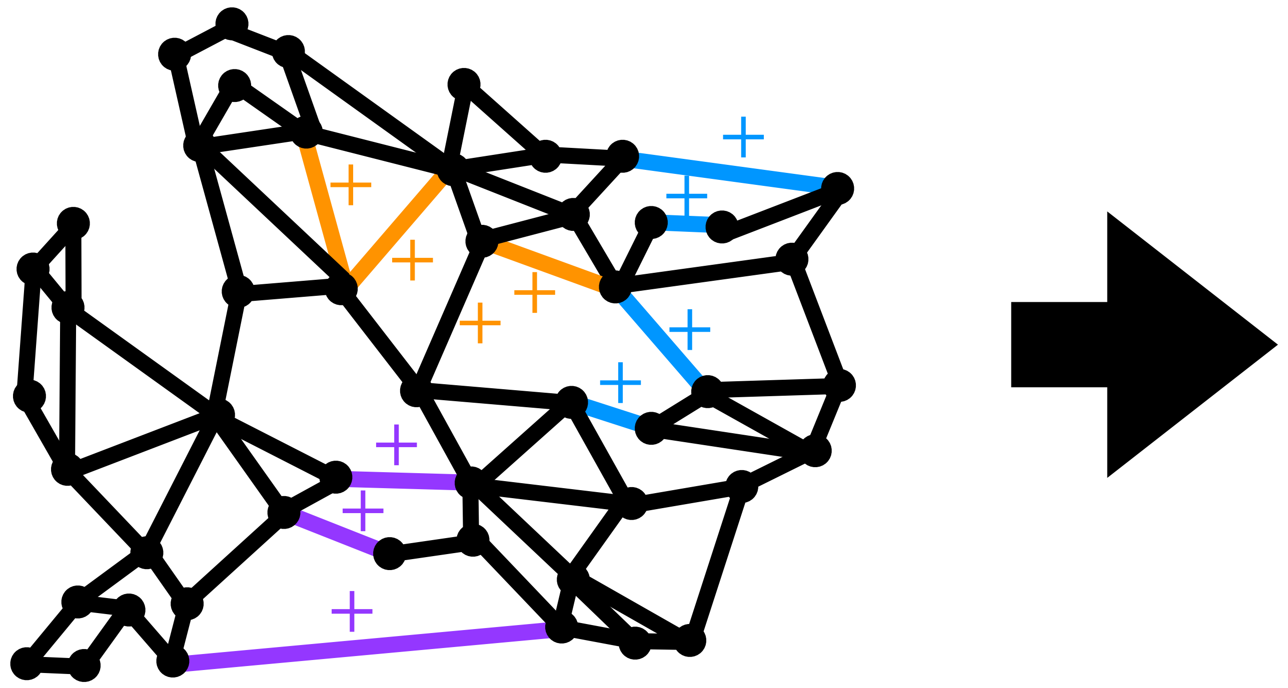
Simple Length-Constrained Expander Decompositions

(SOSA 2026)






Bernhard Haeupler
INSAIT & ETH Zurich

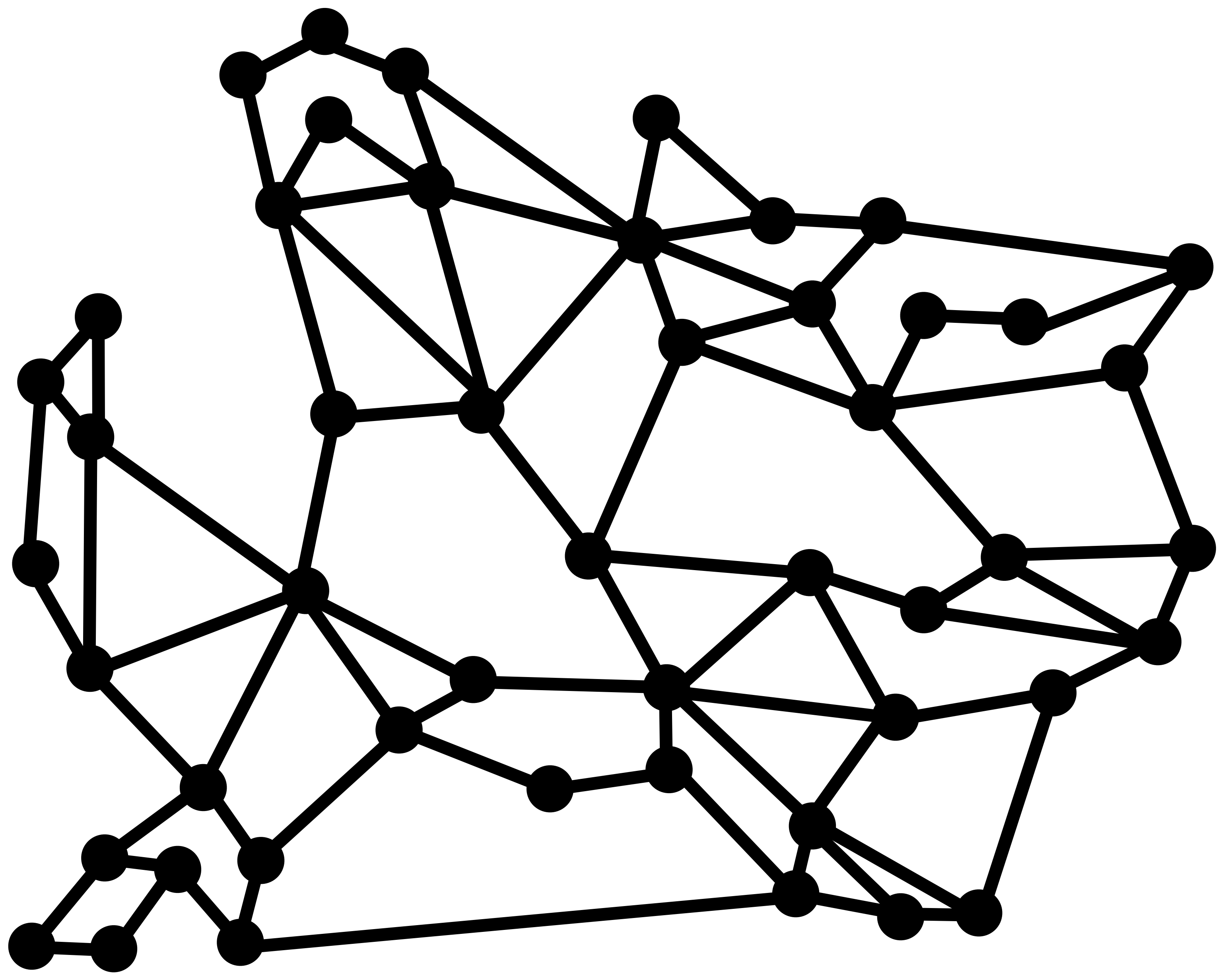
D Ellis Hershkowitz
Brown

Greg Bodwin
U. Michigan

Zihan Tan
U. Minnesota

How to Solve Your Favorite Graph Problem

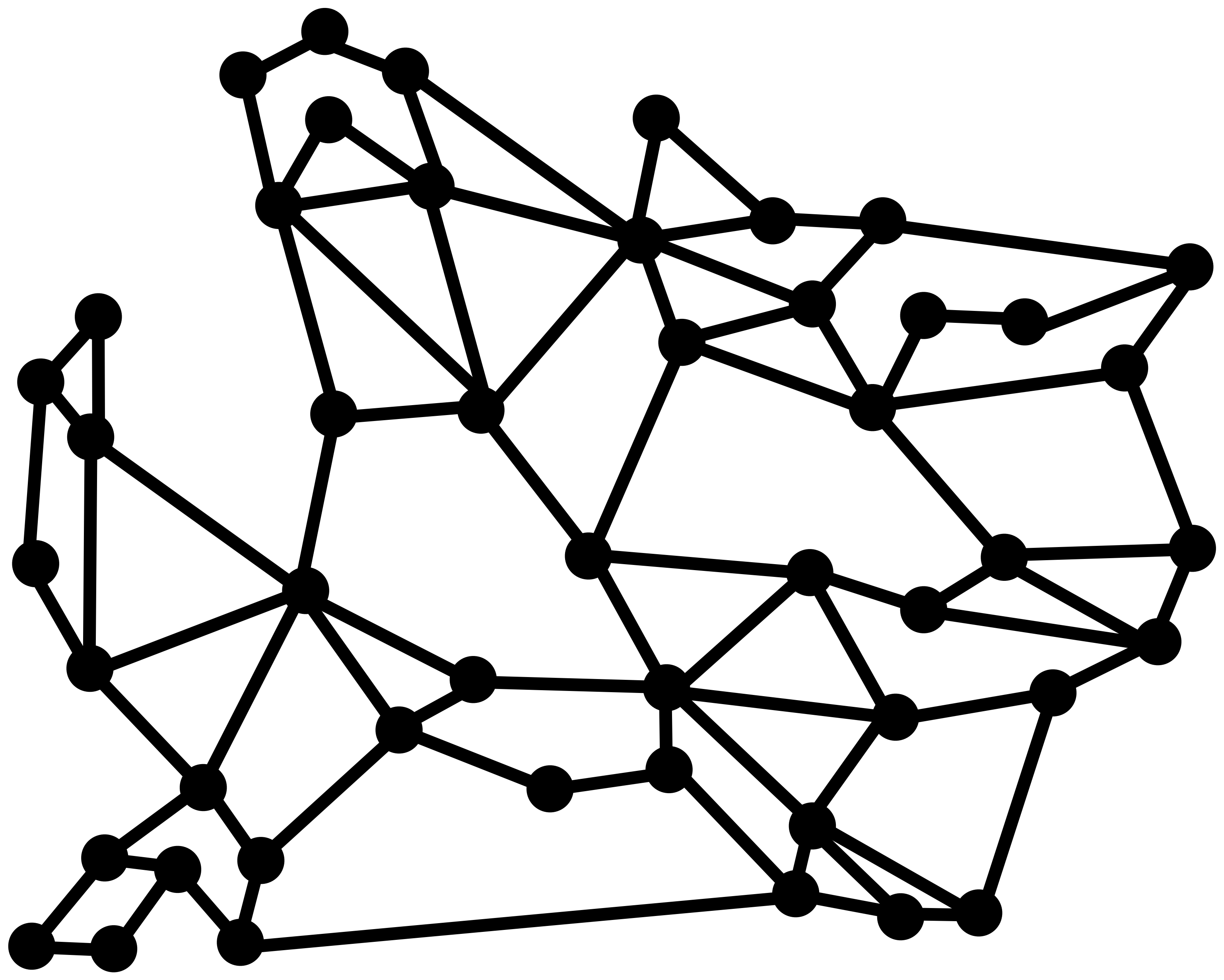
Graph Decomposition Approach



How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

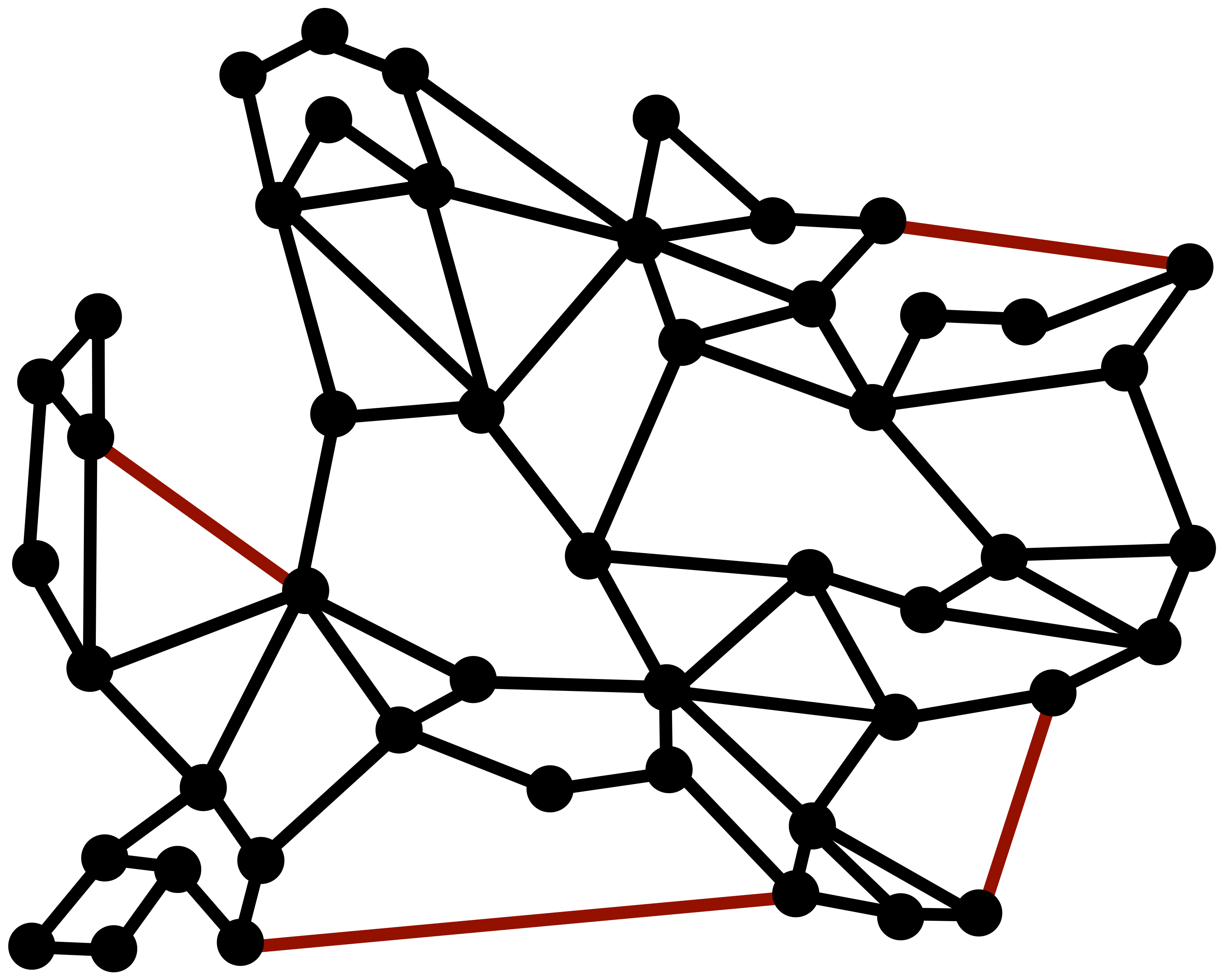
1. **Graph Decomposition**
   add to graph "modifications"
   to make it "nice"



How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

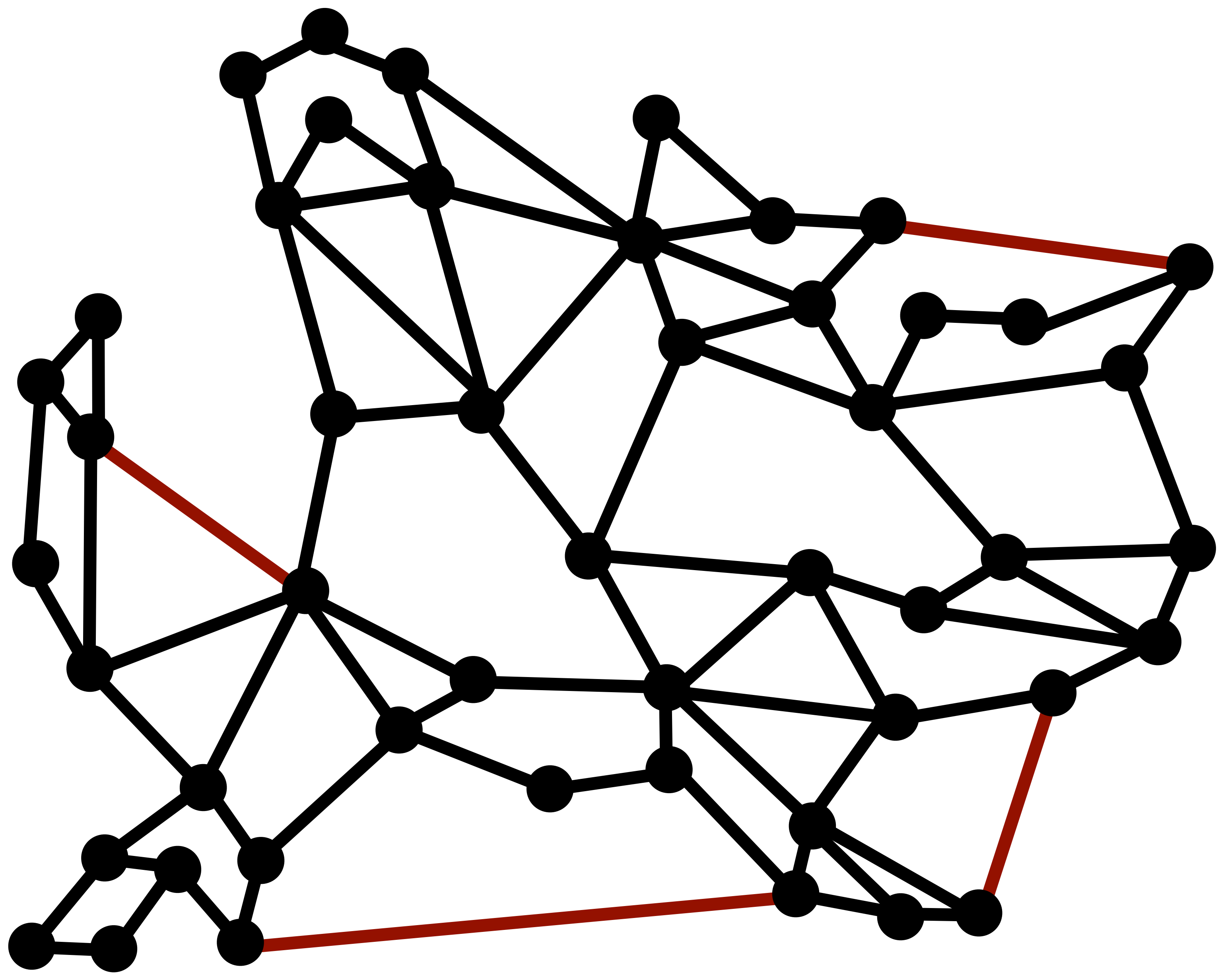
1. **Graph Decomposition**
   add to graph "modifications"
   to make it "nice"



How to Solve Your Favorite Graph Problem

Graph Decomposition Approach

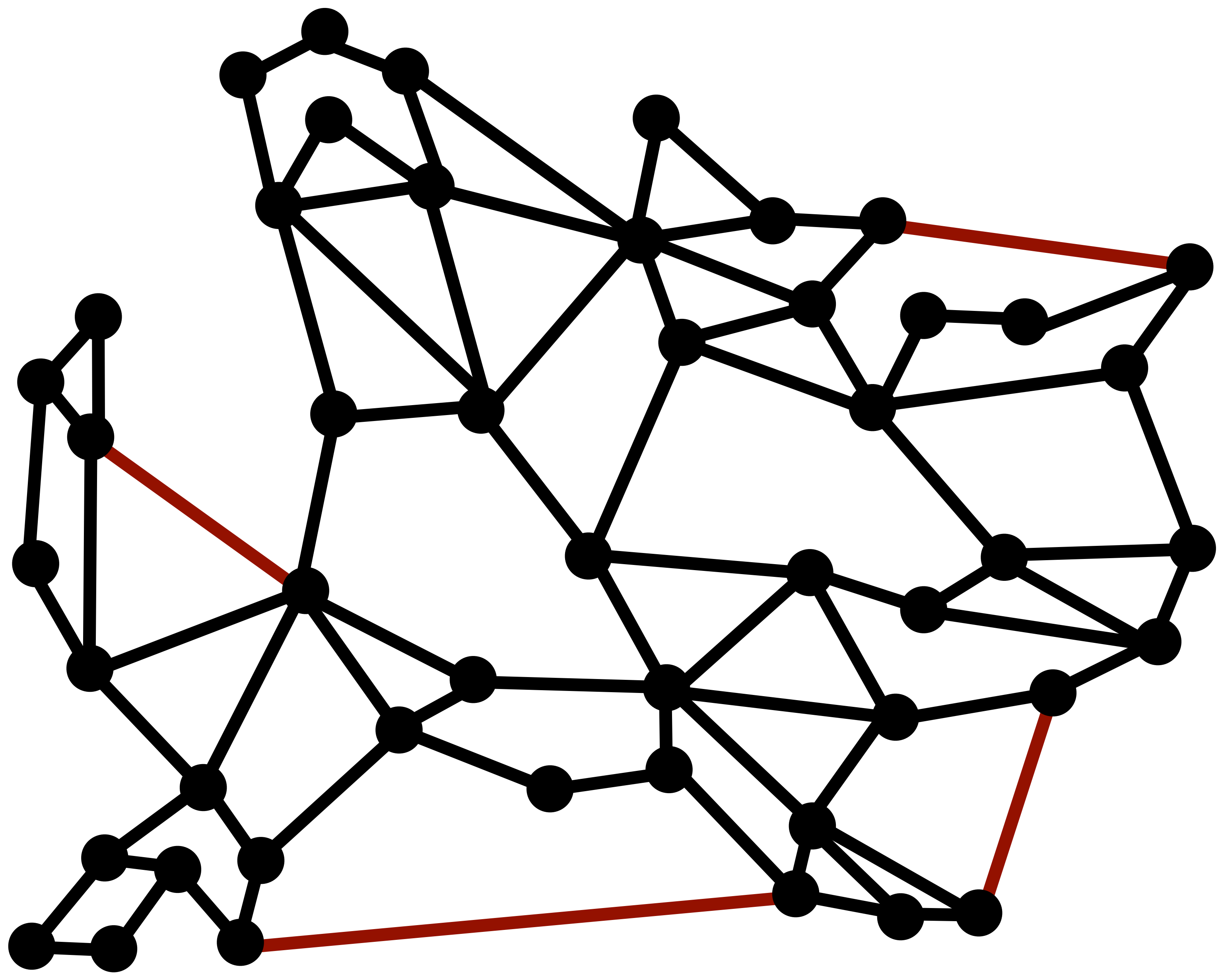
1. **Graph Decomposition**
   add to graph "modifications"
   to make it "nice"
2. **Solve Problem**
   solve problem on nice graph



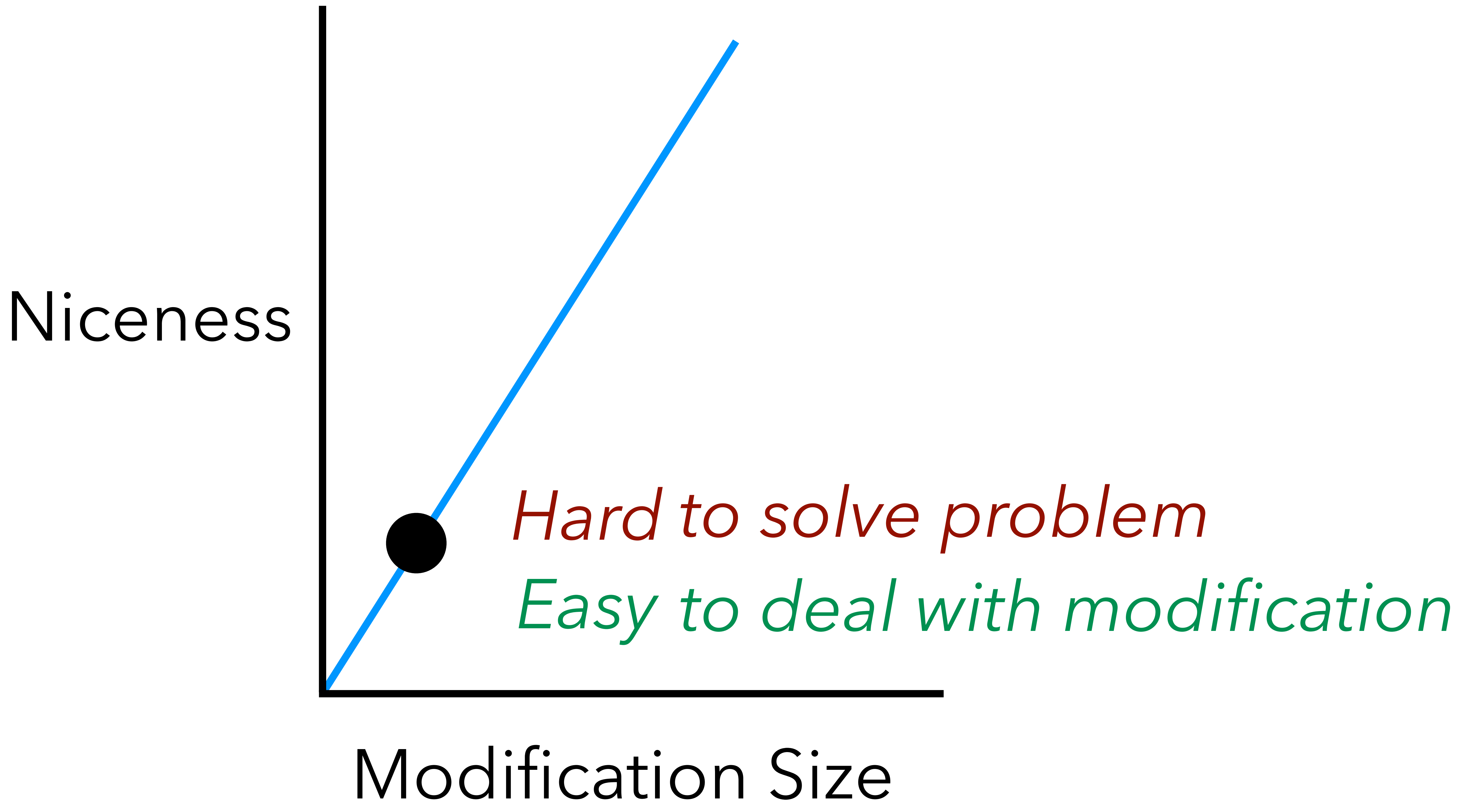
How to Solve Your Favorite Graph Problem

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

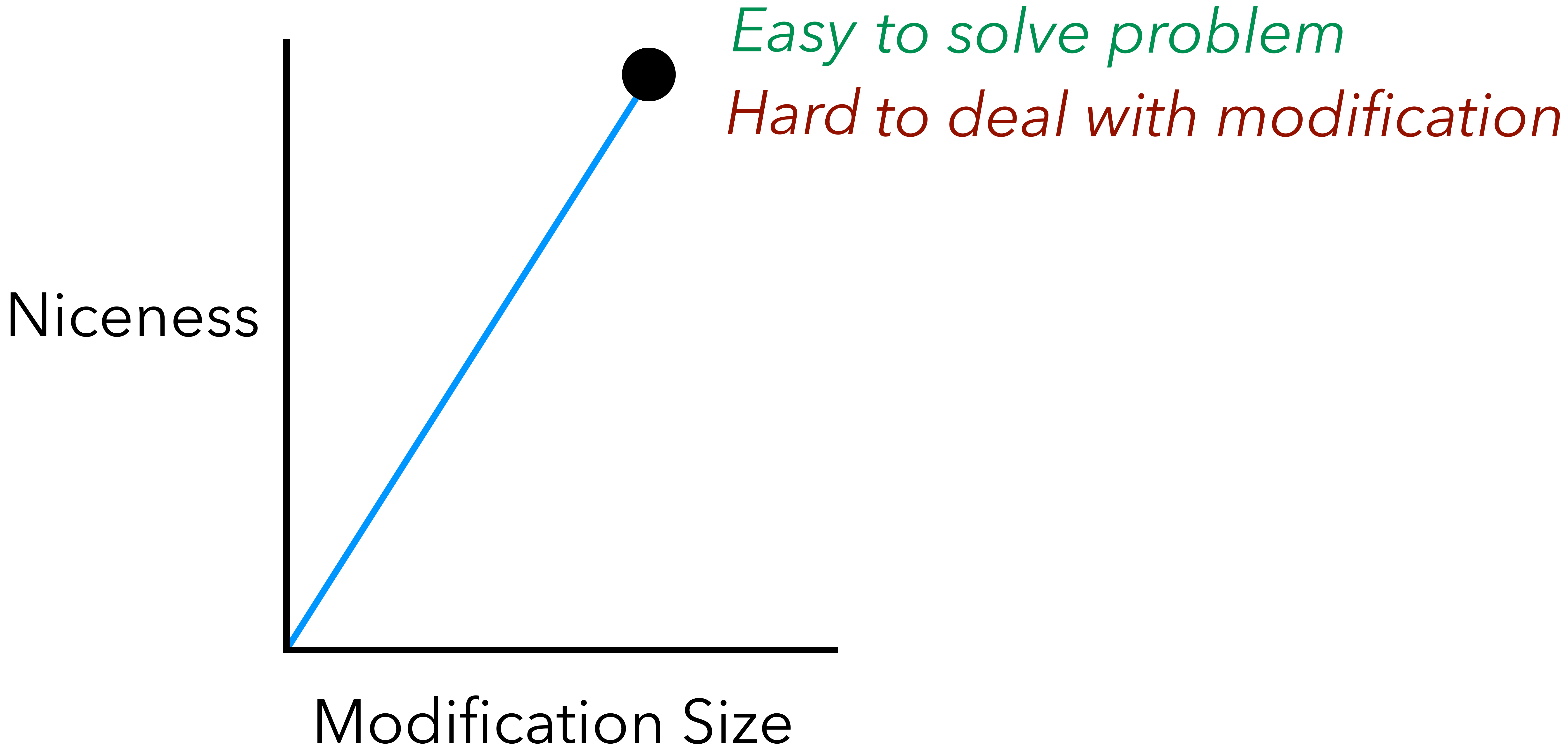


How to Solve Your Favorite Graph Problem



Graph Decomposition Size-Niceness Tradeoff

How to Solve Your Favorite Graph Problem

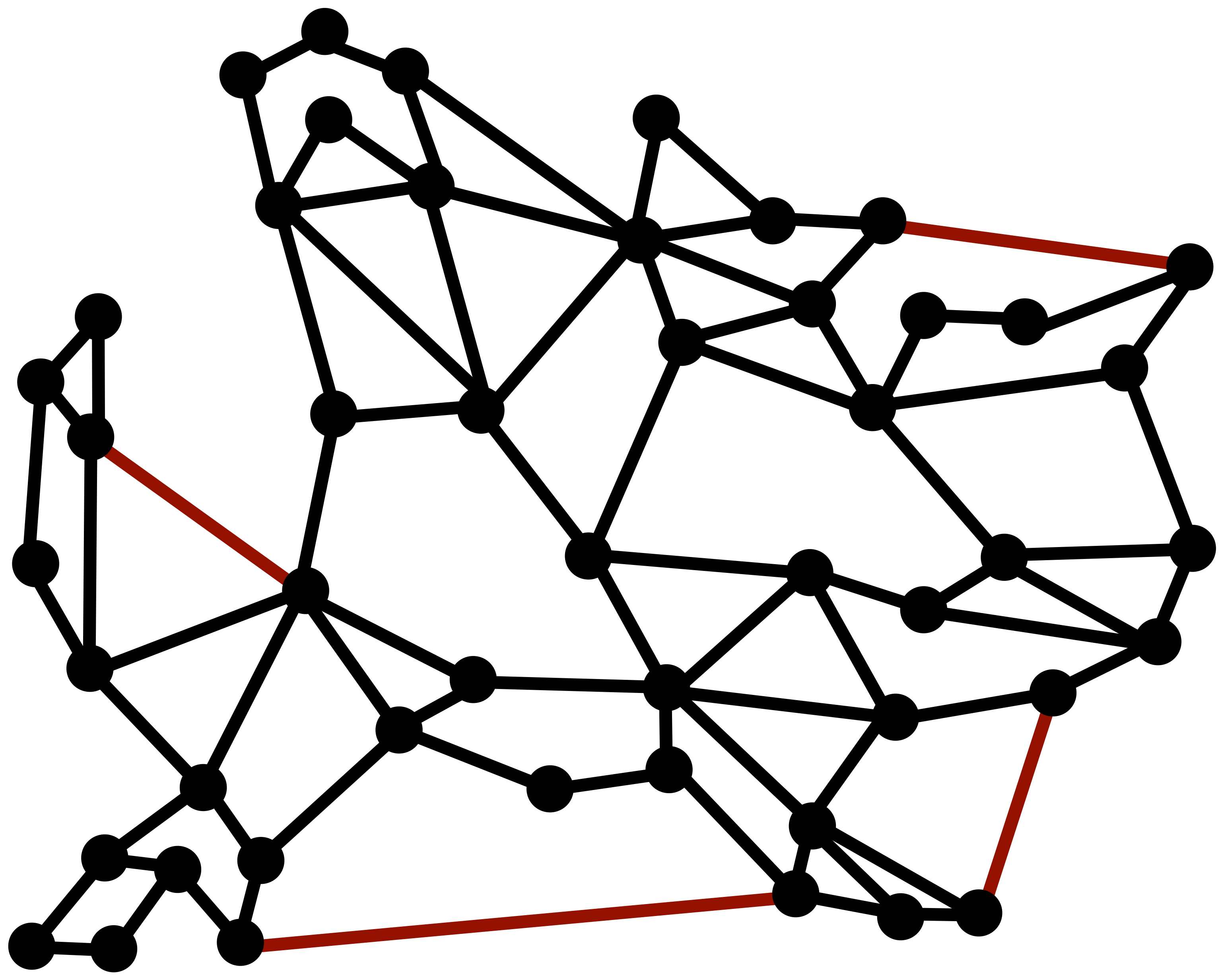


Graph Decomposition Size-Niceness Tradeoff

Length-Constrained Expander Decompositions

Graph Decomposition Approach

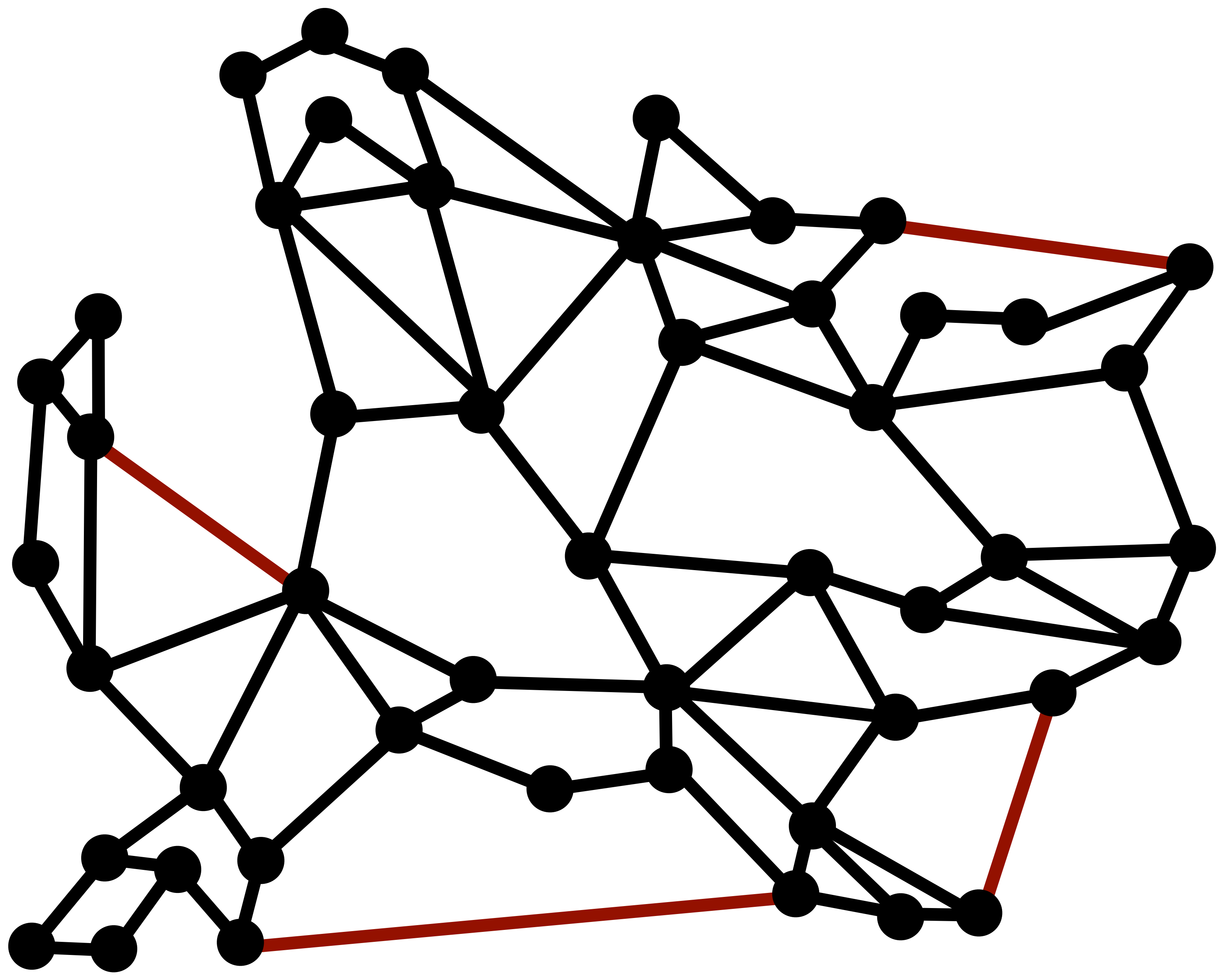
1. **Graph Decomposition**
   add to graph "modifications"
   to make it "nice"
2. **Solve Problem**
   solve problem on nice graph
3. **Clean Up**
   deal with modifications



Length-Constrained Expander Decompositions

Graph Decomposition Approach

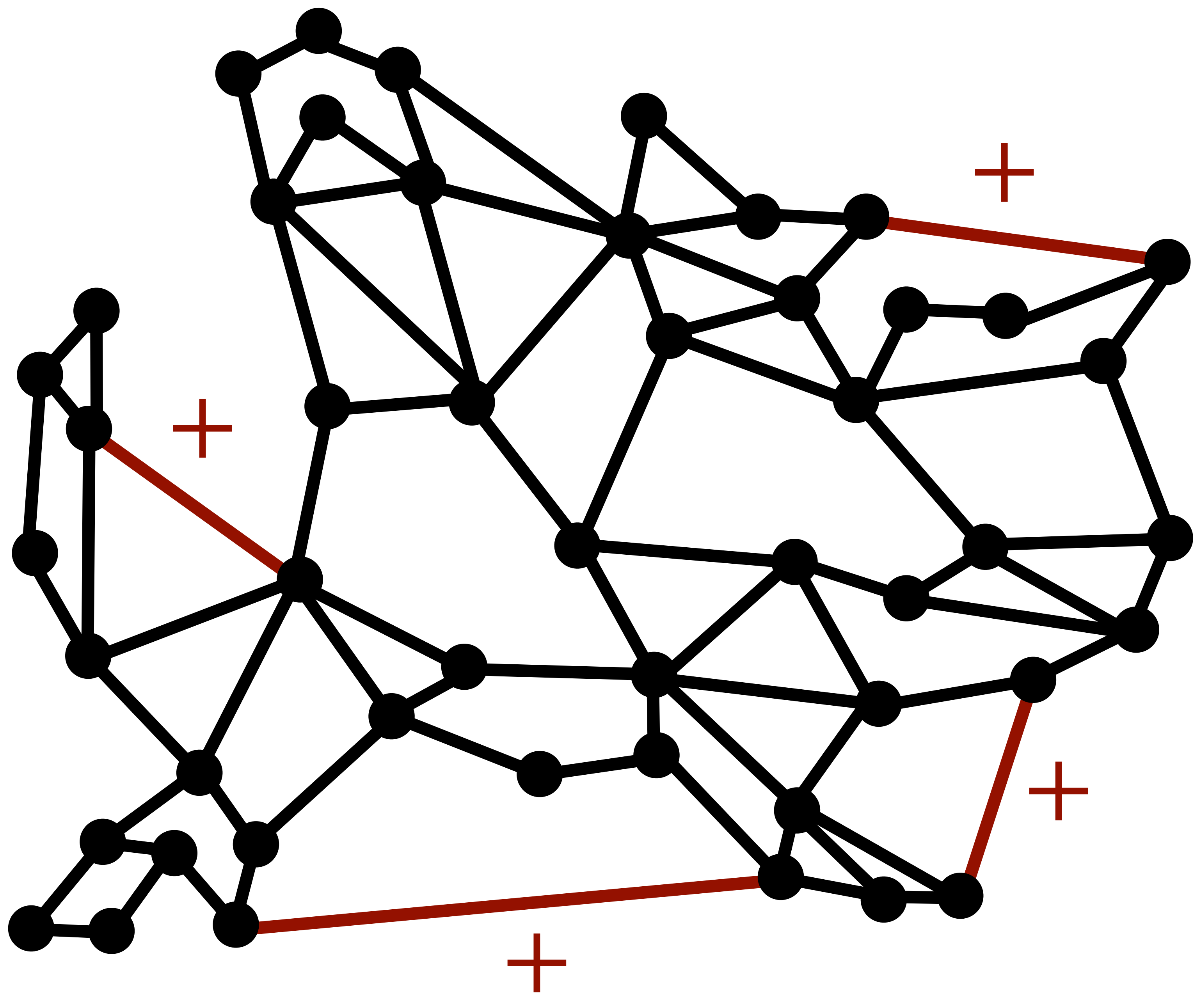
1. **Length-Constrained Expander Decomposition**
   add to graph "modifications"
   to make it "nice"
2. **Solve Problem**
   solve problem on nice graph
3. **Clean Up**
   deal with modifications



Length-Constrained Expander Decompositions

Graph Decomposition Approach

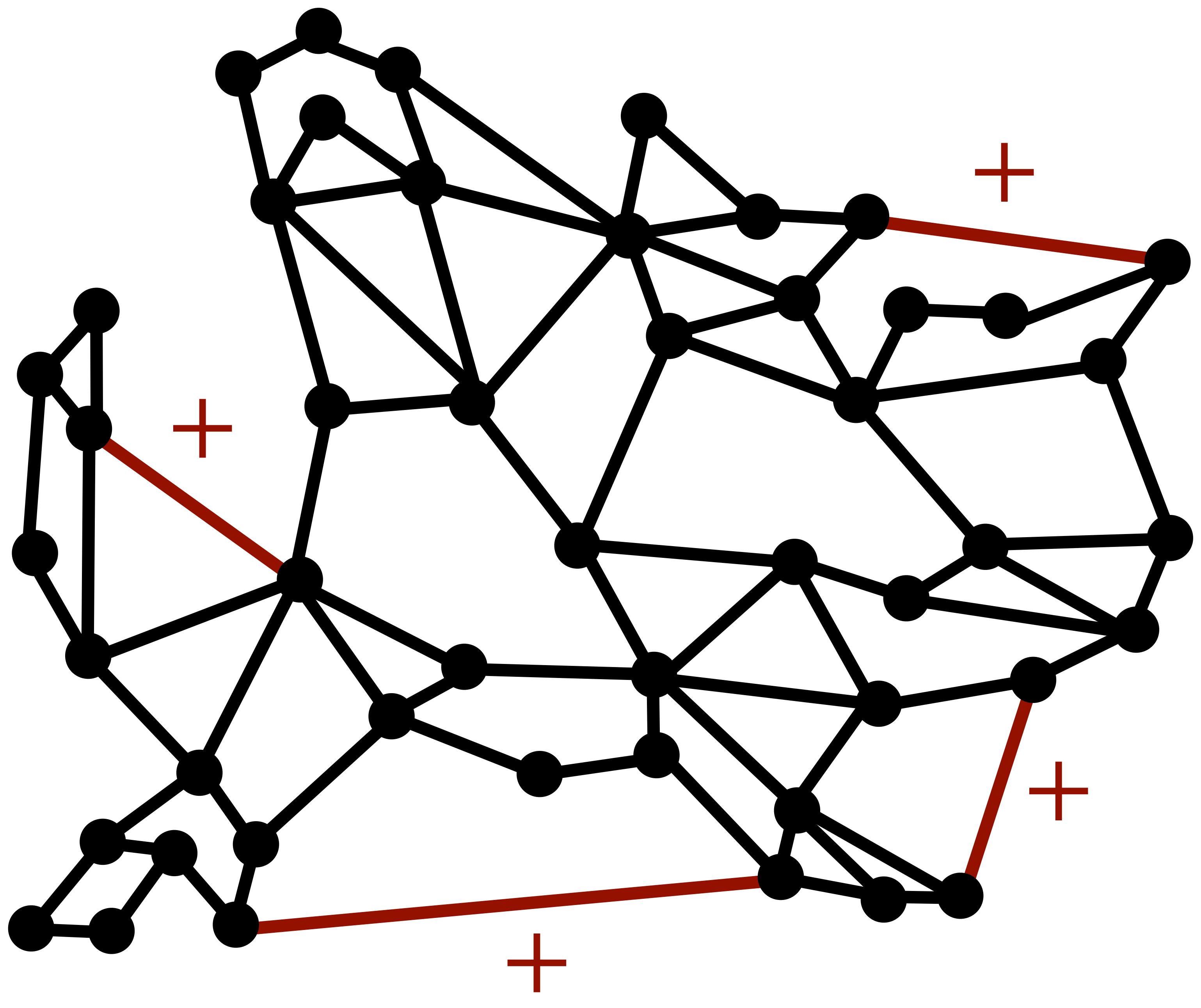
1. **Length-Constrained Expander Decomposition**
   add to graph edge length increases
   to make it "nice"
2. **Solve Problem**
   solve problem on nice graph
3. **Clean Up**
   deal with modifications



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



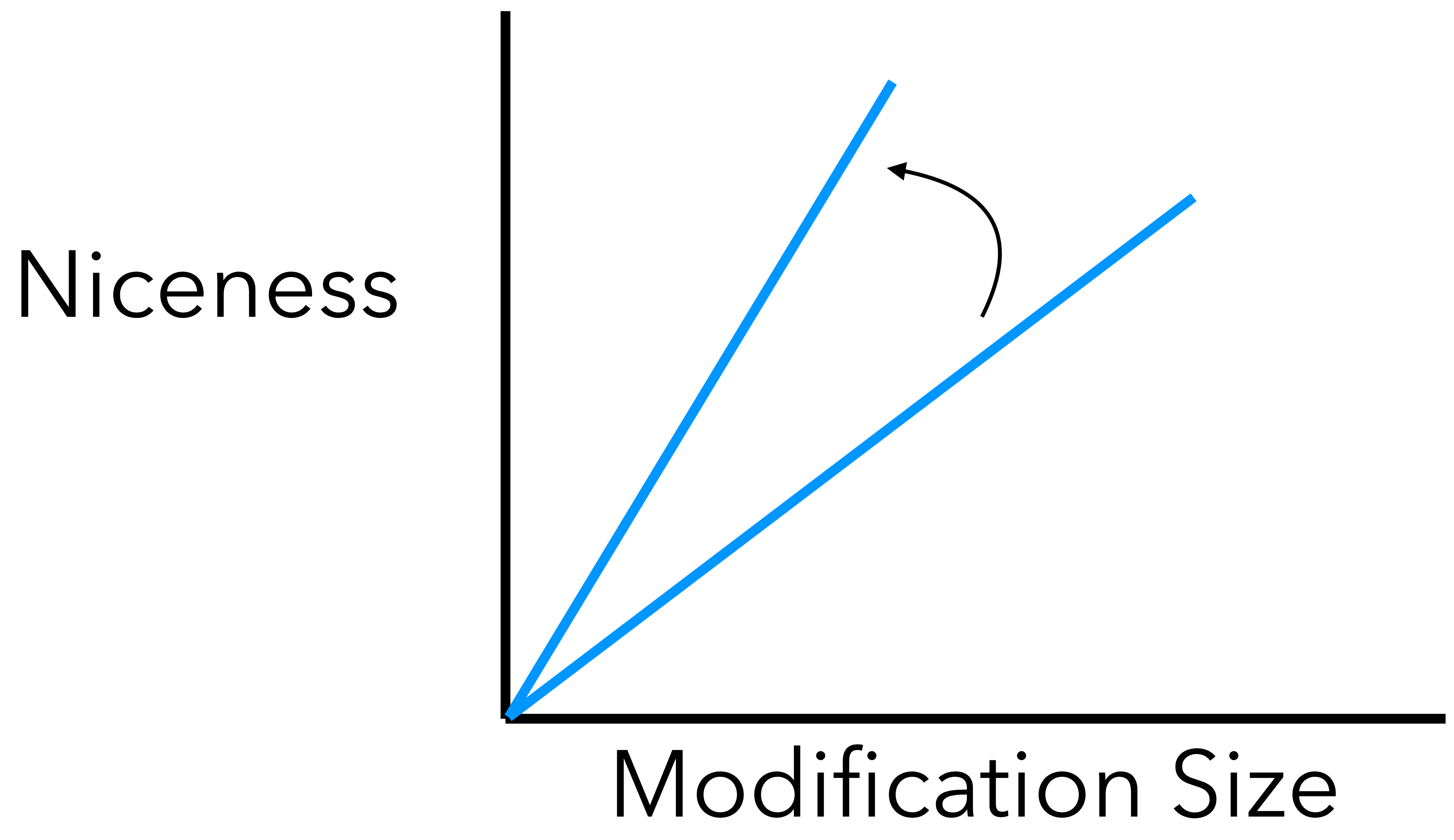
Length-Constrained Expander Decompositions

Graph decomposition approach with LC EDs gives SOTA for:

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

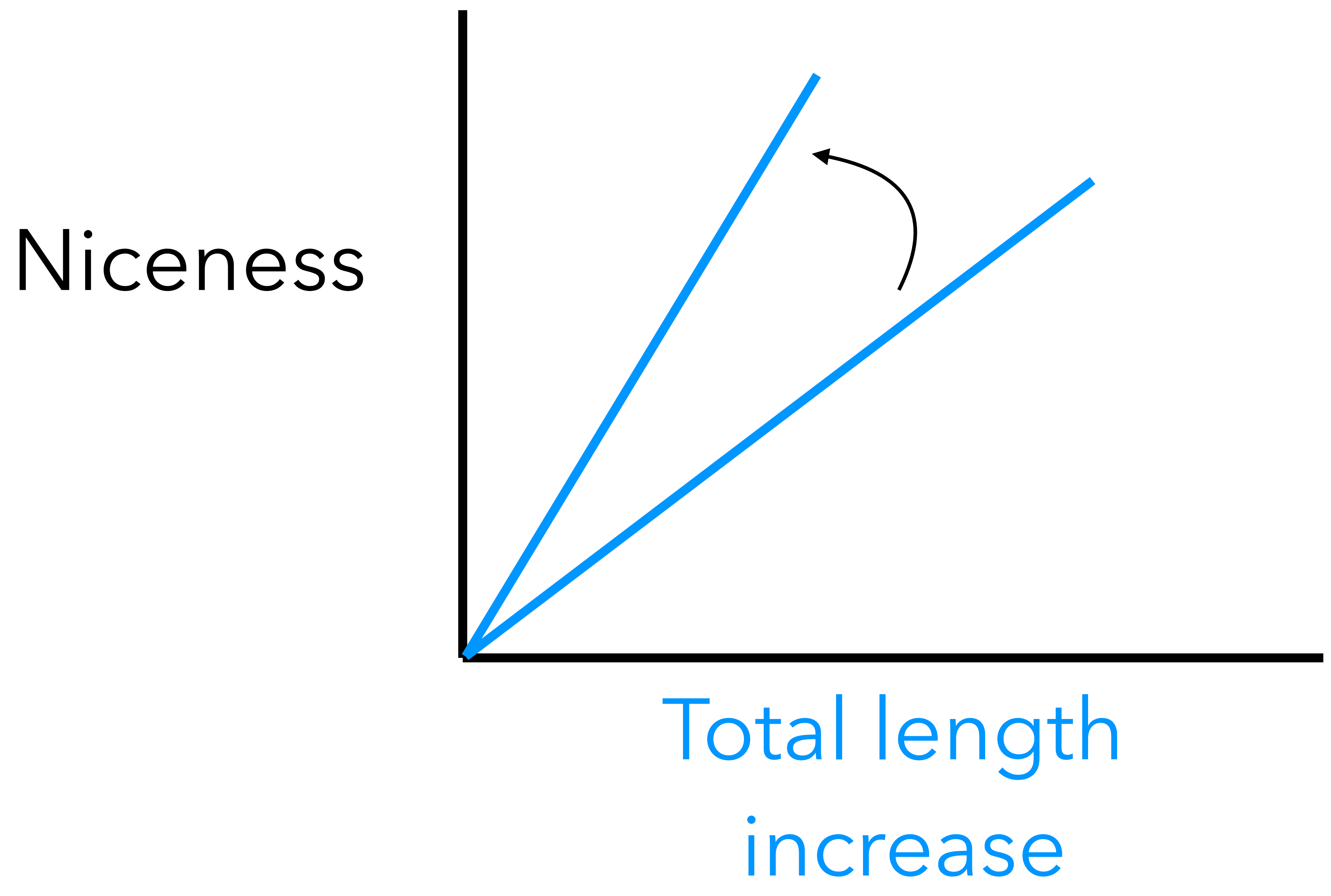
Our Main Result (Informally)

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



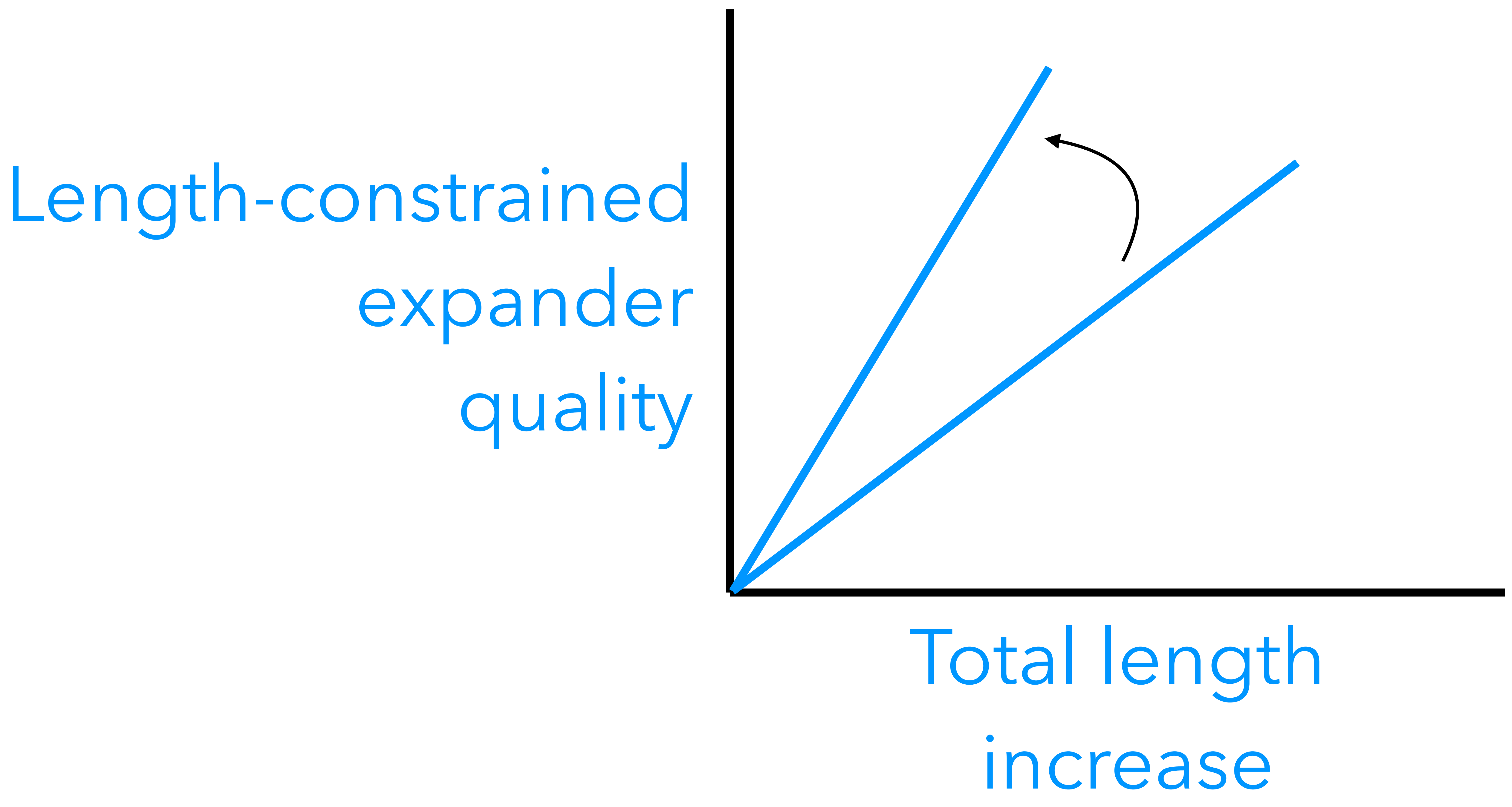
Our Main Result (Informally)

Theorem [BH**H**T]. Simple proof of the existence of length-constrained expander decompositions with improved size-niceness tradeoffs



Our Main Result (Informally)

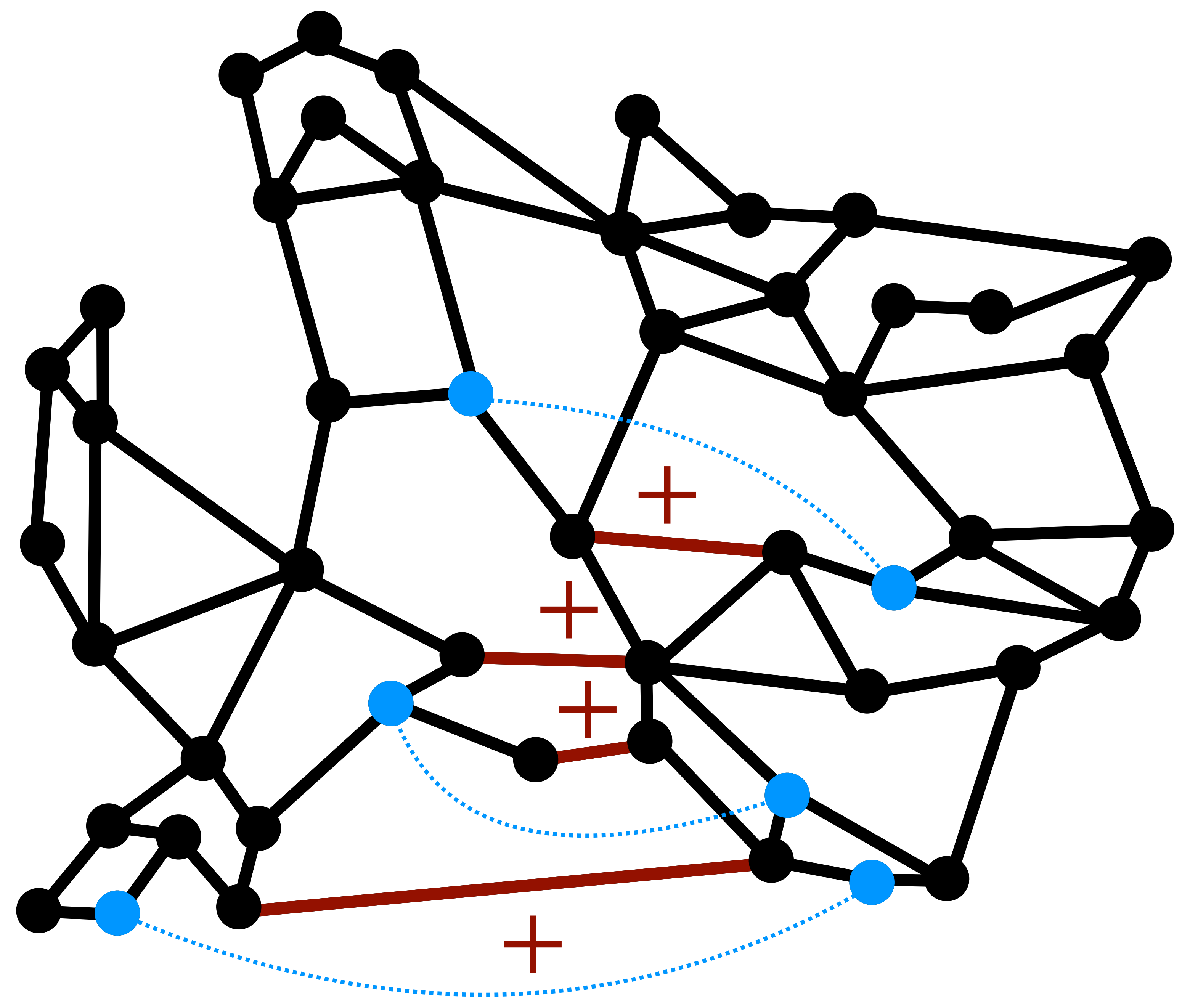
Theorem [BHHT]. Simple proof of the existence of length-constrained expander decompositions with improved size-niceness tradeoffs



Defining Length-Constrained EDs

Sparse Length-Constrained Cuts

Sparse LC Cut Informally.
small total length increase that makes
many close vertex pairs
far

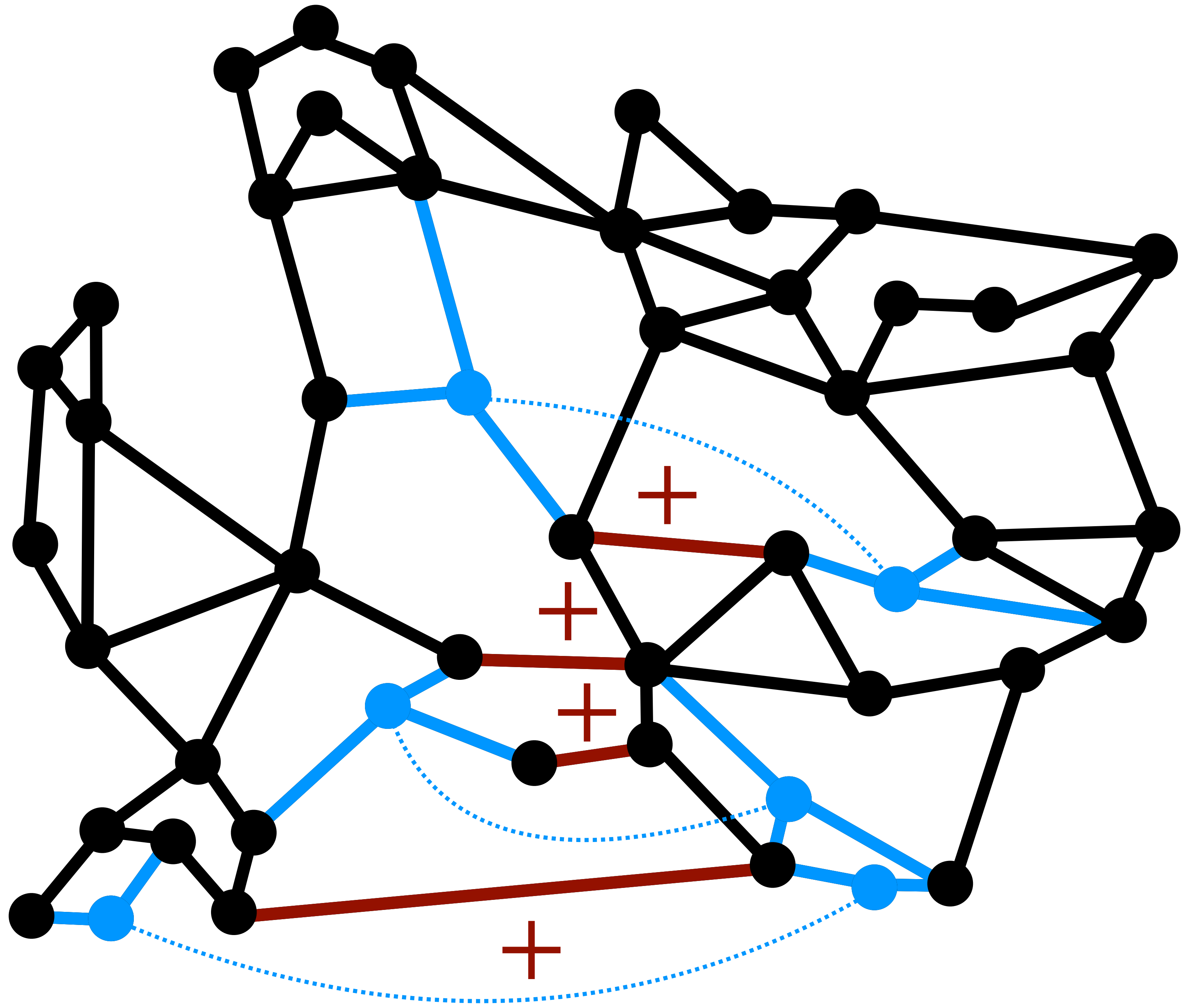


Sparse LC Cut

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

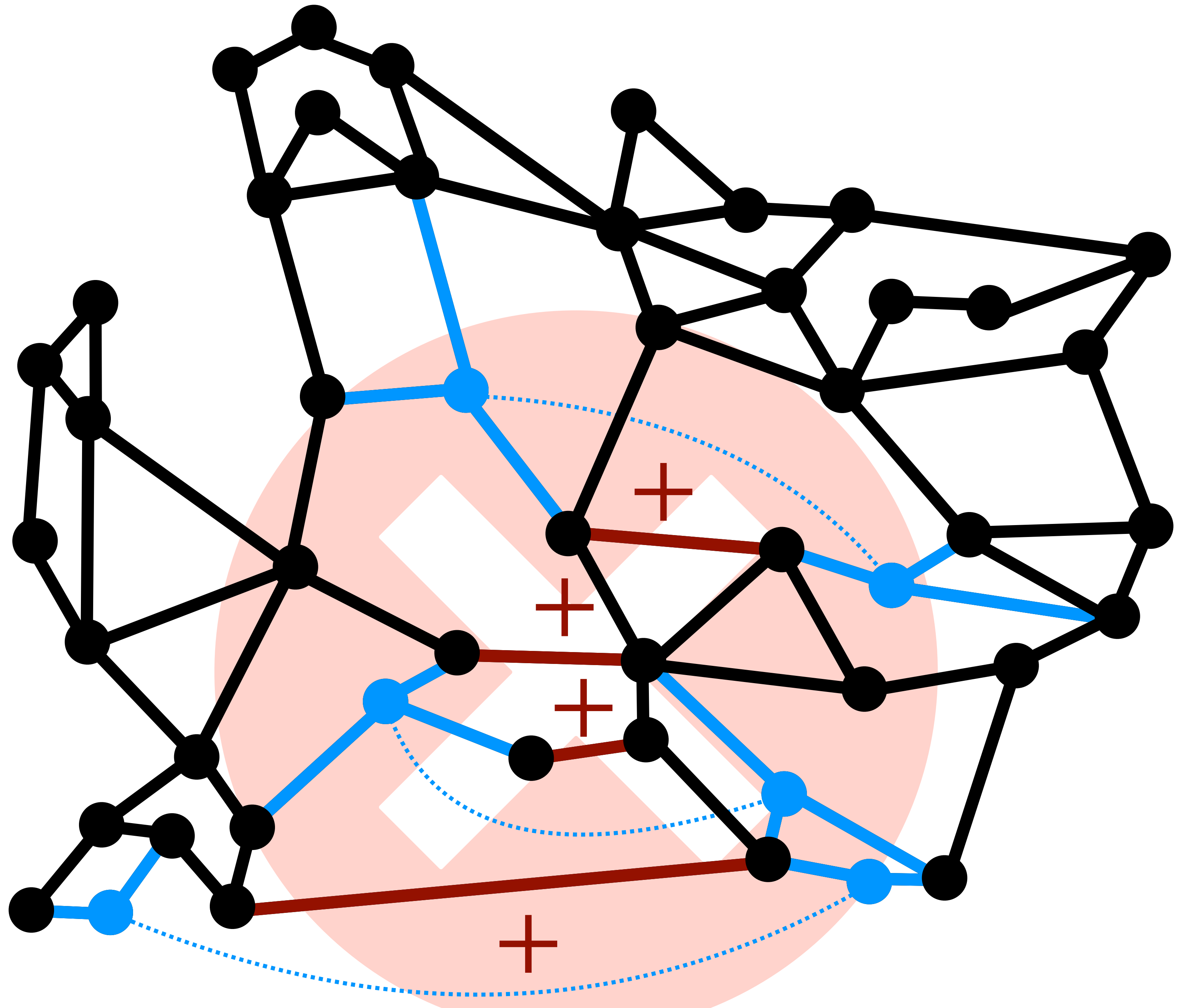


Sparse LC Cut
X length increases
X/ϕ total degree

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

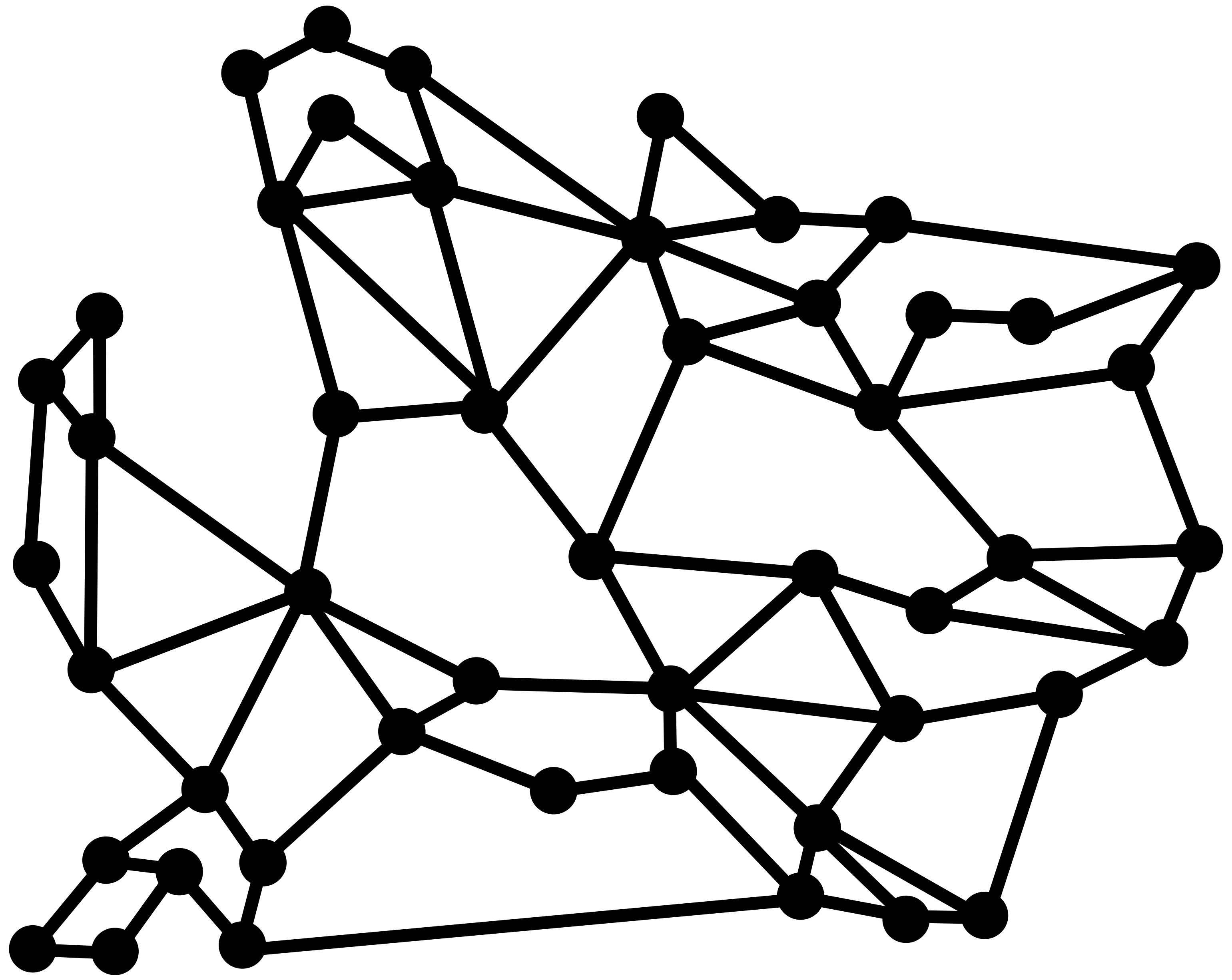


LC Expander

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

Length-Constrained Expander Decompositions

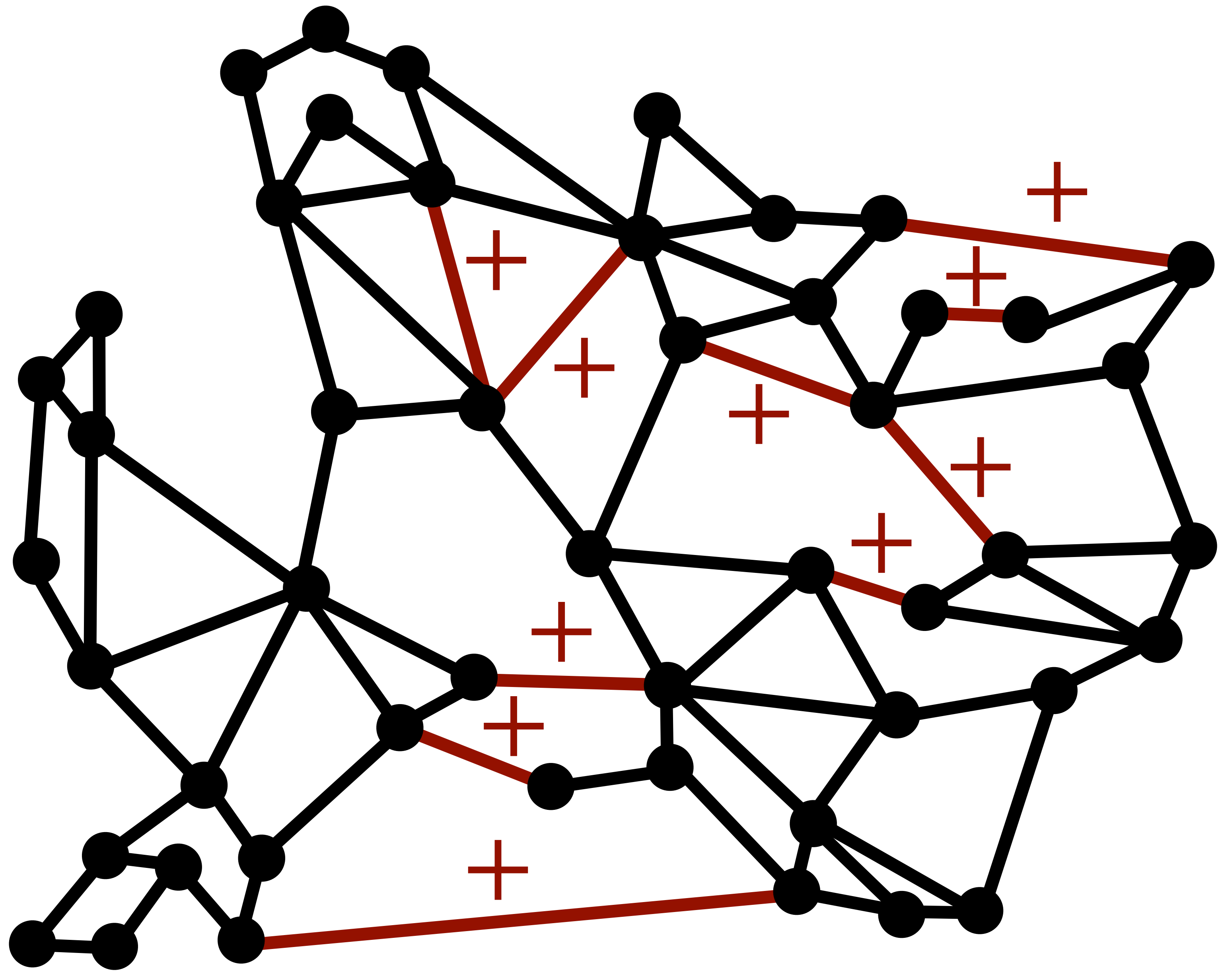
(h, s)**-Length** ϕ**-Expander Decomposition**
length increases that
make graph
an (h, s)-length ϕ-expander



Arbitrary Graph

Length-Constrained Expander Decompositions

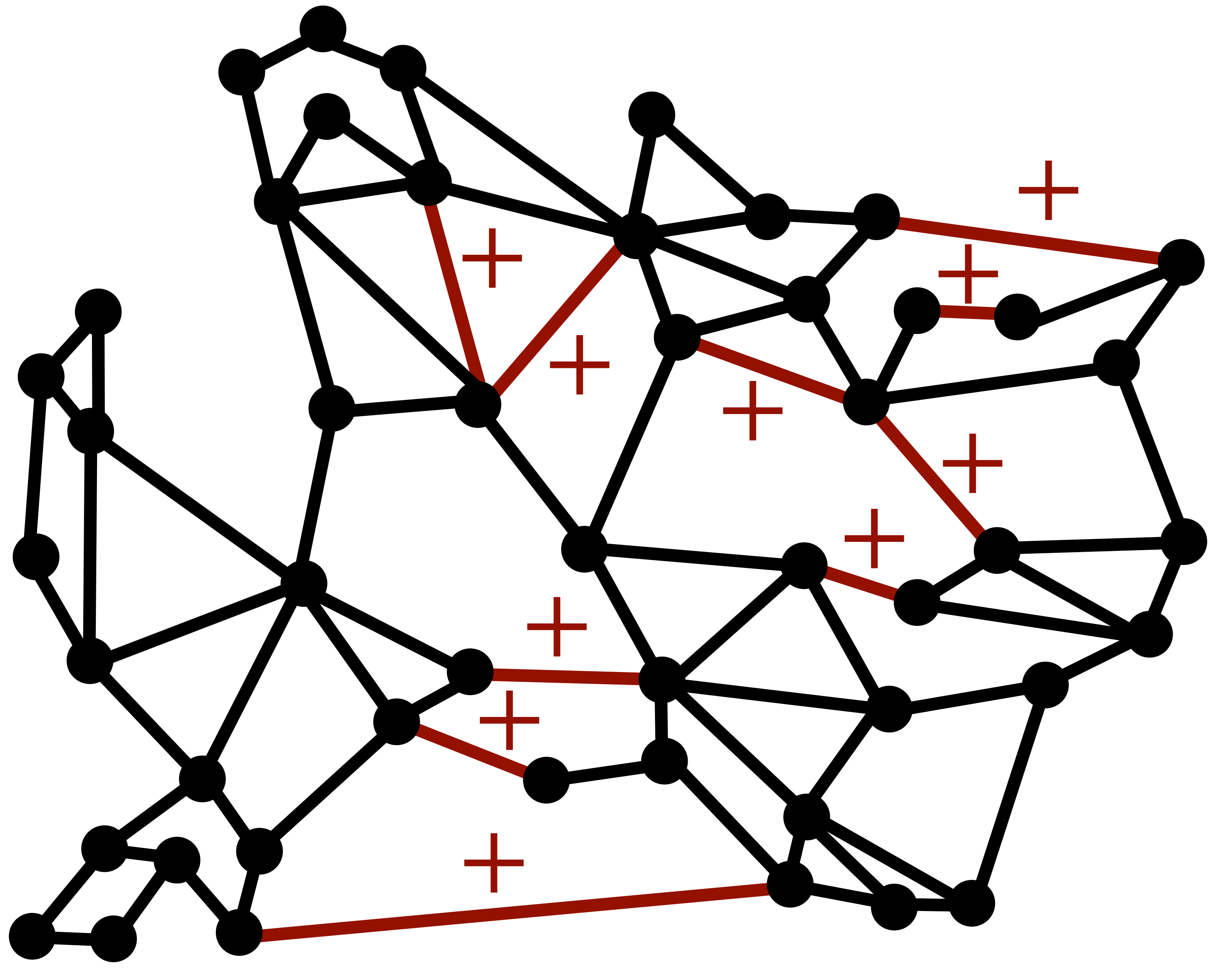
(h, s)-**Length** ϕ-**Expander Decomposition**
length increases that
make graph
an (h, s)-length ϕ-expander



Arbitrary Graph

Length-Constrained Expander Decompositions

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



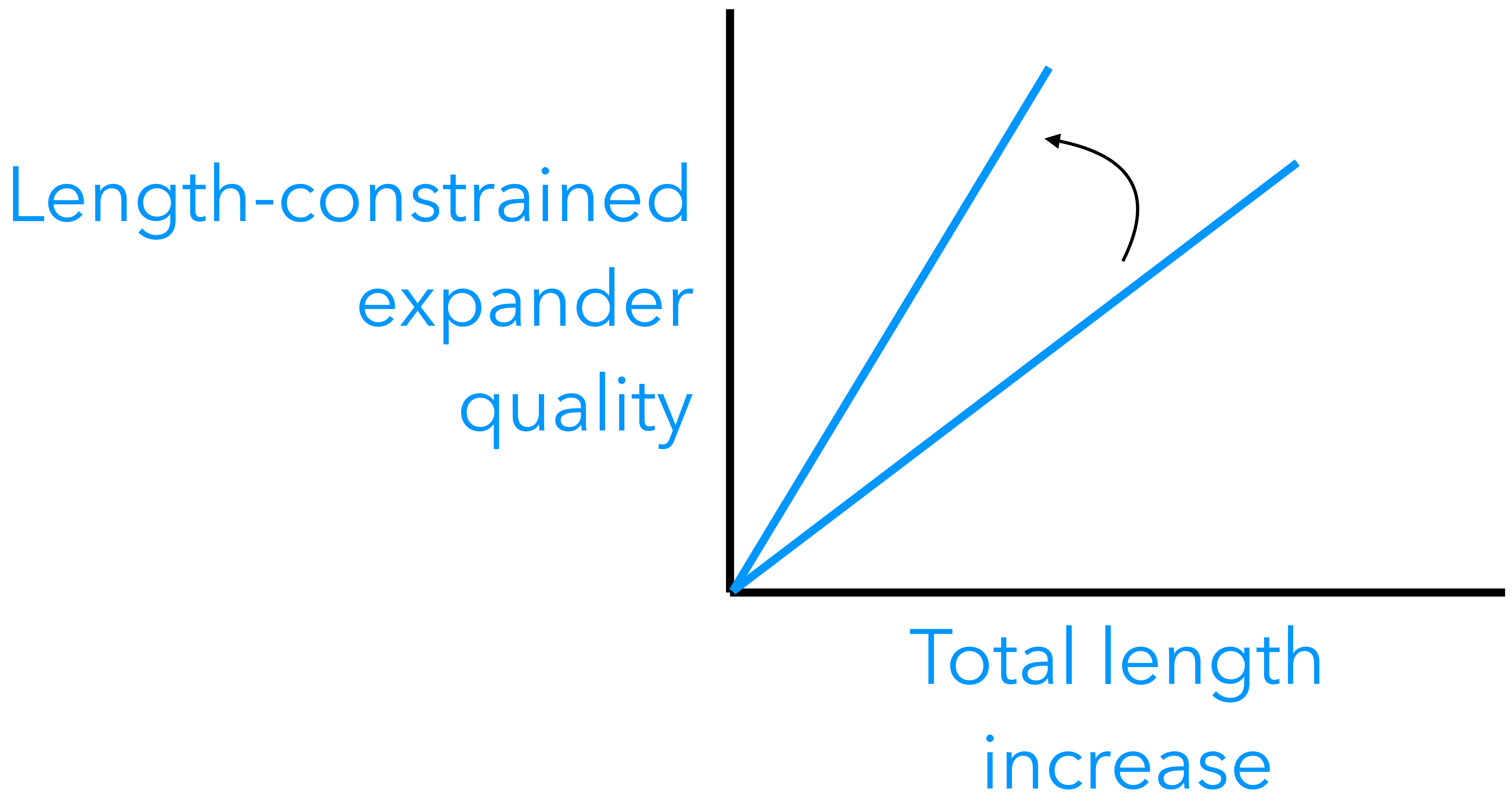
(h, s)-Length ϕ-Expander

Our Result

Our Main Result (Formally)

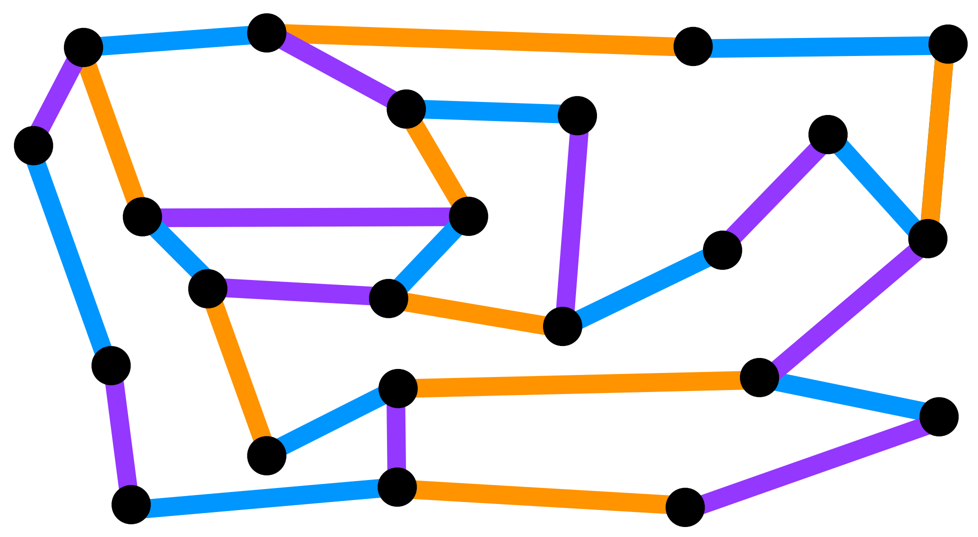
Theorem [BHHT]. Any graph G has an (h, s)-length ϕ-expander decomposition of size $s \cdot n^{O(1/s)} \cdot \phi m$ (proven simply)

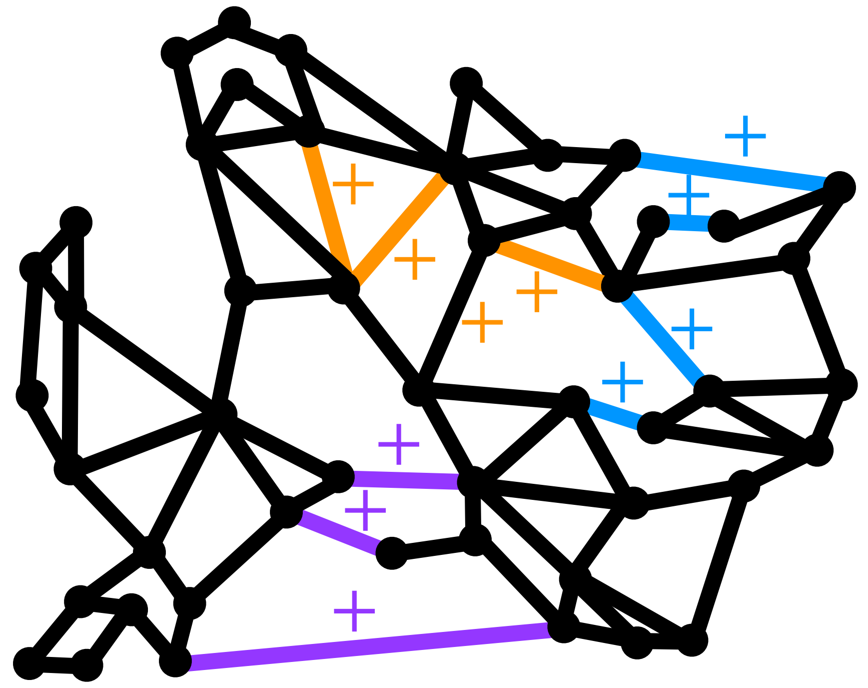
Previously [HHT]. $\log n \cdot s \cdot n^{O(1/s)} \cdot \phi m$ (proven not simply)

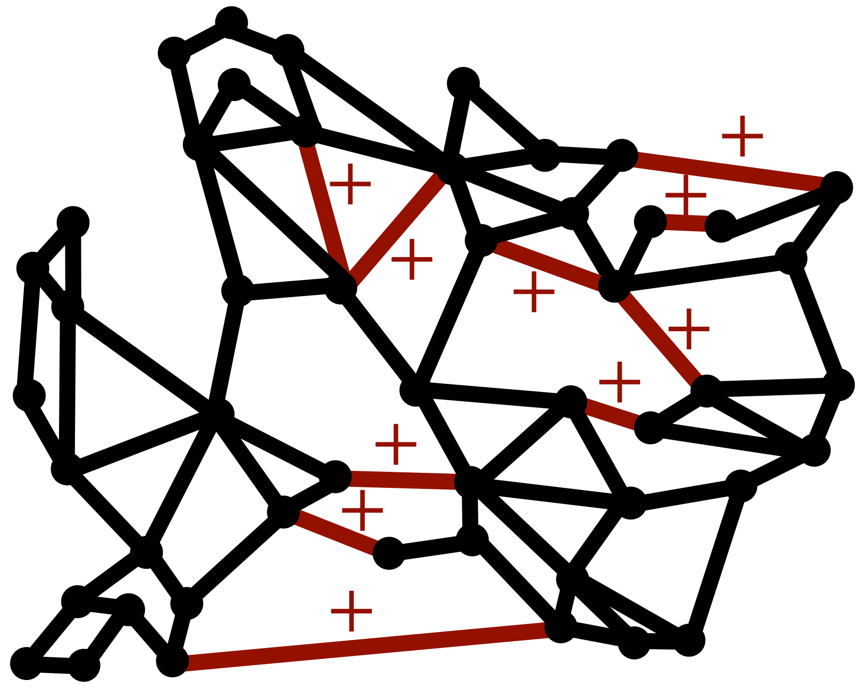


Proof Sketch of Our Result

Outline







Parallel Greedy Arboricity	Easy → (prior)	U of Cuts	Easy →	Existence of LC Decompositions

From ∪ 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)

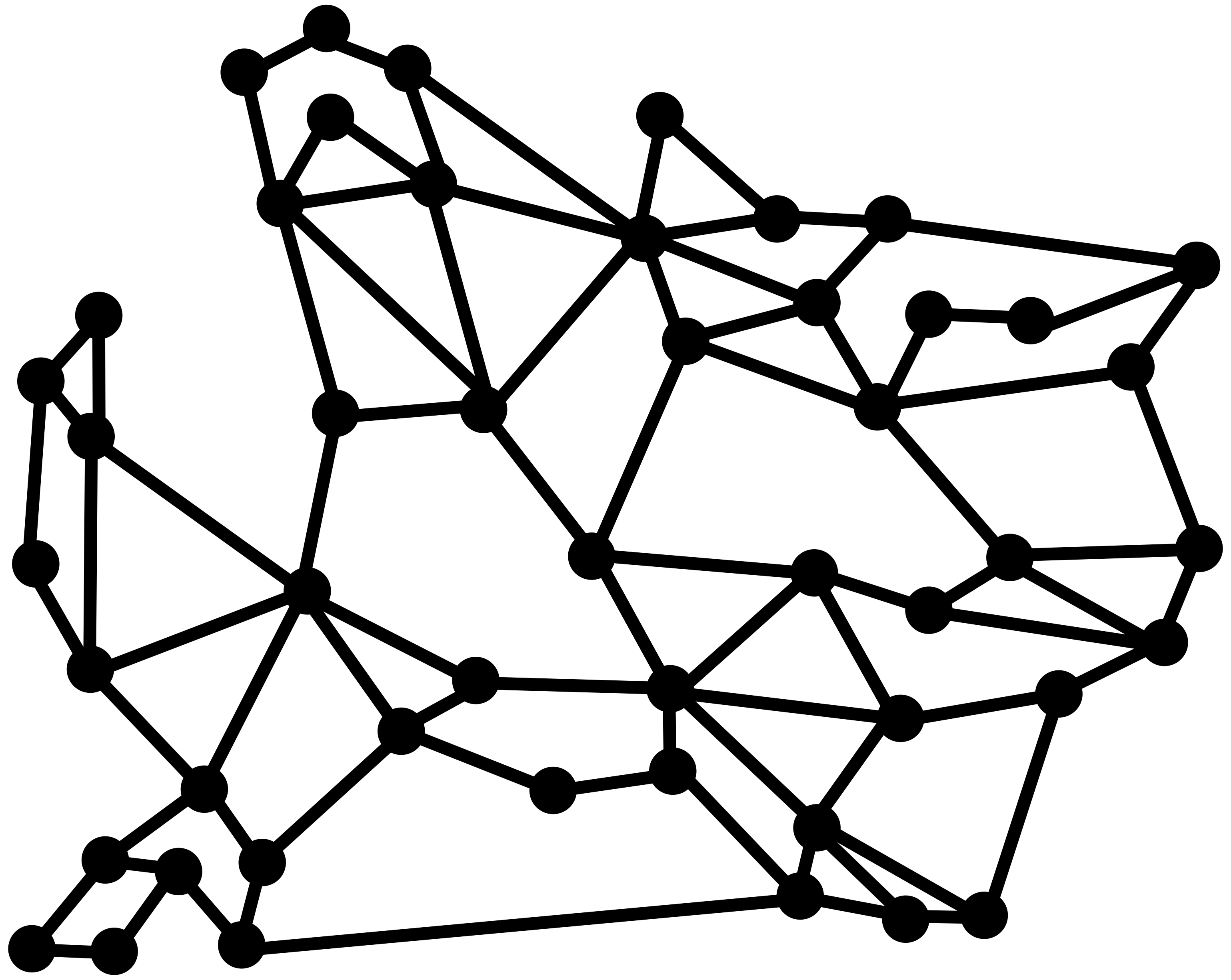
$i \leftarrow 0$, $G_0 \leftarrow G$

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 ∪ 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)

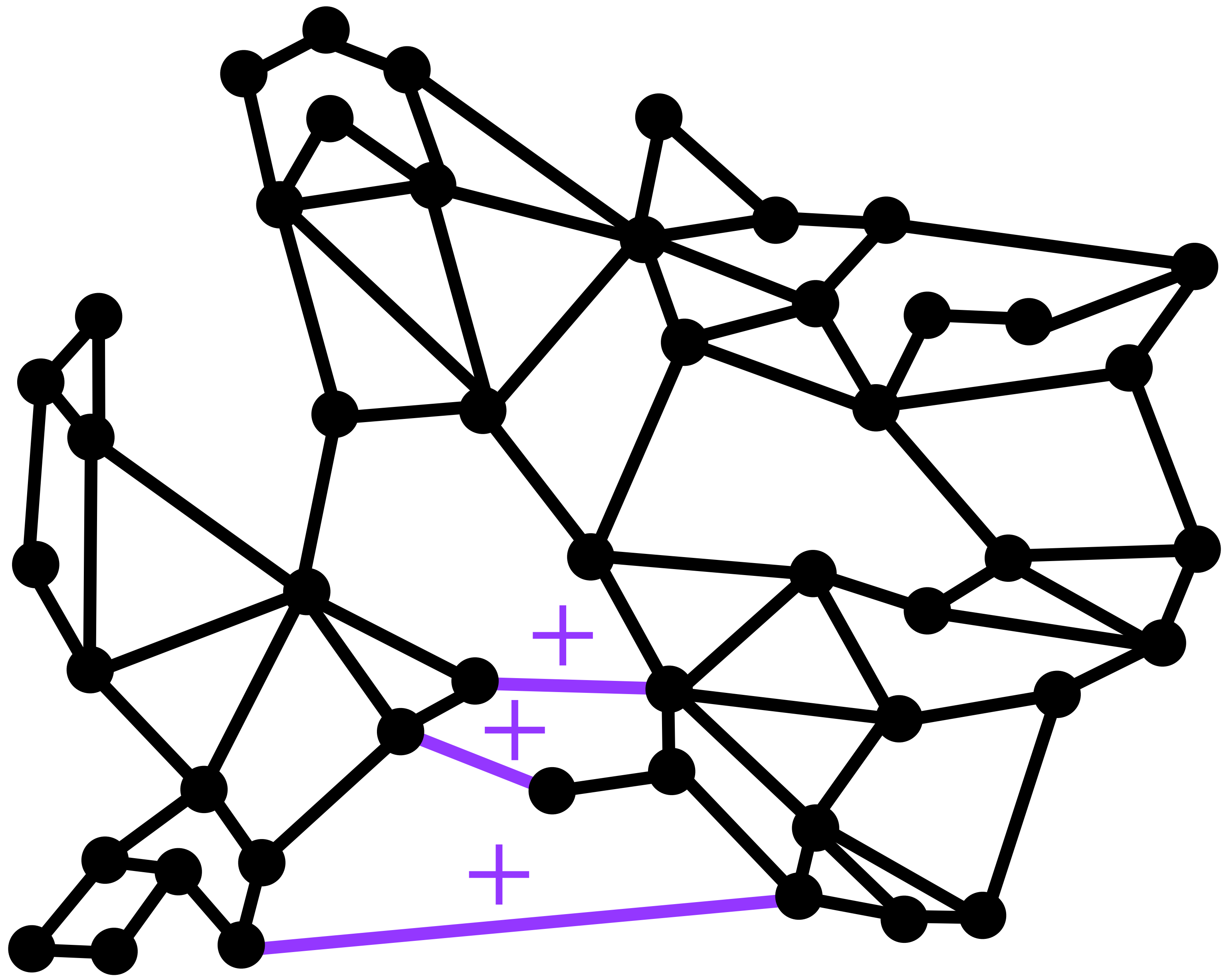
$i \leftarrow 0$, $G_0 \leftarrow G$

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 ∪ 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)

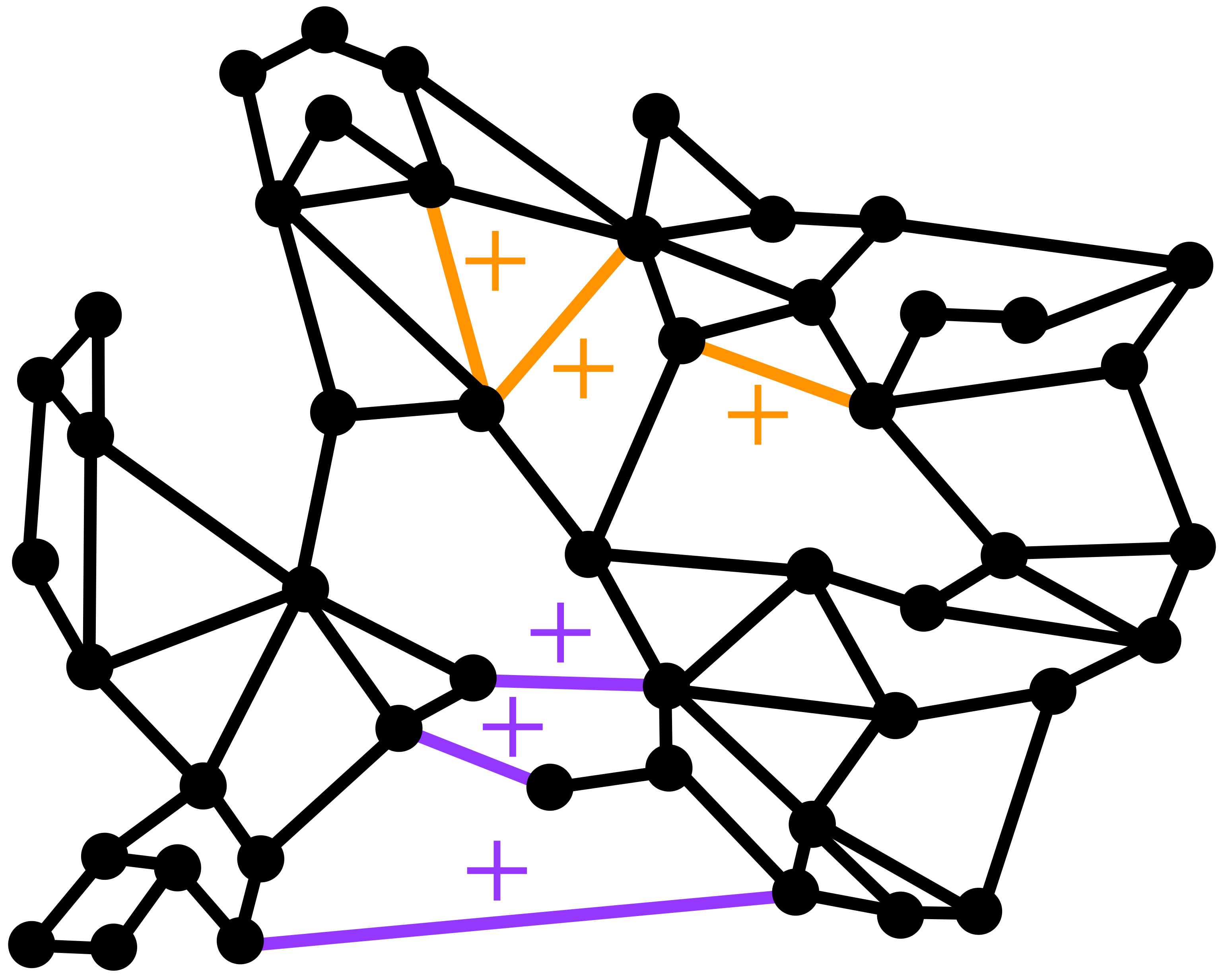
$i \leftarrow 0, G_0 \leftarrow G$

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 ∪ 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)

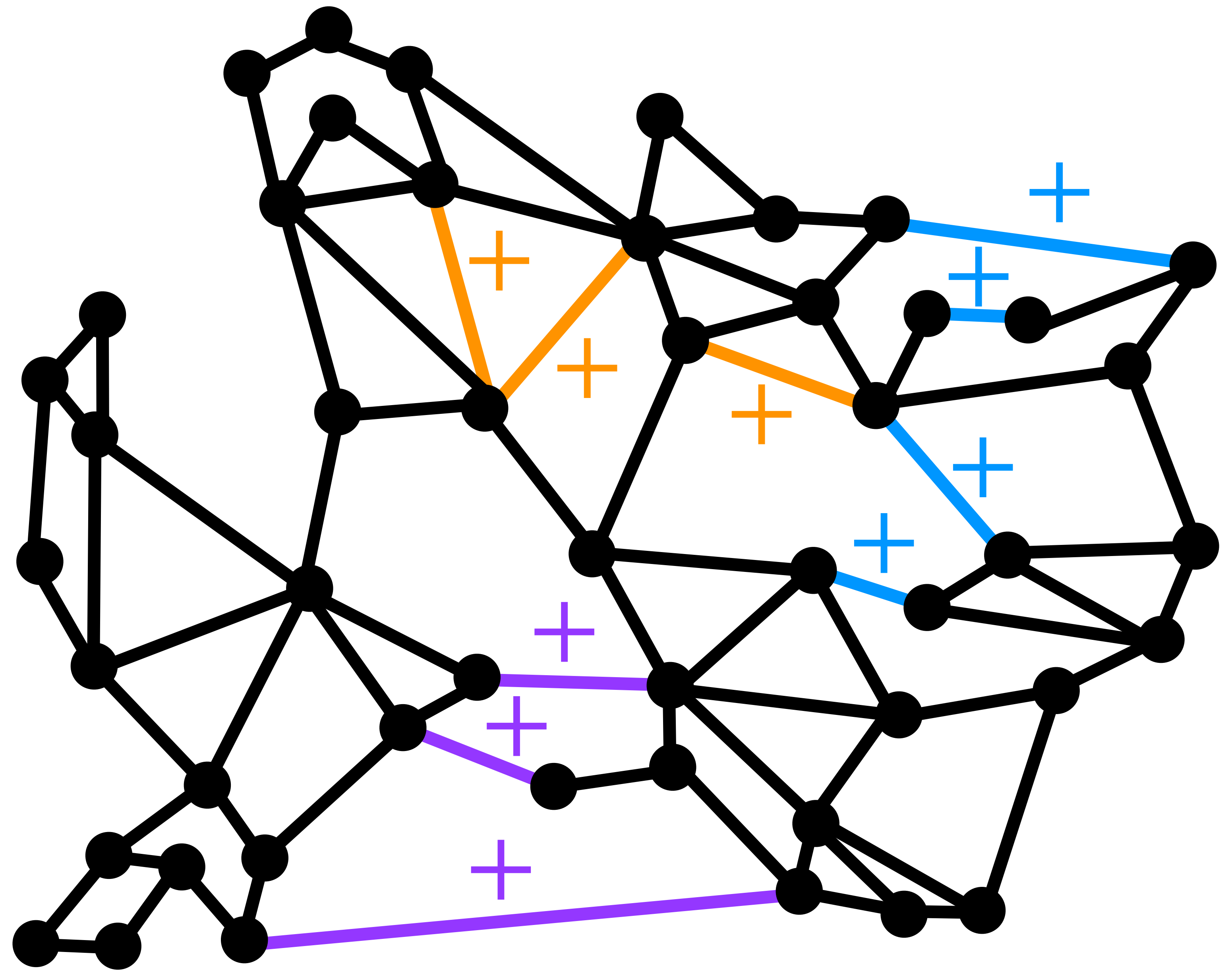
$i \leftarrow 0, G_0 \leftarrow G$

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 ∪ 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)

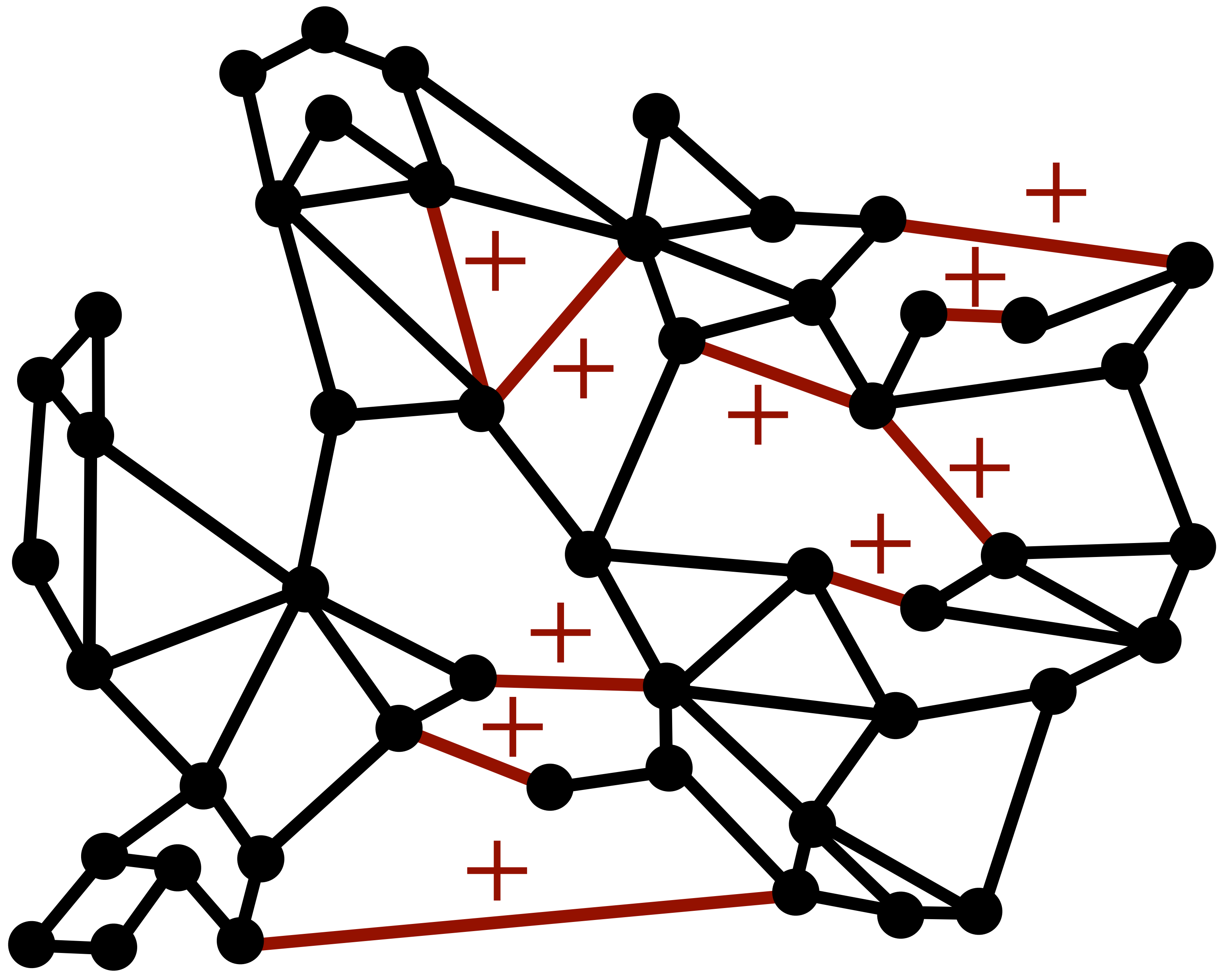
$i \leftarrow 0$, $G_0 \leftarrow G$

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 $C = \sum_i C_i$



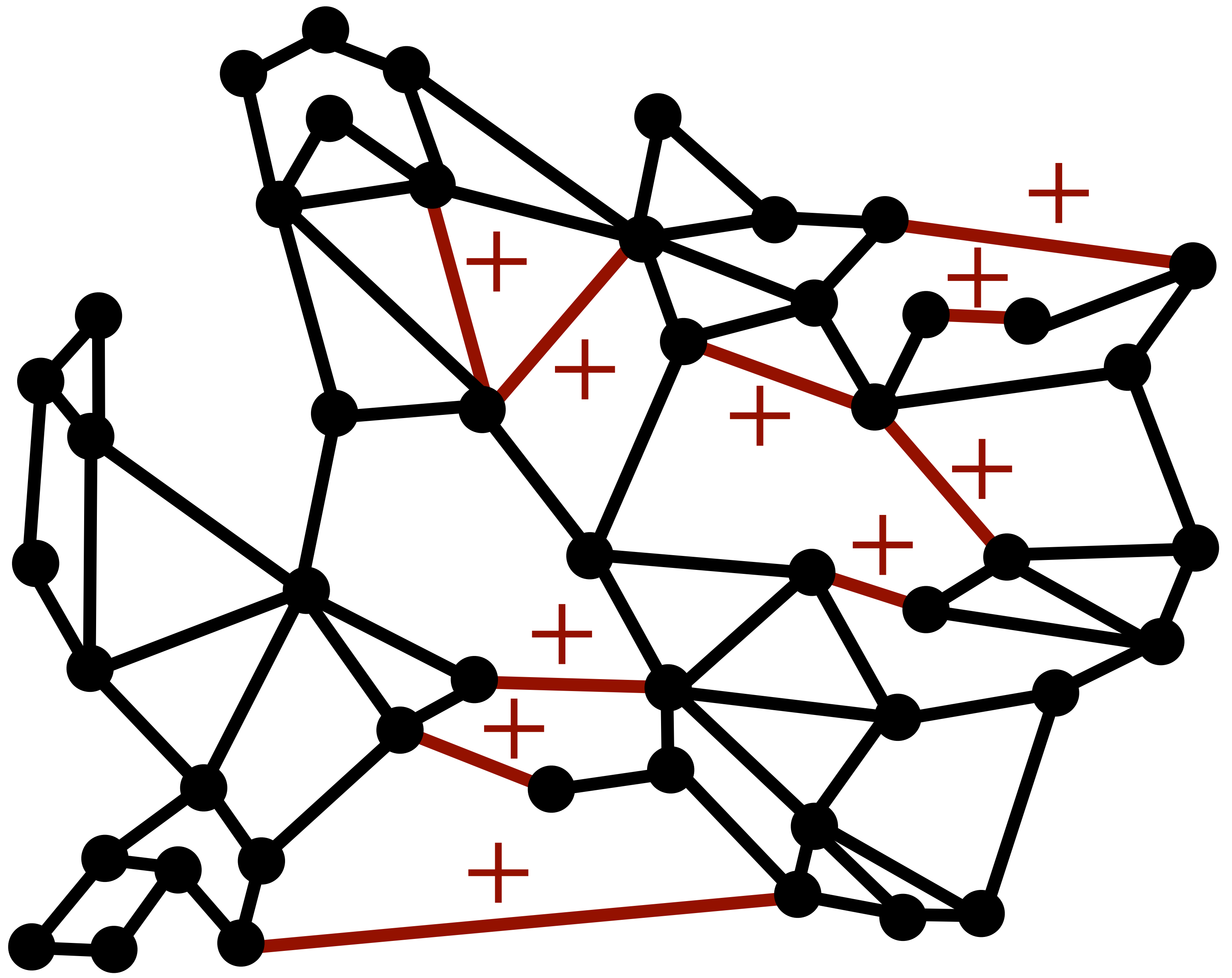
From ∪ of Cuts to LC Expander Decompositions

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)

$\cdots$ Return $C = \sum_i C_i$

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

🤔 But why is it small? 🤔

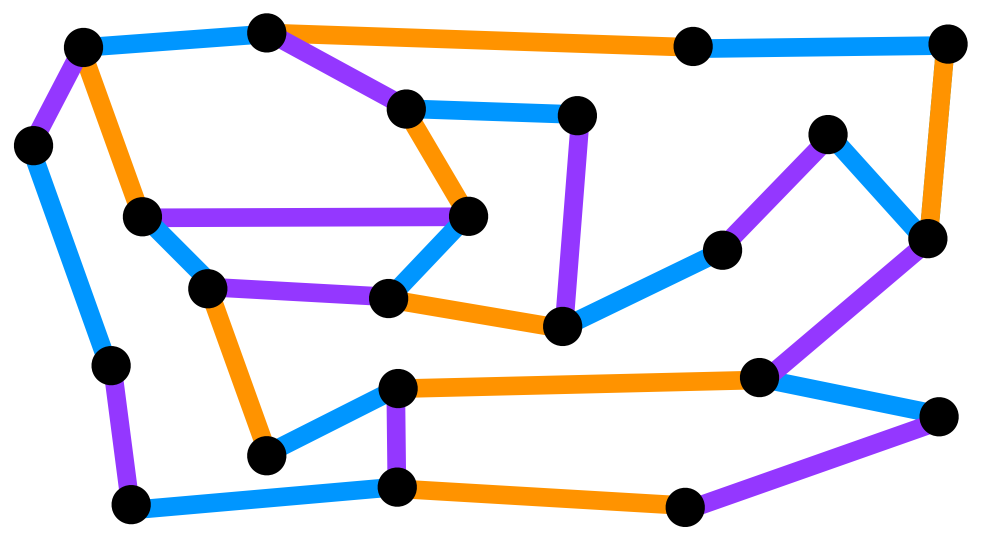


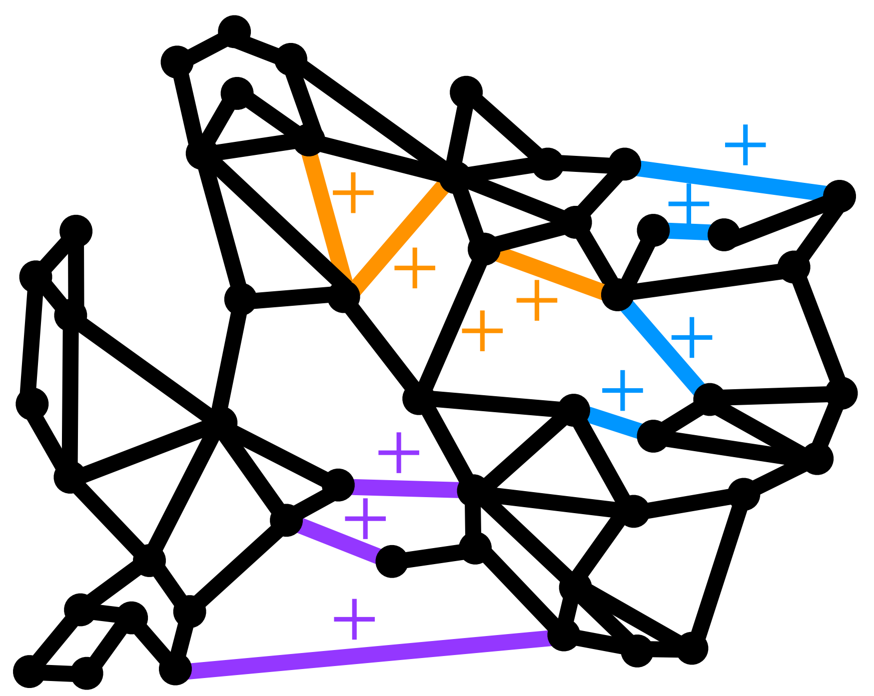
∪ of Cuts [BHHT]. C is an $\sim(h, s)$-length $(sn^{O(1/s)} \cdot \phi)$-sparse cut

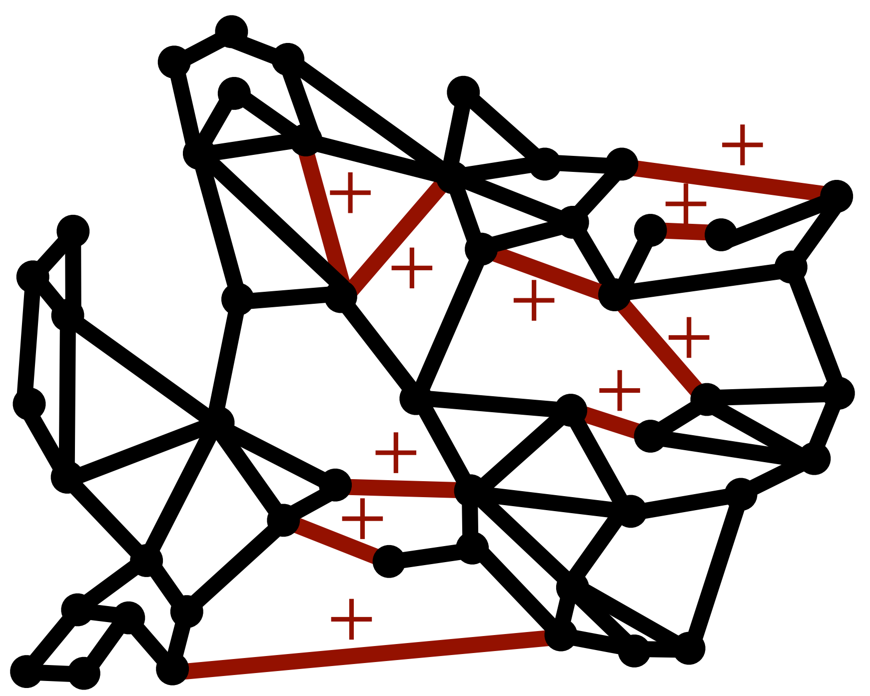
Any (h, s)-length $(sn^{O(1/s)} \cdot \phi)$-sparse cut has size at most $s \cdot n^{O(1/s)} \cdot \phi m$

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

Outline



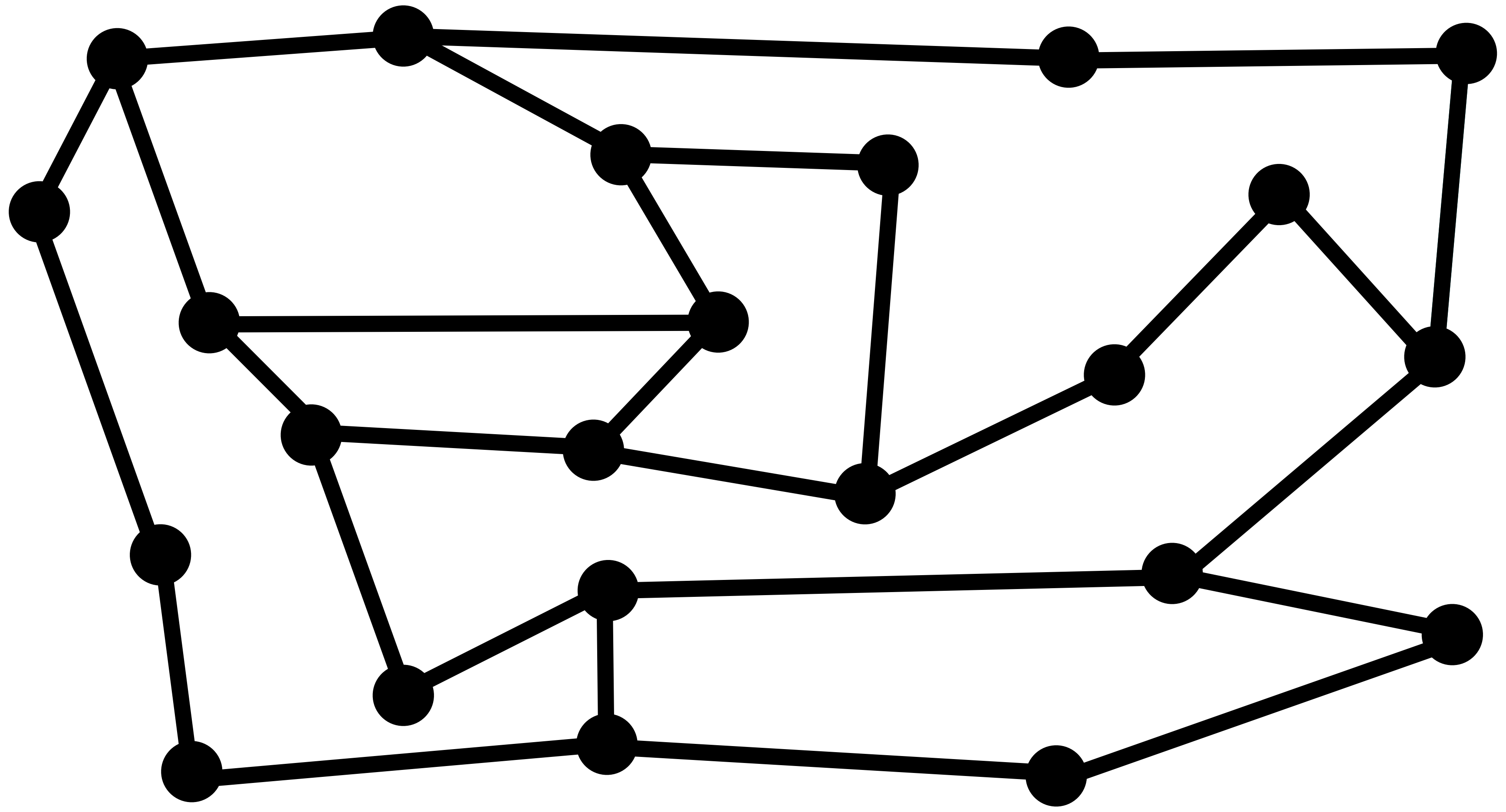




Parallel Greedy Arboricity —Easy (prior)→ U of Cuts —Easy→ Existence of LC Decompositions

From Parallel Greedy Arboricity to ∪ of Cuts

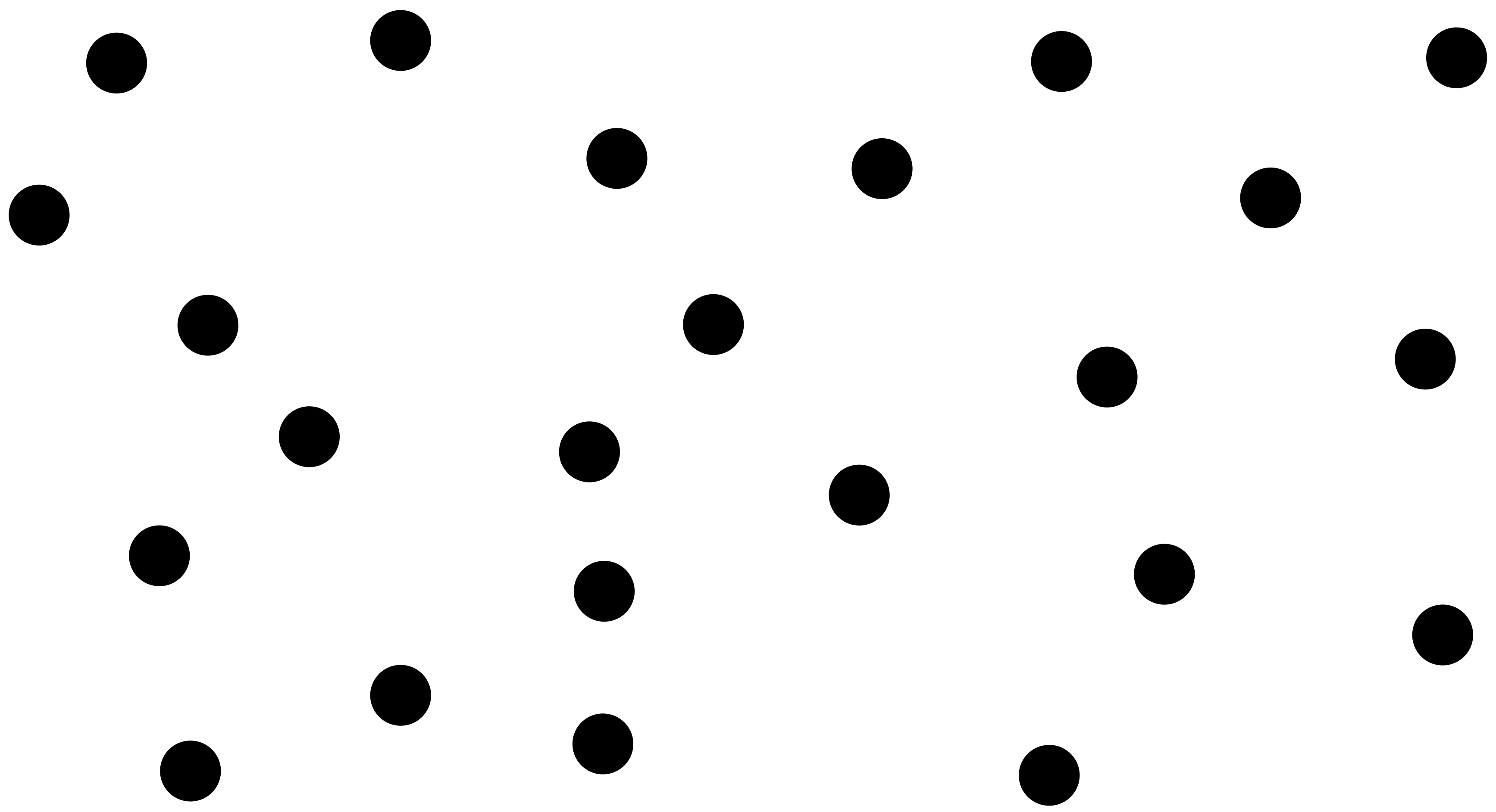
$G = (V, E)$ is an s**-parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \ldots$ where if $\{u, v\} \in M_i$ then u and v at least s-far in $\left(V, \bigcup_{j<i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to ∪ of Cuts

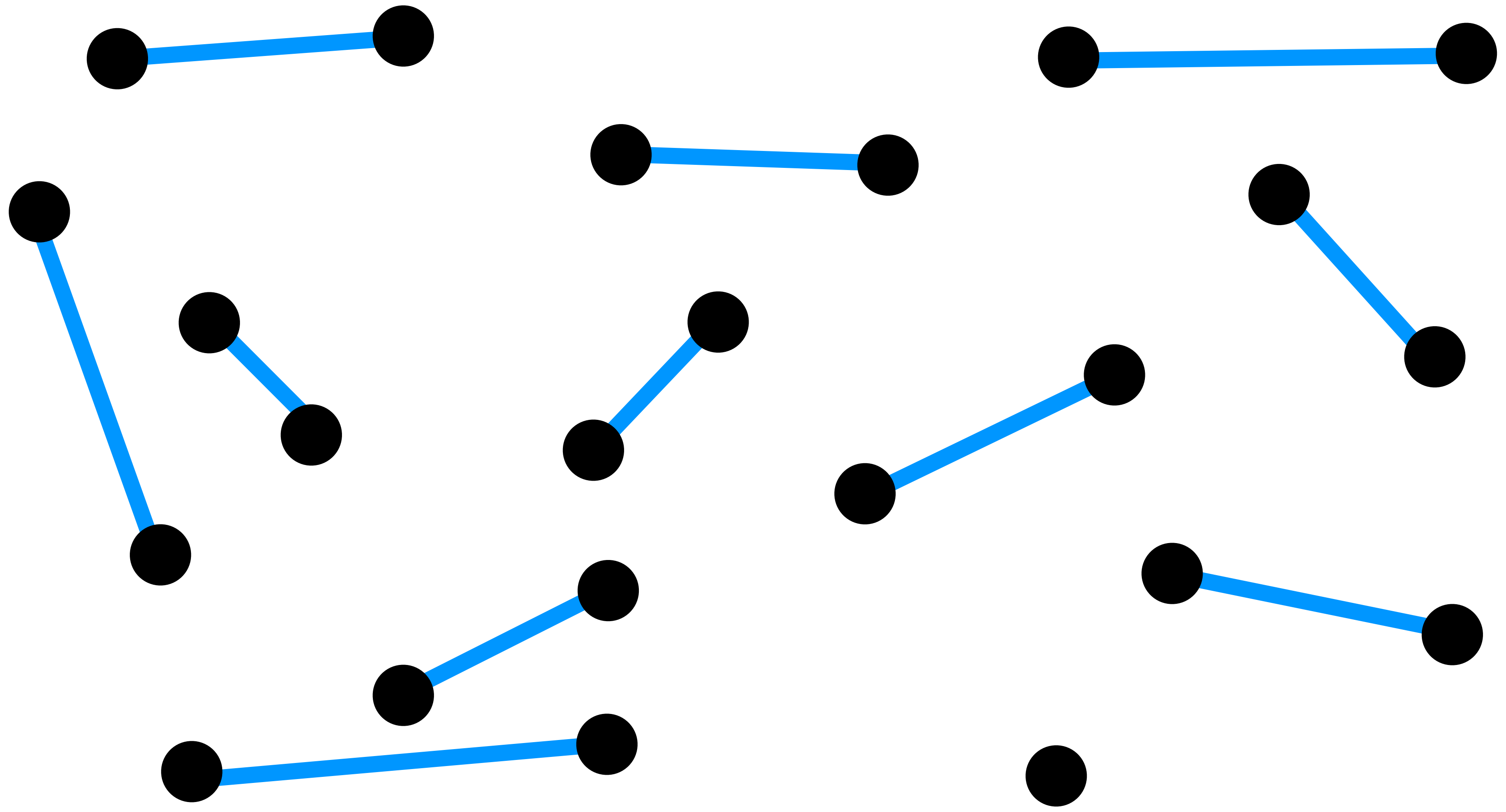
$G = (V, E)$ is an s-**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \ldots$ where if $\{u, v\} \in M_i$ then u and v at least s-far in $\left(V, \bigcup_{j<i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to ∪ of Cuts

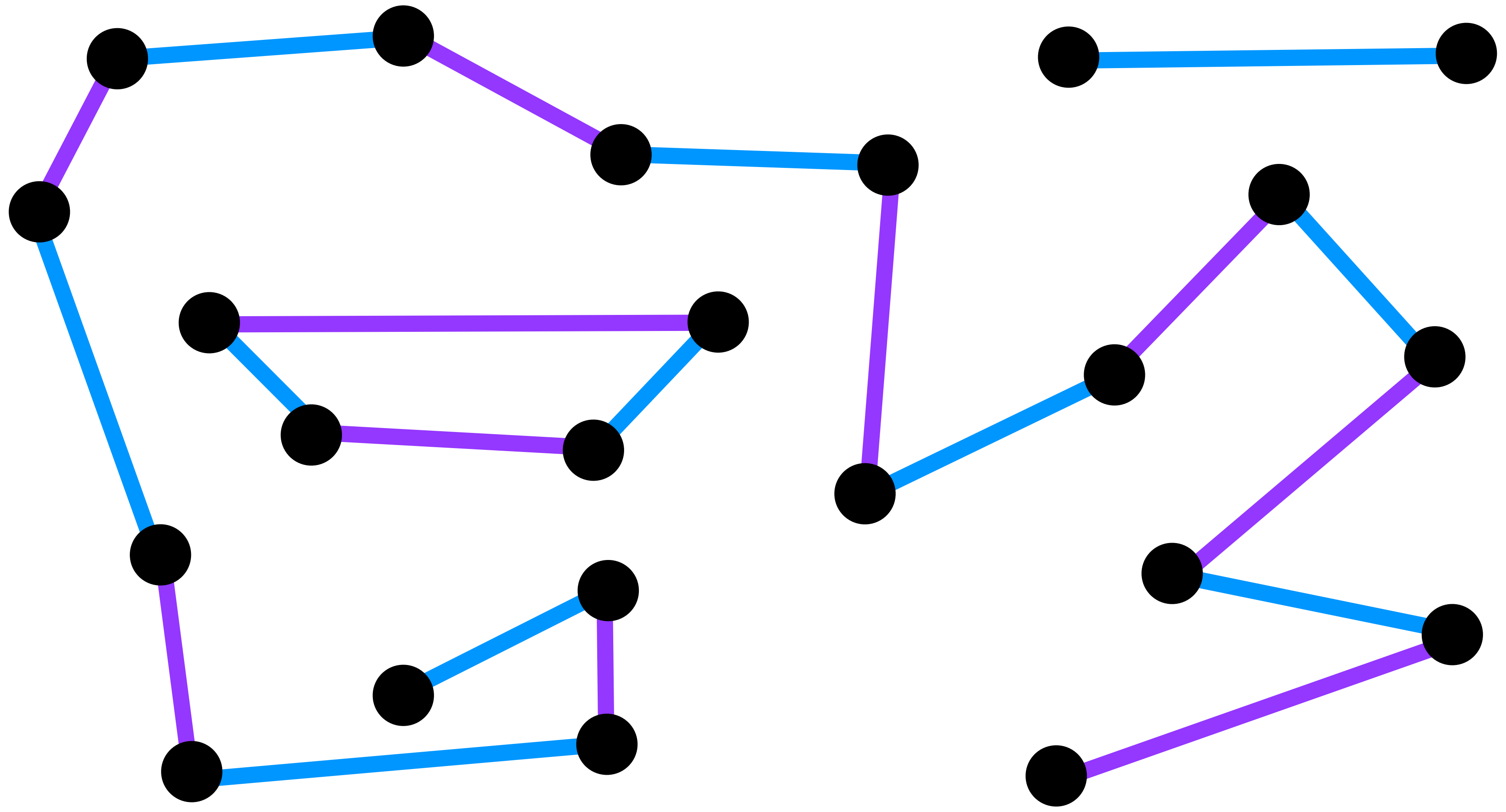
$G = (V, E)$ is an s**-parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \ldots$ where if $\{u, v\} \in M_i$ then u and v at least s-far in $\left(V, \bigcup_{j<i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to ∪ of Cuts

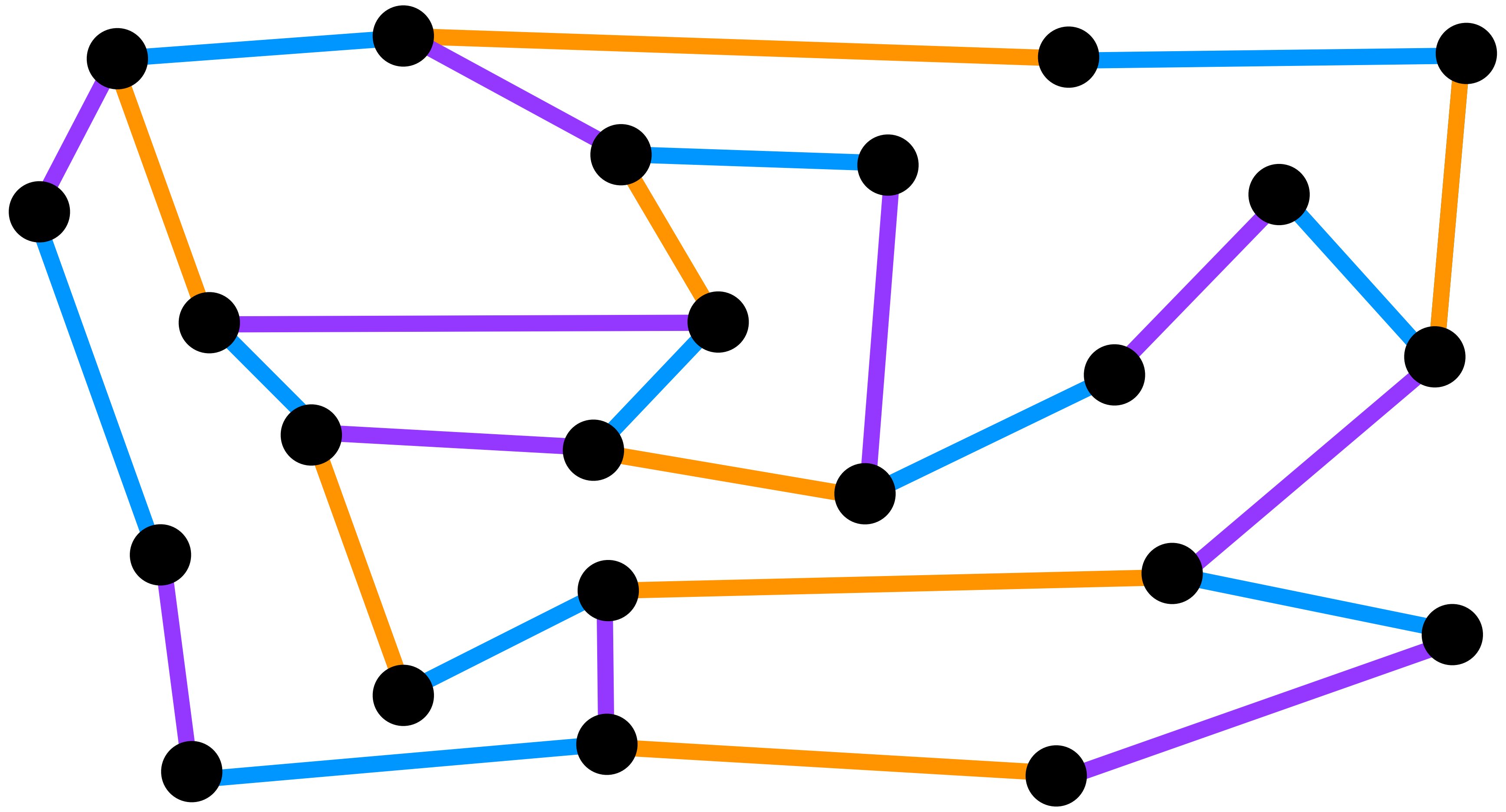
$G = (V, E)$ is an s**-parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \ldots$ where if $\{u, v\} \in M_i$ then u and v at least s-far in $\left(V, \bigcup_{j<i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to $\cup$ of Cuts

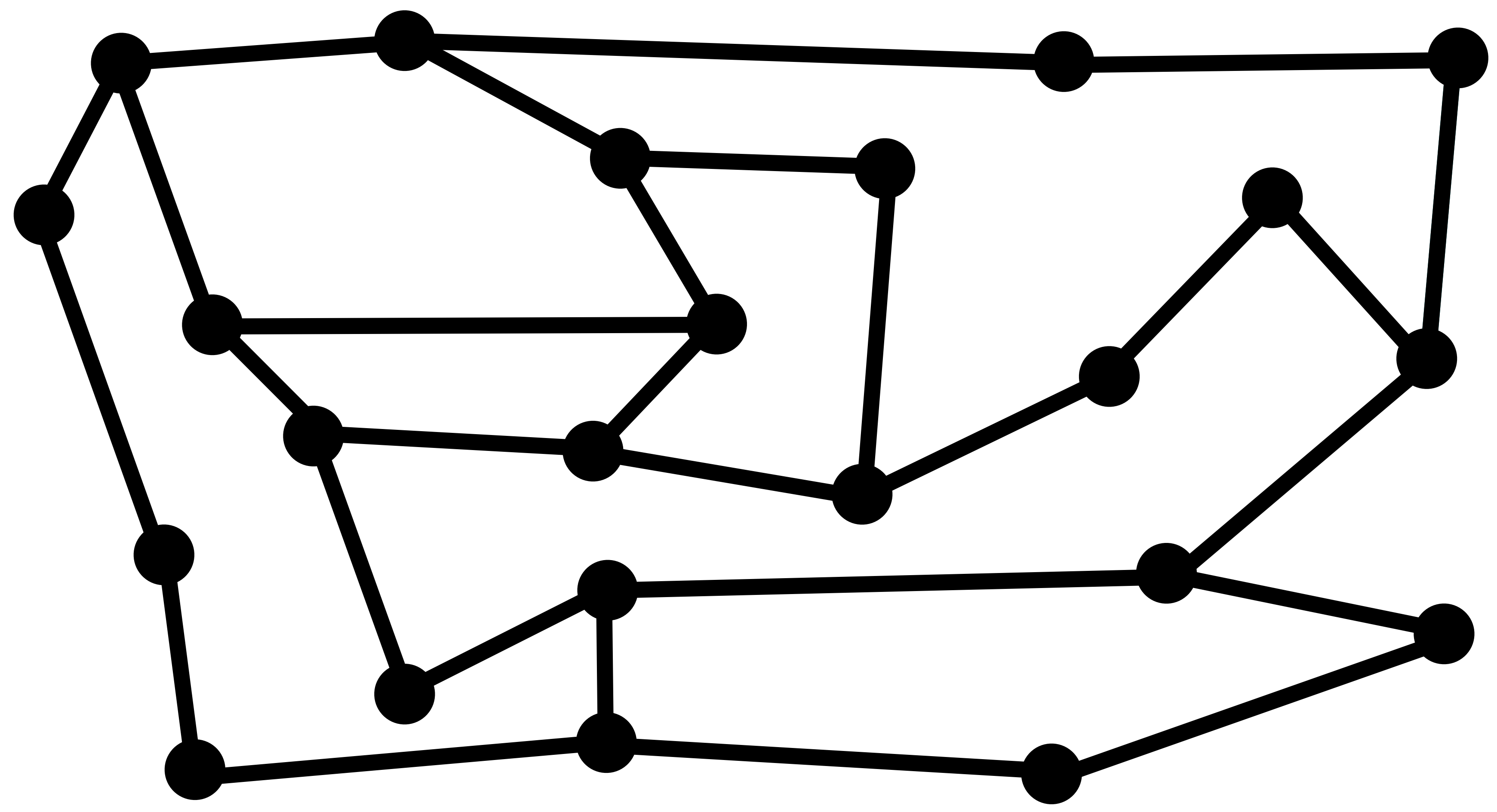
$G = (V, E)$ is an s-**parallel-greedy graph** if its edges decompose into matchings $E = M_1 \sqcup M_2 \sqcup \ldots$ where if $\{u, v\} \in M_i$ then u and v at least s-far in $\left(V, \bigcup_{j<i} M_j\right)$



12-parallel-greedy

From Parallel Greedy Arboricity to ∪ of Cuts

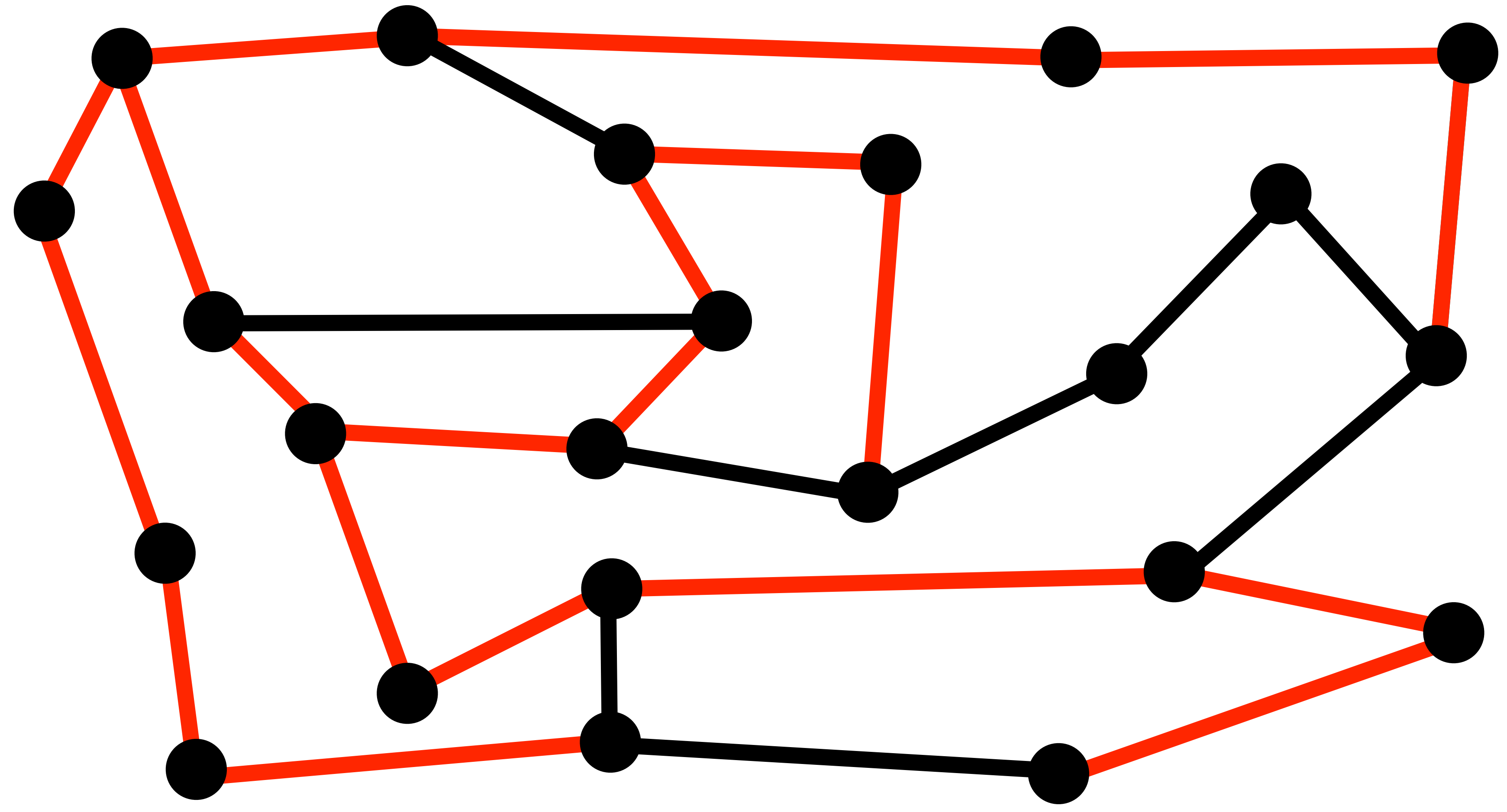
The **arboricity** of a graph is the minimum number of forests needed to cover all edges



Arboricity 2

From Parallel Greedy Arboricity to ∪ of Cuts

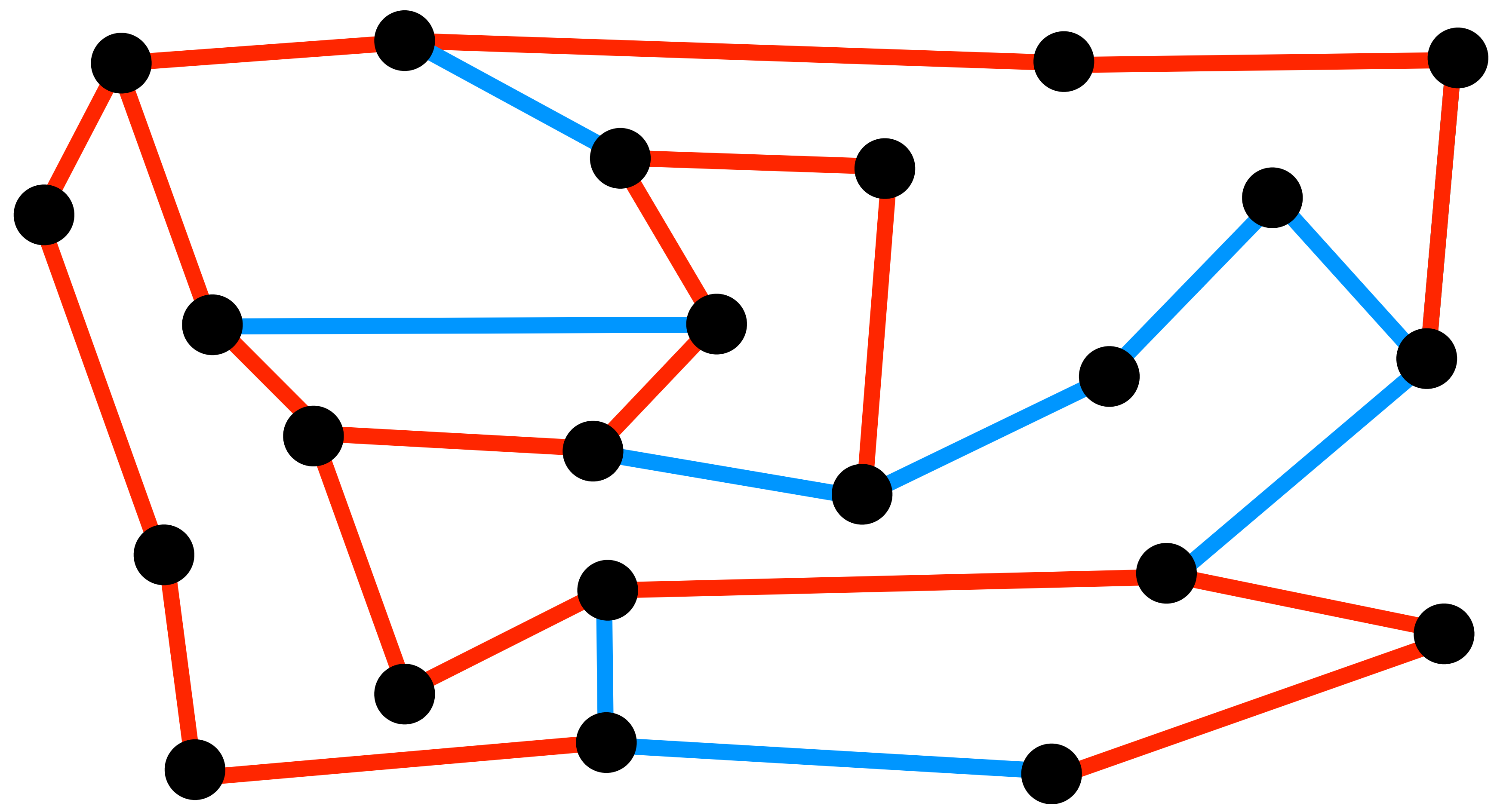
The **arboricity** of a graph is the minimum number of forests needed to cover all edges



Arboricity 2

From Parallel Greedy Arboricity to ∪ of Cuts

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



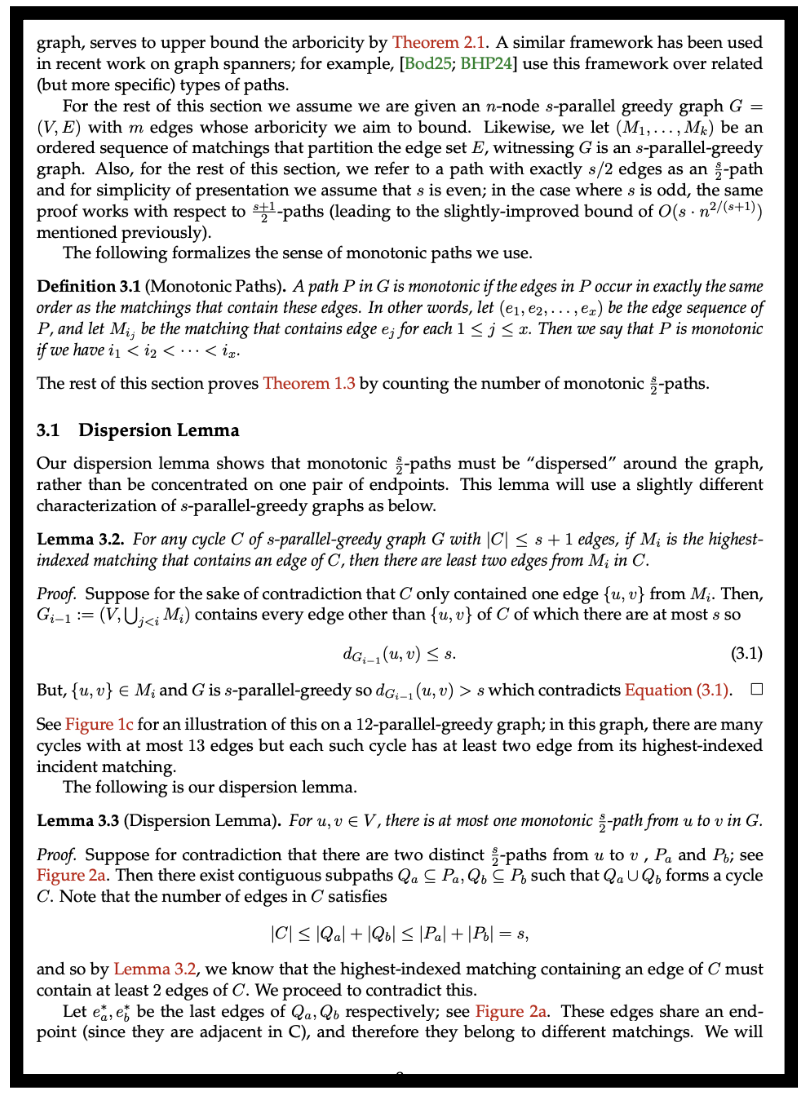
Arboricity 2

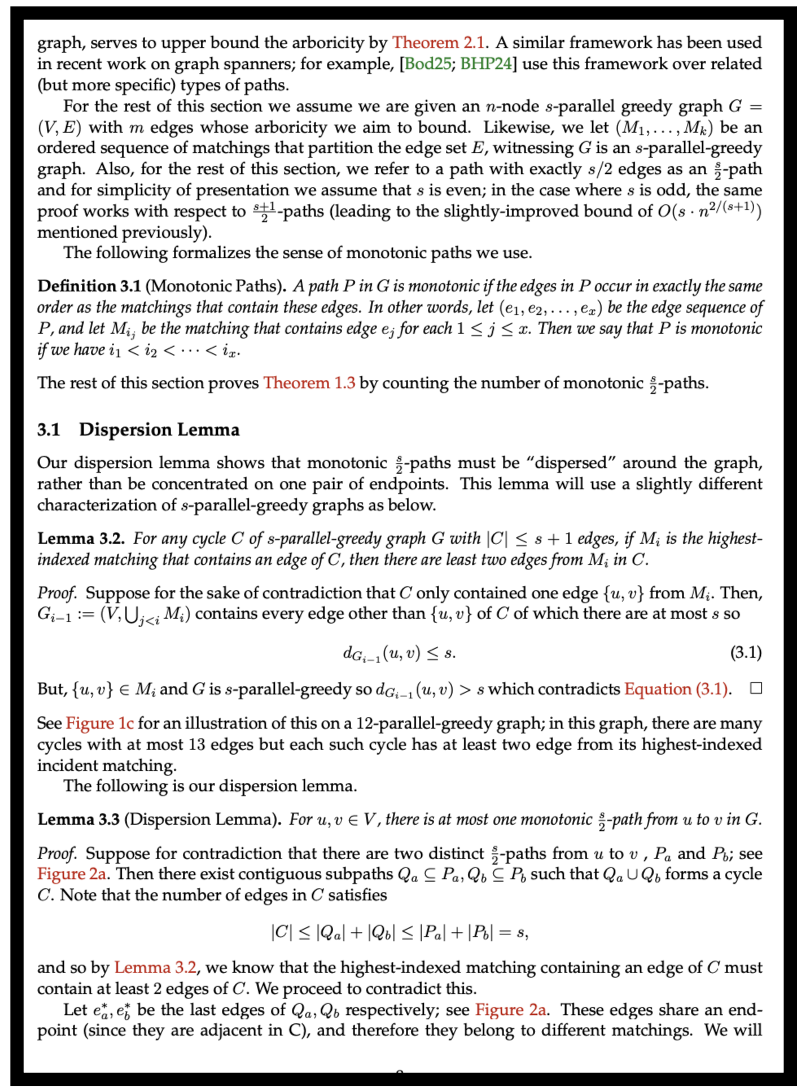
PG Arboricity to ∪ of Cuts [HHT]. C is (h, s)-length $\alpha\phi$-sparse in G where α is the arboricity of s-parallel-greedy graphs

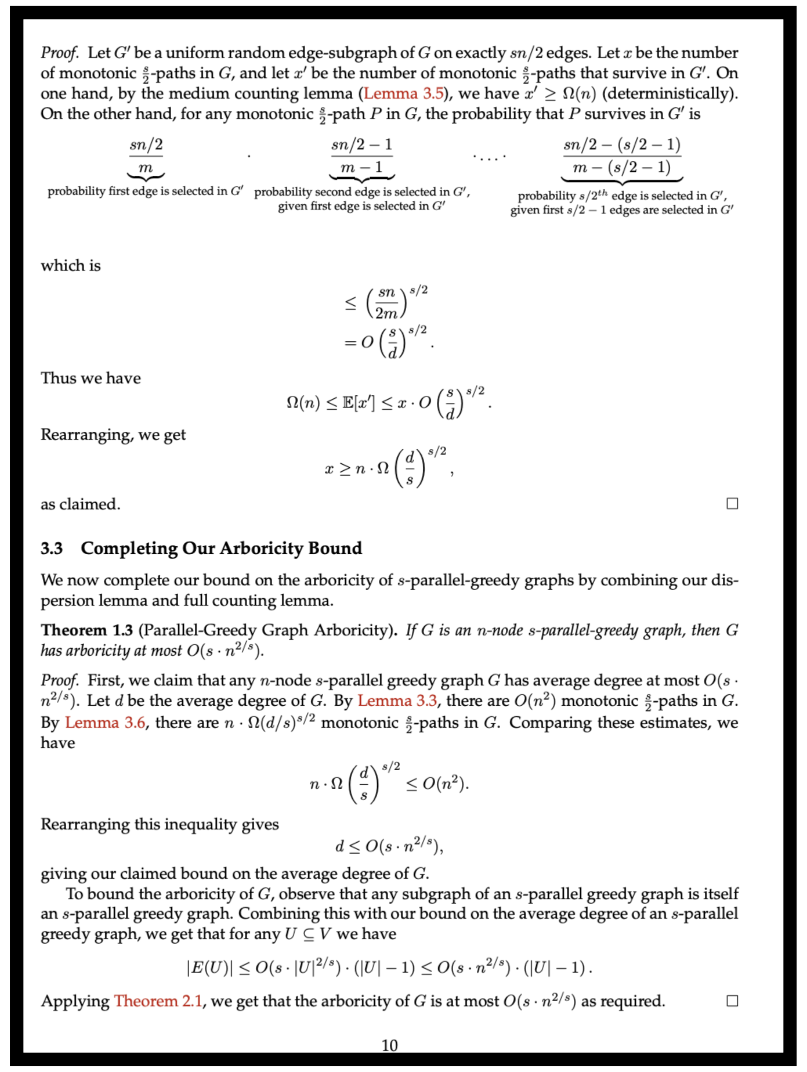
From Parallel Greedy Arboricity to ∪ of Cuts

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.



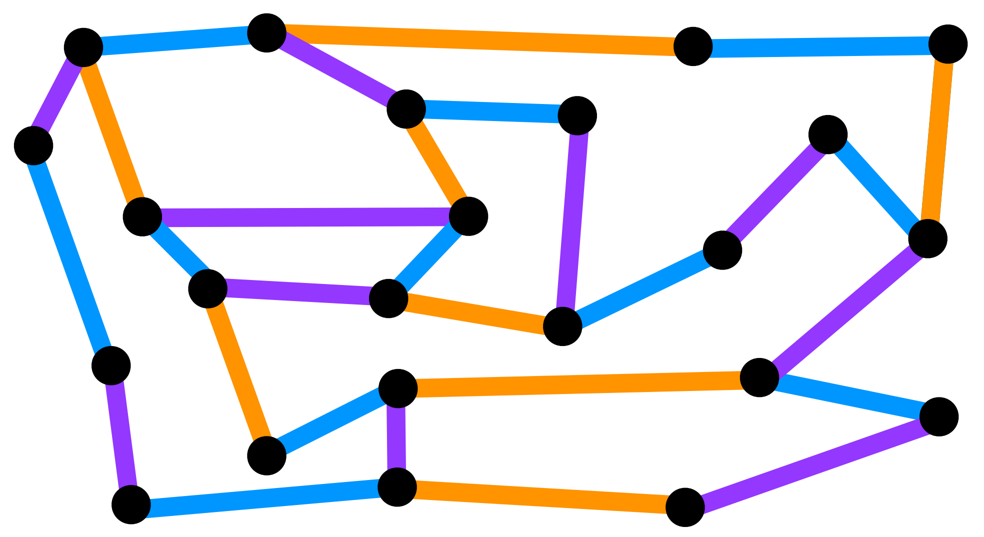


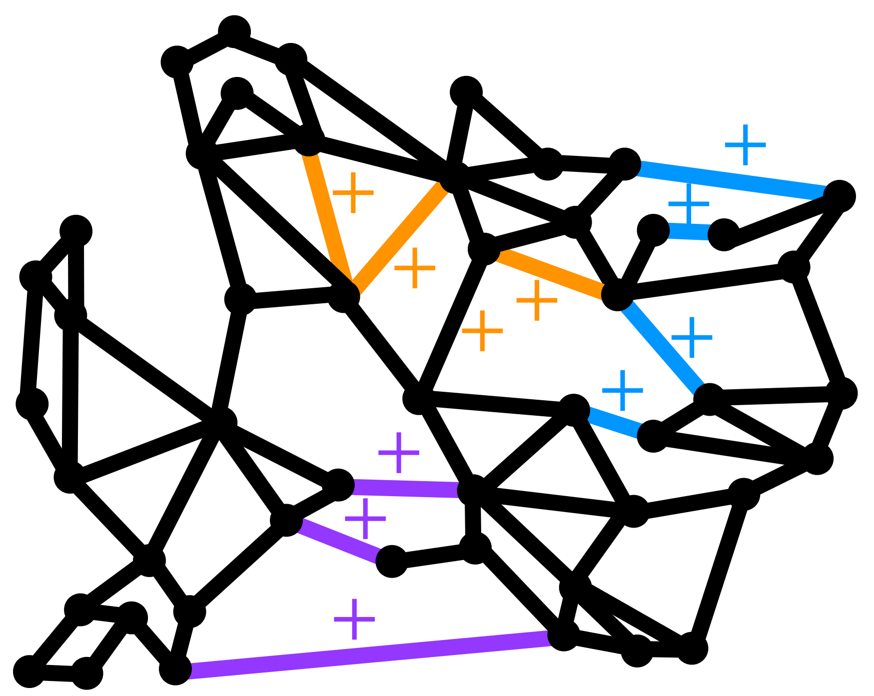


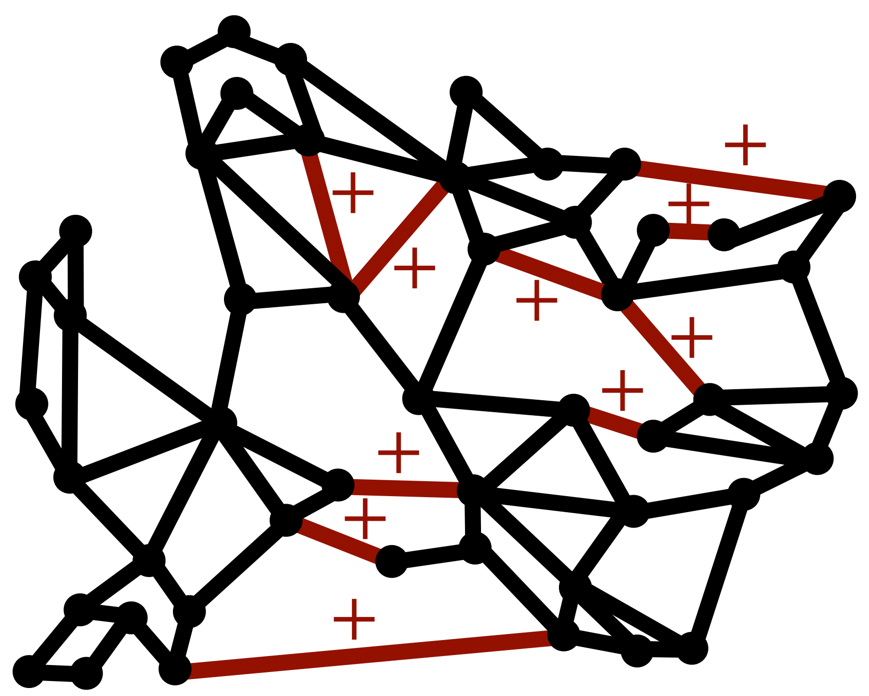
Summarizing

Thanks!

Summary







Parallel Greedy Arboricity → Easy (prior) → ∪ of Cuts → Easy → Existence of LC Decompositions