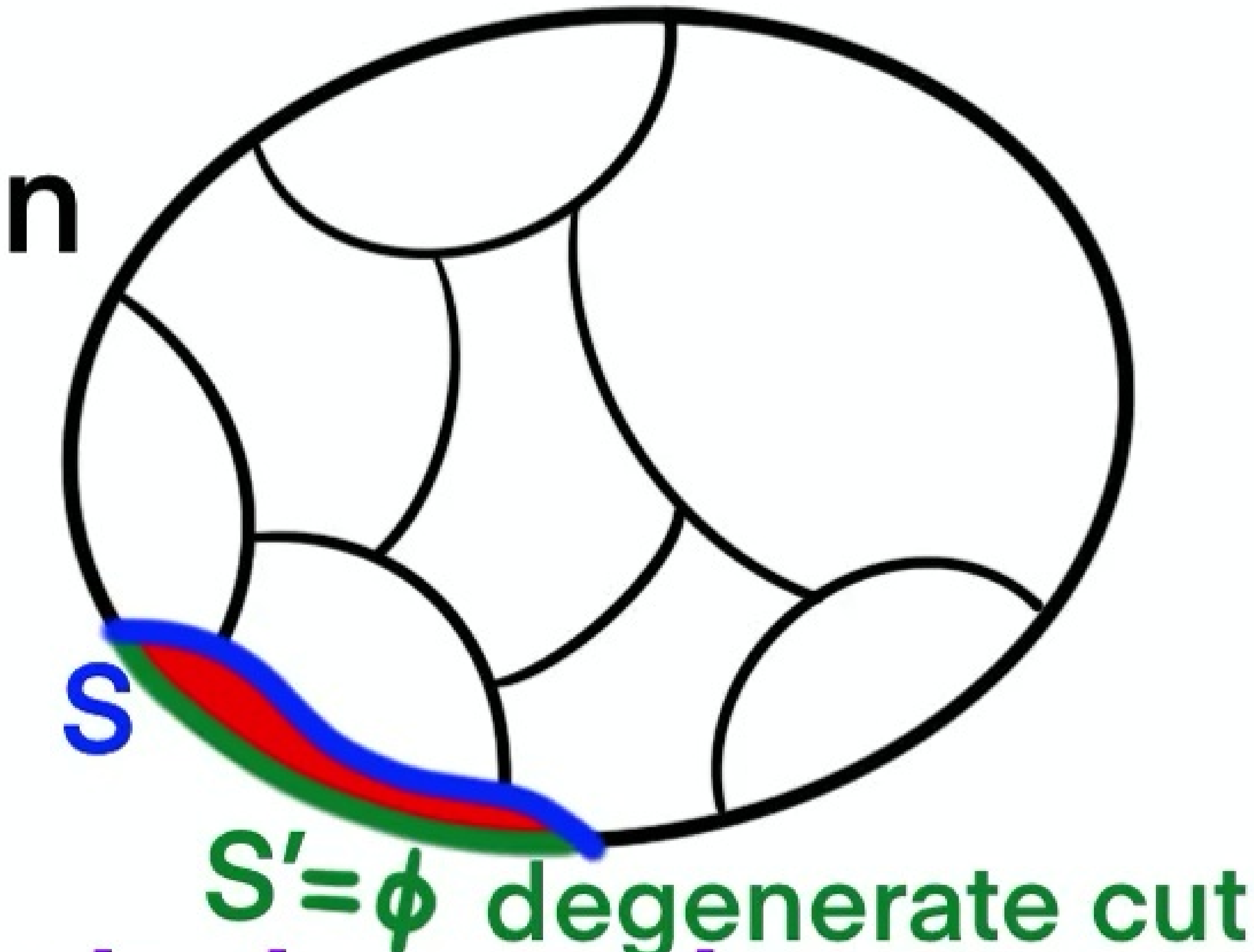


Structural Theorem

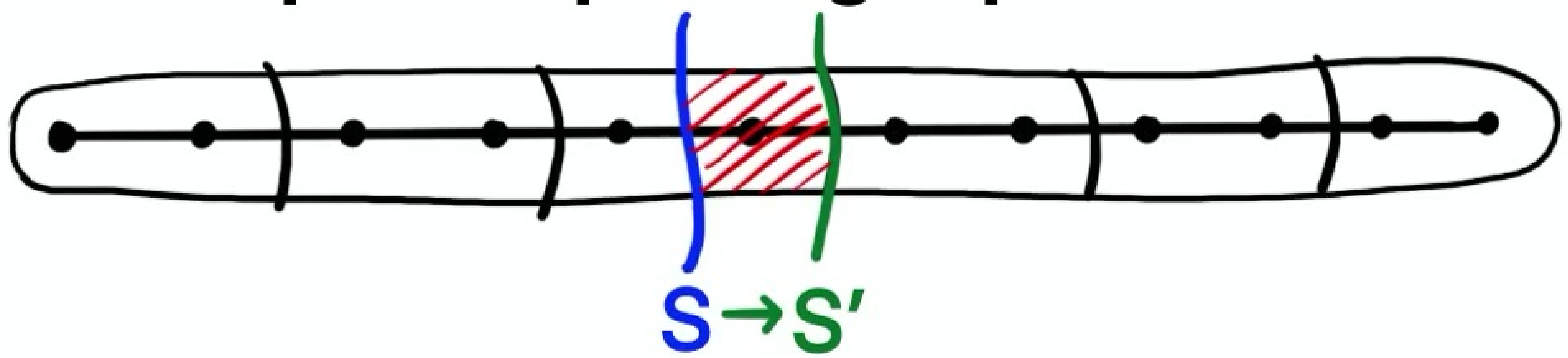
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and (2) S′ is consistent with clustering

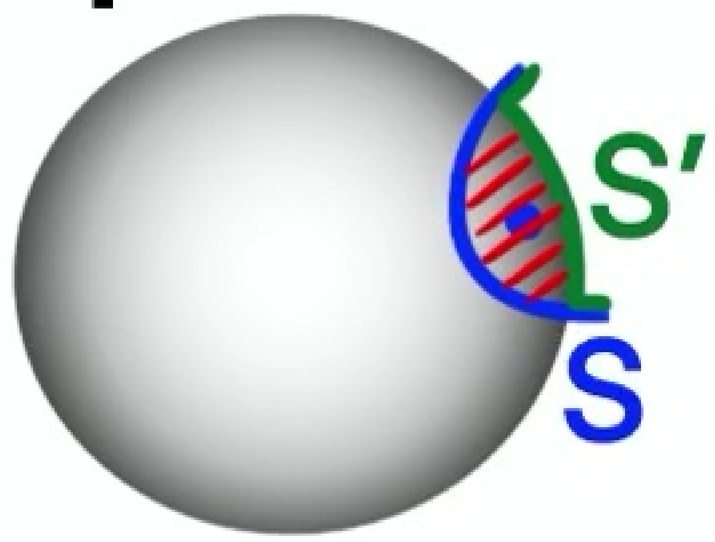
If S′ is degenerate, then S must be unbalanced



Example 1: path graph



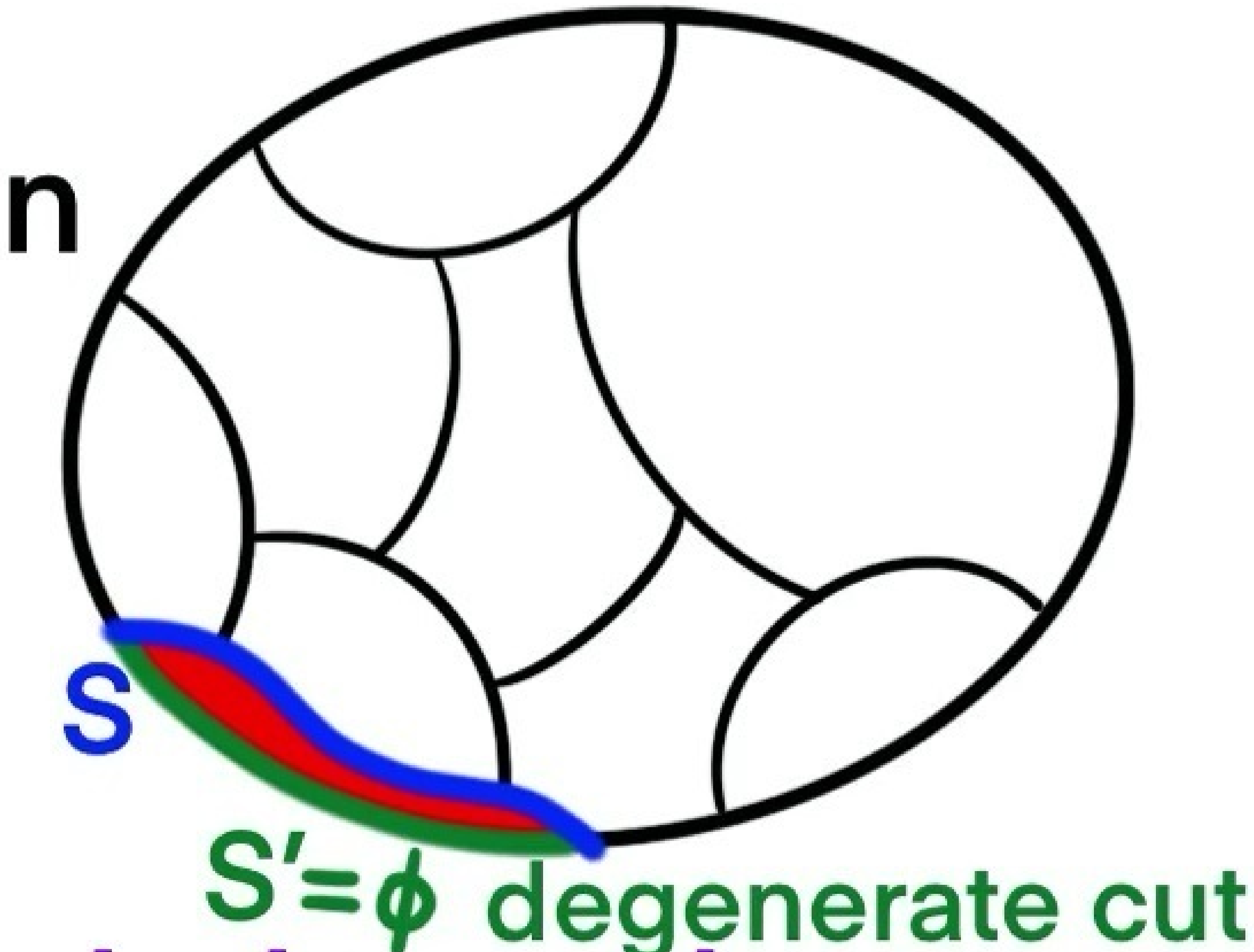
Example 2: clique



Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′ s.t. (1) cost(S′) <= (1+ε) cost(S), and
  (2) S′ is consistent with clustering
  If S′ is degenerate, then S must be unbalanced

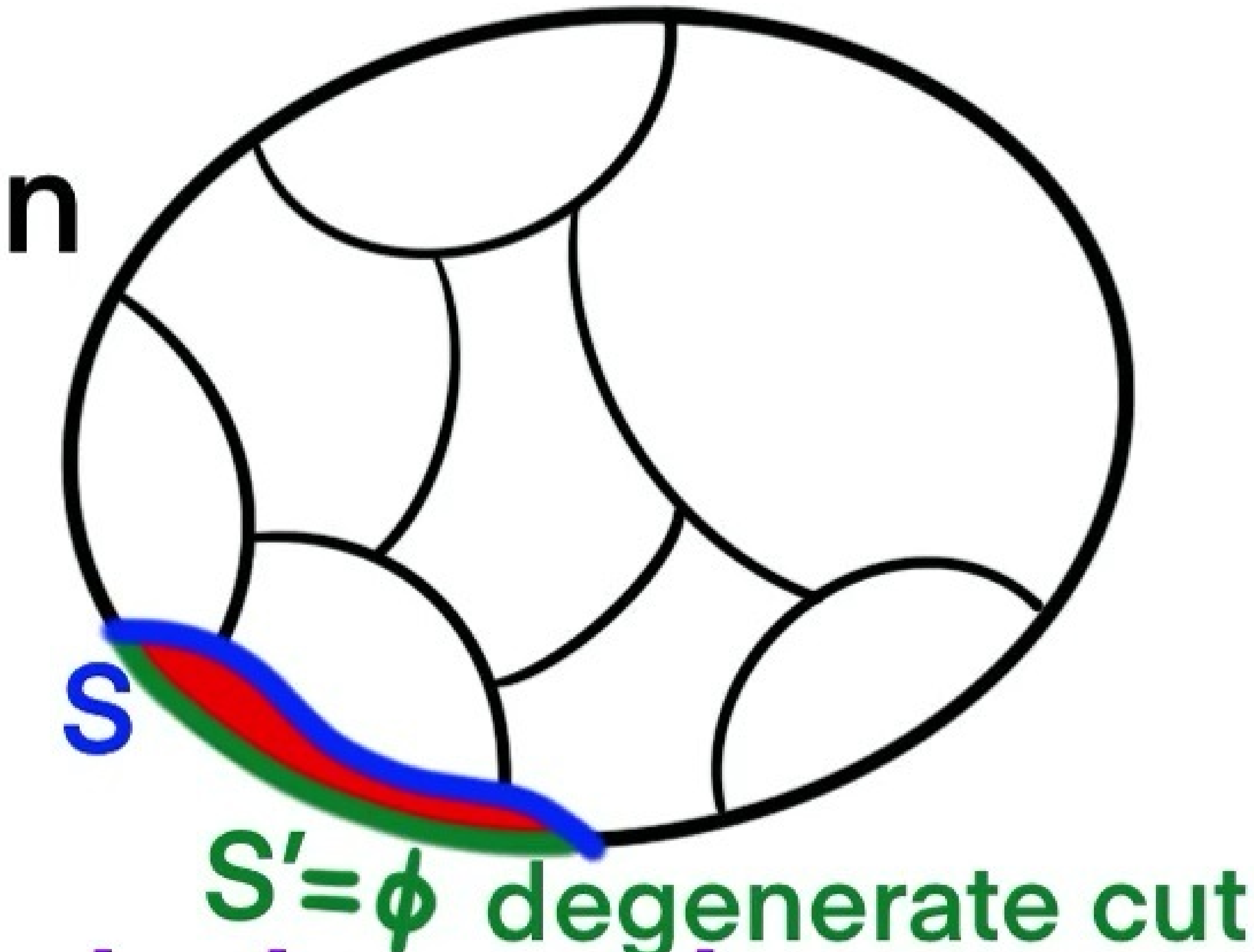


local algorithm [Kawarabayashi-Thorup]:

- if mincut is trivial (single vertex on one side): find min degree
- otherwise, cluster and contract to preserve global mincut

Structural Theorem

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.



- for any 1.01-approximate mincut side S, can add/remove poly(logn,1/ε) vertices → S′
  s.t. (1) cost(S′) <= (1+ε) cost(S), and
  (2) S′ is consistent with clustering
  If S′ is degenerate, then S must be unbalanced

Algorithm

unbalanced <= poly(logn,1/ε) vertices

run local algorithm

- if mincut is ~~trivial~~ unbalanced (~~single vertex~~ on one side): ~~find min degree~~ run local algorithm
- otherwise, cluster and contract to preserve global mincut up to (1+ε) factor

#vertices <= n/2

(1+ε)-approximate mincut

Algorithm: unbalanced <= poly(logn,1/ε) vertices run local algorithm

- if mincut is ~~trivial~~ ~~(single vertex~~ on one side): ~~find min degree~~
- otherwise, cluster and contract to preserve global mincut

#vertices <= n/2 up to (1+ε) factor

$(1+\varepsilon)$-approximate mincut

Algorithm: unbalanced <= poly(logn,1/ε) vertices — run local algorithm

- if mincut is ~~trivial~~ ~~(single vertex~~ on one side): ~~find min degree~~
- otherwise, cluster and contract to preserve global mincut

#vertices <= n/2 — up to $(1+\varepsilon)$ factor

Repeat $\log_2 n$ times: run local algorithm, then cluster+contract

$\rightarrow (1+\varepsilon)^{\log n}$ approximation. Set $\varepsilon \ll 1/\log n \rightarrow (1+o(1))$-approx

$(1+\varepsilon)$-approximate mincut

Algorithm: unbalanced <= poly(logn,1/ε) vertices — run local algorithm

- if mincut is ~~trivial~~ ~~(single vertex~~ on one side): ~~find min degree~~
- otherwise, cluster and contract to preserve global mincut

#vertices <= n/2 — up to $(1+\varepsilon)$ factor

Repeat $\log_2 n$ times: run local algorithm, then cluster+contract

$\rightarrow (1+\varepsilon)^{\log n}$ approximation. Set $\varepsilon \ll 1/\log n \rightarrow (1+o(1))$-approx

"Local Karger contraction" [Nalam-Saranurak'23]:

$mk^2\mathrm{polylog}(n)$ randomized time if k vertices on one side

$k=\mathrm{poly}(\log n,1/\varepsilon)=\mathrm{polylog}(n)$

$(1+\varepsilon)$-approximate mincut

Algorithm: unbalanced <= poly(logn,1/ε) vertices run local algorithm

- if mincut is ~~trivial~~ ~~(single vertex~~ on one side): ~~find min degree~~
- otherwise, cluster and contract to preserve global mincut

#vertices <= n/2 up to (1+ε) factor

Repeat $\log_2 n$ times: run local algorithm, then cluster+contract

→ $(1+\varepsilon)^{\log n}$ approximation. Set $\varepsilon \ll 1/\log n$ → $(1+o(1))$-approx

"Local Karger contraction" [Nalam-Saranurak'23]:

$mk^2\mathrm{polylog}(n)$ randomized time if k vertices on one side

$k=\mathrm{poly}(\log n,1/\varepsilon)=\mathrm{polylog}(n)$

Overall: $m\,\mathrm{polylog}(n)$ randomized for $(1+o(1))$-approx mincut

Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition

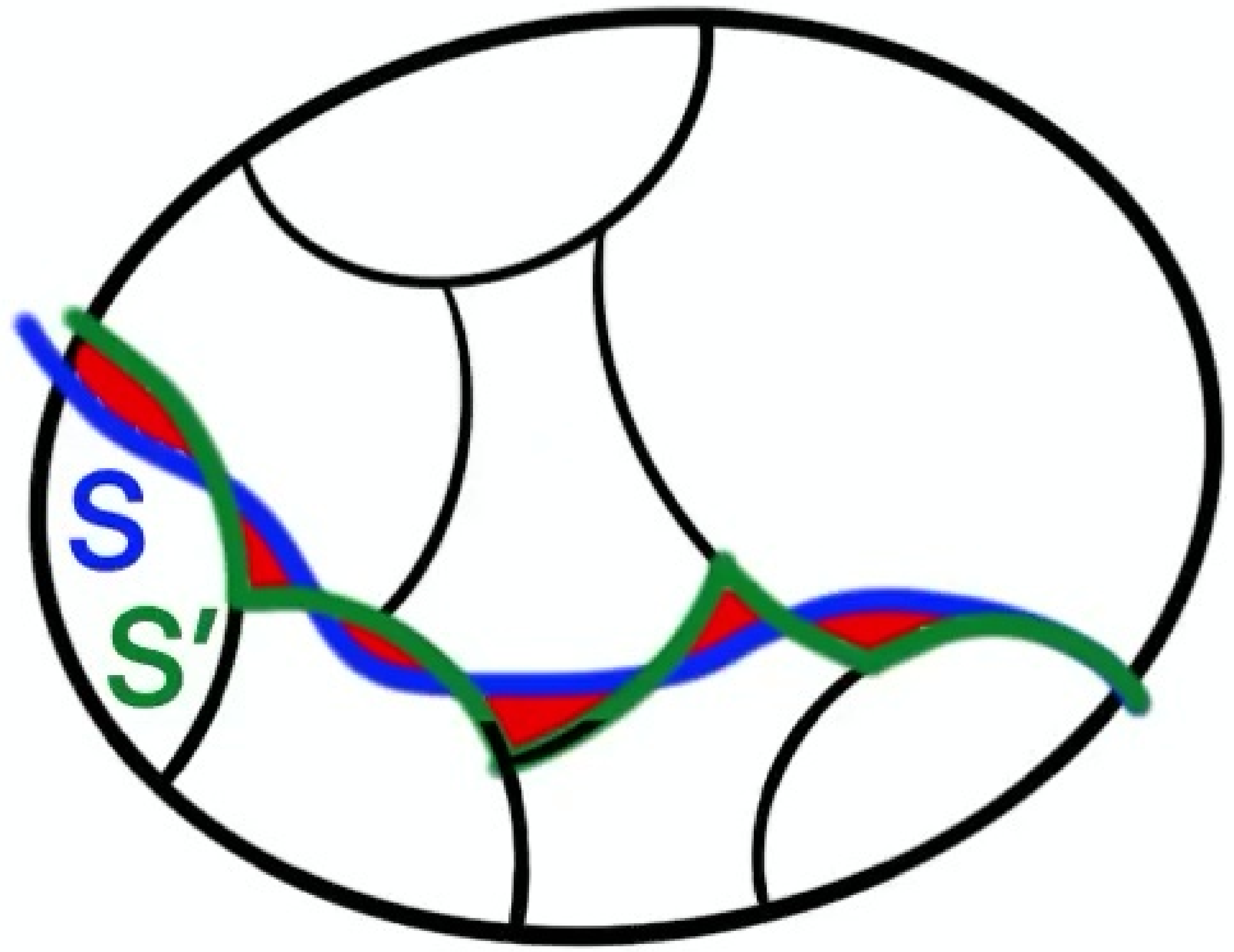
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition



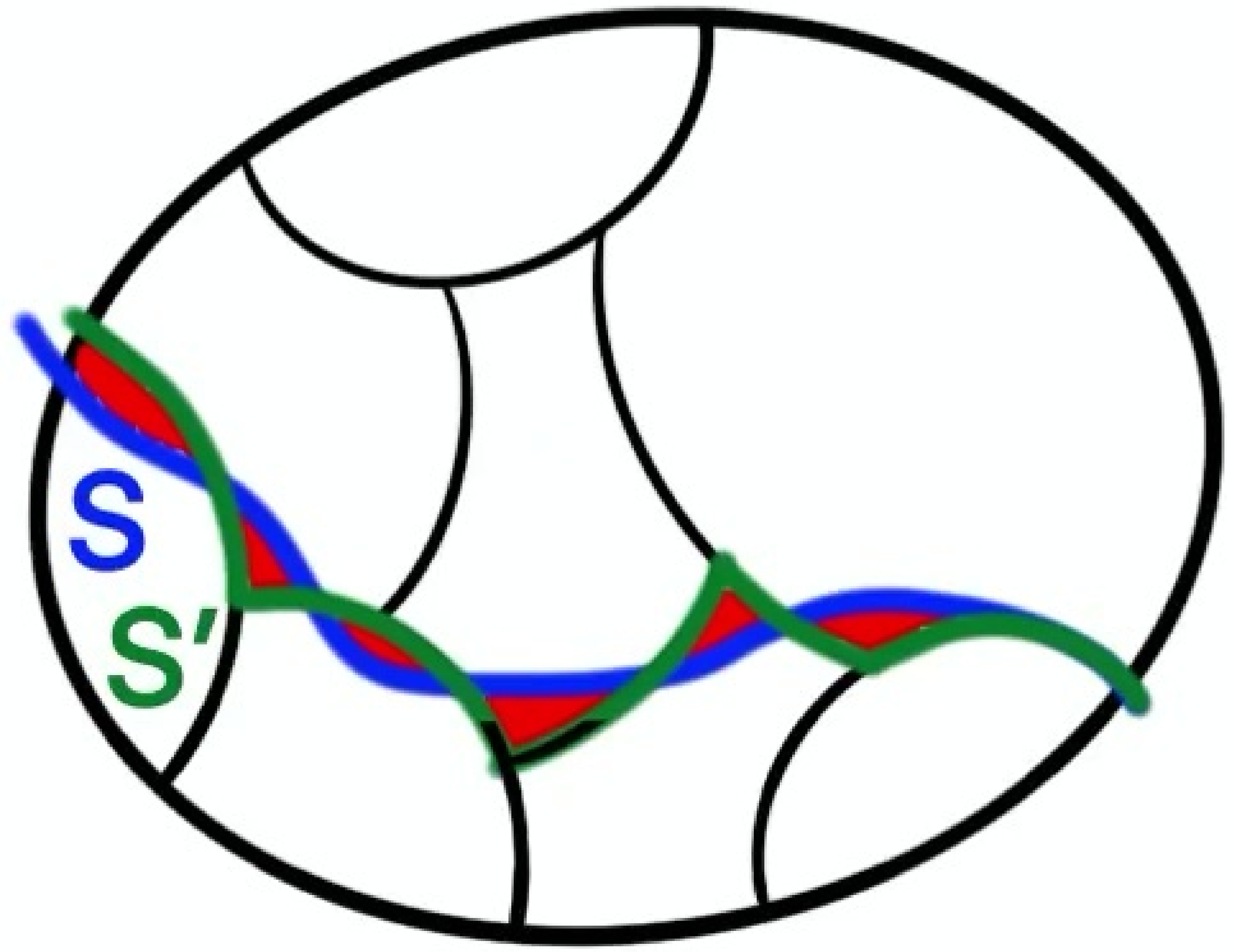
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition



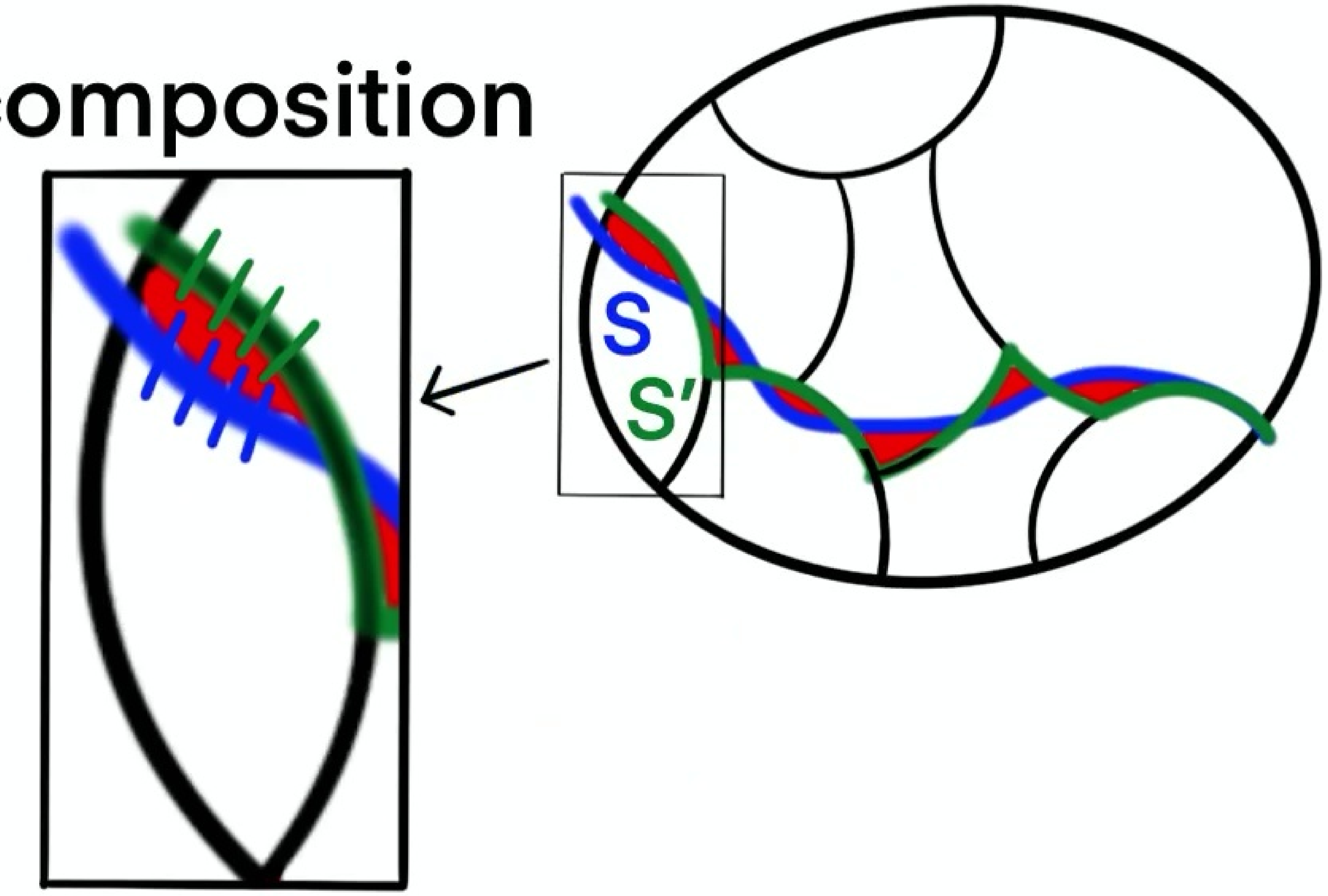
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition



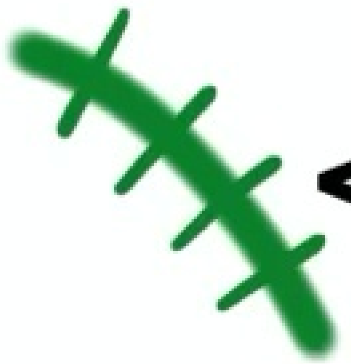
Structure Theorem Proof Outline

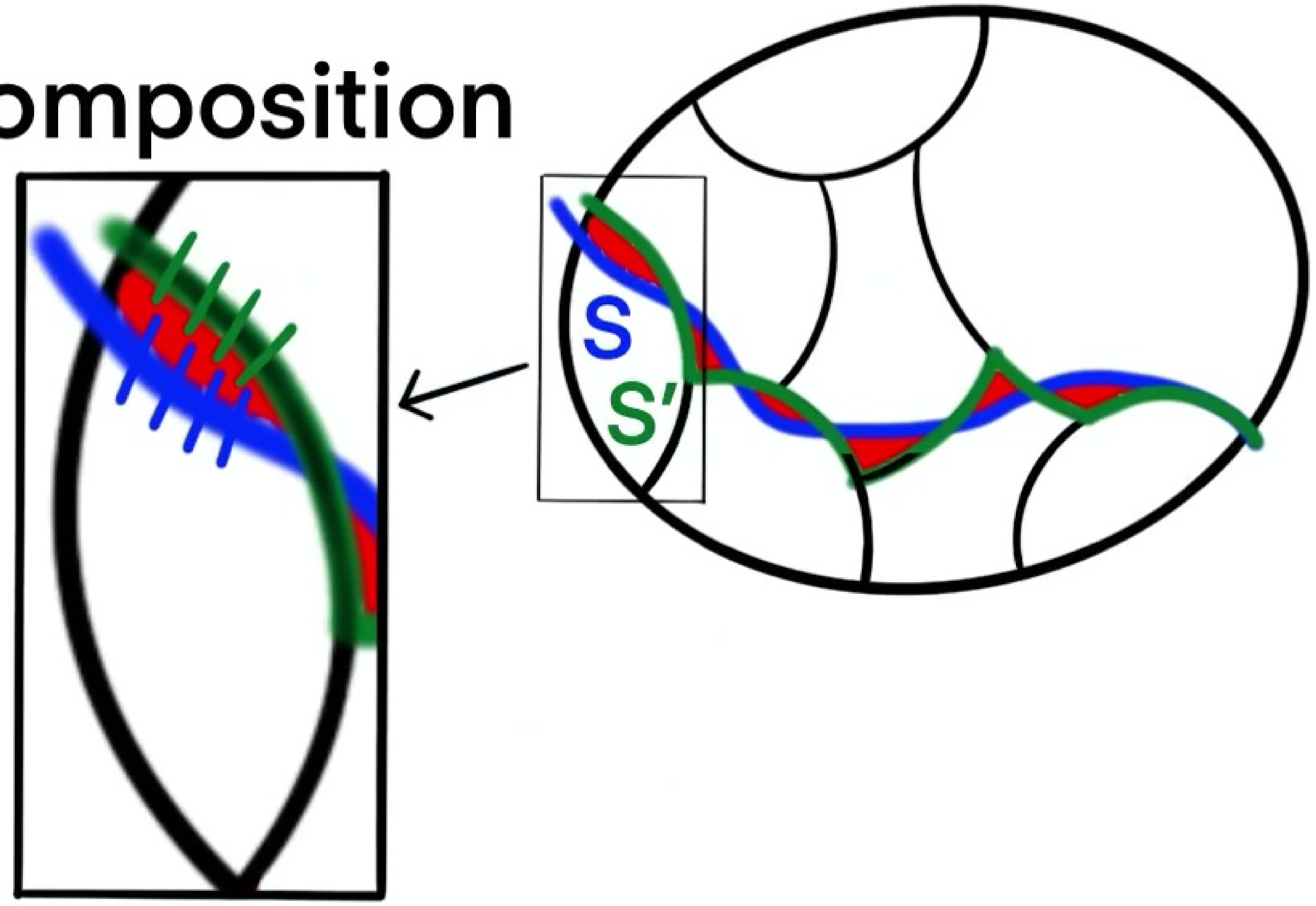
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If  <= (1+ε)x  then



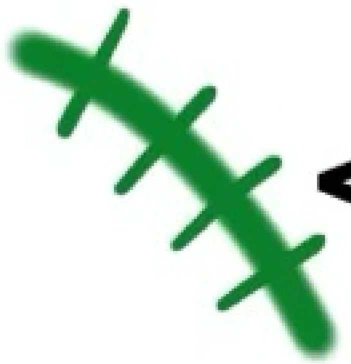
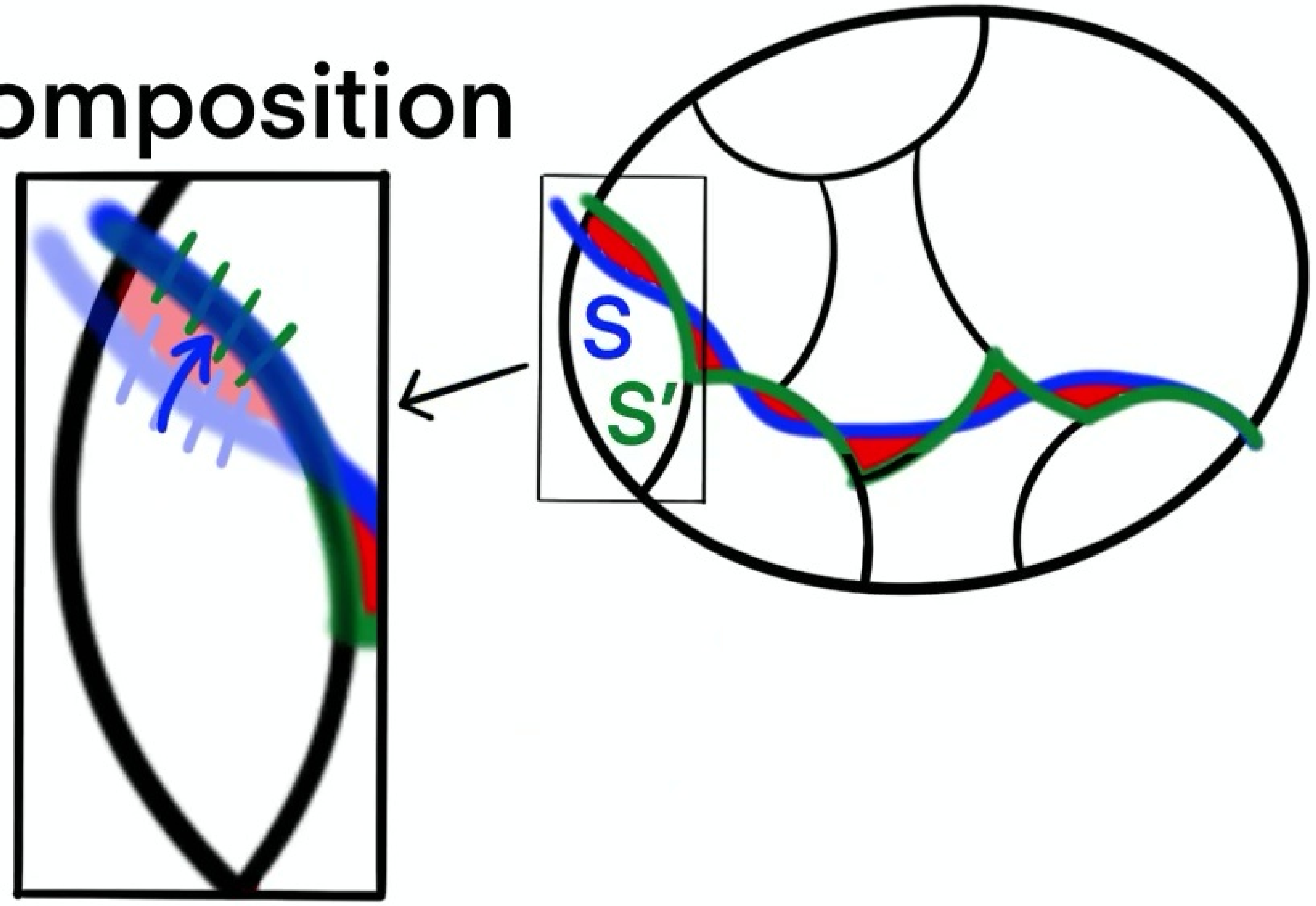
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If  <= (1+ε)x  then 

Structure Theorem Proof Outline

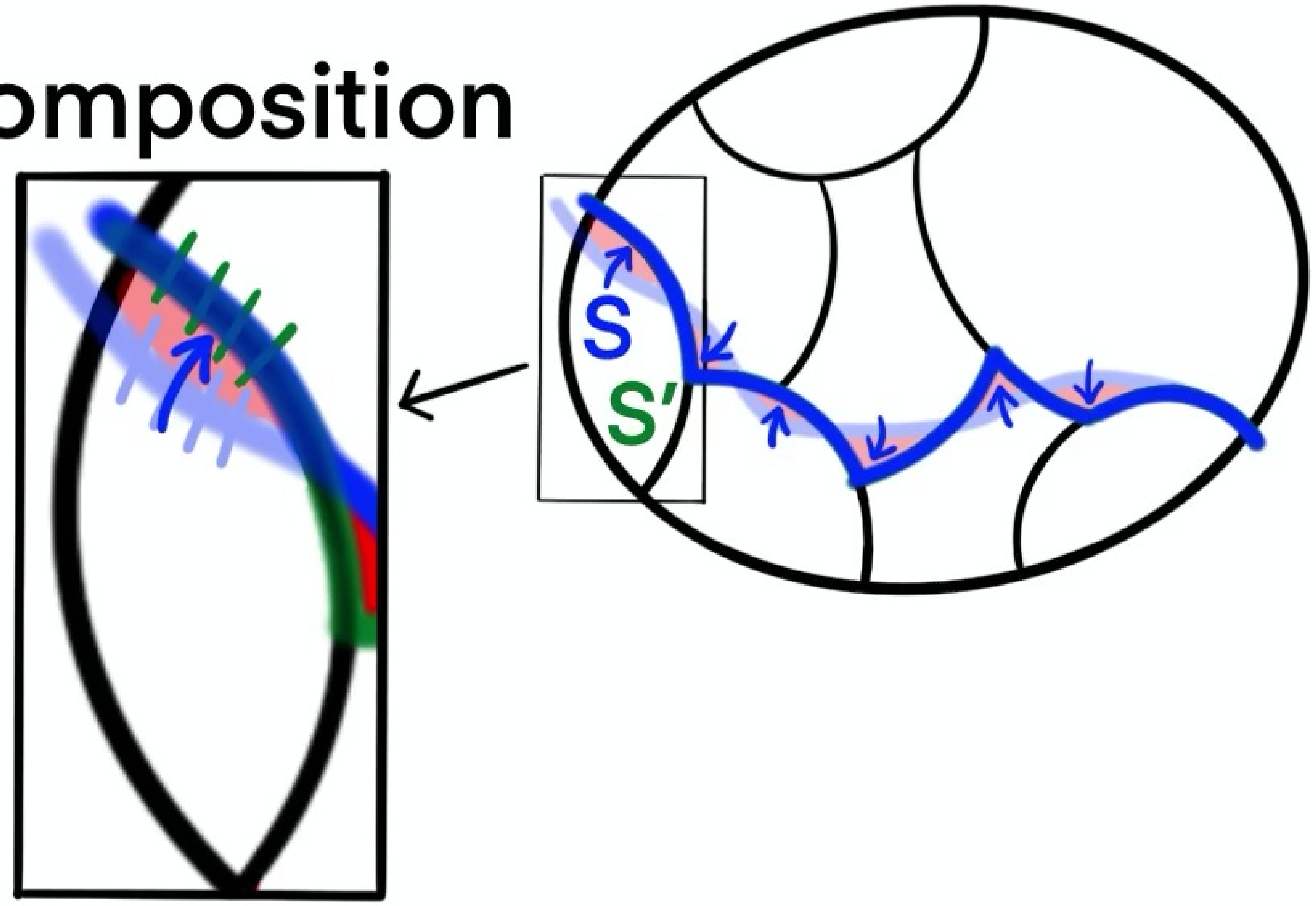
Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If <= (1+ε)x then



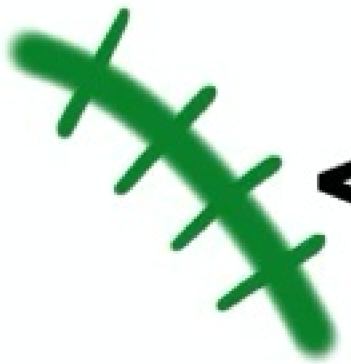
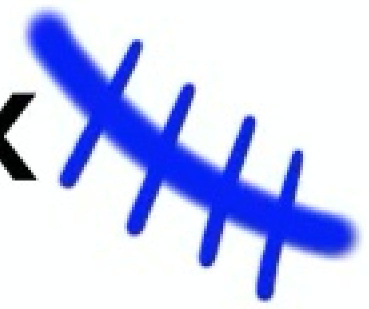
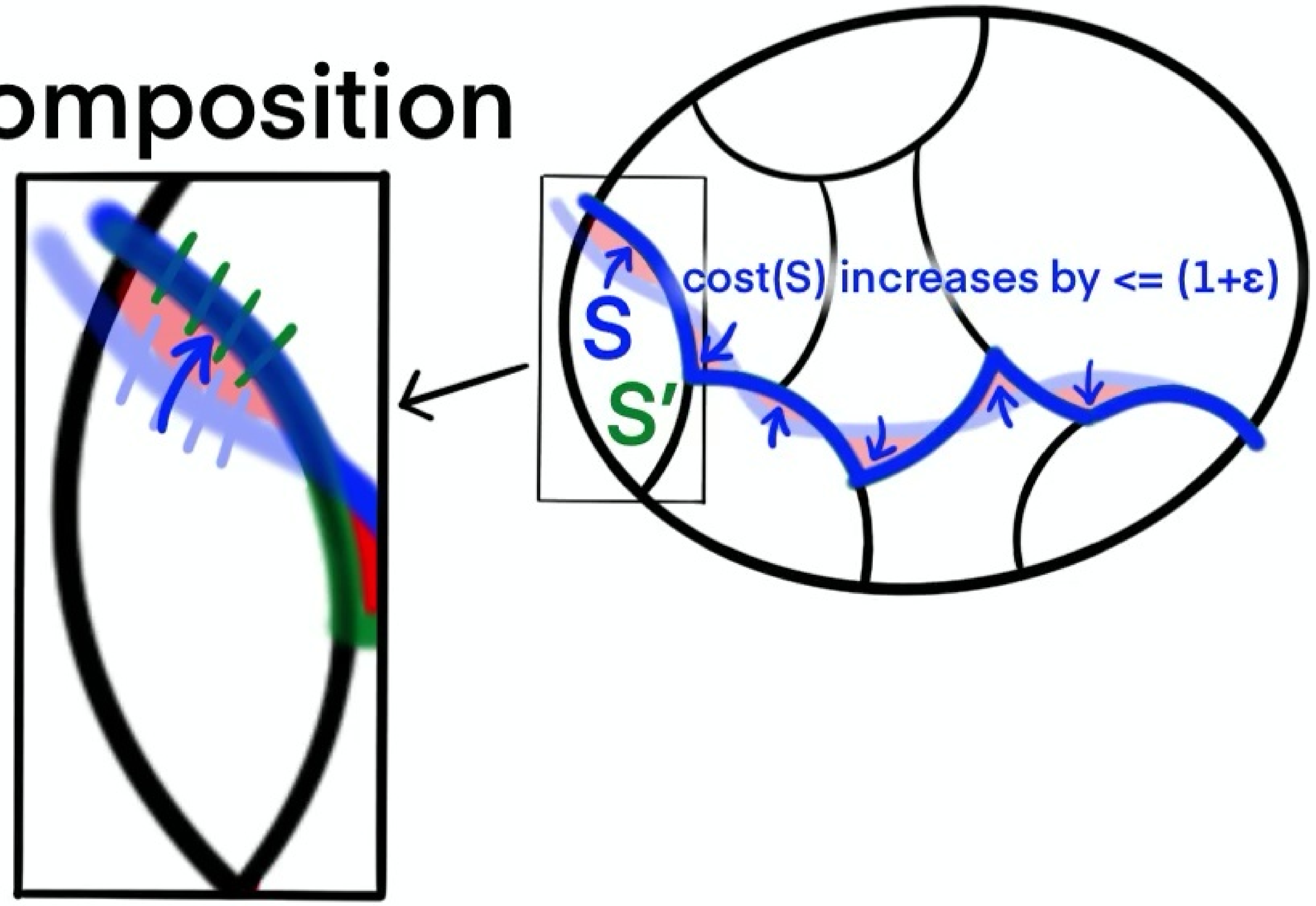
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If  <= (1+ε)x  then 

Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

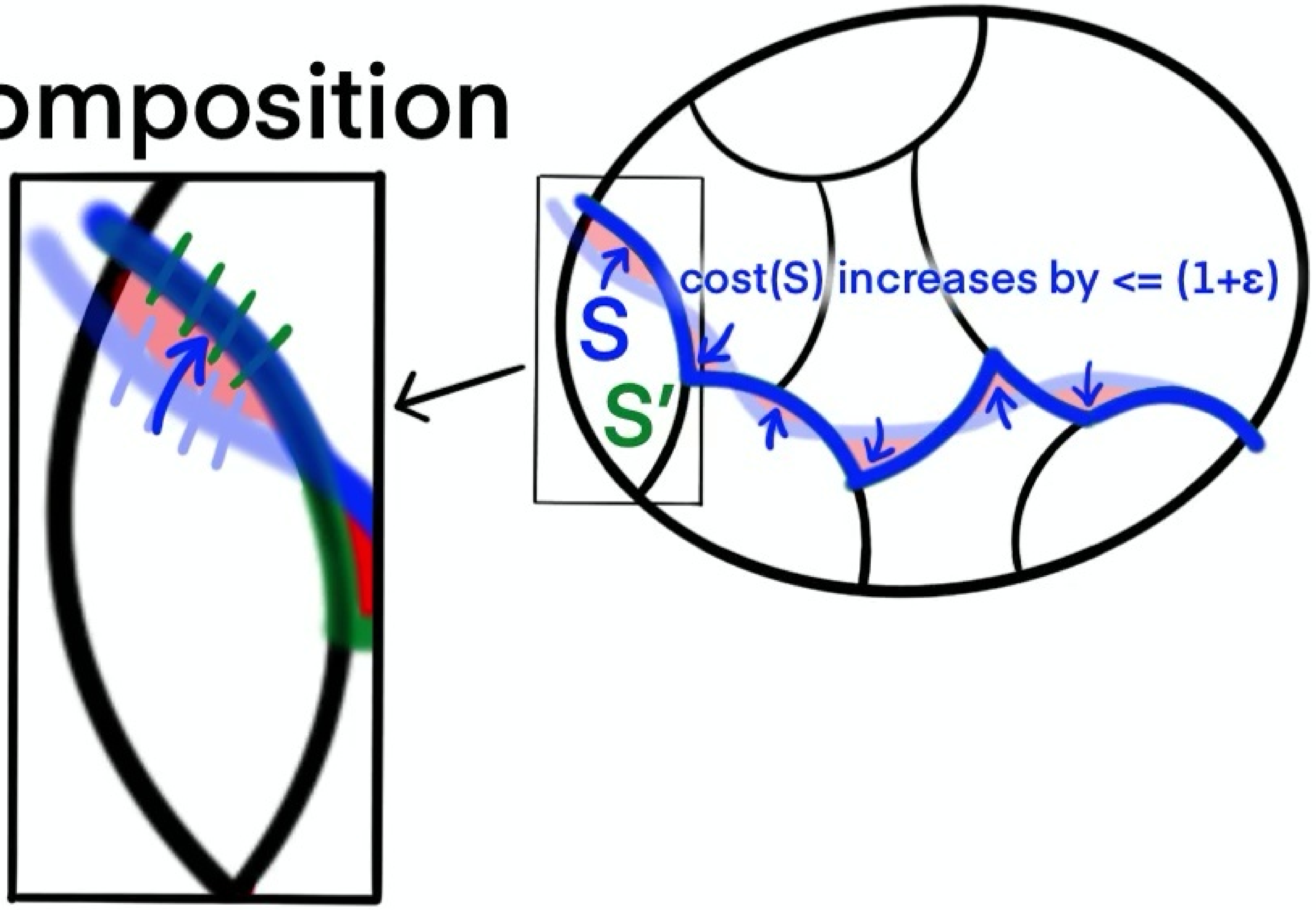
- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If <= (1+ε)x then

Else, split cluster



Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

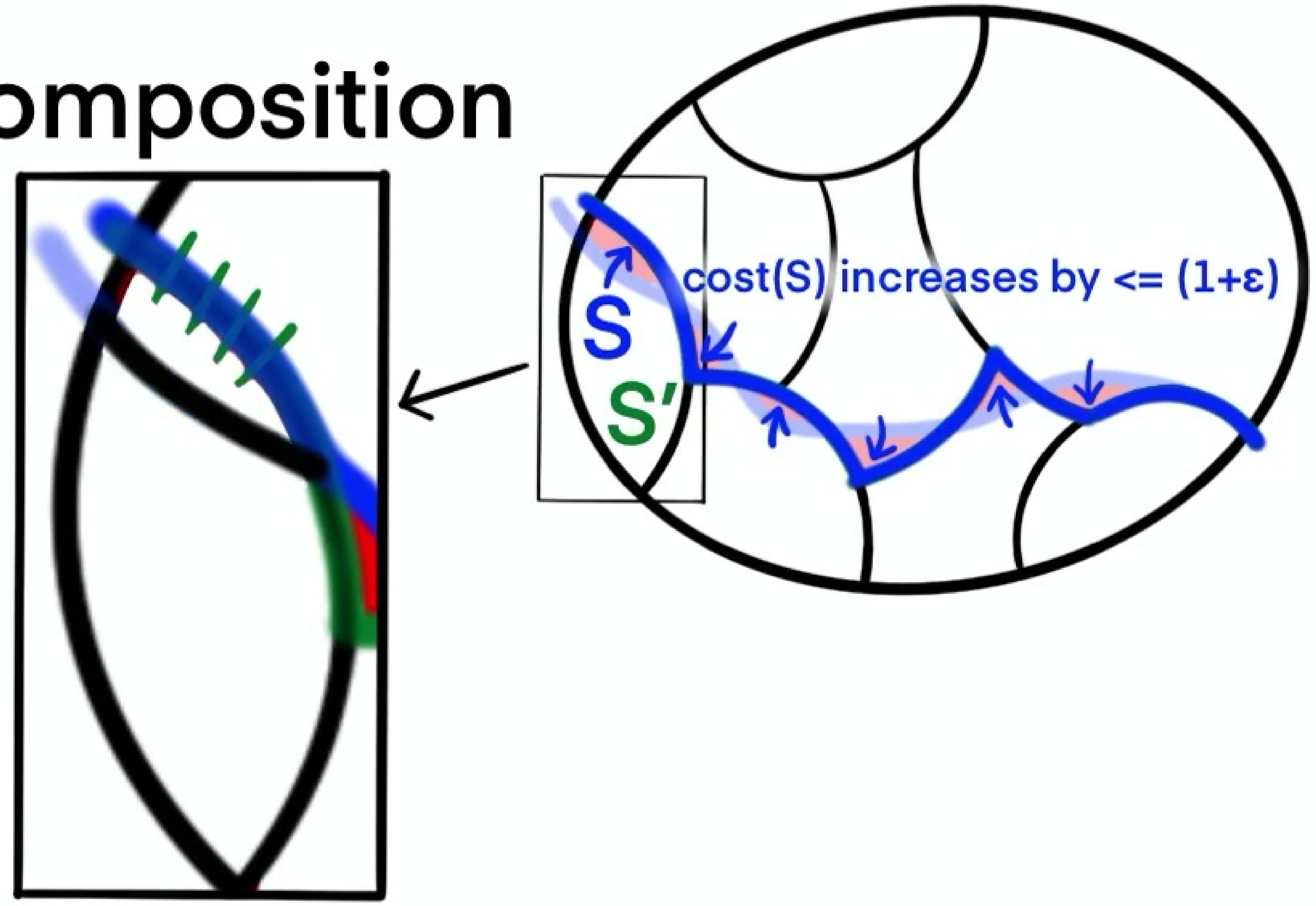
- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If <= (1+ε)x then

Else, split cluster



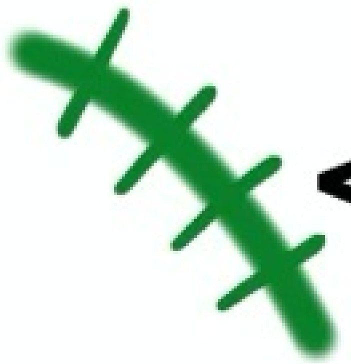
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

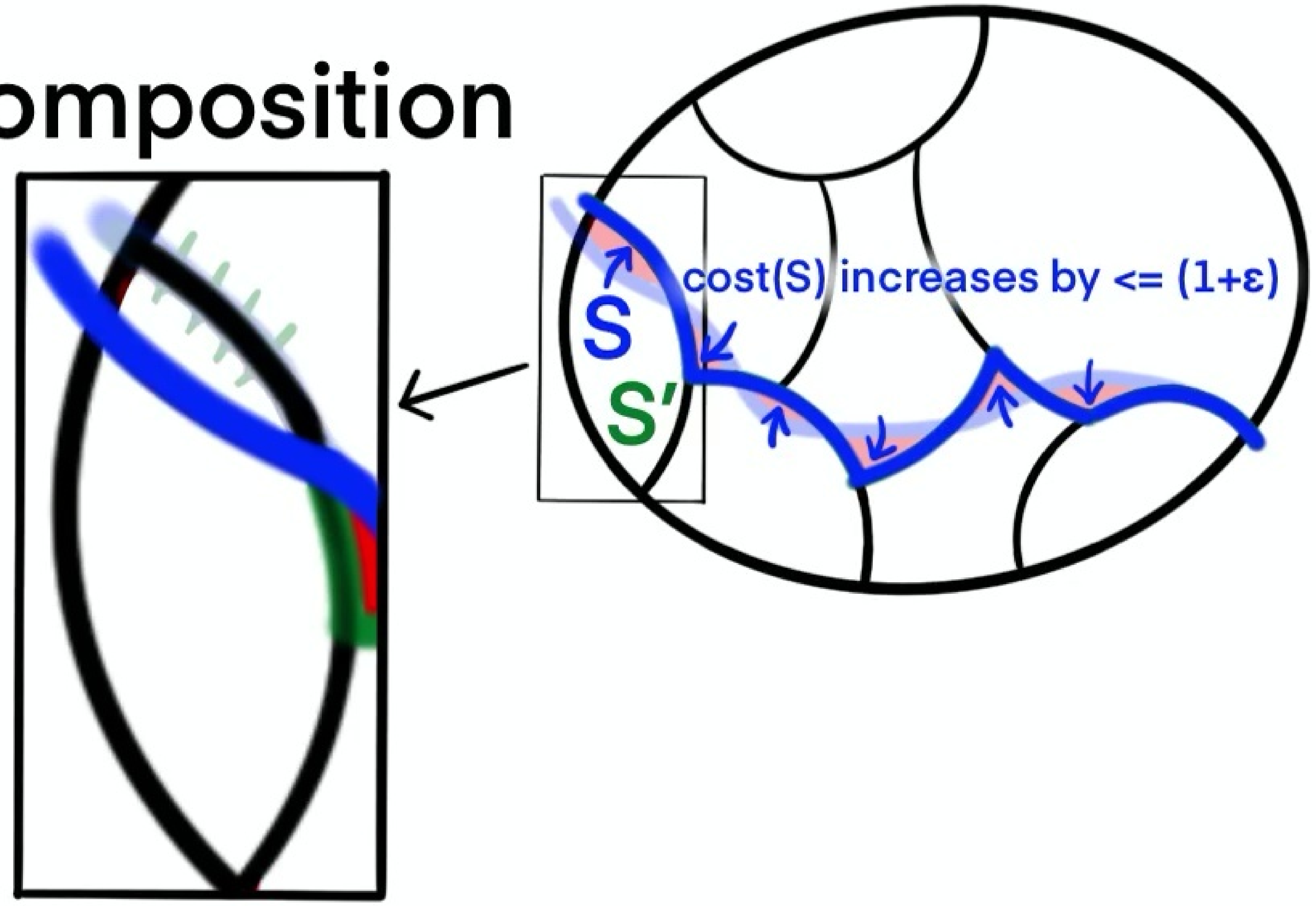
- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S), (2) S′ is consistent with clustering

Step 1: compute expander decomposition

If  <= (1+ε)x  then

Else, split cluster



Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition

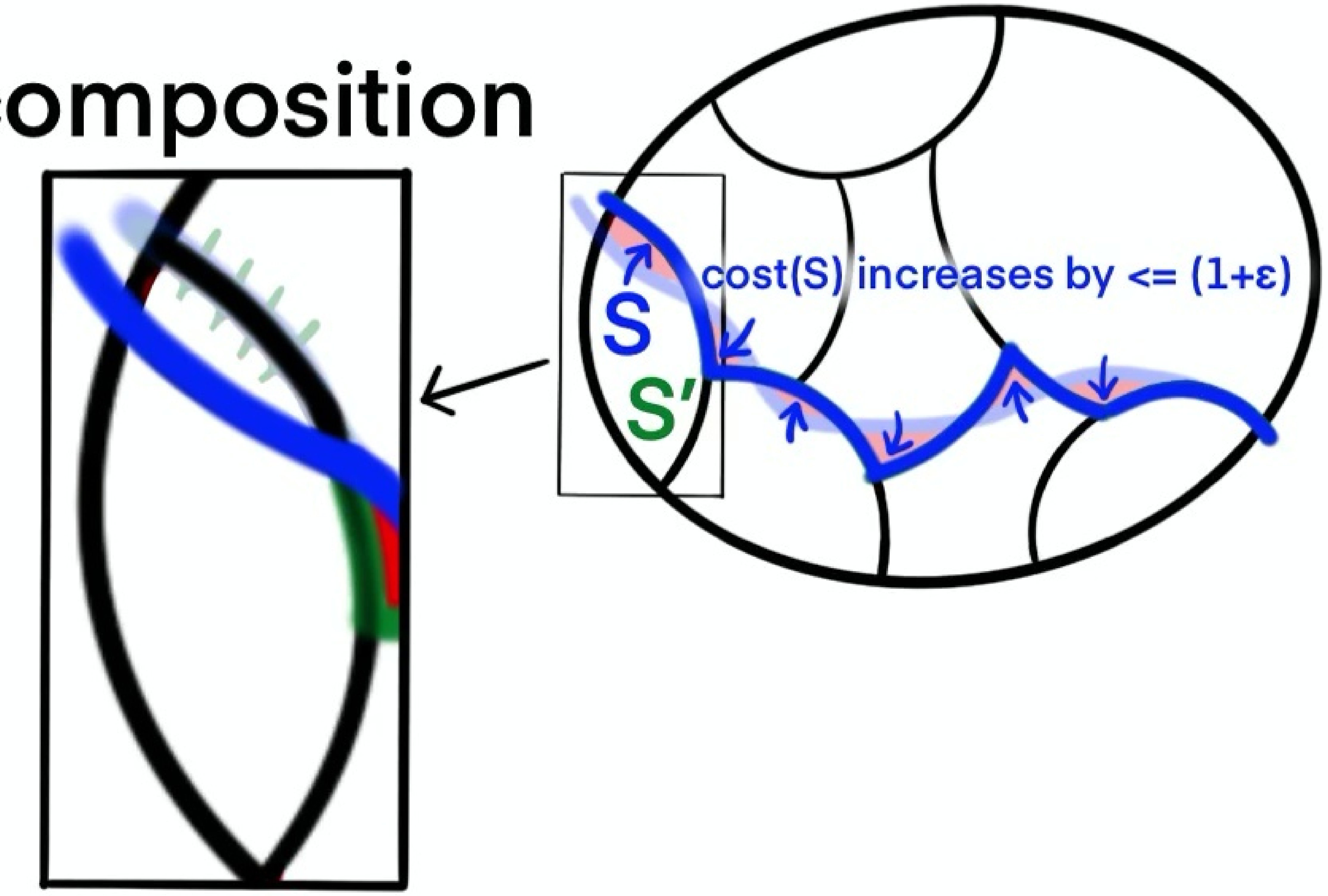
Step 2: while there exists

1.01-approx mincut S

and cluster C with

> (1+ε)x ,

split cluster C along S



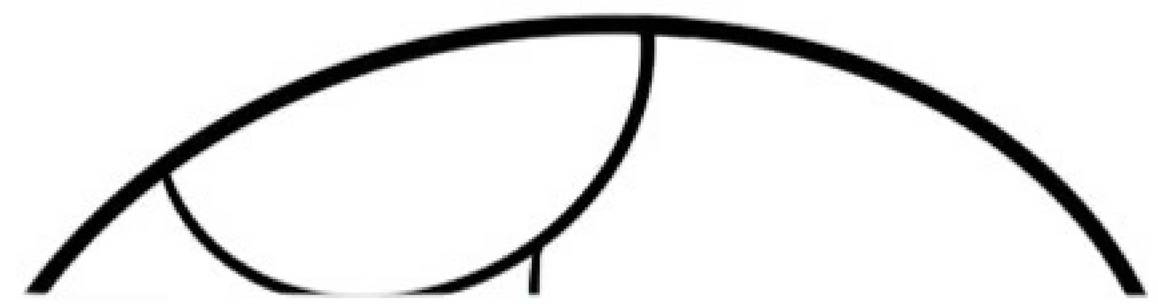
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition



Step 2: while there exists
1.01-approx mincut S
and cluster C with
> (1+ε)x ,
split cluster C along S

This process “converges”:
don’t cut too many edges in total
(analysis is technical)
→ at most n/2 clusters in total

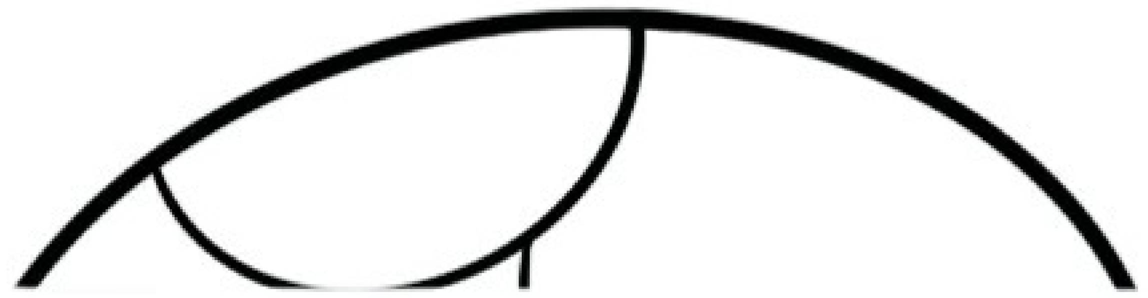
Structure Theorem Proof Outline

Theorem: for any weighted undirected graph, can group the vertices into <= n/2 clusters s.t.

- for any 1.01-approx mincut S, can modify few vertices → S′ s.t.

(1) cost(S′) <= (1+ε) cost(S),   (2) S′ is consistent with clustering

Step 1: compute expander decomposition



Step 2: while there exists

1.01-approx mincut S

and cluster C with

> (1+ε)x ,

split cluster C along S

This process "converges":

don't cut too many edges in total

(analysis is technical)

→ at most n/2 clusters in total

Structure Theorem Algorithm

- Naive algorithm is polynomial time: there are $<= n^2$ many 1.01-approximate mincuts

Structure Theorem Algorithm

- Naive algorithm is polynomial time: there are <=n^2 many 1.01-approximate mincuts
- Need near-linear time and deterministic

Structure Theorem Algorithm

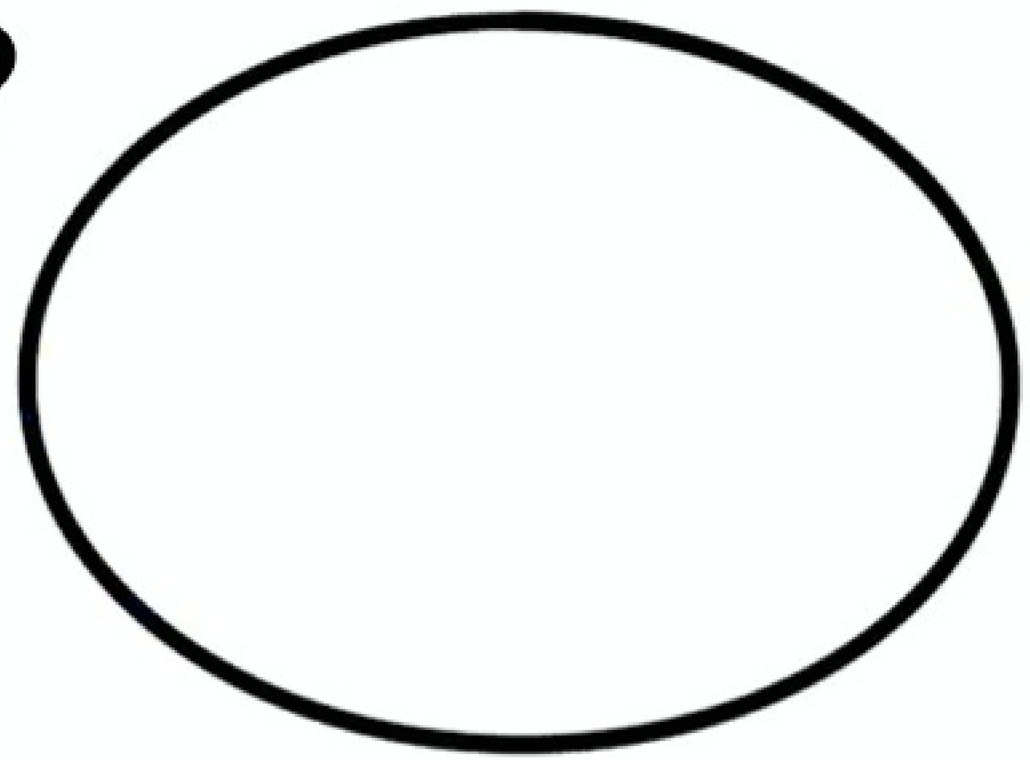
- Naive algorithm is polynomial time: there are <=n^2 many 1.01-approximate mincuts
- Need near-linear time and deterministic
  - Replace expander decomposition with s-strong decomp. [Kawarabayashi-Thorup'15] [Henzinger-Rao-Wang'17]

Structure Theorem Algorithm

- Naive algorithm is polynomial time: there are <=n^2 many 1.01-approximate mincuts
- Need near-linear time and deterministic
  - Replace expander decomposition with s-strong decomp. [Kawarabayashi-Thorup’15] [Henzinger-Rao-Wang’17]
  - If cluster is small (<= polylog vertices), can use poly-time algorithm (“Small Cluster Decomposition”)

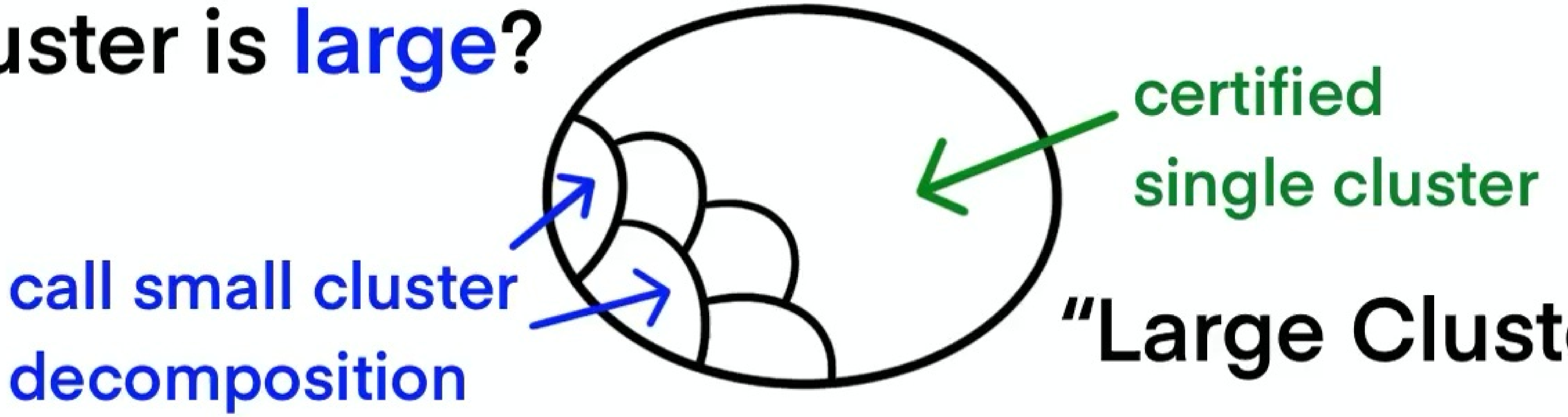
Structure Theorem Algorithm

- Naive algorithm is polynomial time: there are <=n^2 many 1.01-approximate mincuts
- Need near-linear time and deterministic
  - Replace expander decomposition with s-strong decomp. [Kawarabayashi-Thorup'15] [Henzinger-Rao-Wang'17]
  - If cluster is small (<= polylog vertices), can use poly-time algorithm ("Small Cluster Decomposition")
  - If cluster is large?



Structure Theorem Algorithm

- Naive algorithm is polynomial time: there are <=n^2 many 1.01-approximate mincuts
- Need near-linear time and deterministic
  - Replace expander decomposition with s-strong decomp. [Kawarabayashi-Thorup'15] [Henzinger-Rao-Wang'17]
  - If cluster is small (<= polylog vertices), can use poly-time algorithm ("Small Cluster Decomposition")
  - If cluster is large?



"Large Cluster Decomposition"

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)
  - [Karger] Edge sampling gives skeleton graph w.h.p.

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)
  - [Karger] Edge sampling gives skeleton graph w.h.p.
- [Li'21]: deterministic skeleton graph in $m^{1+o(1)}$ time

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)
  - [Karger] Edge sampling gives skeleton graph w.h.p.
- [Li'21]: deterministic skeleton graph in $m^{1+o(1)}$ time
  - Iterative clustering by expander decomposition

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)
  - [Karger] Edge sampling gives skeleton graph w.h.p.
- [Li'21]: deterministic skeleton graph in $m^{1+o(1)}$ time
  - Iterative clustering by expander decomposition
  - Derandomize edge sampling by pessimistic estimator

From Approximate to Exact Mincut

- Only randomized component in Karger's algorithm is building the skeleton graph (mincut sparsifier)
  - [Karger] Edge sampling gives skeleton graph w.h.p.
- [Li'21]: deterministic skeleton graph in $m^{1+o(1)}$ time
  - Iterative clustering by expander decomposition
  - Derandomize edge sampling by pessimistic estimator
- [This work]: replace expander decomposition by Structure Theorem

Conclusion

- Structure Theorem: new local approach to global mincut

Conclusion

- Structure Theorem: new local approach to global mincut
  - Generalization of Kawarabayashi-Thorup sparsification to weighted graphs

Conclusion

- Structure Theorem: new local approach to global mincut
  - Generalization of Kawarabayashi-Thorup sparsification to weighted graphs
  - Only $(1+\varepsilon)$-approx. algorithm that doesn't use max-flow or Karger's tree packing (global)

Conclusion

- Structure Theorem: new local approach to global mincut
  - Generalization of Kawarabayashi-Thorup sparsification to weighted graphs
  - Only $(1+\varepsilon)$-approx. algorithm that doesn't use max-flow or Karger's tree packing (global)
- Should be useful for global mincut in other settings (dynamic/streaming/distributed)