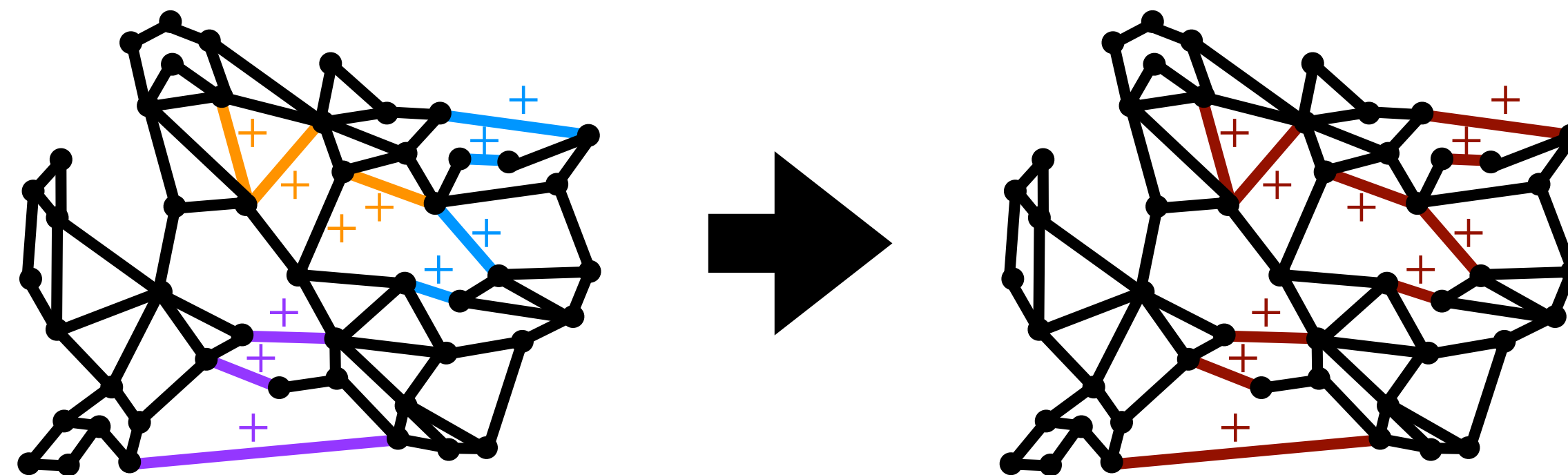


Simple Length-Constrained Expander Decompositions

(SOSA 2026)



Bernhard Haeupler
INSAIT & ETH Zurich



D Ellis Hershkowitz
Brown



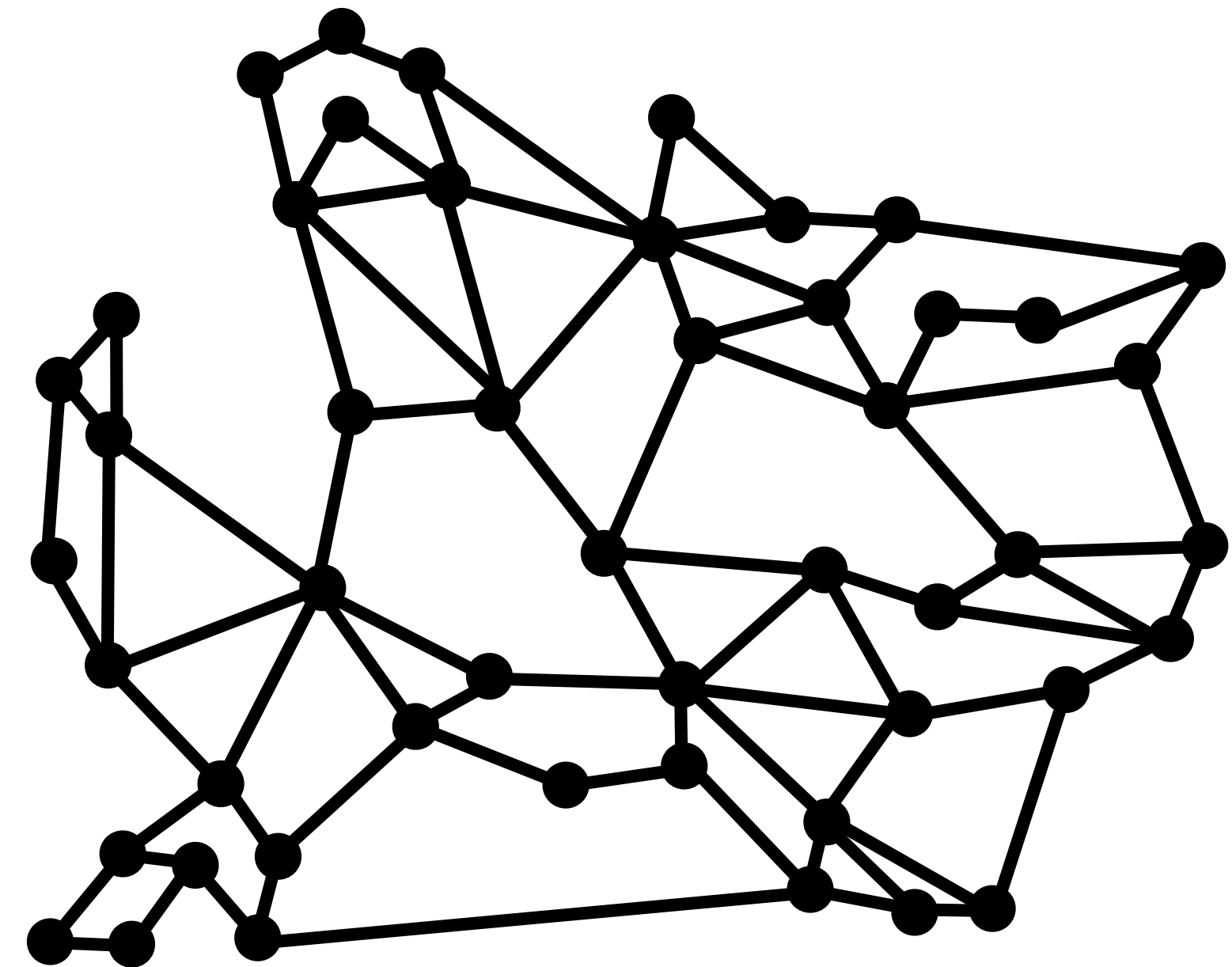
Greg Bodwin
U. Michigan



Zihan Tan
U. Minnesota

How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

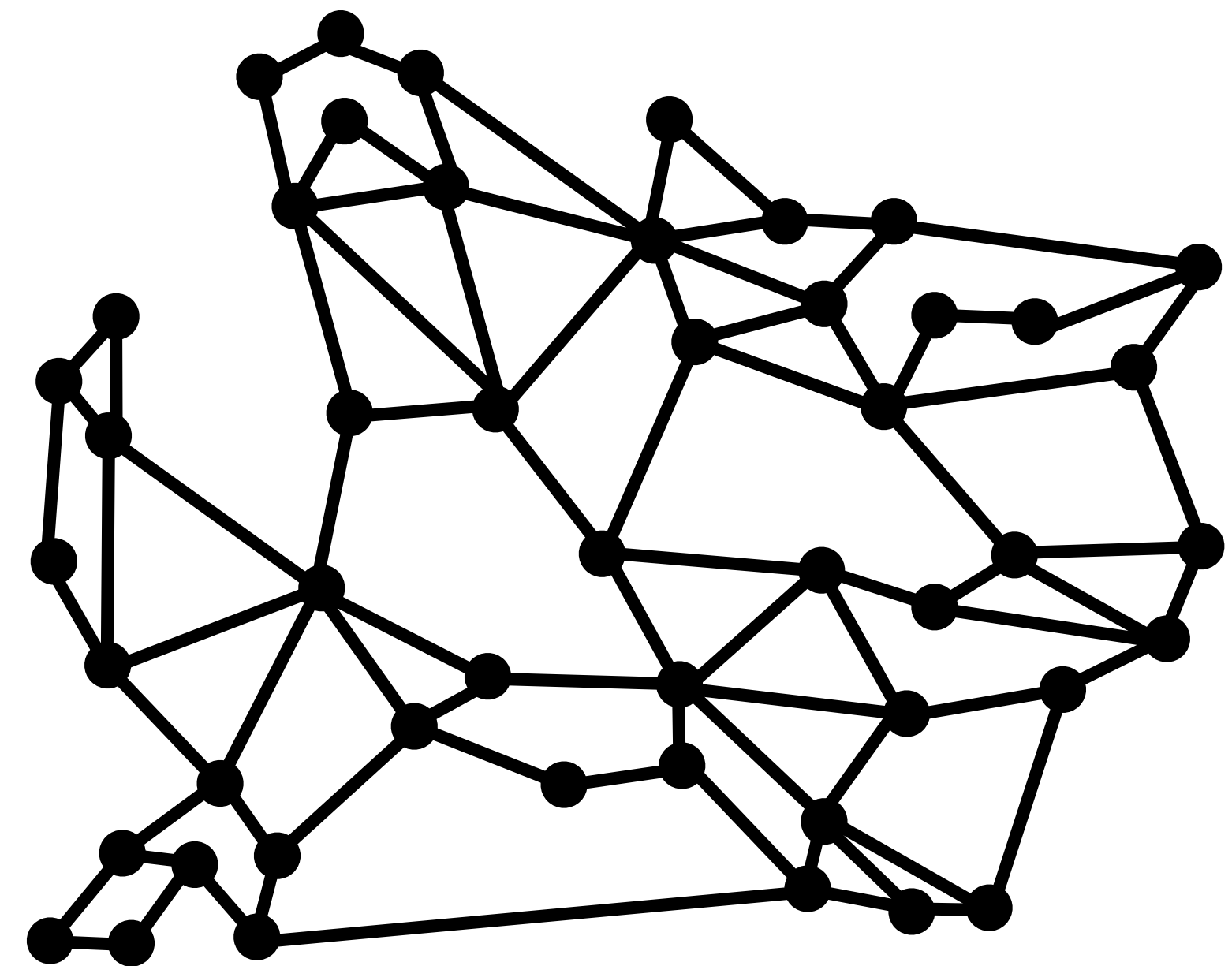


How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

1. **Graph Decomposition**

add to graph "modifications"
to make it "nice"

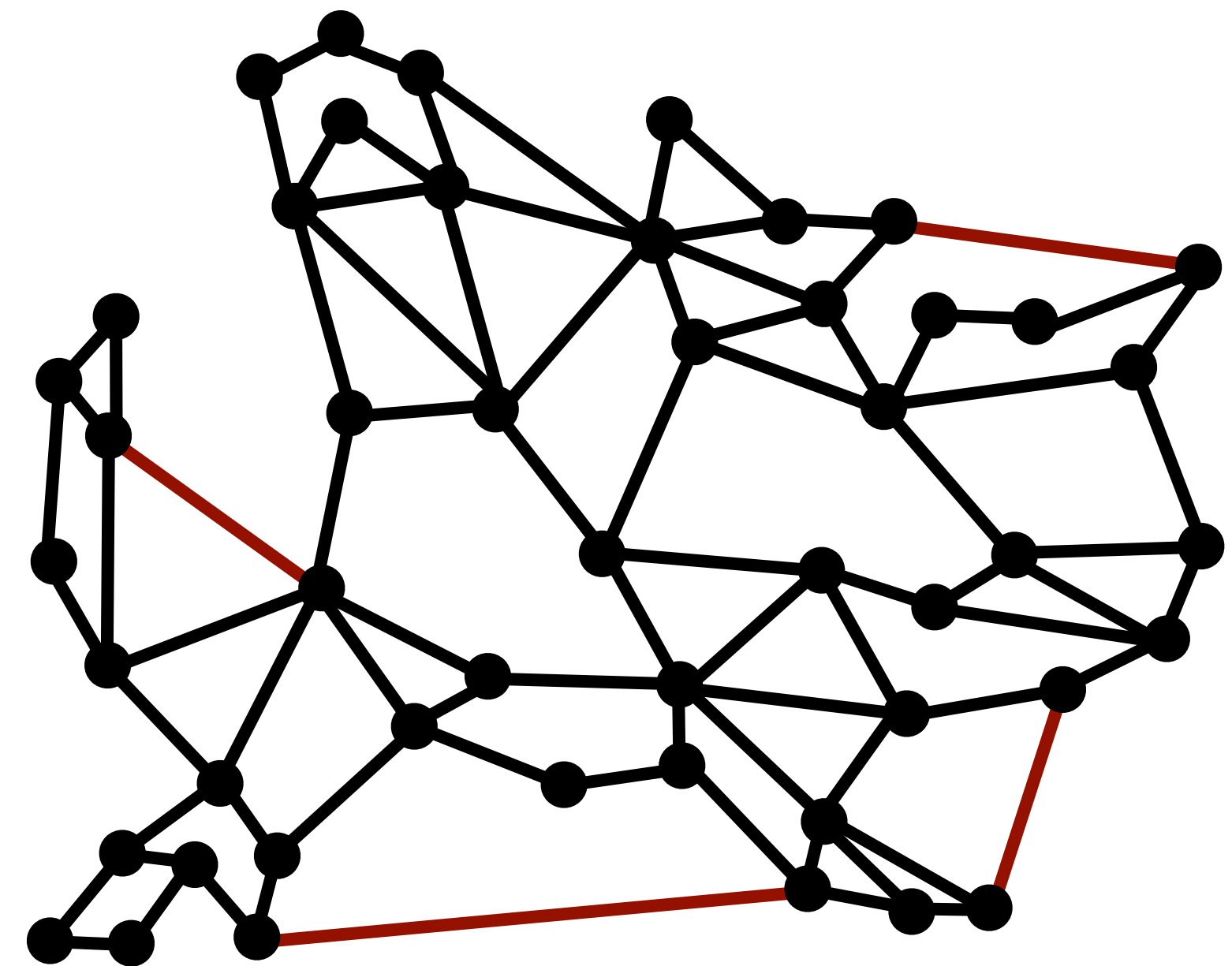


How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

1. Graph Decomposition

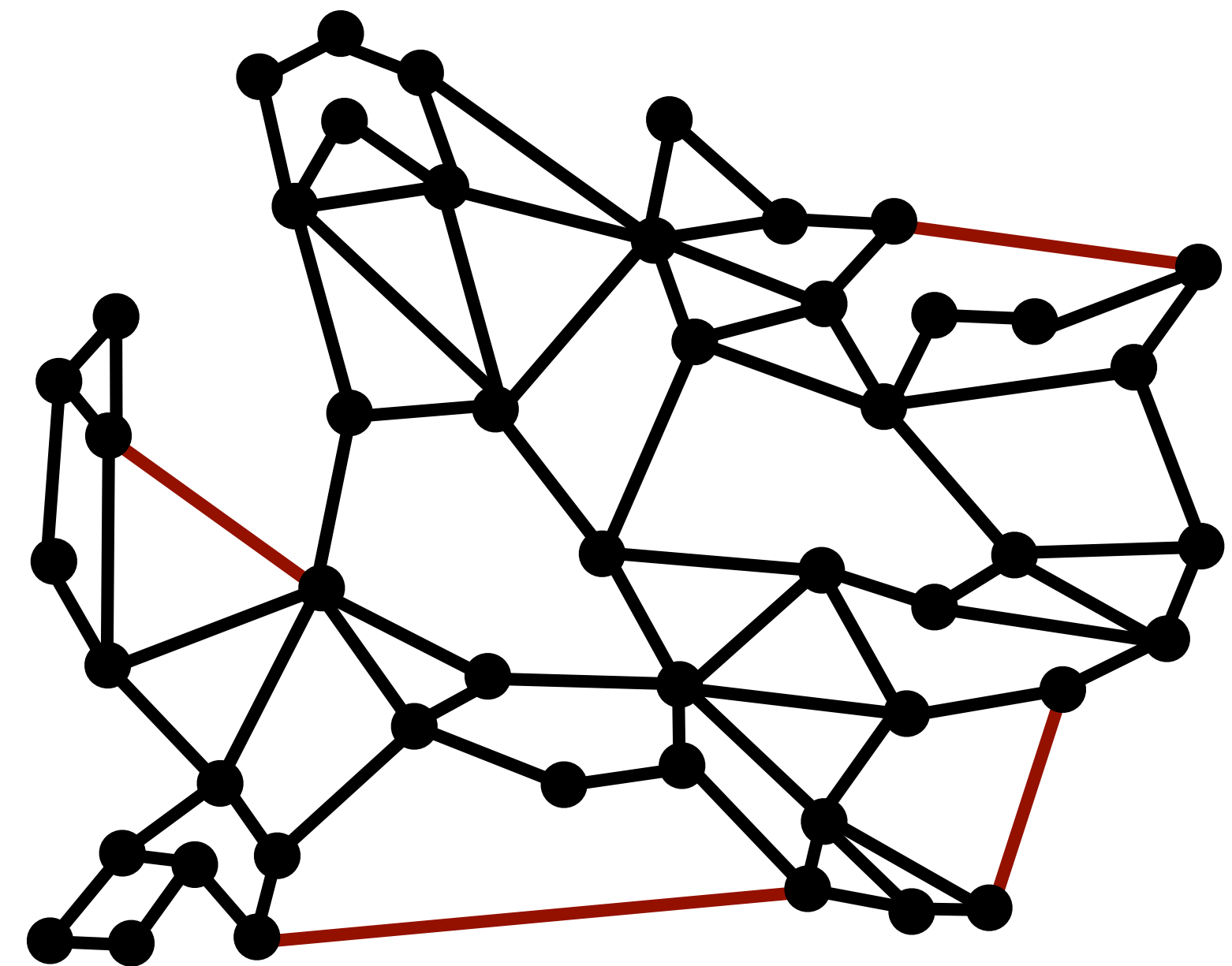
add to graph "modifications"
to make it "nice"



How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

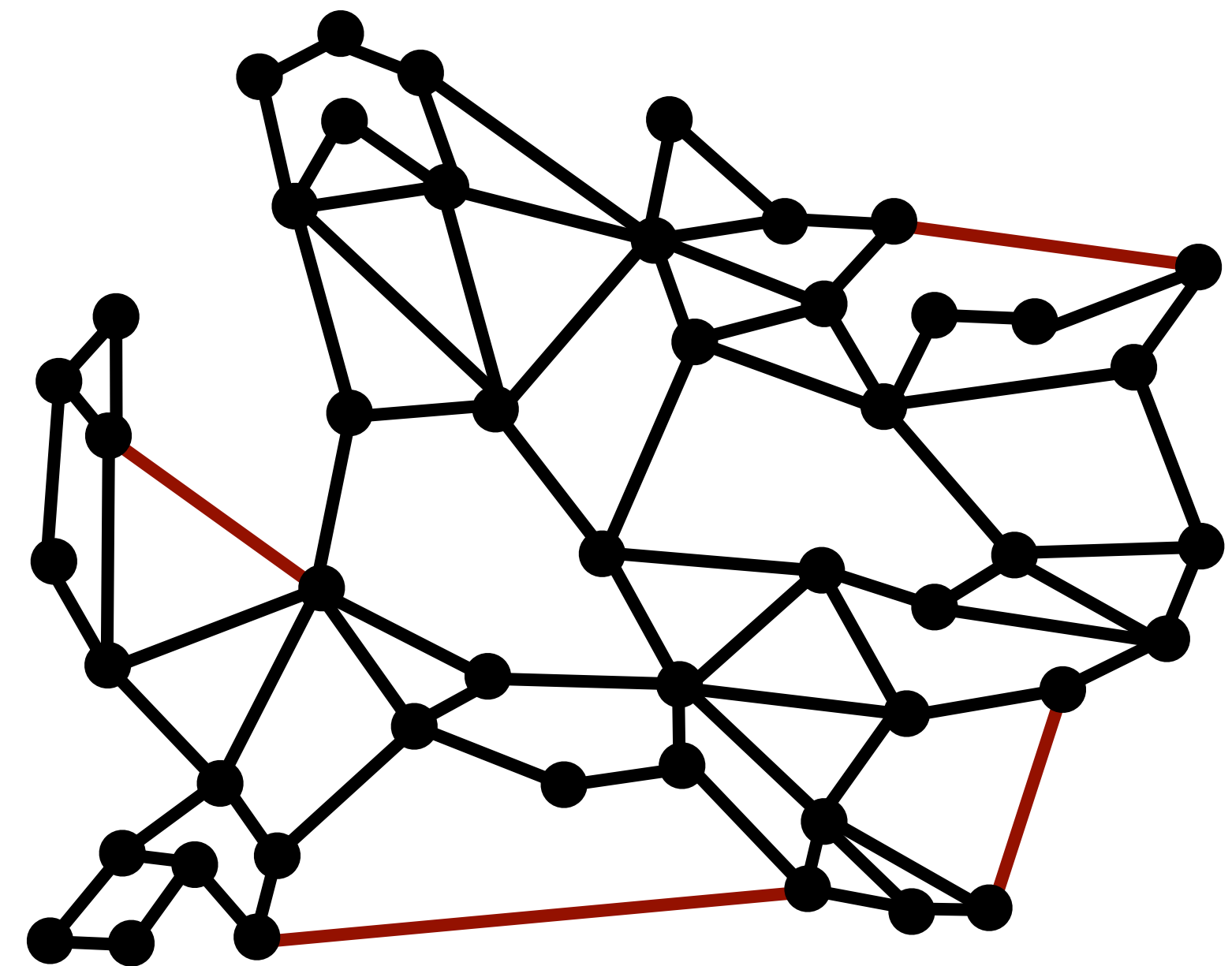
1. **Graph Decomposition**
add to graph "modifications"
to make it "nice"
2. **Solve Problem**
solve problem on nice graph



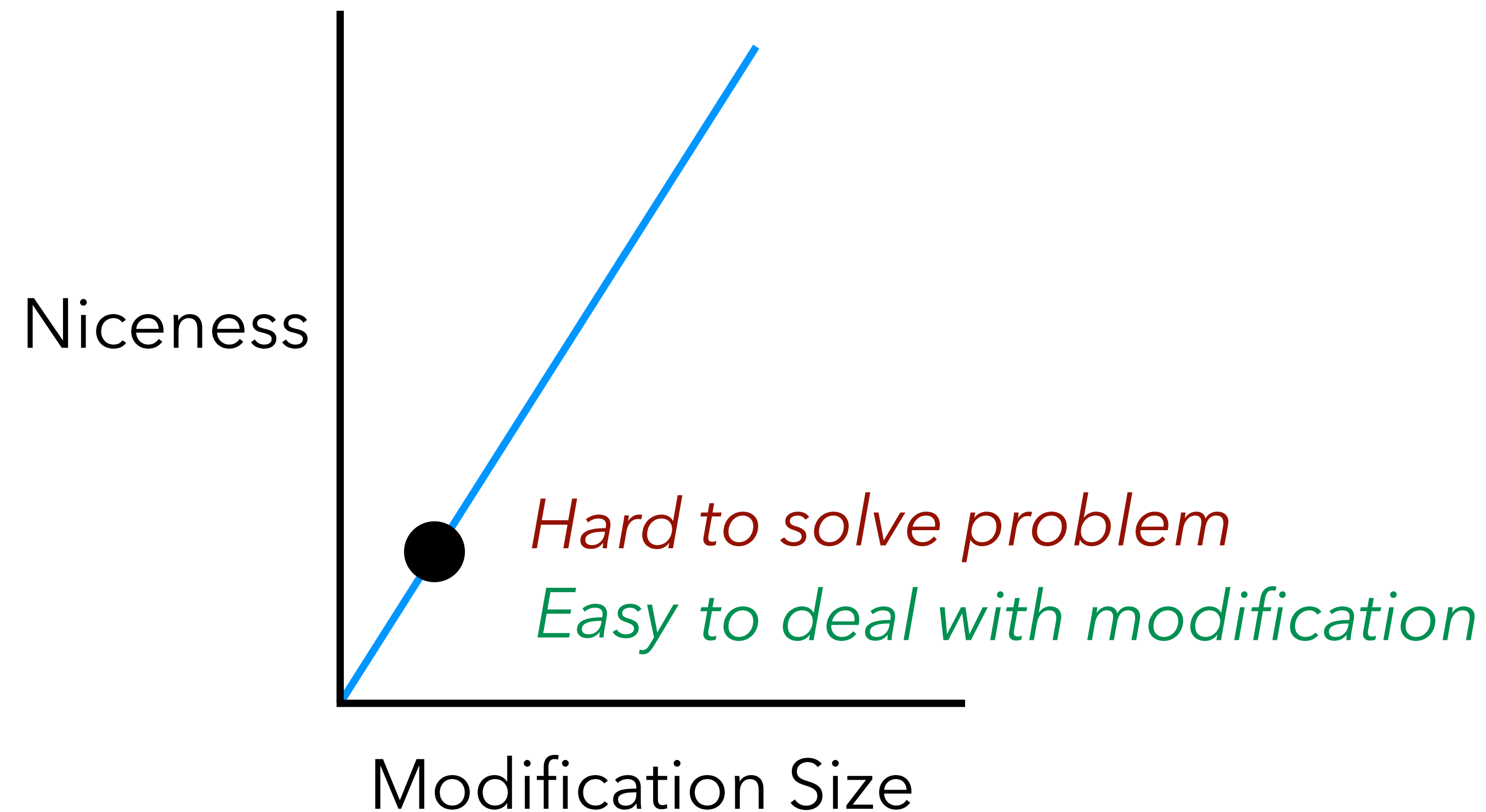
How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

1. **Graph Decomposition**
add to graph "modifications"
to make it "nice"
2. **Solve Problem**
solve problem on nice graph
3. **Clean Up**
deal with modifications

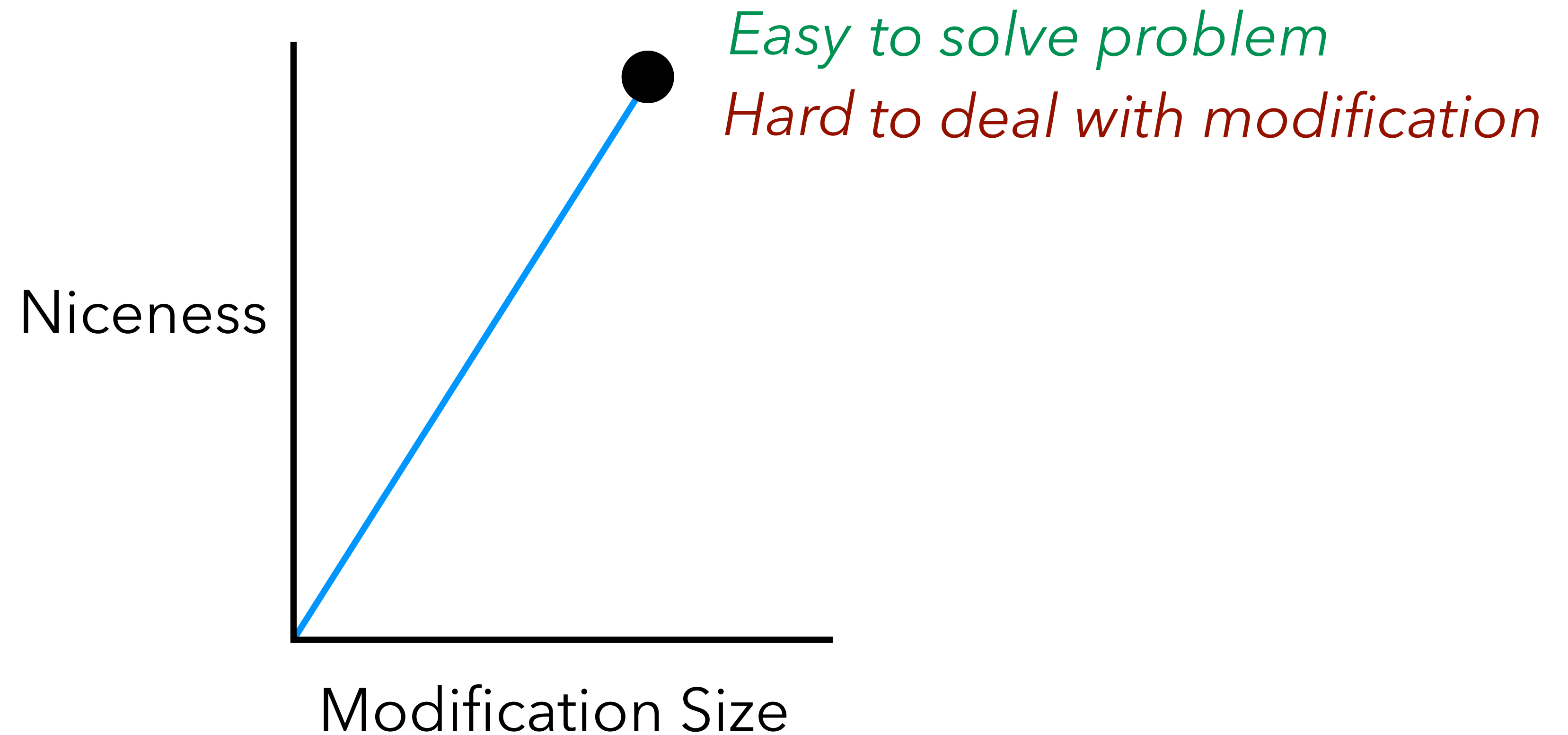


How to Solve Your Favorite Graph Problem



Graph Decomposition Size-Niceness Tradeoff

How to Solve Your Favorite Graph Problem

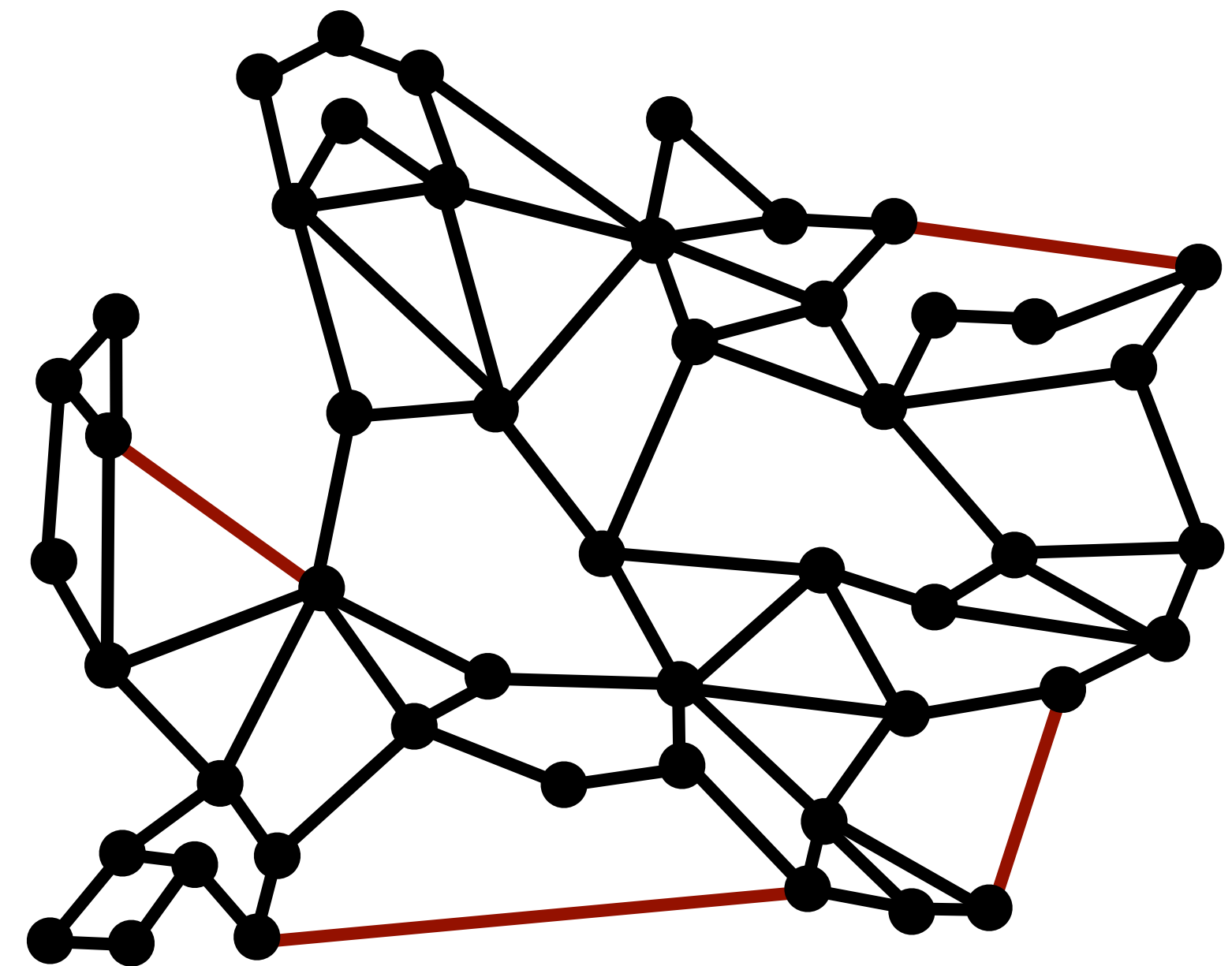


Graph Decomposition Size-Niceness Tradeoff

Length-Constrained Expander Decompositions

Graph Decomposition Approach

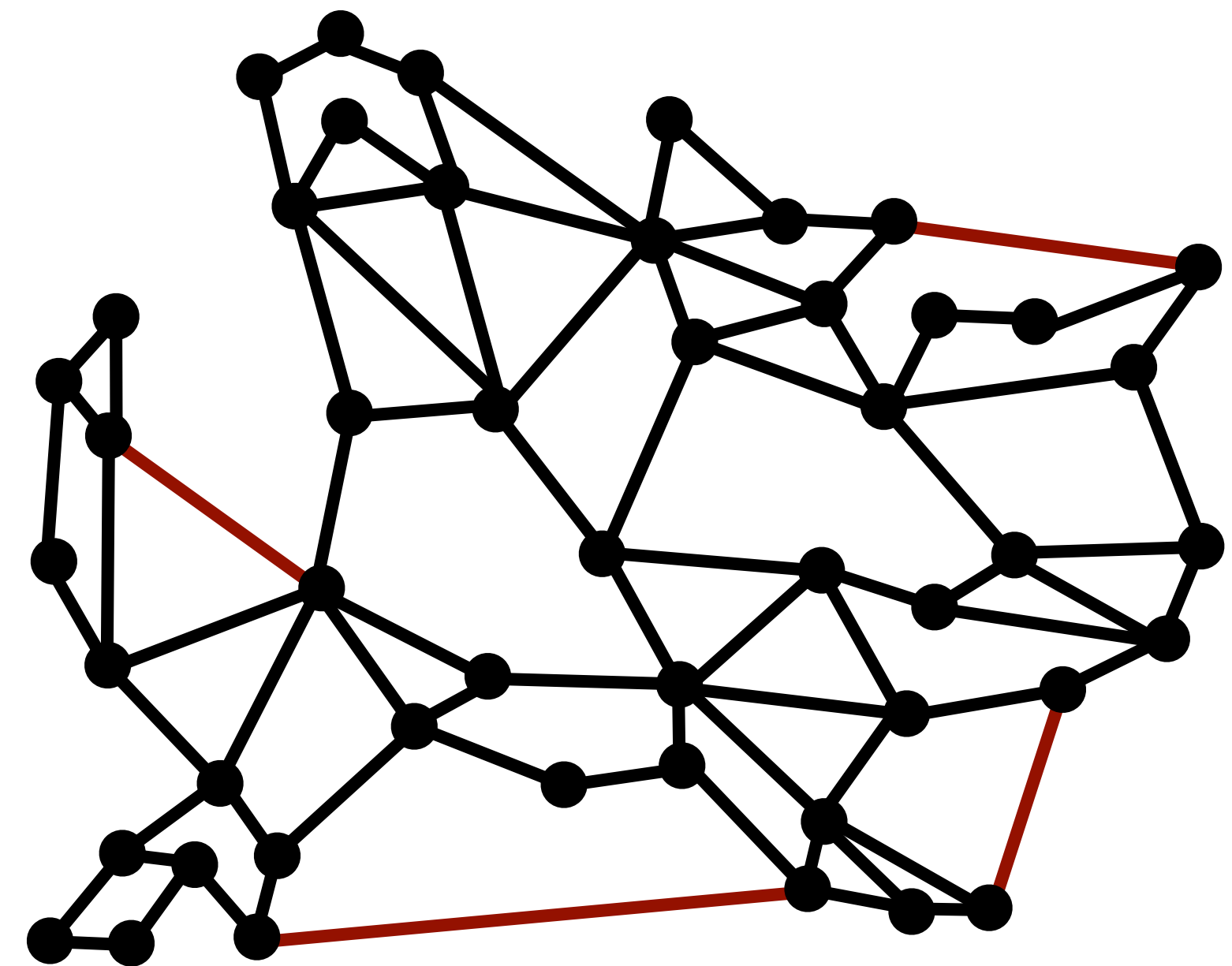
1. **Graph Decomposition**
add to graph "modifications"
to make it "nice"
2. **Solve Problem**
solve problem on nice graph
3. **Clean Up**
deal with modifications



Length-Constrained Expander Decompositions

Graph Decomposition Approach

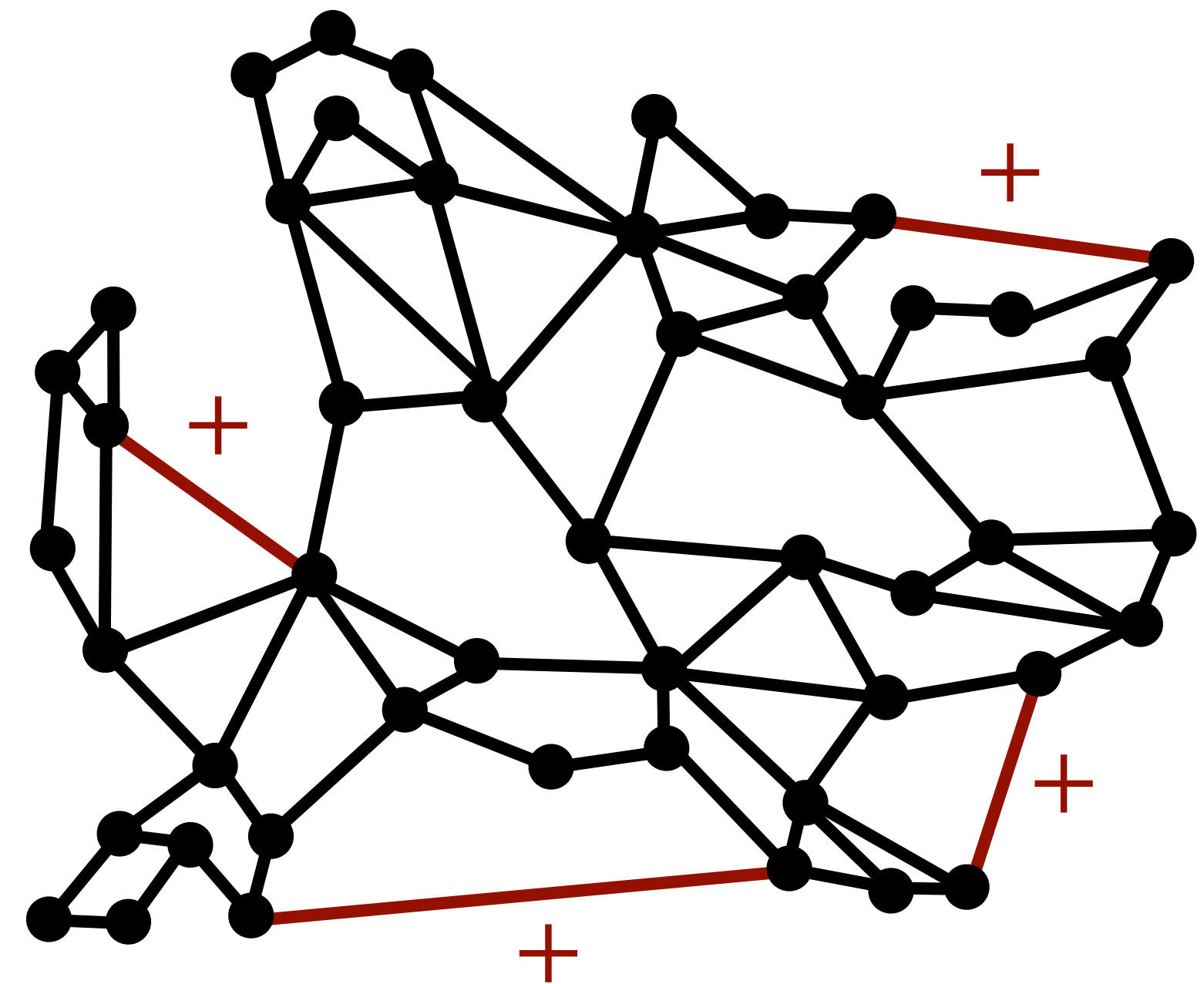
1. **Length-Constrained Expander Decomposition**
add to graph "modifications"
to make it "nice"
2. **Solve Problem**
solve problem on nice graph
3. **Clean Up**
deal with modifications



Length-Constrained Expander Decompositions

Graph Decomposition Approach

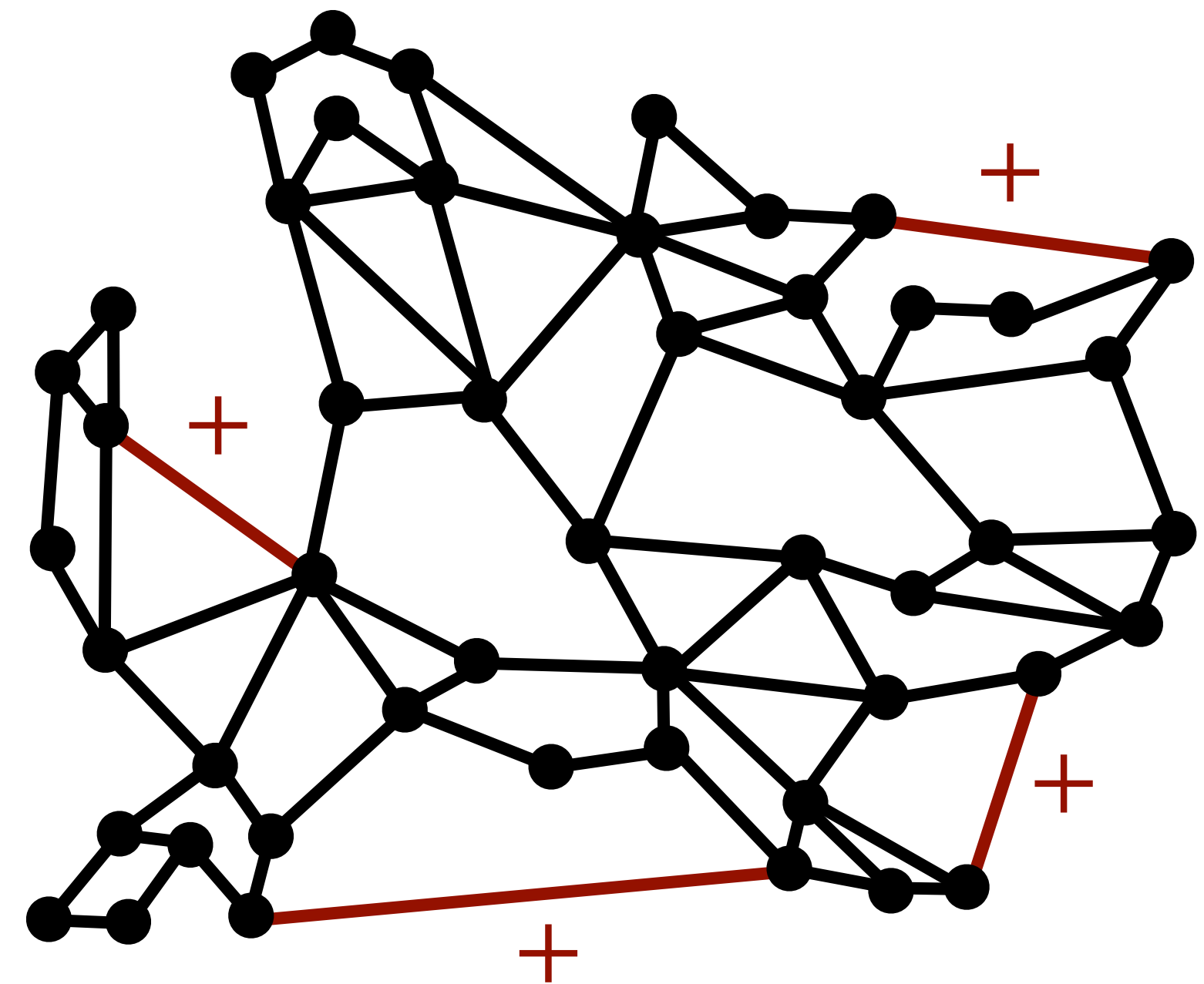
1. **Length-Constrained Expander Decomposition**
add to graph **edge length increases**
to make it "nice"
2. **Solve Problem**
solve problem on nice graph
3. **Clean Up**
deal with modifications



Length-Constrained Expander Decompositions

Graph Decomposition Approach

1. **Length-Constrained Expander Decomposition**
add to graph **edge length increases**
to make it **a length-constrained expander**
2. **Solve Problem**
solve problem on nice graph
3. **Clean Up**
deal with modifications



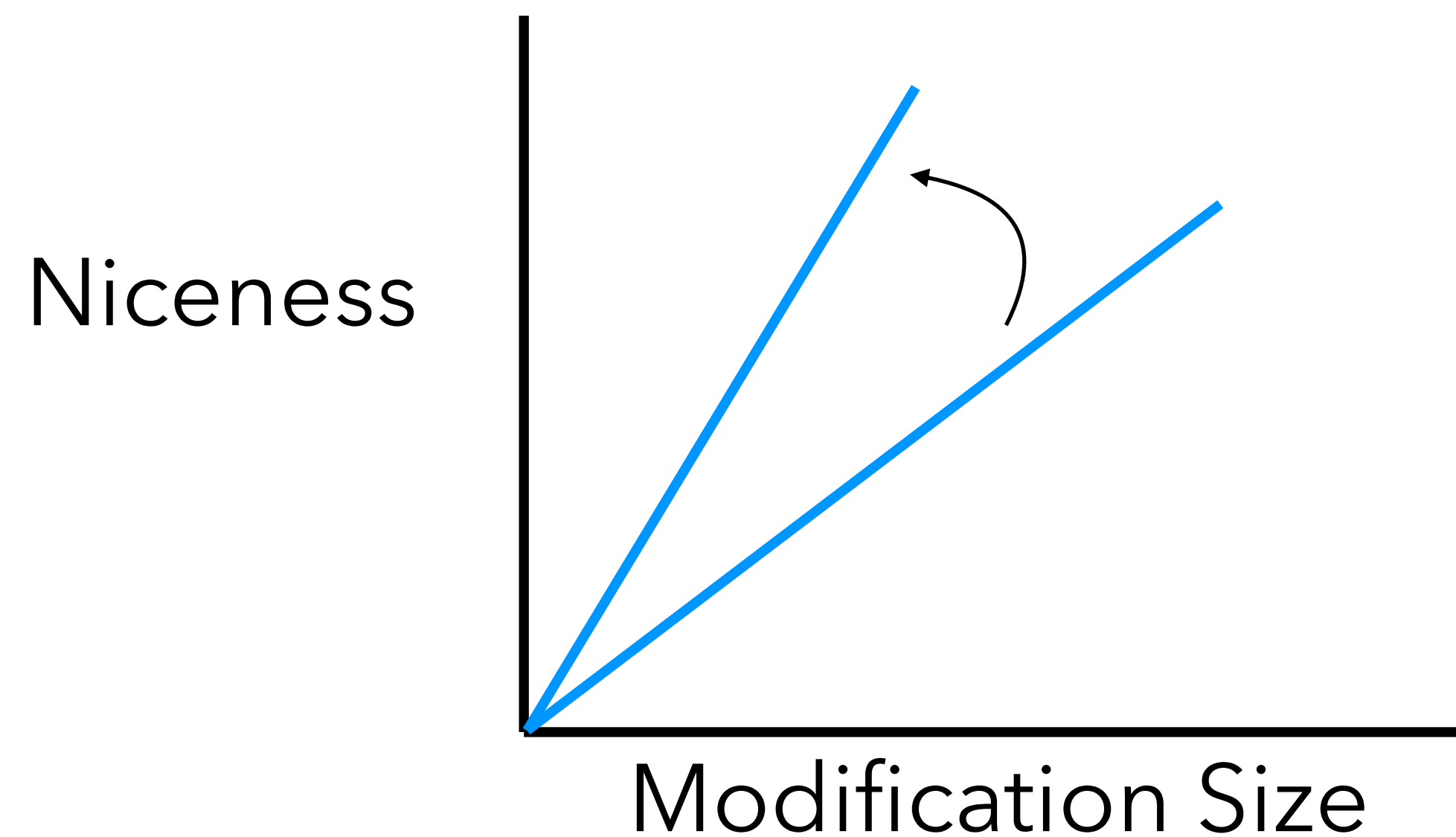
Length-Constrained Expander Decompositions

Graph decomposition approach with LC EDs gives SOTA for:

- Approximate Min-Cost Multi-Commodity Flow [HHLRS STOC24]
- Deterministic Distance Oracles [HLS FOCS24]
- $(1 + \epsilon)$ -Approximate Parallel Min Cost Flow [HJLSW FOCS25]

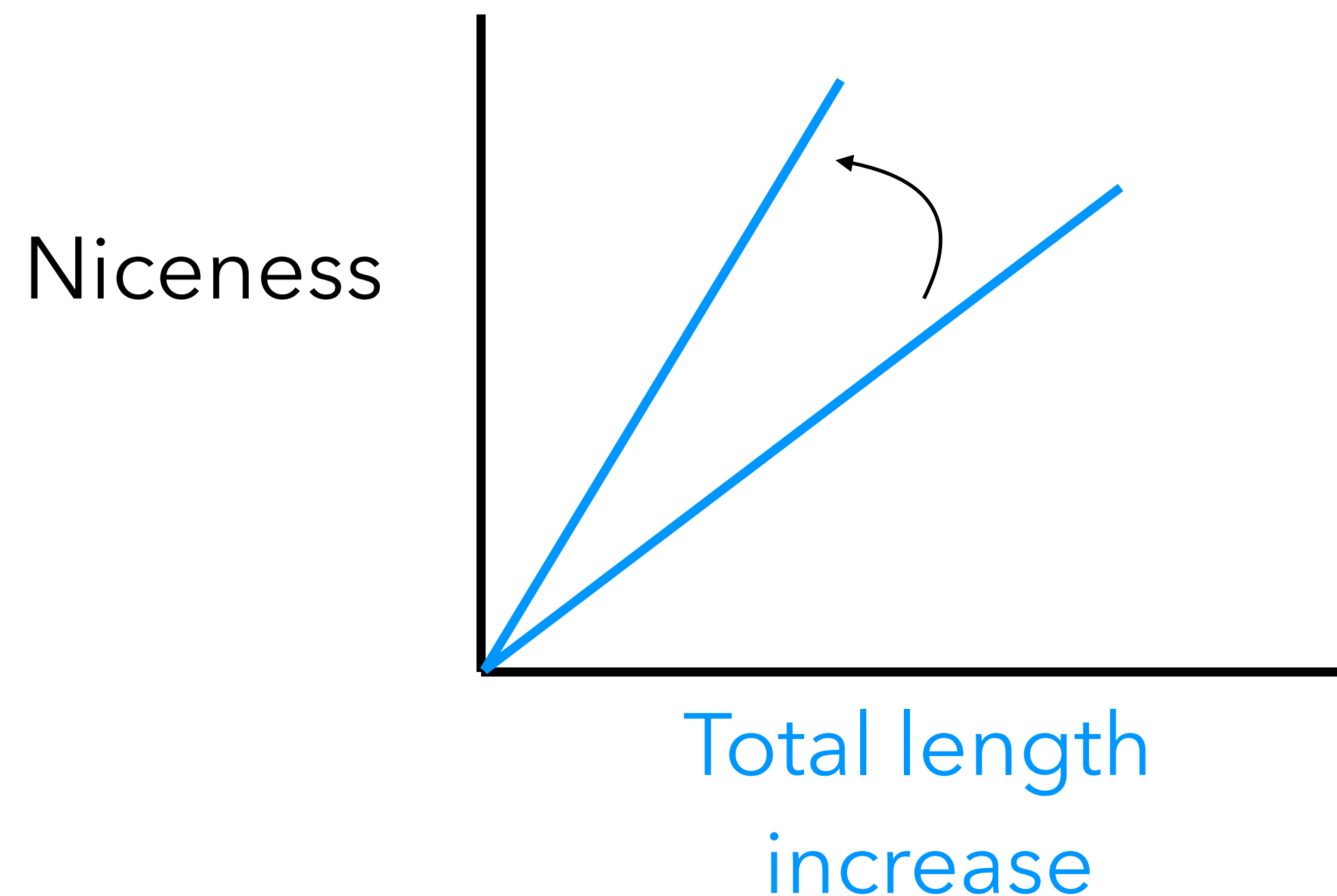
Our Main Result (Informally)

Theorem [BHHT]. Simple proof of the existence of length-constrained expander decompositions with improved size-niceness tradeoffs



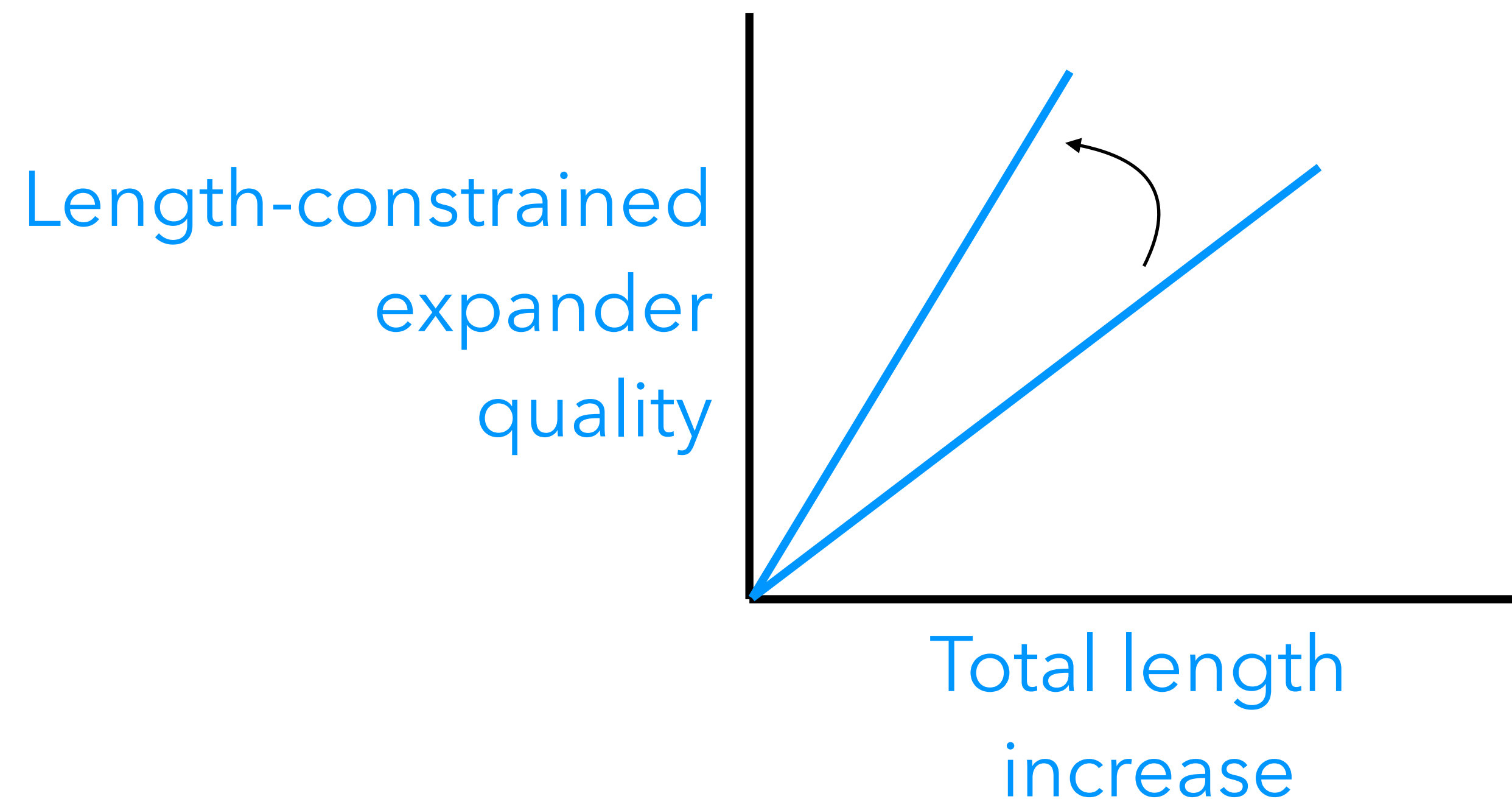
Our Main Result (Informally)

Theorem [BHHT]. Simple proof of the existence of length-constrained expander decompositions with improved size-niceness tradeoffs



Our Main Result (Informally)

Theorem [BHHT]. Simple proof of the existence of length-constrained expander decompositions with improved size-niceness tradeoffs

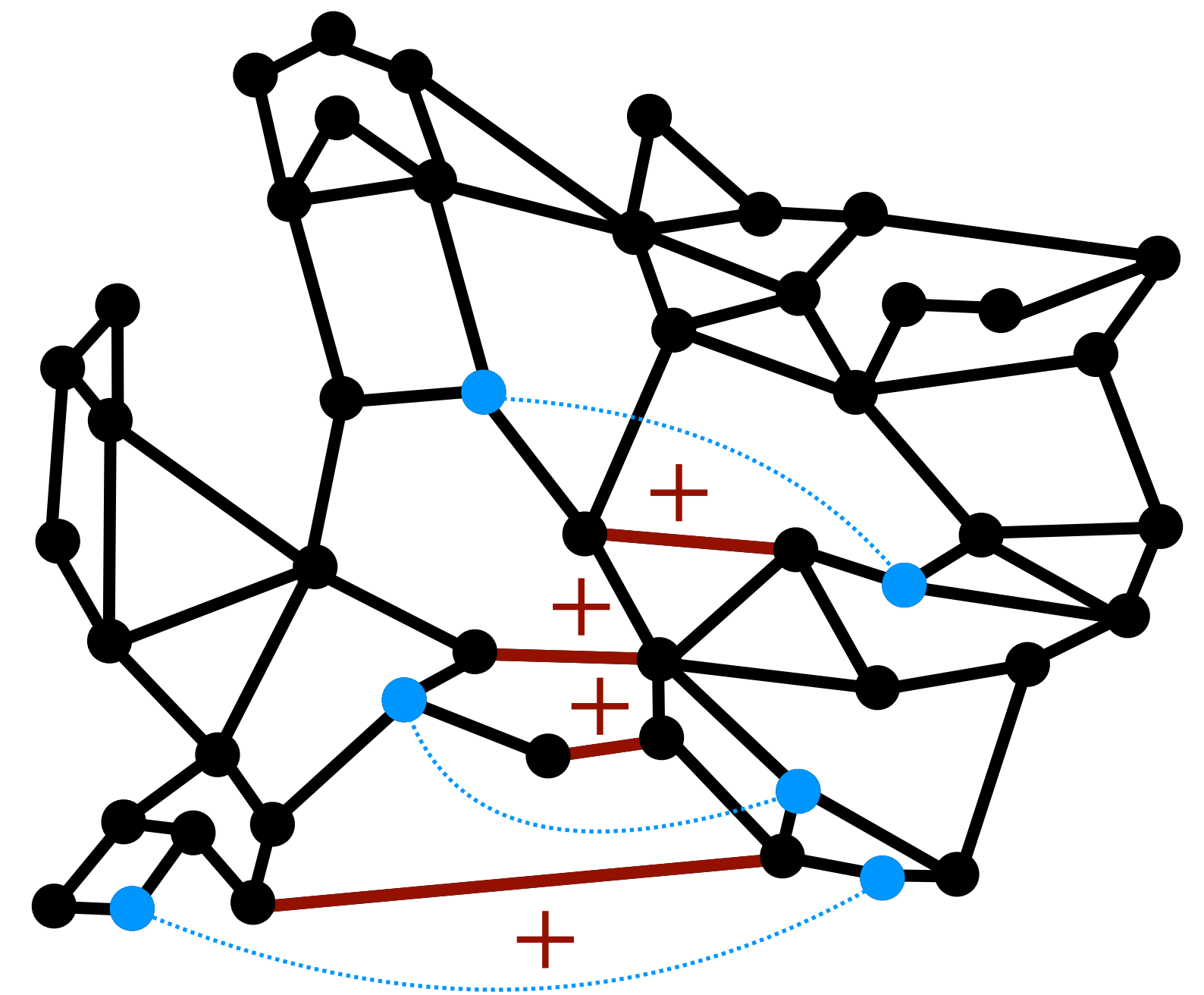


Defining Length-Constrained EDs

Sparse Length-Constrained Cuts

Sparse LC Cut Informally.

small total length increase that makes
many close vertex pairs
far



Sparse LC Cut

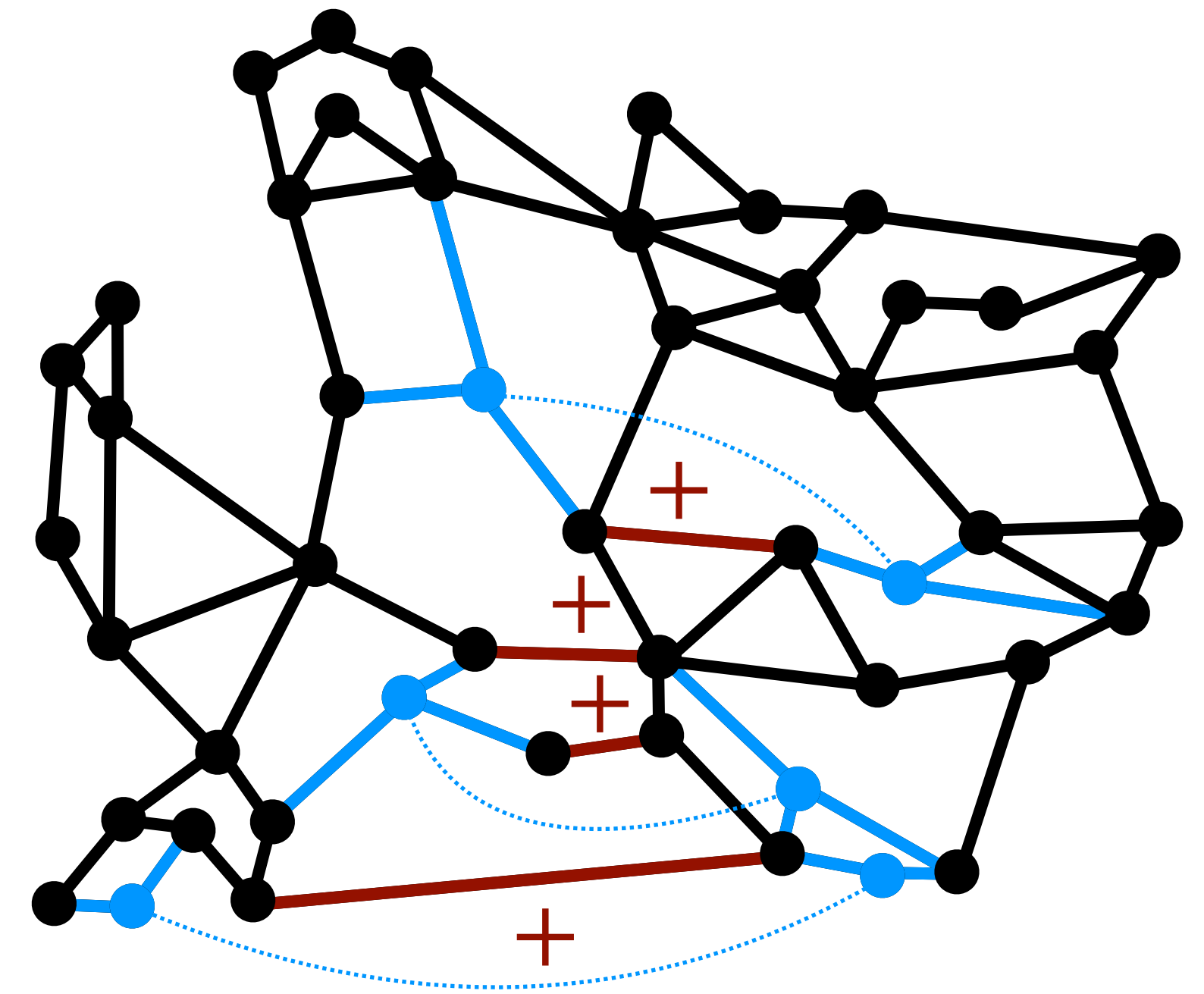
Sparse Length-Constrained Cuts

Sparse LC Cut Informally.

small total length increase that makes
many close vertex pairs
far

(h, s) -Length ϕ -Sparse Cut Formallyish.

X total length increase for some X that makes
some h -near disjoint vertex pairs w/ degree X/ϕ
at least hs -far



Sparse LC Cut
 X length increases
 X/ϕ total degree

Note. Any (h, s) -length ϕ -sparse cut has size (i.e. X) at most $\sim \phi m$ (since $\sim X/\phi \leq m$)

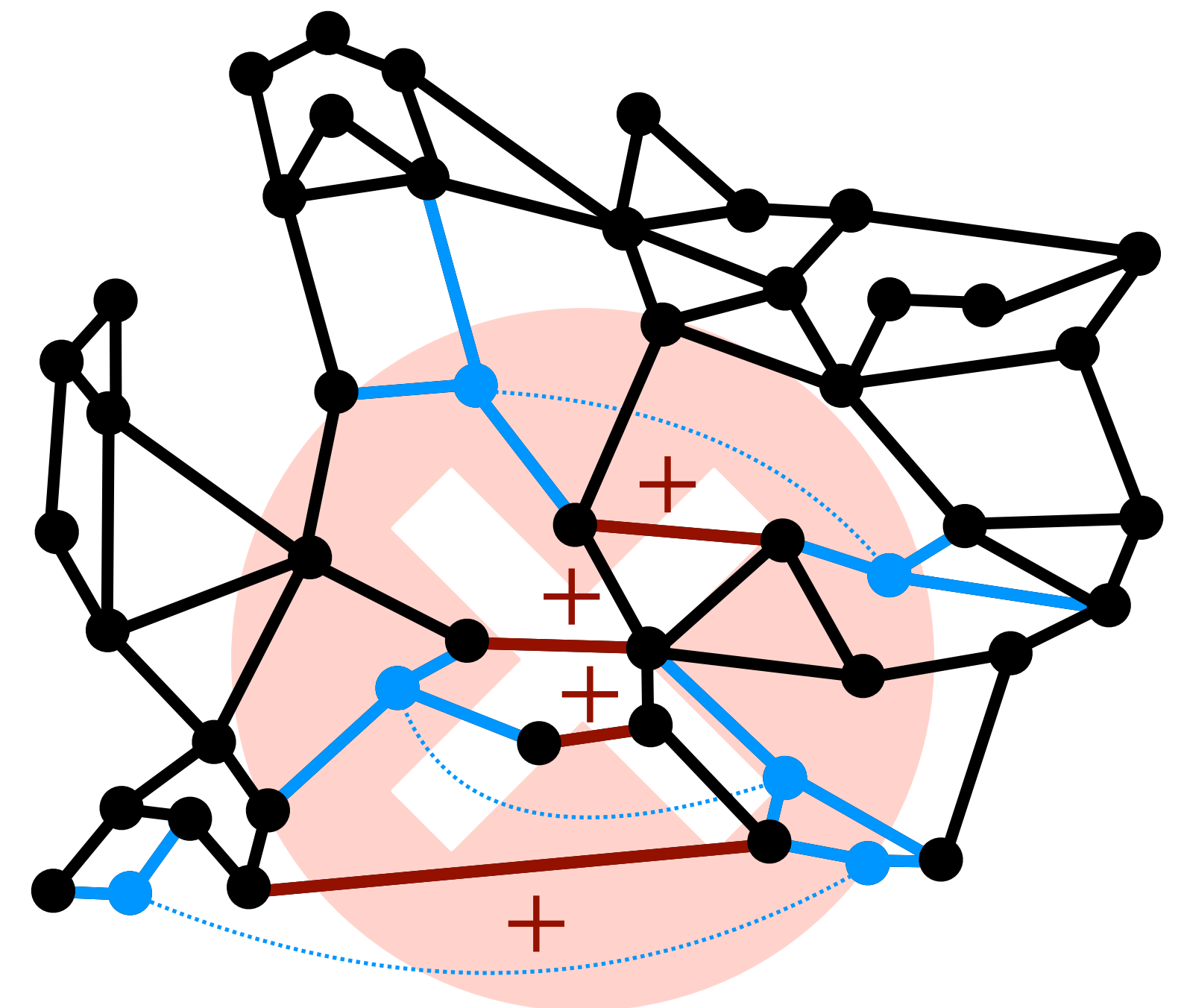
Length-Constrained Expanders

Length-Constrained Expanders Informally.

hard to make
nearby nodes
far

(h, s) -Length ϕ -Expanders Formallyish.

no (h, s) -length
 ϕ -sparse
cuts



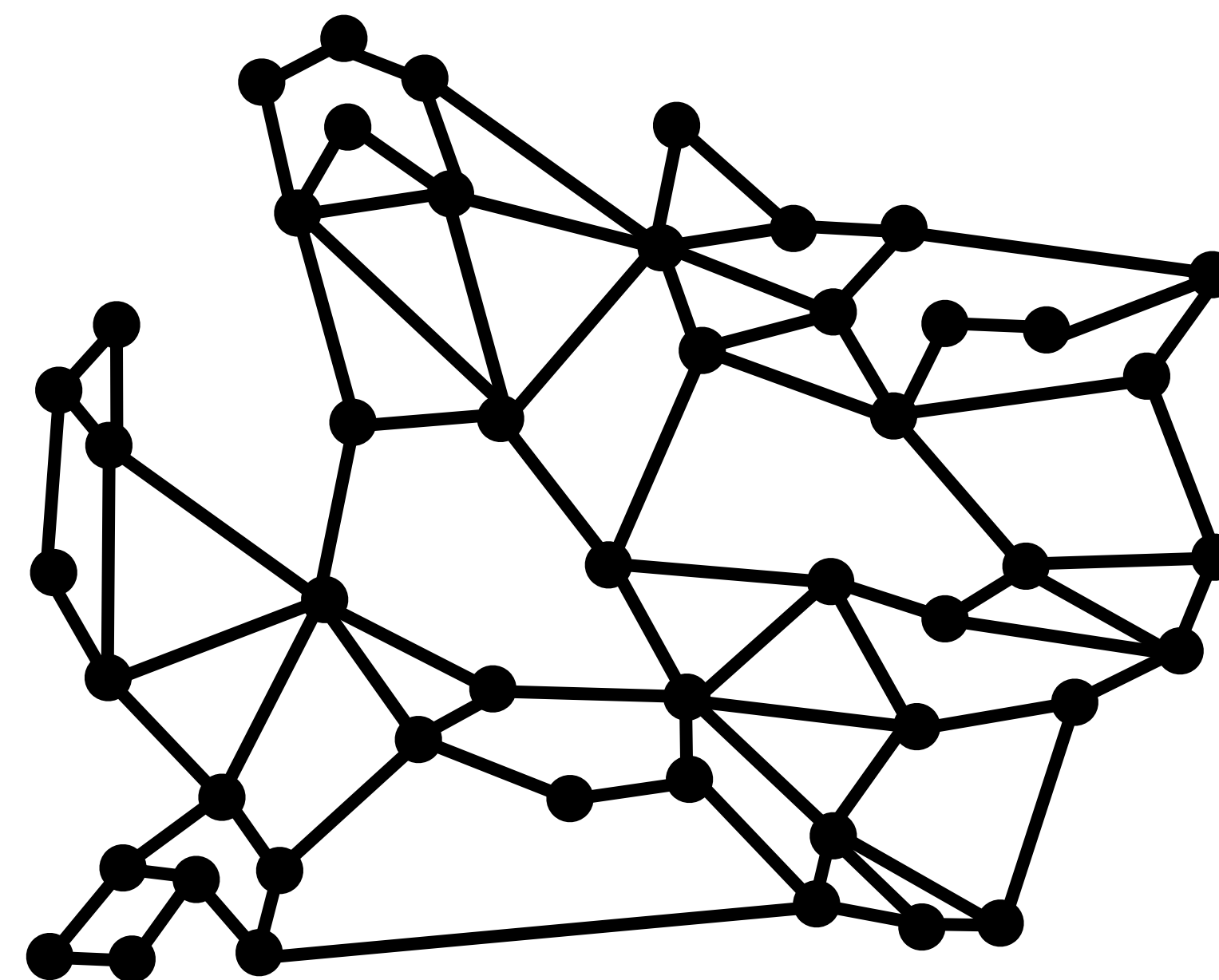
LC Expander

Flow View Informally. Easy for nearby nodes to send flow over short paths

Length-Constrained Expander Decompositions

(h, s) -Length ϕ -Expander Decomposition

length increases that
make graph
an (h, s) -length ϕ -expander

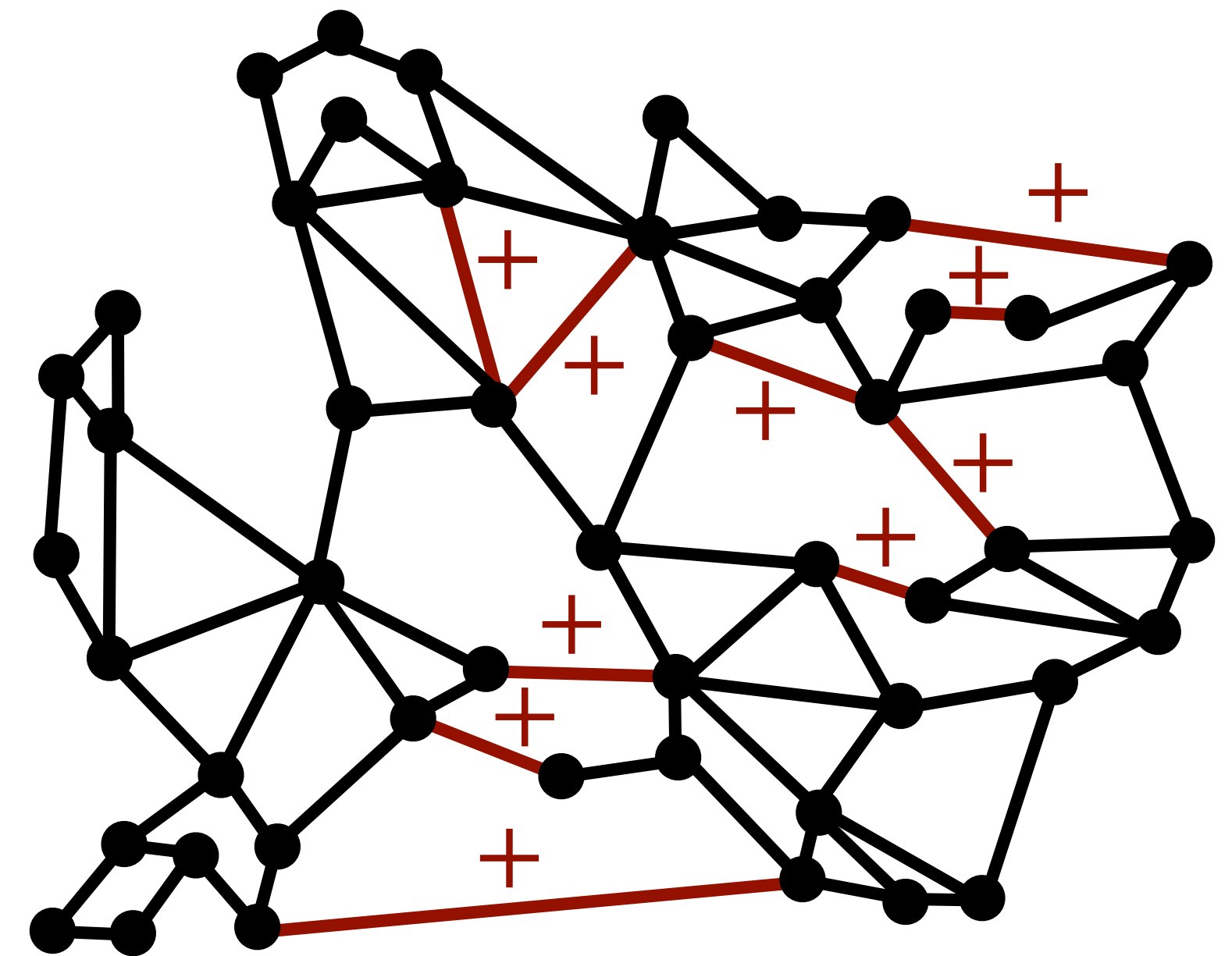


Arbitrary Graph

Length-Constrained Expander Decompositions

(h, s) -Length ϕ -Expander Decomposition

length increases that
make graph
an (h, s) -length ϕ -expander

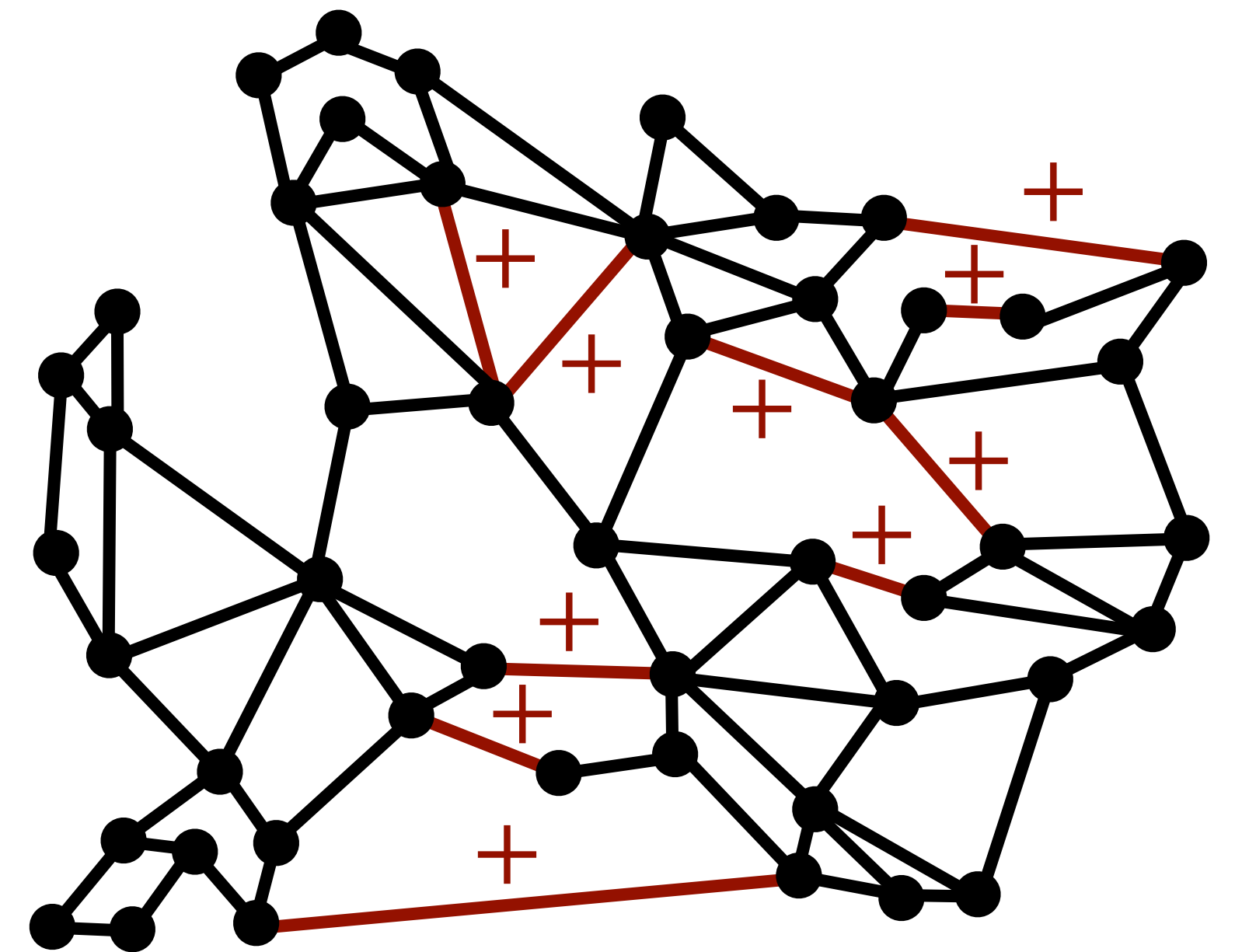


Arbitrary Graph

Length-Constrained Expander Decompositions

(h, s) -Length ϕ -Expander Decomposition

length increases that
make graph
an (h, s) -length ϕ -expander



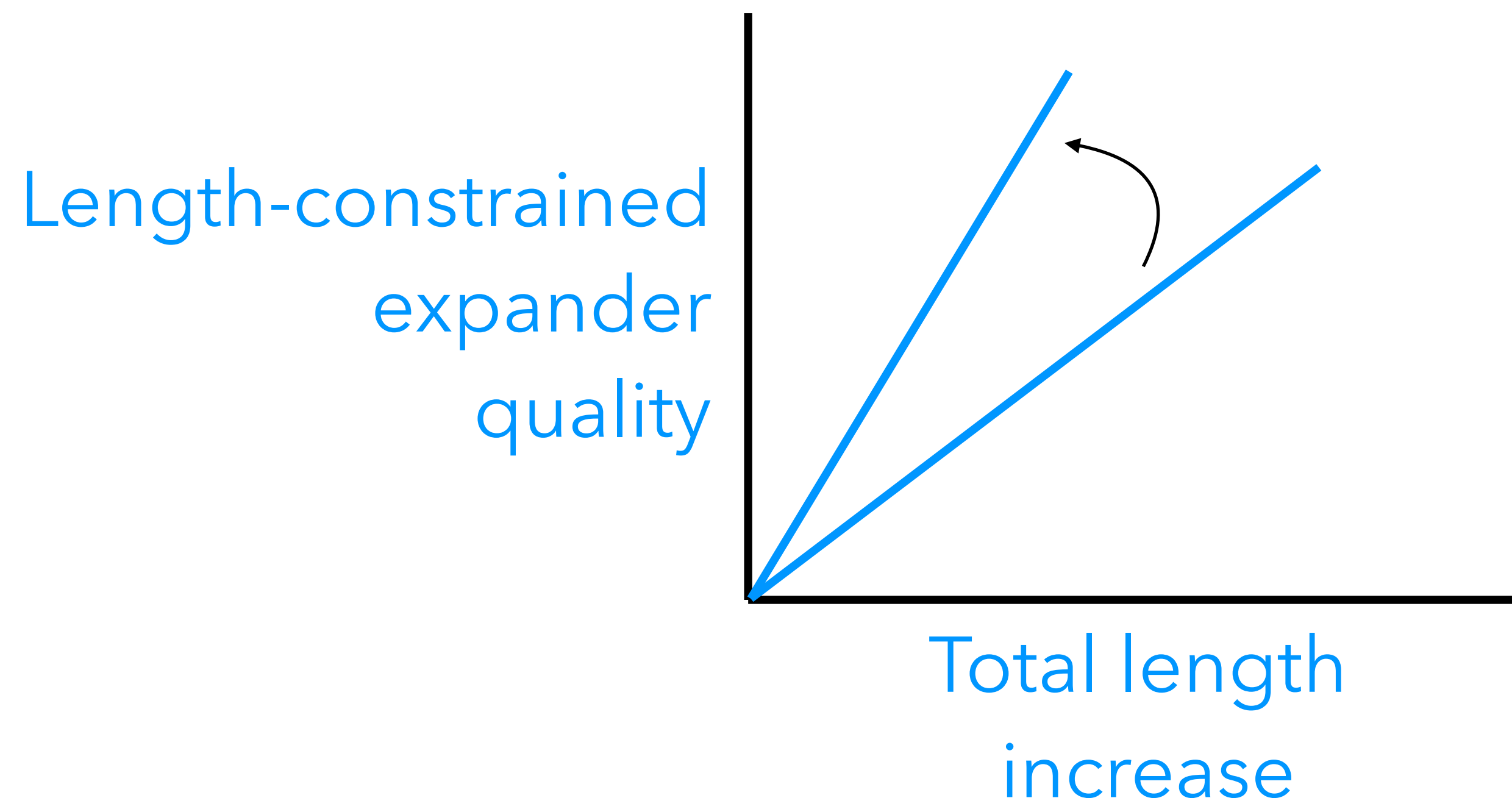
(h, s) -Length ϕ -Expander

Our Result

Our Main Result (Formally)

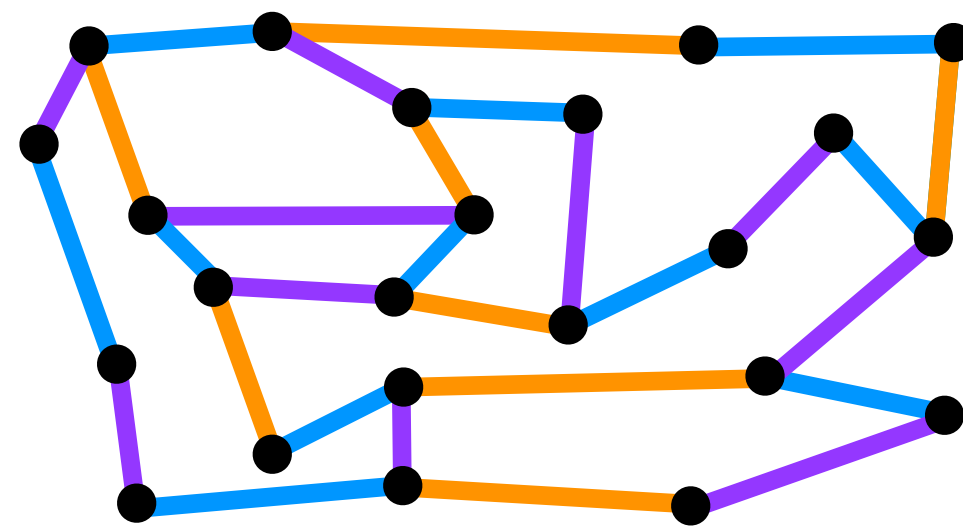
Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

Previously [HHT]. $\log n \cdot s \cdot n^{O(1/s)} \cdot \phi m$ (proven not simply)



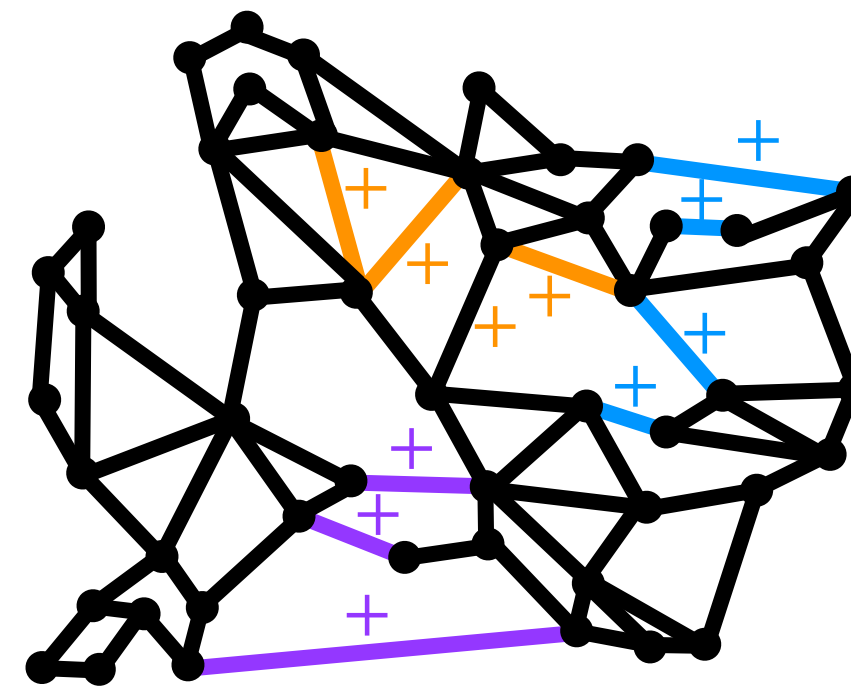
Proof Sketch of Our Result

Outline



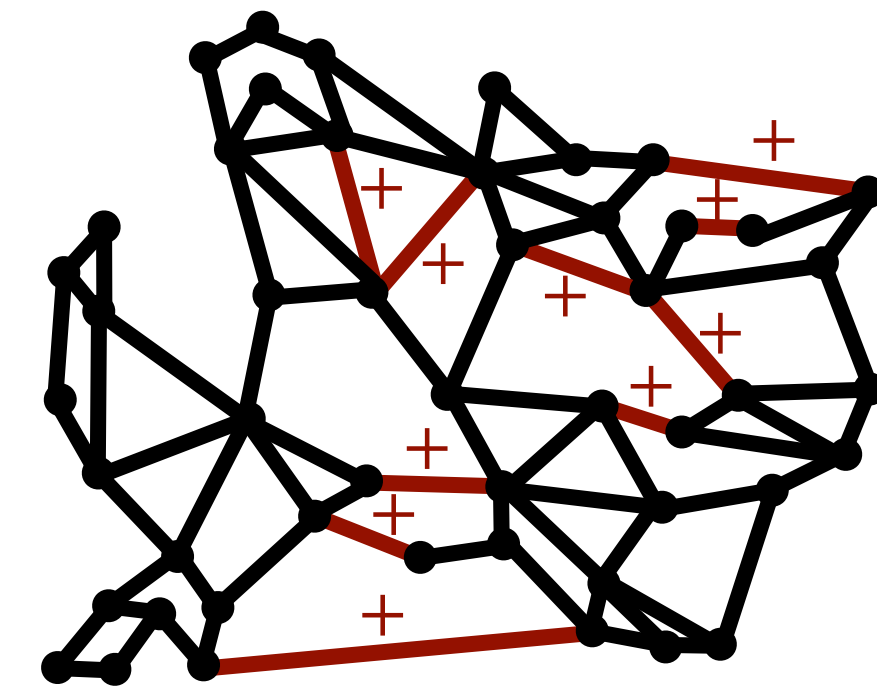
Parallel Greedy
Arboricity

Easy
→
(prior)



U of Cuts

Easy
→



Existence of LC
Decompositions

From \cup of Cuts to LC Expander Decompositions

Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

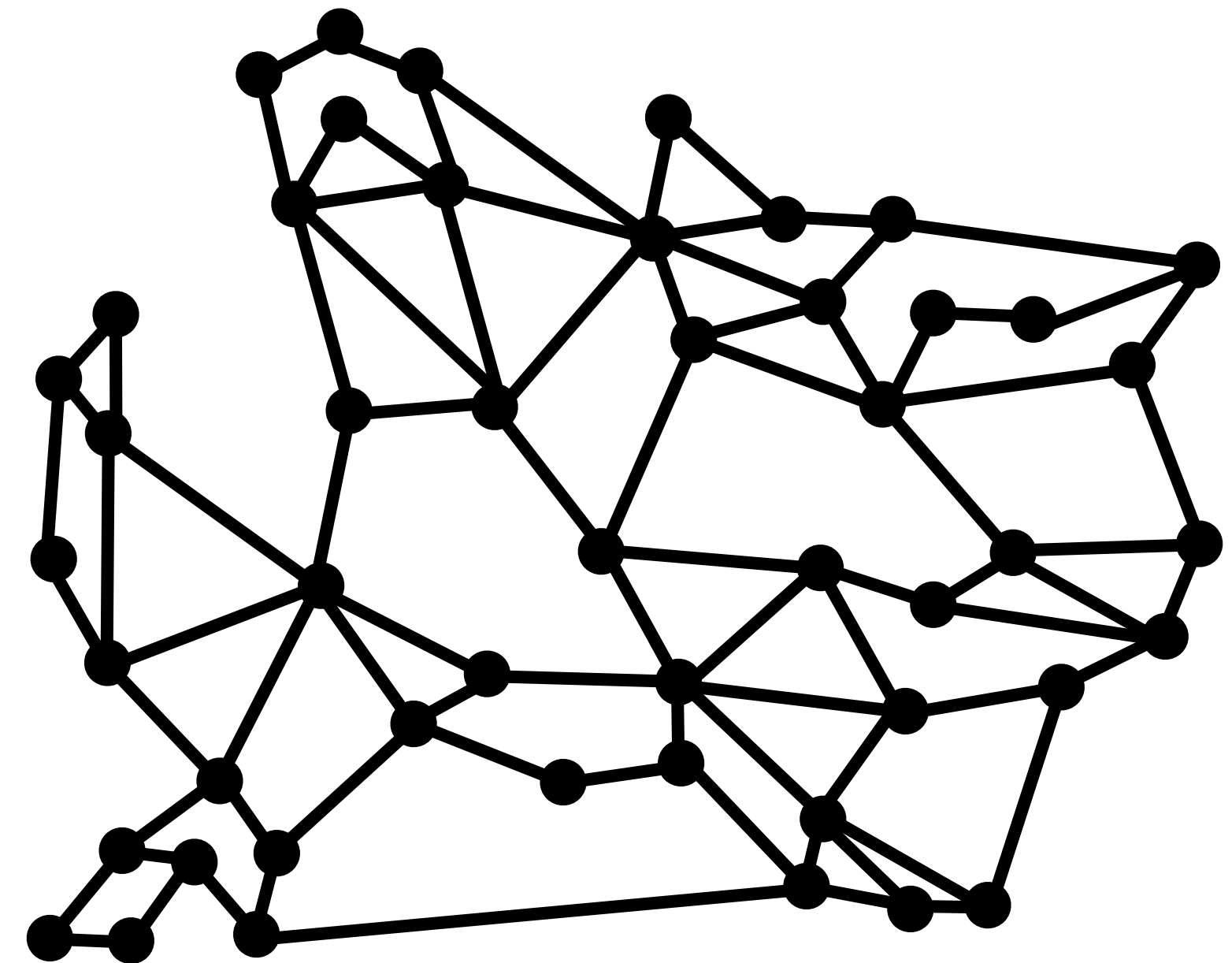
$i \leftarrow 0, G_0 \leftarrow G$

While G_i has an (h, s) -length ϕ -sparse cut C_i

$G_{i+1} \leftarrow G_i$ with C_i applied

$i \leftarrow i + 1$

Return $C = \sum_i C_i$



From \cup of Cuts to LC Expander Decompositions

Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

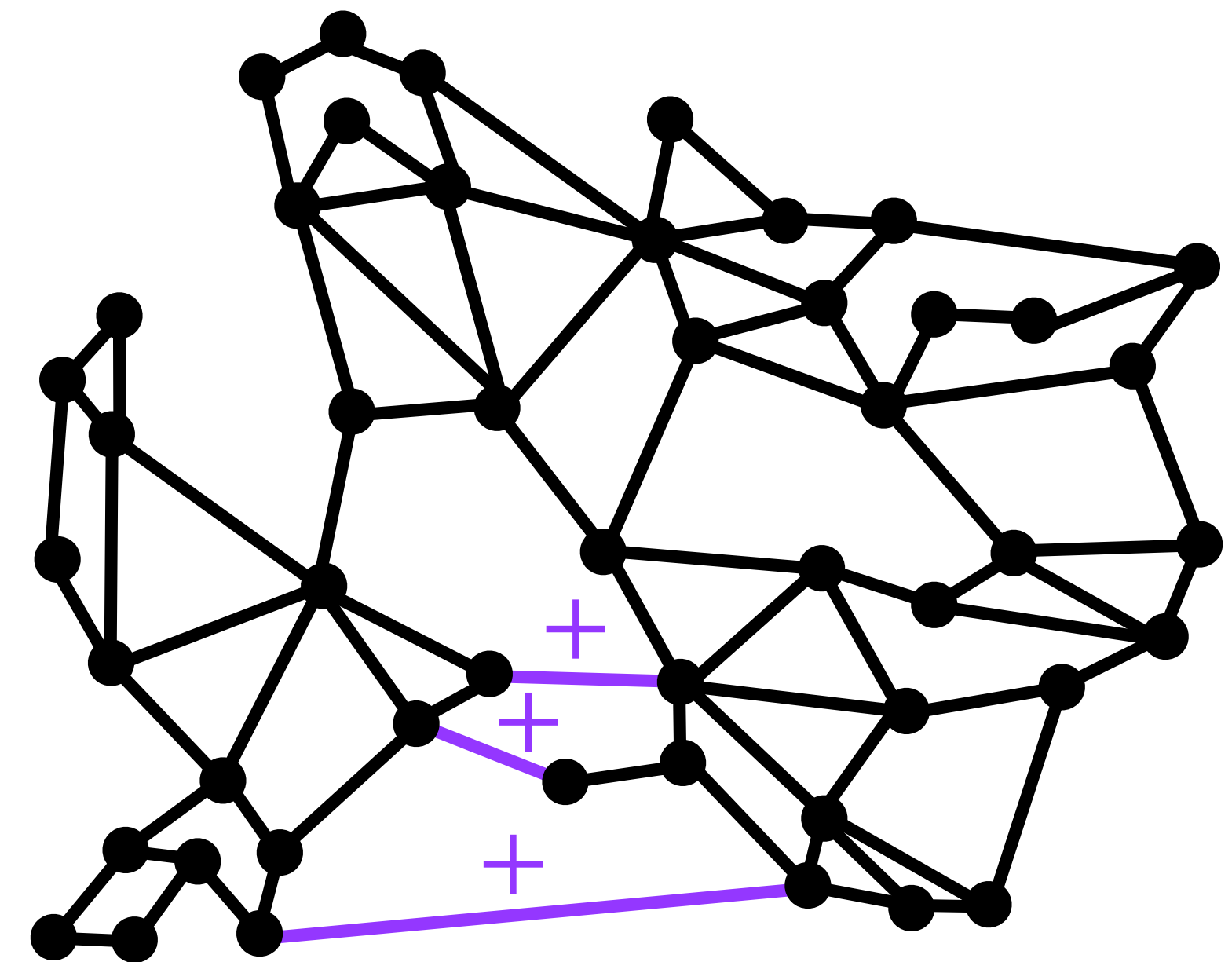
$i \leftarrow 0, G_0 \leftarrow G$

While G_i has an (h, s) -length ϕ -sparse cut C_i

$G_{i+1} \leftarrow G_i$ with C_i applied

$i \leftarrow i + 1$

Return $C = \sum_i C_i$



From \cup of Cuts to LC Expander Decompositions

Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

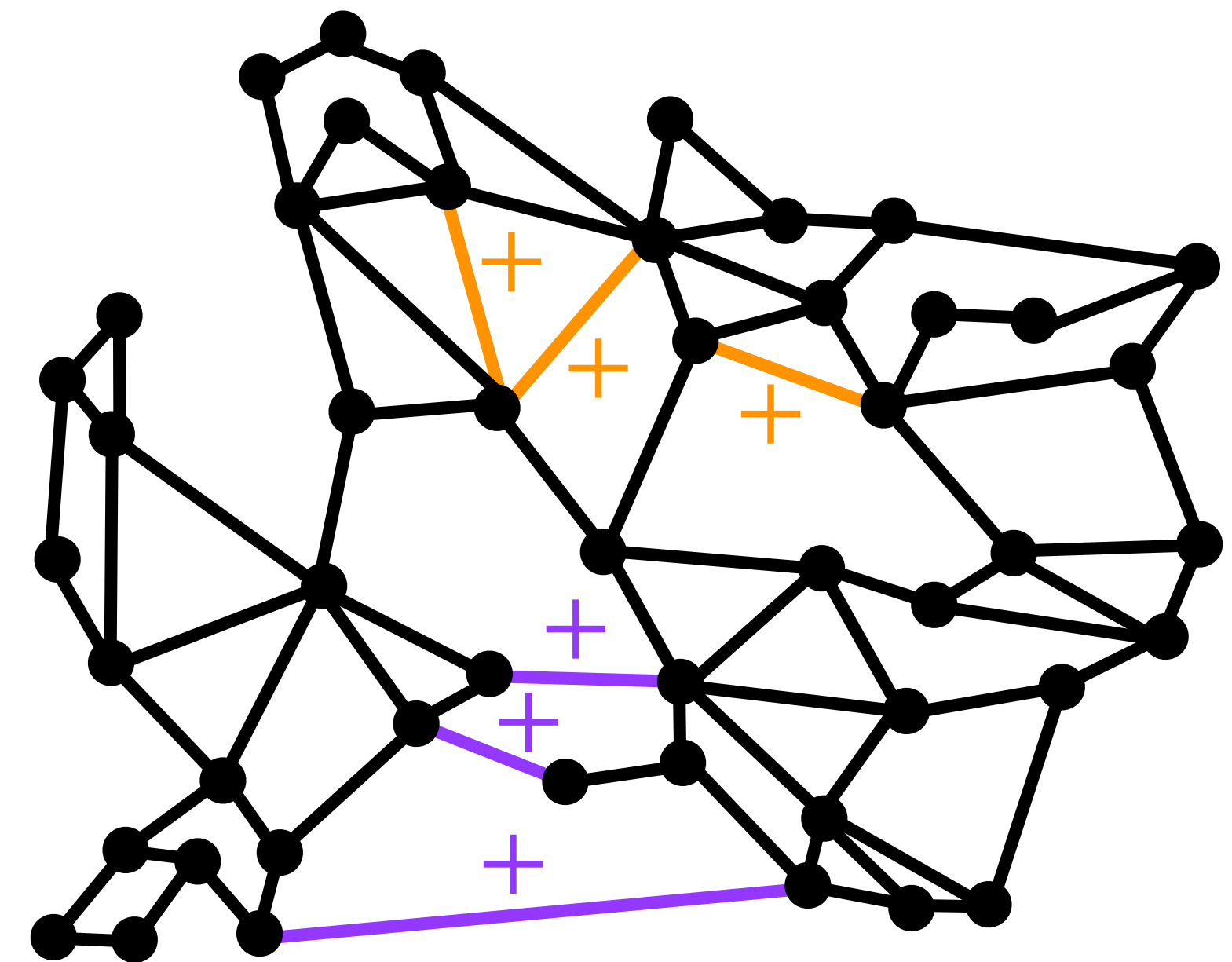
$i \leftarrow 0, G_0 \leftarrow G$

While G_i has an (h, s) -length ϕ -sparse cut C_i

$G_{i+1} \leftarrow G_i$ with C_i applied

$i \leftarrow i + 1$

Return $C = \sum_i C_i$



From \cup of Cuts to LC Expander Decompositions

Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

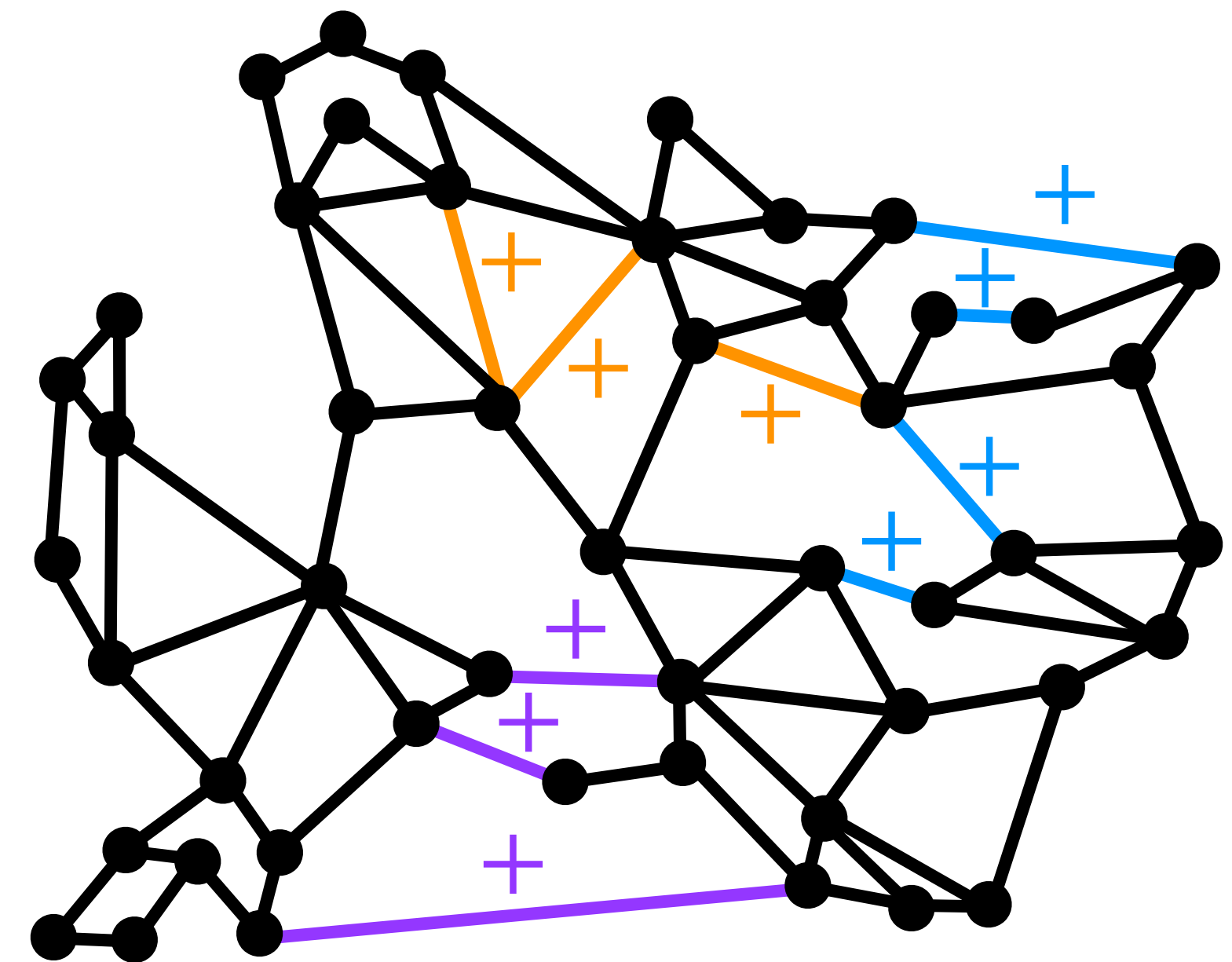
$i \leftarrow 0, G_0 \leftarrow G$

While G_i has an (h, s) -length ϕ -sparse cut C_i

$G_{i+1} \leftarrow G_i$ with C_i applied

$i \leftarrow i + 1$

Return $C = \sum_i C_i$



From \cup of Cuts to LC Expander Decompositions

Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

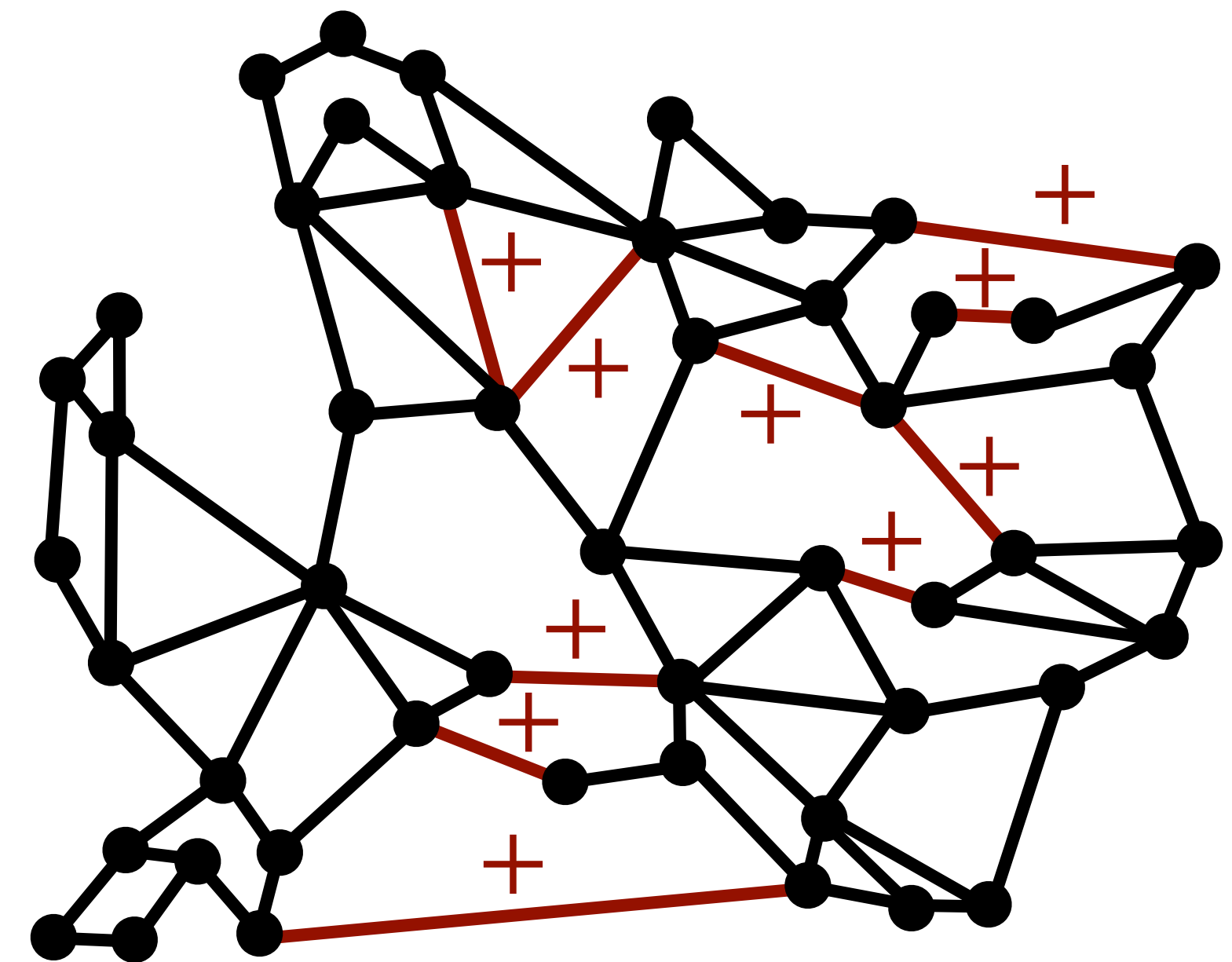
$i \leftarrow 0, G_0 \leftarrow G$

While G_i has an (h, s) -length ϕ -sparse cut C_i

$G_{i+1} \leftarrow G_i$ with C_i applied

$i \leftarrow i + 1$

Return $C = \sum_i C_i$



From \cup of Cuts to LC Expander Decompositions

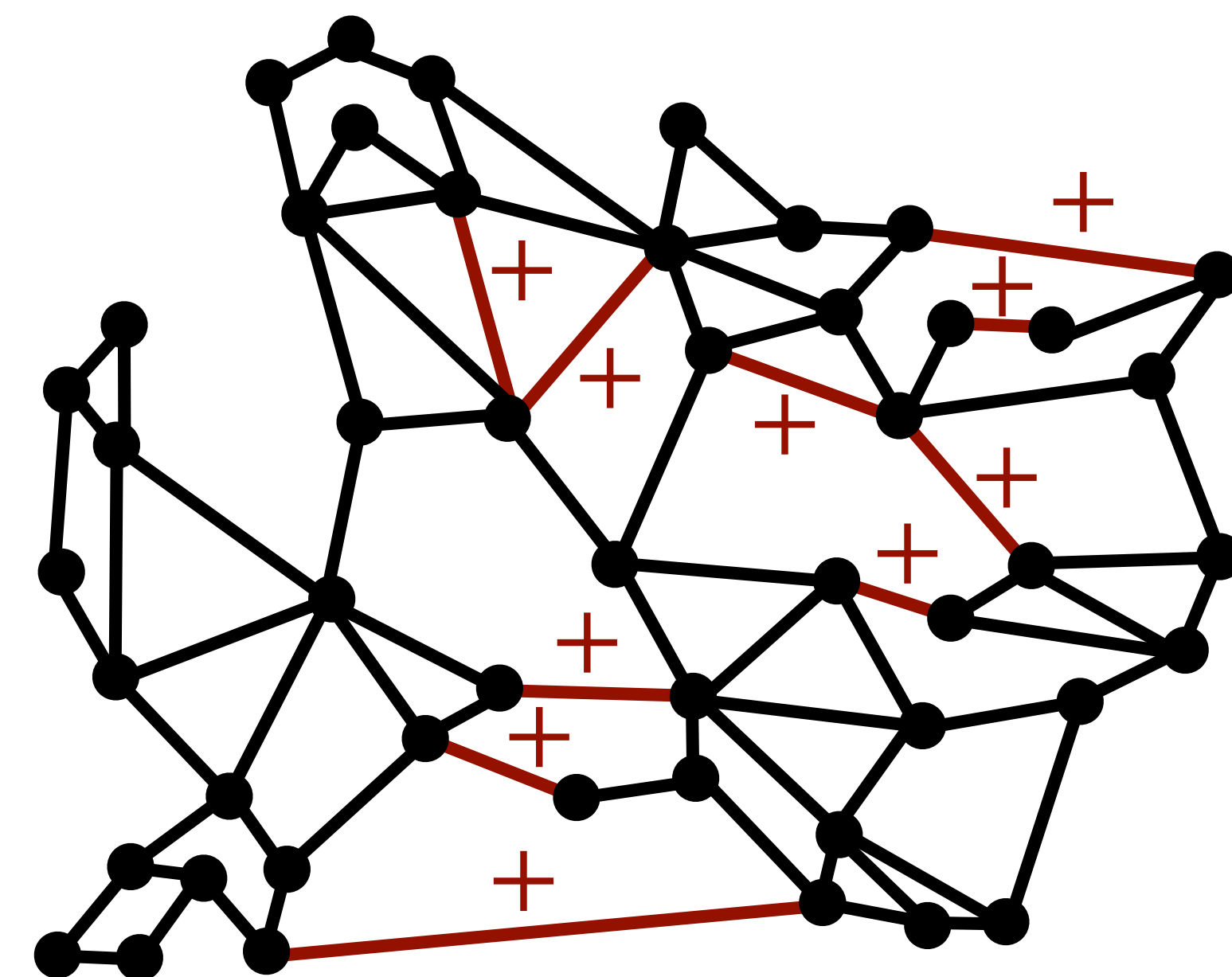
Theorem [BHHT]. Any graph G has an (h, s) -length ϕ -expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

... Return $C = \sum_i C_i$

C is an (h, s) -length ϕ -ED (no sparse cuts left)

🤔 But why is it small? 🤔

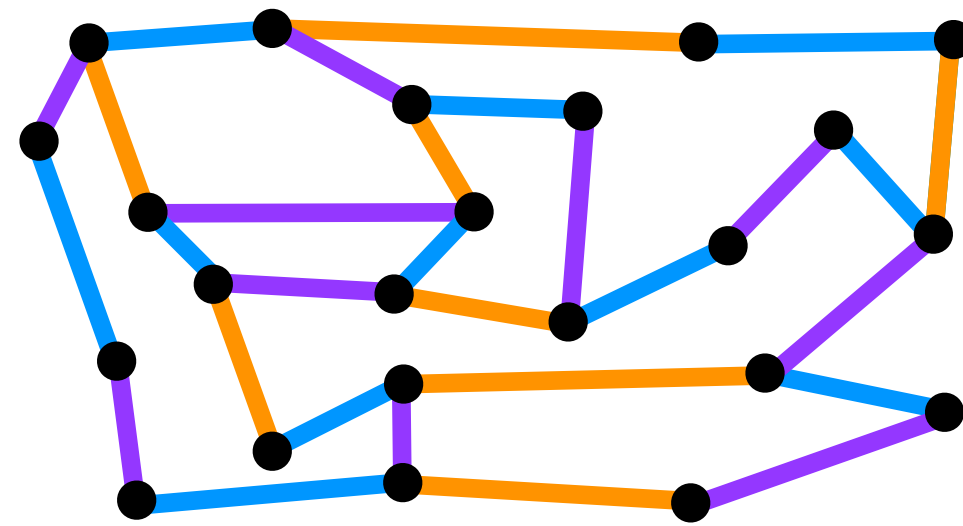
\cup of Cuts [BHHT]. C is an $\sim(h, s)$ -length $(sn^{O(1/s)} \cdot \phi)$ -sparse cut



Any (h, s) -length $(sn^{O(1/s)} \cdot \phi)$ -sparse cut has size at most $s \cdot n^{O(1/s)} \cdot \phi m$

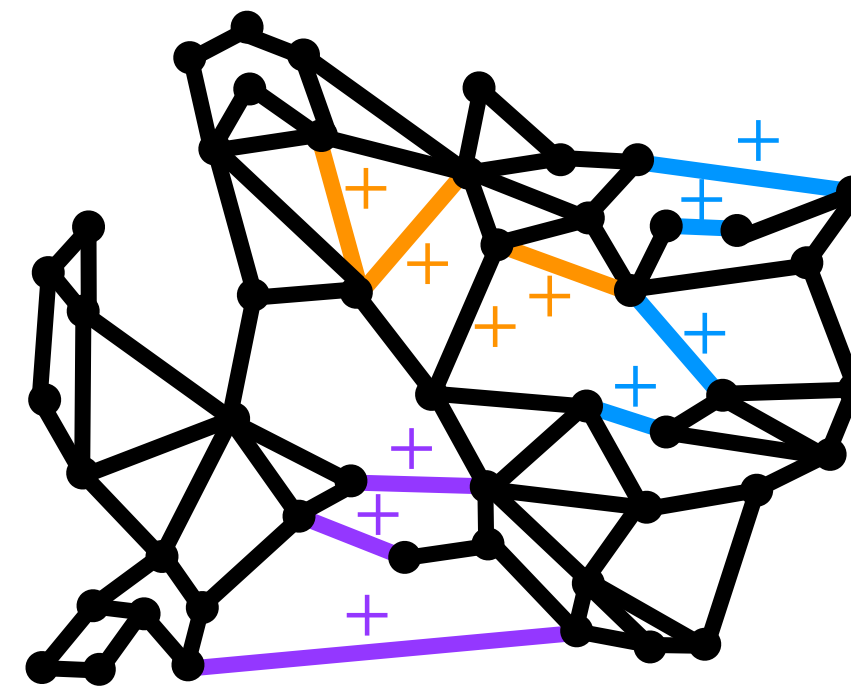
So C has size at most $s \cdot n^{O(1/s)} \cdot \phi m$

Outline



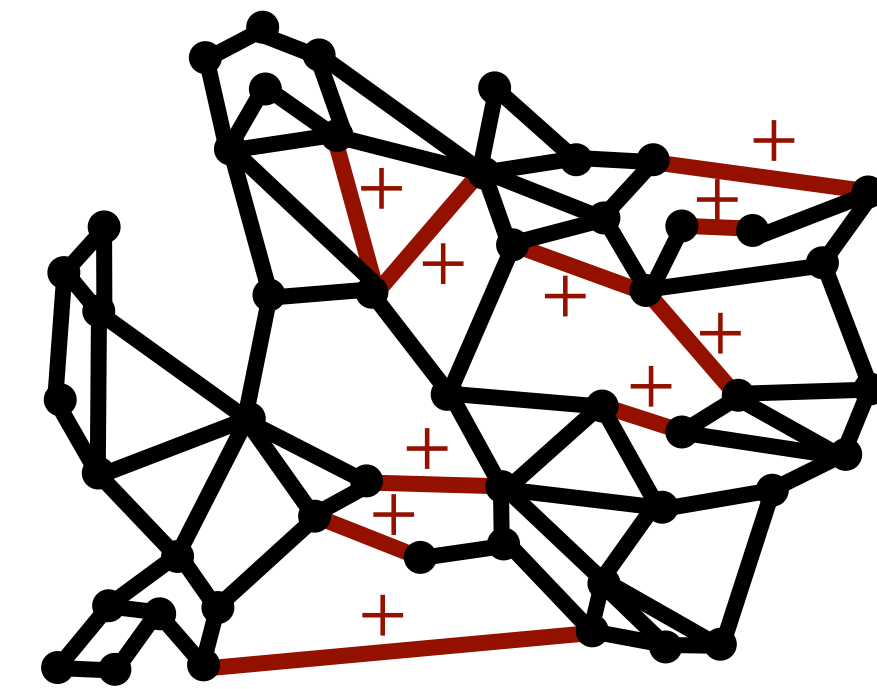
Parallel Greedy
Arboricity

Easy
→
(prior)



U of Cuts

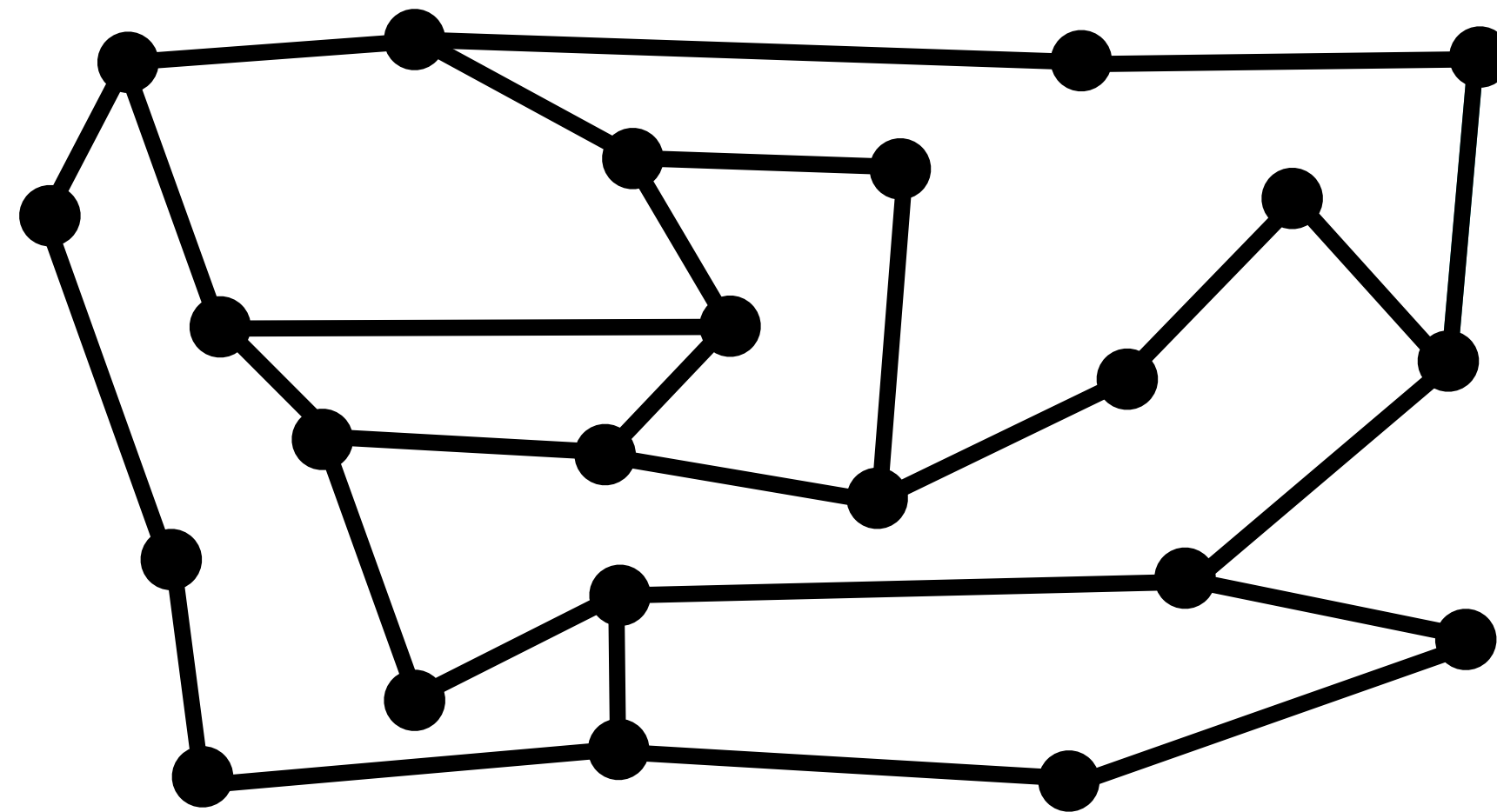
Easy
→
✓



Existence of LC
Decompositions

From Parallel Greedy Arboricity to \cup of Cuts

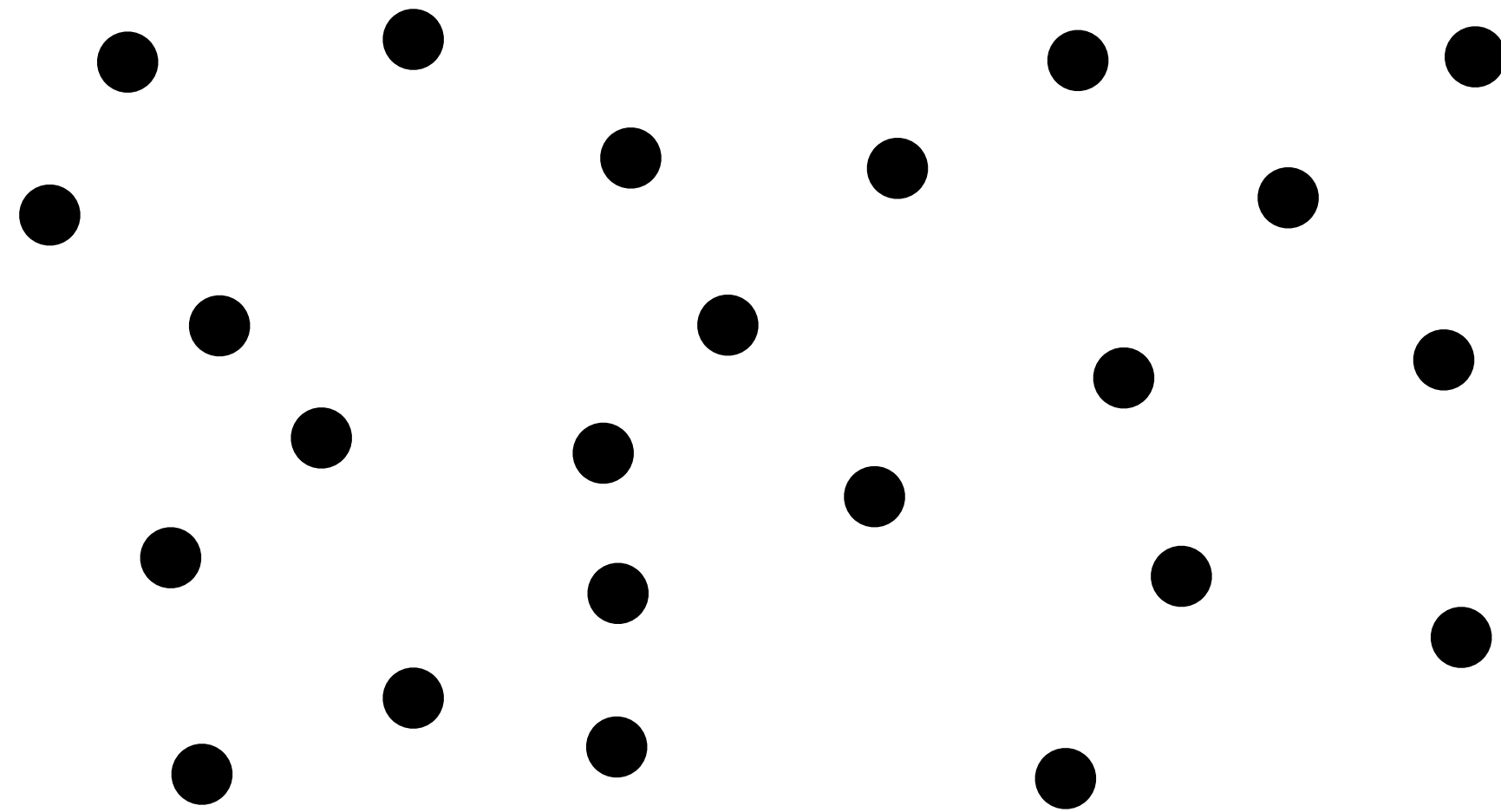
$G = (V, E)$ is an s -**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \dots$ where if $\{u, v\} \in M_i$ then u and v at least s -far in $\left(V, \bigcup_{j < i} M_j \right)$



12-parallel-greedy

From Parallel Greedy Arboricity to \cup of Cuts

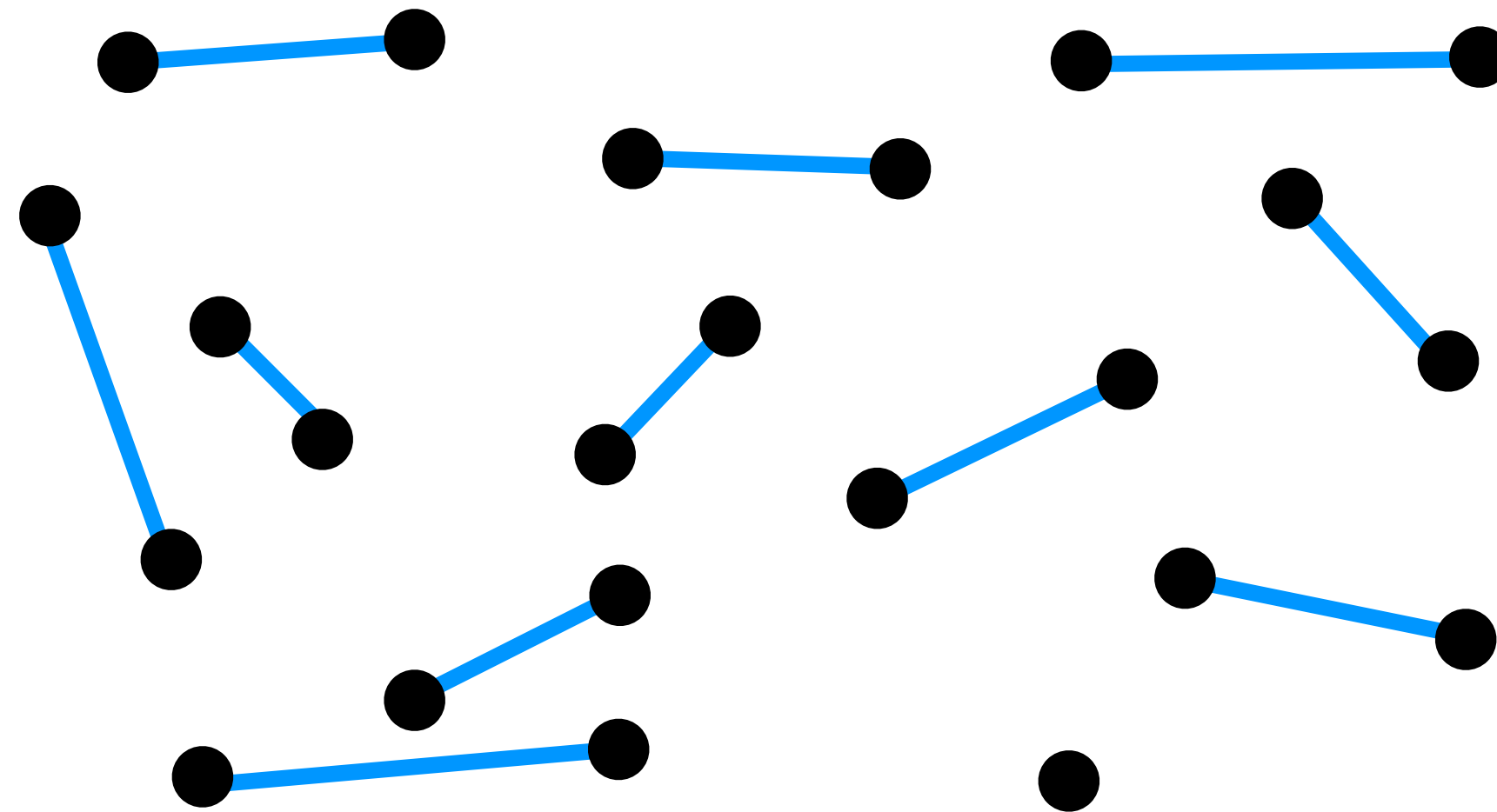
$G = (V, E)$ is an s -**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \dots$ where if $\{u, v\} \in M_i$ then u and v at least s -far in $\left(V, \bigcup_{j < i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to \cup of Cuts

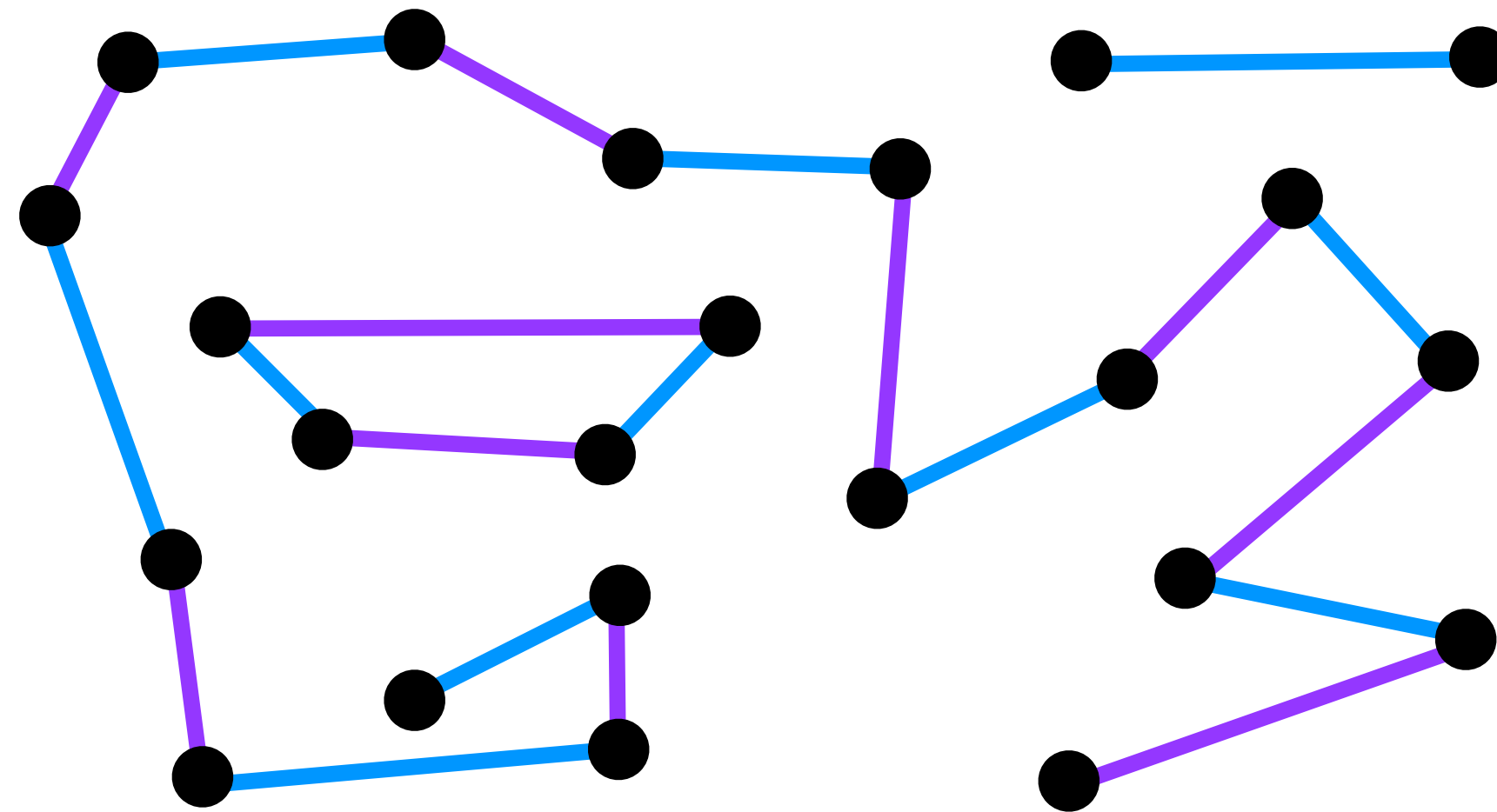
$G = (V, E)$ is an s -**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \dots$ where if $\{u, v\} \in M_i$ then u and v at least s -far in $\left(V, \bigcup_{j < i} M_j \right)$



12-parallel-greedy

From Parallel Greedy Arboricity to \cup of Cuts

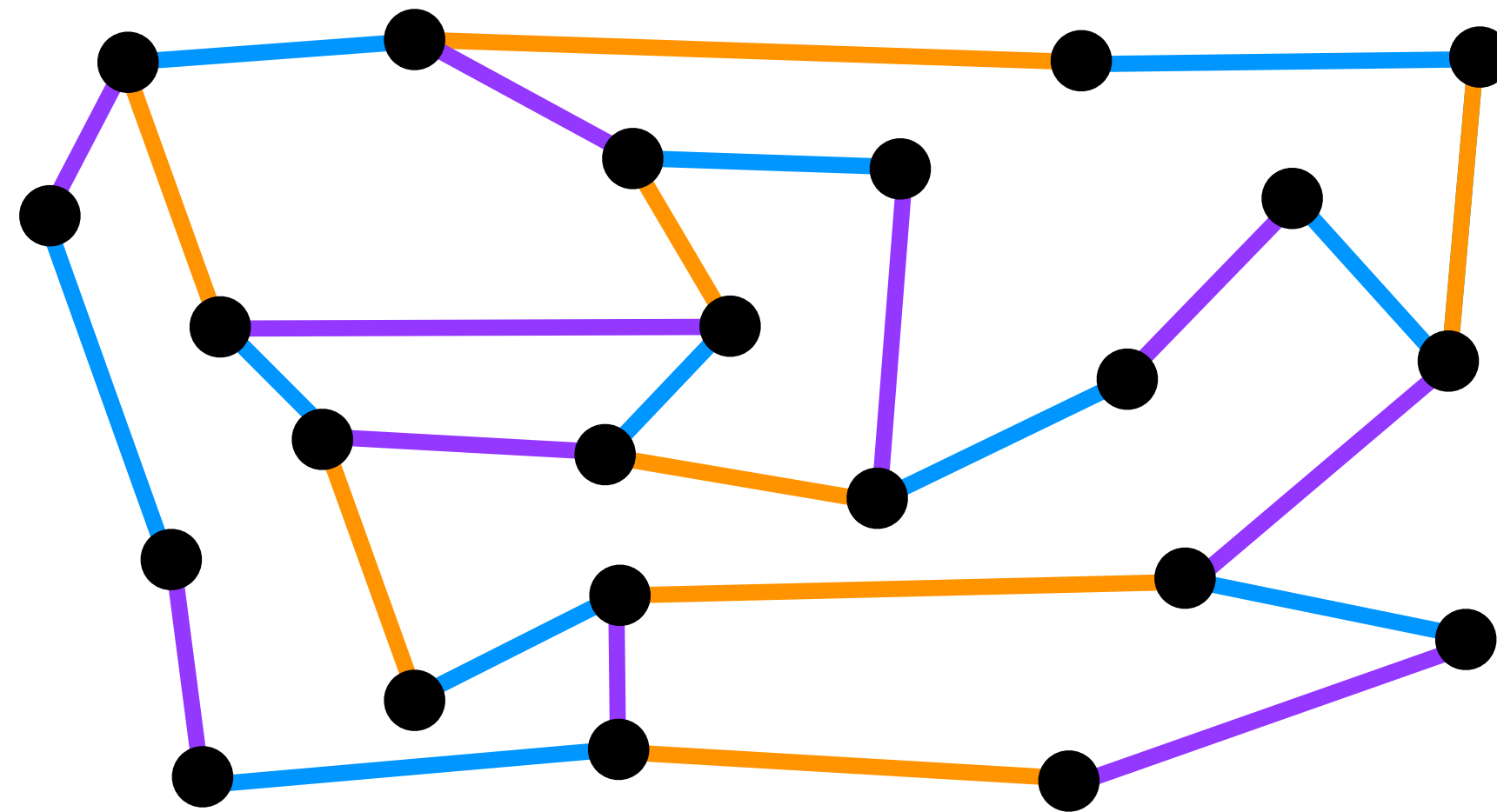
$G = (V, E)$ is an s -**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \dots$ where if $\{u, v\} \in M_i$ then u and v at least s -far in $\left(V, \bigcup_{j < i} M_j \right)$



12-parallel-greedy

From Parallel Greedy Arboricity to \cup of Cuts

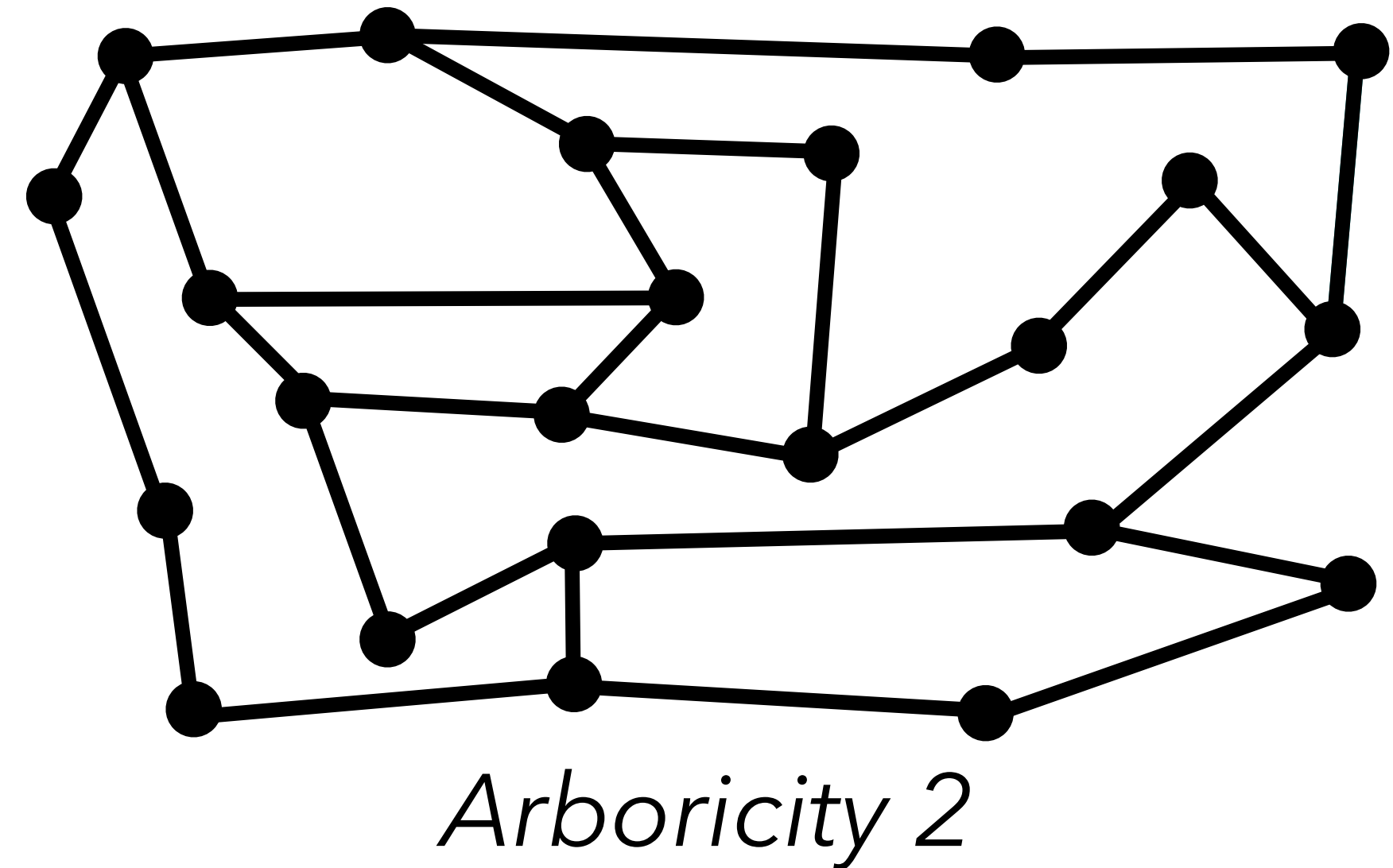
$G = (V, E)$ is an s -**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \dots$ where if $\{u, v\} \in M_i$ then u and v at least s -far in $\left(V, \bigcup_{j < i} M_j \right)$



12-parallel-greedy

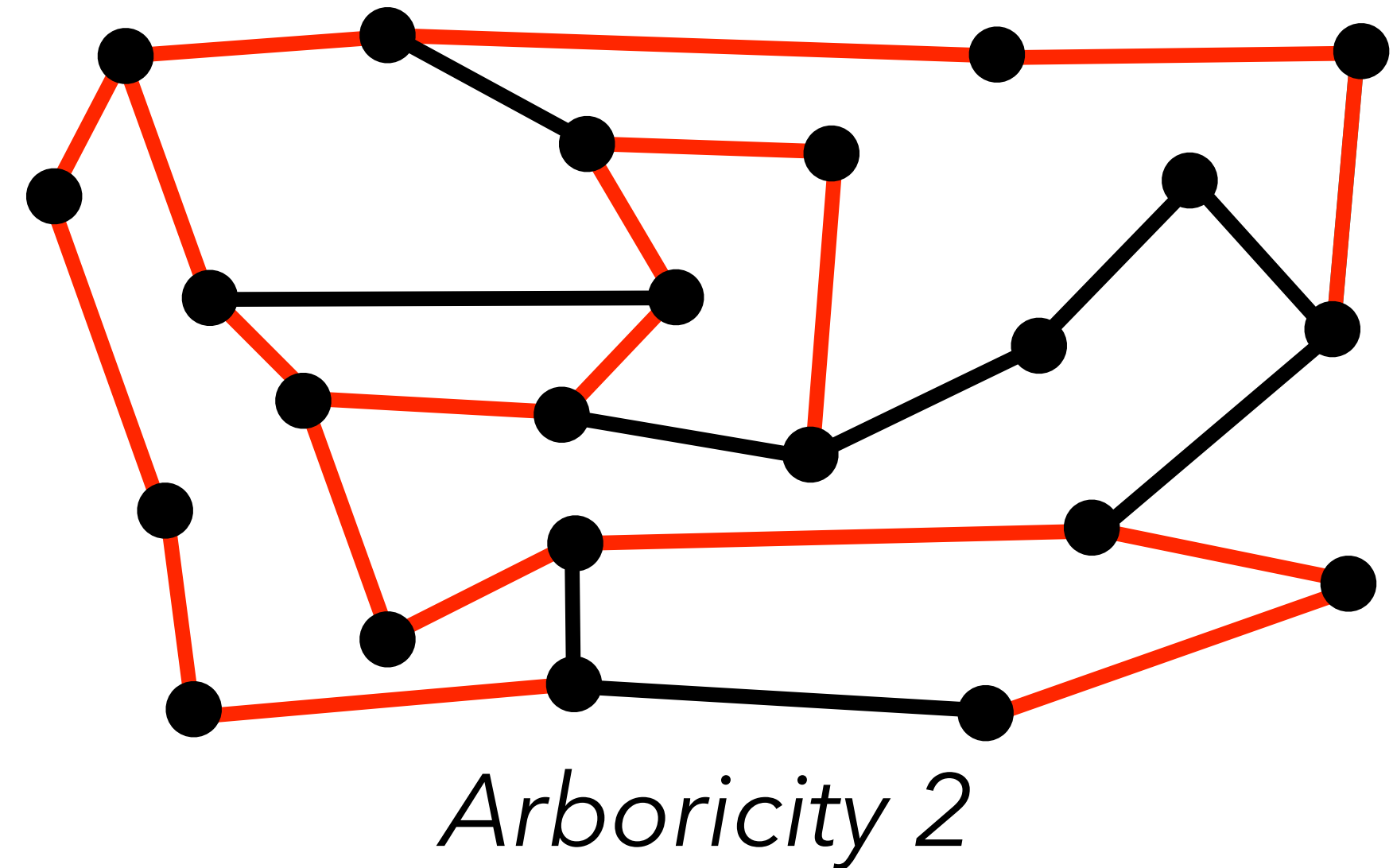
From Parallel Greedy Arboricity to \cup of Cuts

The **arboricity** of a graph is the minimum number of forests needed to cover all edges



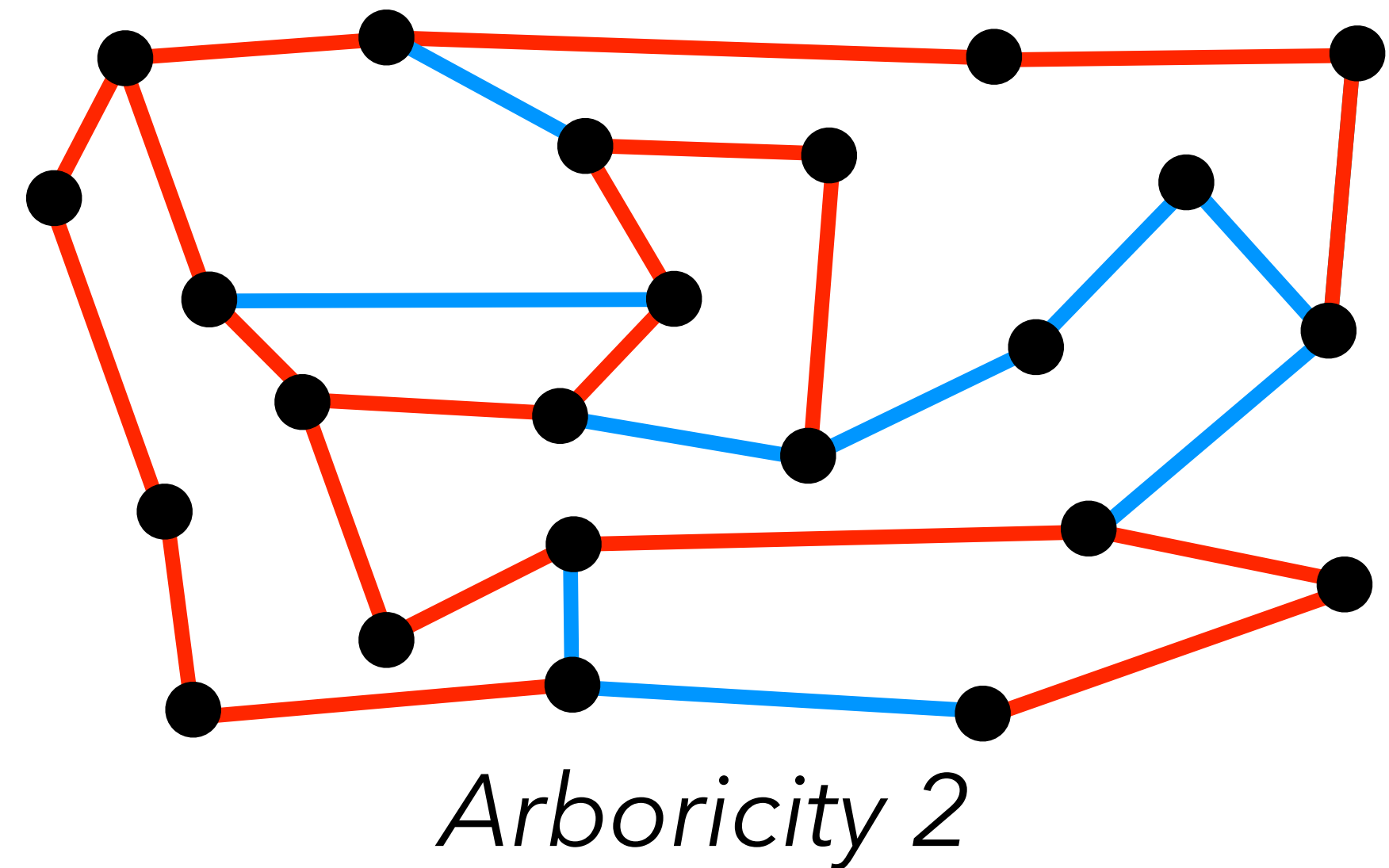
From Parallel Greedy Arboricity to \cup of Cuts

The **arboricity** of a graph is the minimum number of forests needed to cover all edges



From Parallel Greedy Arboricity to \cup of Cuts

The **arboricity** of a graph is the minimum number of forests needed to cover all edges



PG Arboricity to \cup of Cuts [HHT]. C is (h, s) -length $\alpha\phi$ -sparse in G where α is the arboricity of s -parallel-greedy graphs

From Parallel Greedy Arboricity to \cup of Cuts

Parallel Greedy Arboricity [BHHT]. $\alpha \leq s \cdot n^{O(1/s)}$
where α is the arboricity of s -parallel-greedy graphs

~3 pages based on “dispersion/counting” framework.

graph, serves to upper bound the arboricity by [Theorem 2.1](#). A similar framework has been used in recent work on graph spanners; for example, [Bod25; BHP24] use this framework over related (but more specific) types of paths.

For the rest of this section we assume we are given an n -node s -parallel greedy graph $G = (V, E)$ with m edges whose arboricity we aim to bound. Likewise, we let (M_1, \dots, M_k) be an ordered sequence of matchings that partition the edge set E , witnessing G is an s -parallel-greedy graph. Also, for the rest of this section, we refer to a path with exactly $s/2$ edges as an $\frac{s}{2}$ -path and for simplicity of presentation we assume that s is even; in the case where s is odd, the same proof works with respect to $\frac{s+1}{2}$ -paths (leading to the slightly-improved bound of $O(s \cdot n^{2/(s+1)})$ mentioned previously).

The following formalizes the sense of monotonic paths we use.

Definition 3.1 (Monotonic Paths). A path P in G is monotonic if the edges in P occur in exactly the same order as the matchings that contain these edges. In other words, let (e_1, e_2, \dots, e_s) be the edge sequence of P , and let M_{i_j} be the matching that contains edge e_j for each $1 \leq j \leq s$. Then we say that P is monotonic if we have $i_1 < i_2 < \dots < i_s$.

The rest of this section proves [Theorem 1.3](#) by counting the number of monotonic $\frac{s}{2}$ -paths.

3.1 Dispersion Lemma

Our dispersion lemma shows that monotonic $\frac{s}{2}$ -paths must be “dispersed” around the graph, rather than be concentrated on one pair of endpoints. This lemma will use a slightly different characterization of s -parallel-greedy graphs as below.

Lemma 3.2. For any cycle C of s -parallel-greedy graph G with $|C| \leq s + 1$ edges, if M_i is the highest-indexed matching that contains an edge of C , then there are least two edges from M_i in C .

Proof. Suppose for the sake of contradiction that C only contained one edge $\{u, v\}$ from M_i . Then, $G_{i-1} := (V, \bigcup_{j < i} M_j)$ contains every edge other than $\{u, v\}$ of C of which there are at most s so

$$d_{G_{i-1}}(u, v) \leq s. \quad (3.1)$$

But, $\{u, v\} \in M_i$ and G is s -parallel-greedy so $d_{G_{i-1}}(u, v) > s$ which contradicts [Equation \(3.1\)](#). \square

See [Figure 1c](#) for an illustration of this on a 12-parallel-greedy graph; in this graph, there are many cycles with at most 13 edges but each such cycle has at least two edge from its highest-indexed incident matching.

The following is our dispersion lemma.

Lemma 3.3 (Dispersion Lemma). For $u, v \in V$, there is at most one monotonic $\frac{s}{2}$ -path from u to v in G .

Proof. Suppose for contradiction that there are two distinct $\frac{s}{2}$ -paths from v to v , P_a and P_b ; see [Figure 2a](#). Then there exist contiguous subpaths $Q_a \subseteq P_a, Q_b \subseteq P_b$ such that $Q_a \cup Q_b$ forms a cycle C . Note that the number of edges in C satisfies

$$|C| \leq |Q_a| + |Q_b| \leq |P_a| + |P_b| = s,$$

and so by [Lemma 3.2](#), we know that the highest-indexed matching containing an edge of C must contain at least 2 edges of C . We proceed to contradict this.

Let e_a^*, e_b^* be the last edges of Q_a, Q_b respectively; see [Figure 2a](#). These edges share an end-point (since they are adjacent in C), and therefore they belong to different matchings. We will

graph, serves to upper bound the arboricity by [Theorem 2.1](#). A similar framework has been used in recent work on graph spanners; for example, [Bod25; BHP24] use this framework over related (but more specific) types of paths.

For the rest of this section we assume we are given an n -node s -parallel greedy graph $G = (V, E)$ with m edges whose arboricity we aim to bound. Likewise, we let (M_1, \dots, M_k) be an ordered sequence of matchings that partition the edge set E , witnessing G is an s -parallel-greedy graph. Also, for the rest of this section, we refer to a path with exactly $s/2$ edges as an $\frac{s}{2}$ -path and for simplicity of presentation we assume that s is even; in the case where s is odd, the same proof works with respect to $\frac{s+1}{2}$ -paths (leading to the slightly-improved bound of $O(s \cdot n^{2/(s+1)})$ mentioned previously).

The following formalizes the sense of monotonic paths we use.

Definition 3.1 (Monotonic Paths). A path P in G is monotonic if the edges in P occur in exactly the same order as the matchings that contain these edges. In other words, let (e_1, e_2, \dots, e_s) be the edge sequence of P , and let M_{i_j} be the matching that contains edge e_j for each $1 \leq j \leq s$. Then we say that P is monotonic if we have $i_1 < i_2 < \dots < i_s$.

The rest of this section proves [Theorem 1.3](#) by counting the number of monotonic $\frac{s}{2}$ -paths.

3.1 Dispersion Lemma

Our dispersion lemma shows that monotonic $\frac{s}{2}$ -paths must be “dispersed” around the graph, rather than be concentrated on one pair of endpoints. This lemma will use a slightly different characterization of s -parallel-greedy graphs as below.

Lemma 3.2. For any cycle C of s -parallel-greedy graph G with $|C| \leq s + 1$ edges, if M_i is the highest-indexed matching that contains an edge of C , then there are least two edges from M_i in C .

Proof. Suppose for the sake of contradiction that C only contained one edge $\{u, v\}$ from M_i . Then, $G_{i-1} := (V, \bigcup_{j < i} M_j)$ contains every edge other than $\{u, v\}$ of C of which there are at most s so

$$d_{G_{i-1}}(u, v) \leq s. \quad (3.1)$$

But, $\{u, v\} \in M_i$ and G is s -parallel-greedy so $d_{G_{i-1}}(u, v) > s$ which contradicts [Equation \(3.1\)](#). \square

See [Figure 1c](#) for an illustration of this on a 12-parallel-greedy graph; in this graph, there are many cycles with at most 13 edges but each such cycle has at least two edge from its highest-indexed incident matching.

The following is our dispersion lemma.

Lemma 3.3 (Dispersion Lemma). For $u, v \in V$, there is at most one monotonic $\frac{s}{2}$ -path from u to v in G .

Proof. Suppose for contradiction that there are two distinct $\frac{s}{2}$ -paths from v to v , P_a and P_b ; see [Figure 2a](#). Then there exist contiguous subpaths $Q_a \subseteq P_a, Q_b \subseteq P_b$ such that $Q_a \cup Q_b$ forms a cycle C . Note that the number of edges in C satisfies

$$|C| \leq |Q_a| + |Q_b| \leq |P_a| + |P_b| = s,$$

and so by [Lemma 3.2](#), we know that the highest-indexed matching containing an edge of C must contain at least 2 edges of C . We proceed to contradict this.

Let e_a^*, e_b^* be the last edges of Q_a, Q_b respectively; see [Figure 2a](#). These edges share an end-point (since they are adjacent in C), and therefore they belong to different matchings. We will

Proof. Let G' be a uniform random edge-subgraph of G on exactly $sn/2$ edges. Let x be the number of monotonic $\frac{s}{2}$ -paths in G , and let x' be the number of monotonic $\frac{s}{2}$ -paths that survive in G' . On one hand, by the medium counting lemma ([Lemma 3.5](#)), we have $x' \geq \Omega(n)$ (deterministically). On the other hand, for any monotonic $\frac{s}{2}$ -path P in G , the probability that P survives in G' is

$$\underbrace{\frac{sn/2}{m}}_{\text{probability first edge is selected in } G'} \cdot \underbrace{\frac{sn/2-1}{m-1}}_{\text{probability second edge is selected in } G', \text{ given first edge is selected in } G'} \cdot \dots \cdot \underbrace{\frac{sn/2-(s/2-1)}{m-(s/2-1)}}_{\text{probability } s/2^{\text{th}} \text{ edge is selected in } G', \text{ given first } s/2-1 \text{ edges are selected in } G'}$$

which is

$$\leq \left(\frac{sn}{2m} \right)^{s/2} \\ = O \left(\frac{s}{d} \right)^{s/2}.$$

Thus we have

$$\Omega(n) \leq \mathbb{E}[x'] \leq x \cdot O \left(\frac{s}{d} \right)^{s/2}.$$

Rearranging, we get

$$x \geq n \cdot \Omega \left(\frac{d}{s} \right)^{s/2},$$

as claimed. \square

3.3 Completing Our Arboricity Bound

We now complete our bound on the arboricity of s -parallel-greedy graphs by combining our dispersion lemma and full counting lemma.

Theorem 1.3 (Parallel-Greedy Graph Arboricity). If G is an n -node s -parallel-greedy graph, then G has arboricity at most $O(s \cdot n^{2/s})$.

Proof. First, we claim that any n -node s -parallel greedy graph G has average degree at most $O(s \cdot n^{2/s})$. Let d be the average degree of G . By [Lemma 3.3](#), there are $O(n^2)$ monotonic $\frac{s}{2}$ -paths in G . By [Lemma 3.6](#), there are $n \cdot \Omega(d/s)^{s/2}$ monotonic $\frac{s}{2}$ -paths in G . Comparing these estimates, we have

$$n \cdot \Omega \left(\frac{d}{s} \right)^{s/2} \leq O(n^2).$$

Rearranging this inequality gives

$$d \leq O(s \cdot n^{2/s}),$$

giving our claimed bound on the average degree of G .

To bound the arboricity of G , observe that any subgraph of an s -parallel greedy graph is itself an s -parallel greedy graph. Combining this with our bound on the average degree of an s -parallel greedy graph, we get that for any $U \subseteq V$ we have

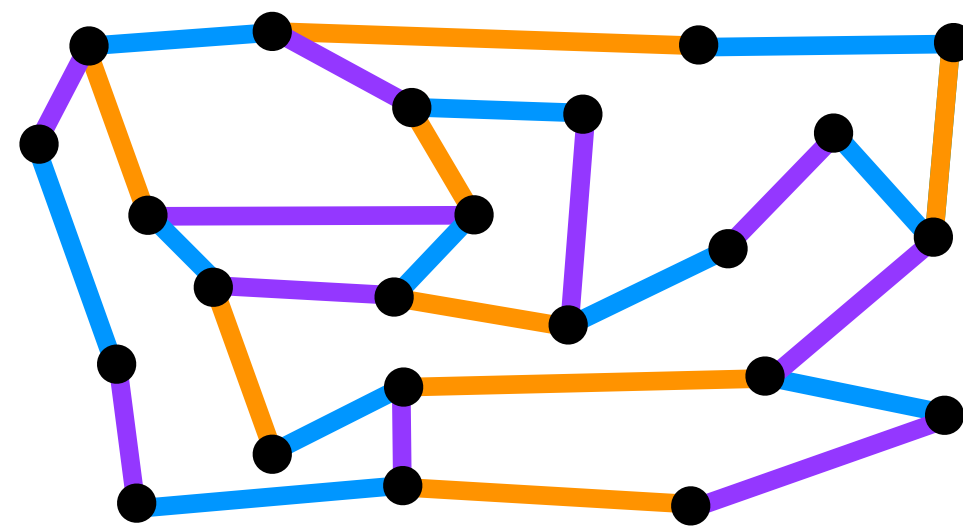
$$|E(U)| \leq O(s \cdot |U|^{2/s}) \cdot (|U| - 1) \leq O(s \cdot n^{2/s}) \cdot (|U| - 1).$$

Applying [Theorem 2.1](#), we get that the arboricity of G is at most $O(s \cdot n^{2/s})$ as required. \square

Summarizing

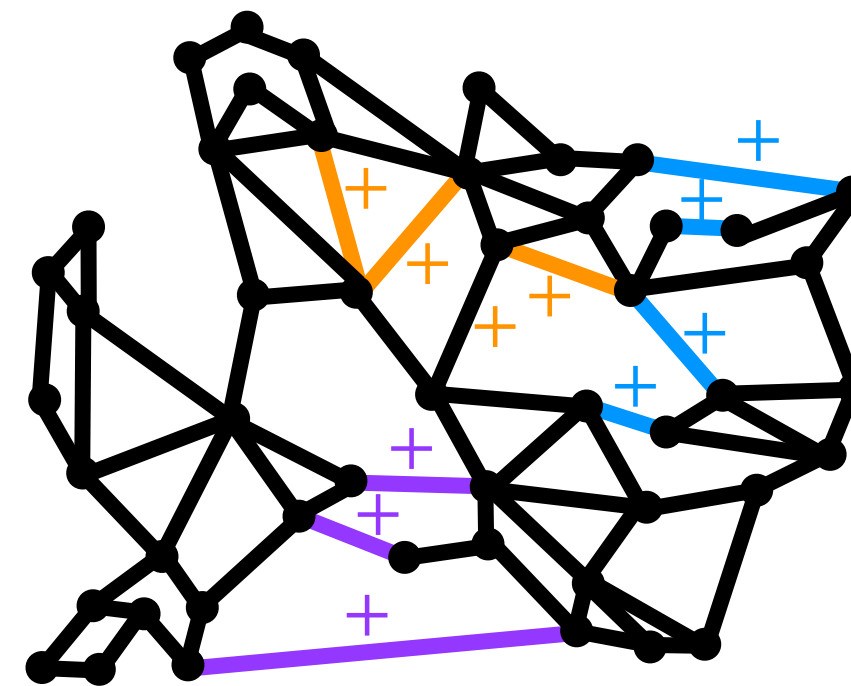
Summary

Thanks!



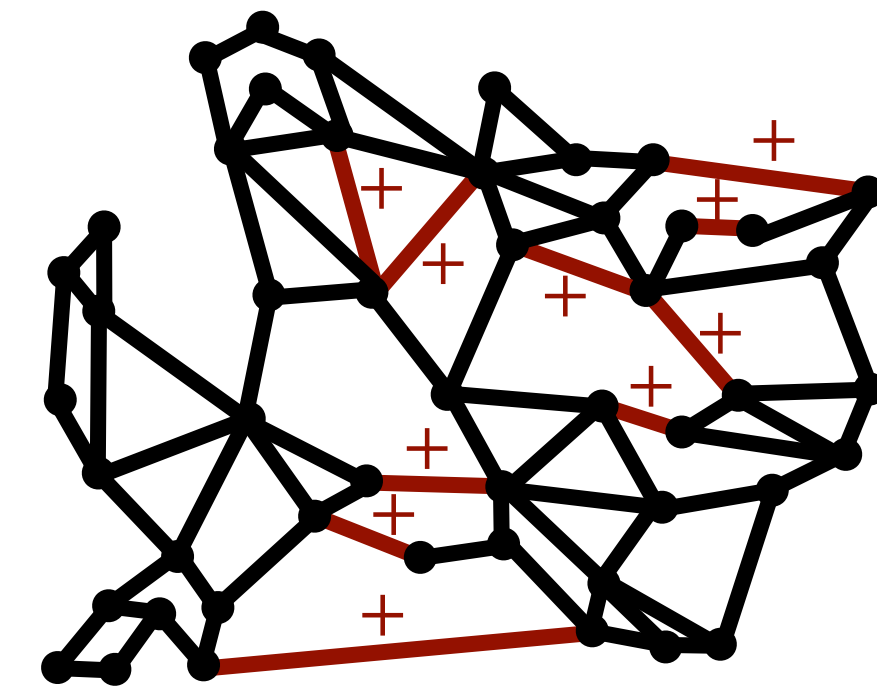
Parallel Greedy
Arboricity

Easy
→
(prior)



U of Cuts

Easy
→



Existence of LC
Decompositions