Minimum Cut and Minimum *k*-Cut in Hypergraphs via Branching Contractions

Kyle Fox

joint with

Debmalya Panigrahi and Fred Zhang
Duke University

Talk Outline

- A (short) lecture, some recent work, and a new result…

Talk Outline

- A (short) lecture, some recent work, and a new result…

  1. Minimum cuts in graphs via branching random contractions [Karger, Stein '96]

Talk Outline

- A (short) lecture, some recent work, and a new result…

1. Minimum cuts in graphs via branching random contractions [Karger, Stein ’96]

2. Minimum cuts in hypergraphs via *serial* random contractions
[Ghaffari *et al.* ’17; Chandrasekharan *et al.* ’18]

Talk Outline

- A (short) lecture, some recent work, and a new result…

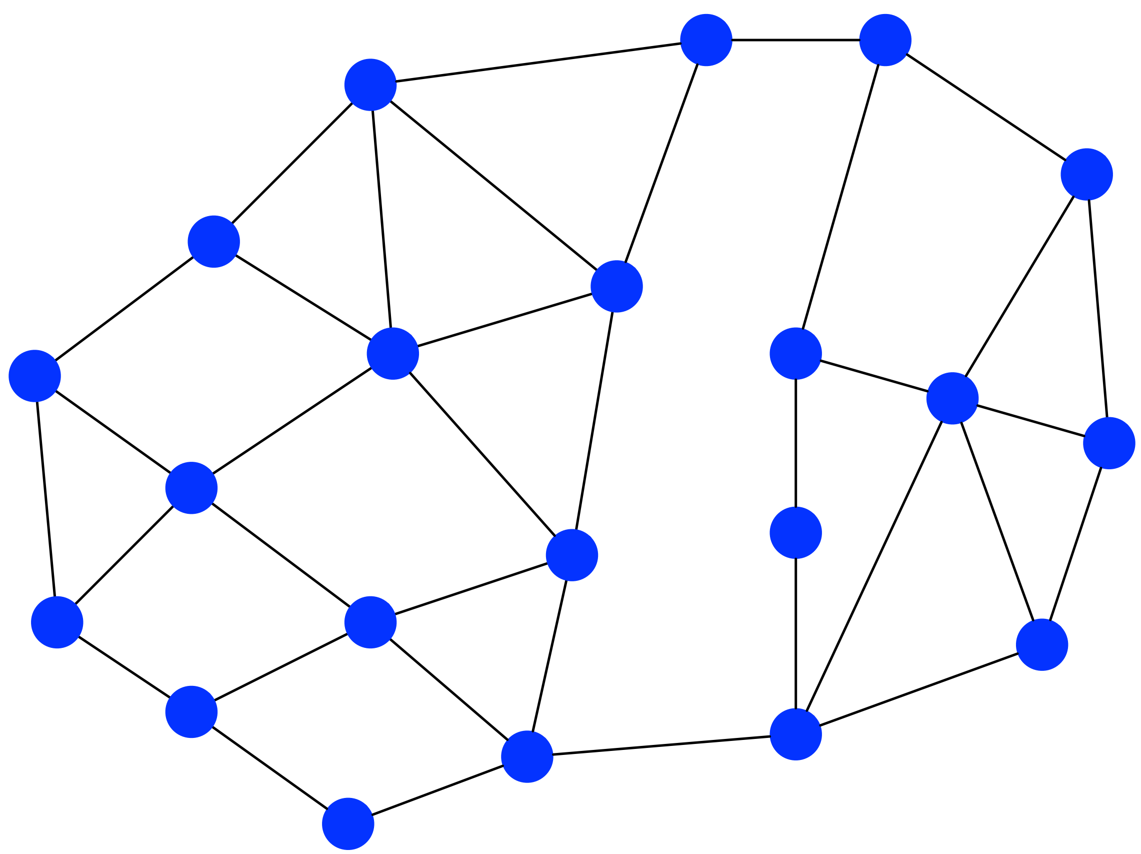
1. Minimum cuts in graphs via branching random contractions [Karger, Stein ’96]

2. Minimum cuts in hypergraphs via *serial* random contractions [Ghaffari *et al.* ’17; Chandrasekharan *et al.* ’18]

3. Faster minimum cuts in hypergraphs via *branching* contractions [**F**PZ]

Minimum Cuts

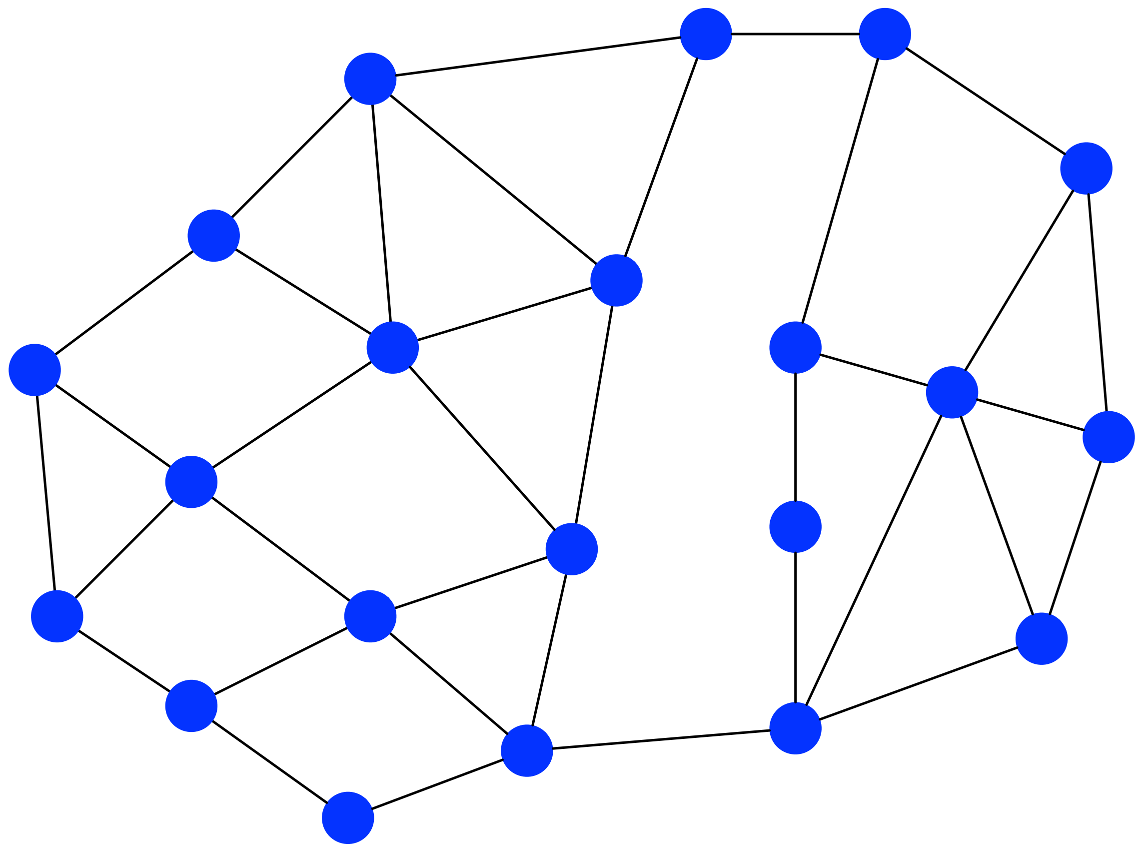
- A ***cut*** is a set of edges whose removal creates at least 2 connected components; a ***k-cut*** creates at least *k* components



Minimum Cuts

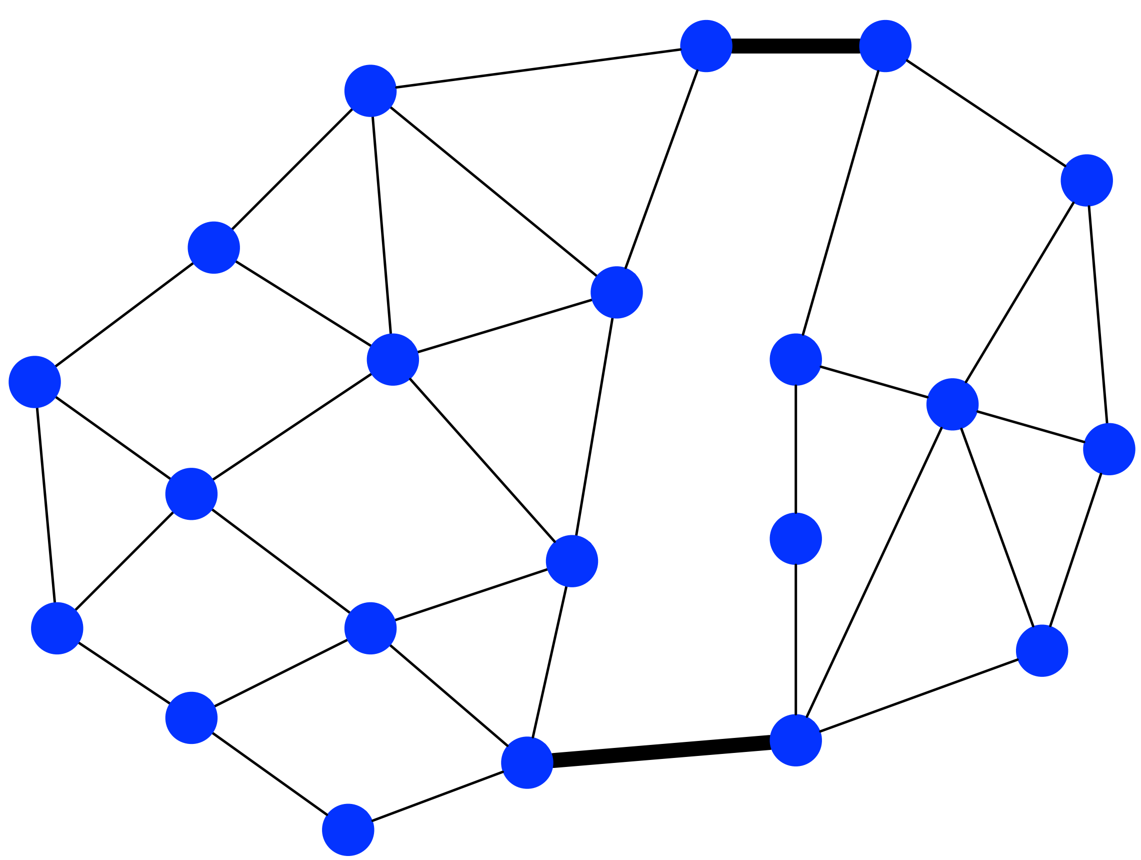
- A ***cut*** is a set of edges whose removal creates at least 2 connected components; a ***k-cut*** creates at least *k* components

- A ***minimum (k-)cut*** has as few edges as possible



Minimum Cuts

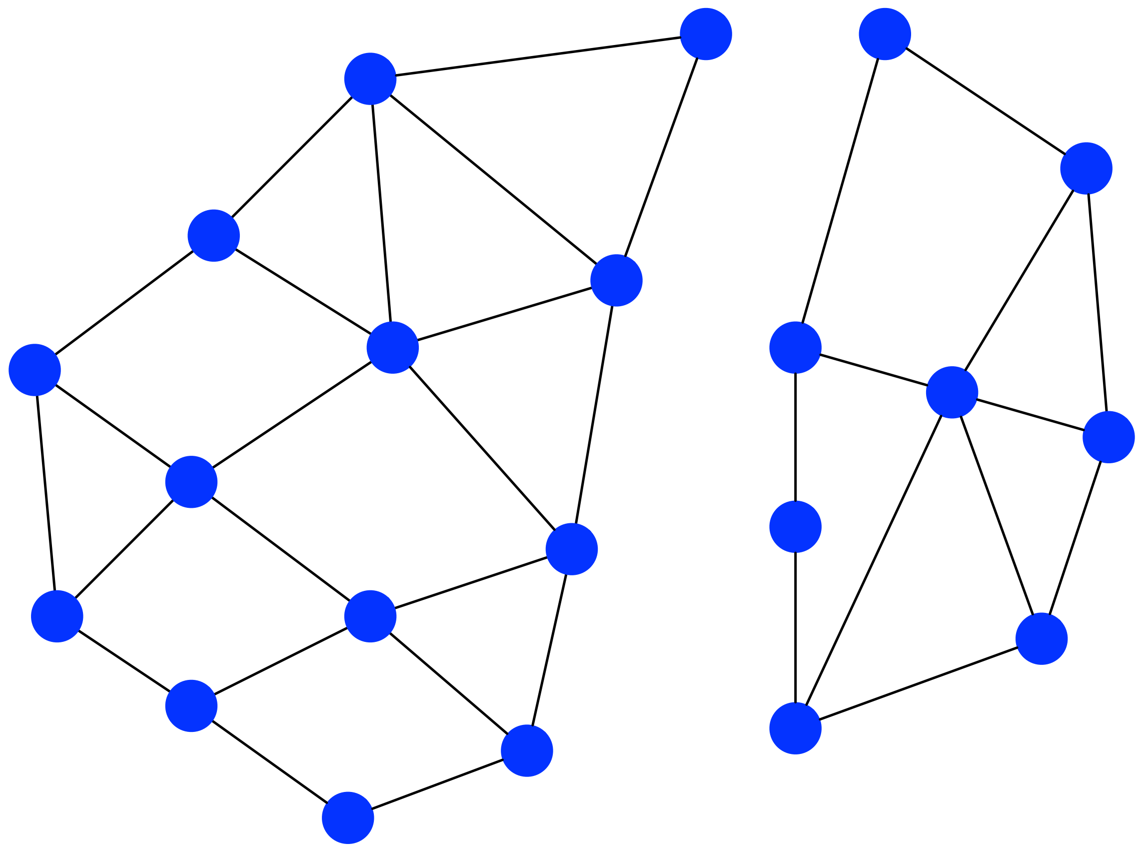
- A ***cut*** is a set of edges whose removal creates at least 2 connected components; a ***k-cut*** creates at least *k* components
- A ***minimum (k-)cut*** has as few edges as possible



Minimum Cuts

- A ***cut*** is a set of edges whose removal creates at least 2 connected components; a ***k-cut*** creates at least *k* components

- A ***minimum (k-)cut*** has as few edges as possible



Strategies

- We'll focus on finding minimum (2-)cuts (*i.e.*, minimum cuts)

Strategies

- We'll focus on finding minimum (2-)cuts (*i.e.*, minimum cuts)
- Given a graph with n vertices and m edges…

Strategies

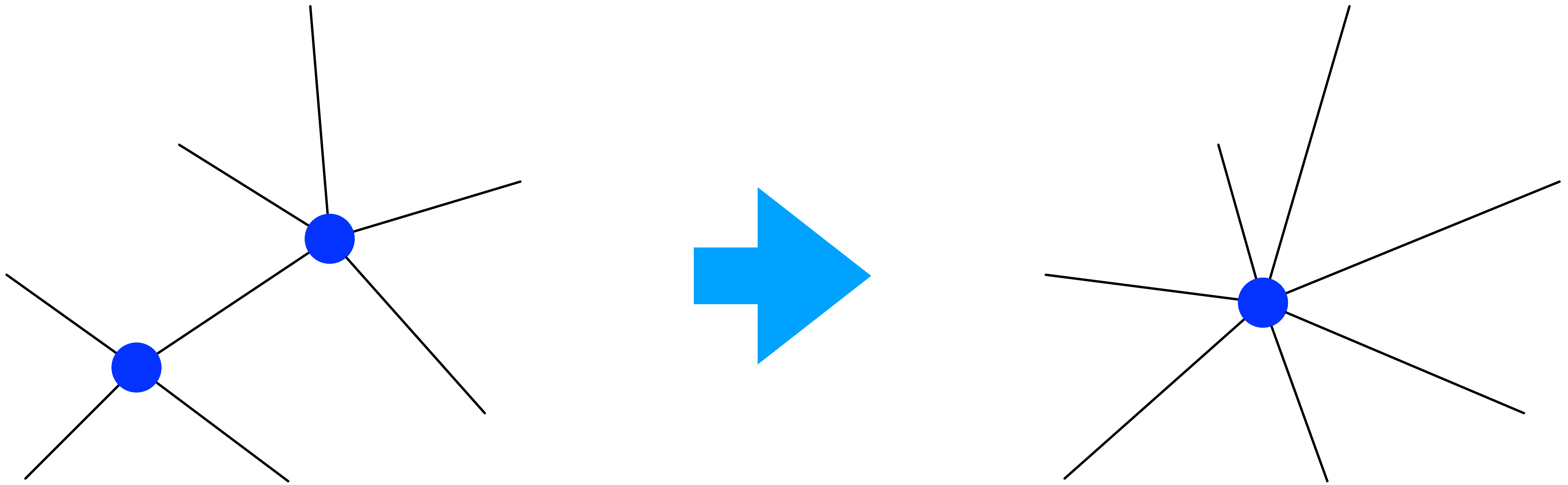
- We'll focus on finding minimum (2-)cuts (*i.e.*, minimum cuts)
- Given a graph with n vertices and m edges…
  - Compute n - 1 minimum s,t-cuts in O(mn^2) time

Strategies

- We'll focus on finding minimum (2-)cuts (*i.e.*, minimum cuts)

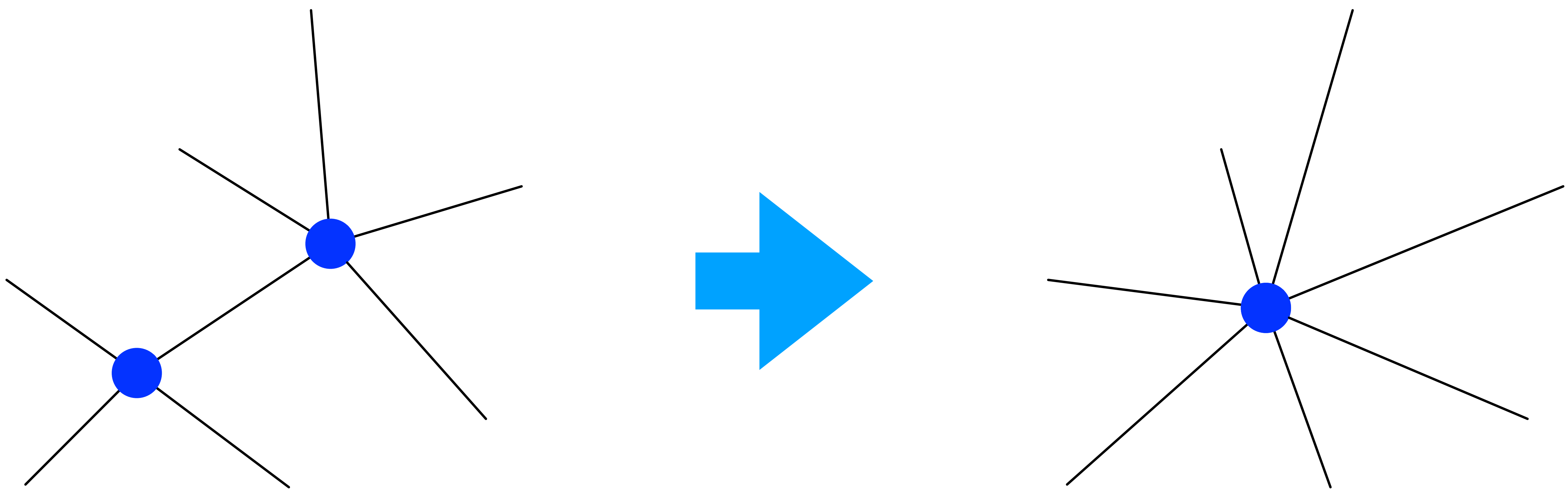
- Given a graph with n vertices and m edges…
  - Compute $n - 1$ minimum s,t-cuts in $O(mn^2)$ time
  - Guess an edge outside a minimum cut (uniformly at random) and ***contract***…

Edge Contraction



- Identify endpoints and remove loops

Edge Contraction



- Identify endpoints and remove loops
- Like committing to *not* separating the endpoints with a cut

Contractions Preserve Cuts (Probably)

- **Lemma:** For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$

Contractions Preserve Cuts (Probably)

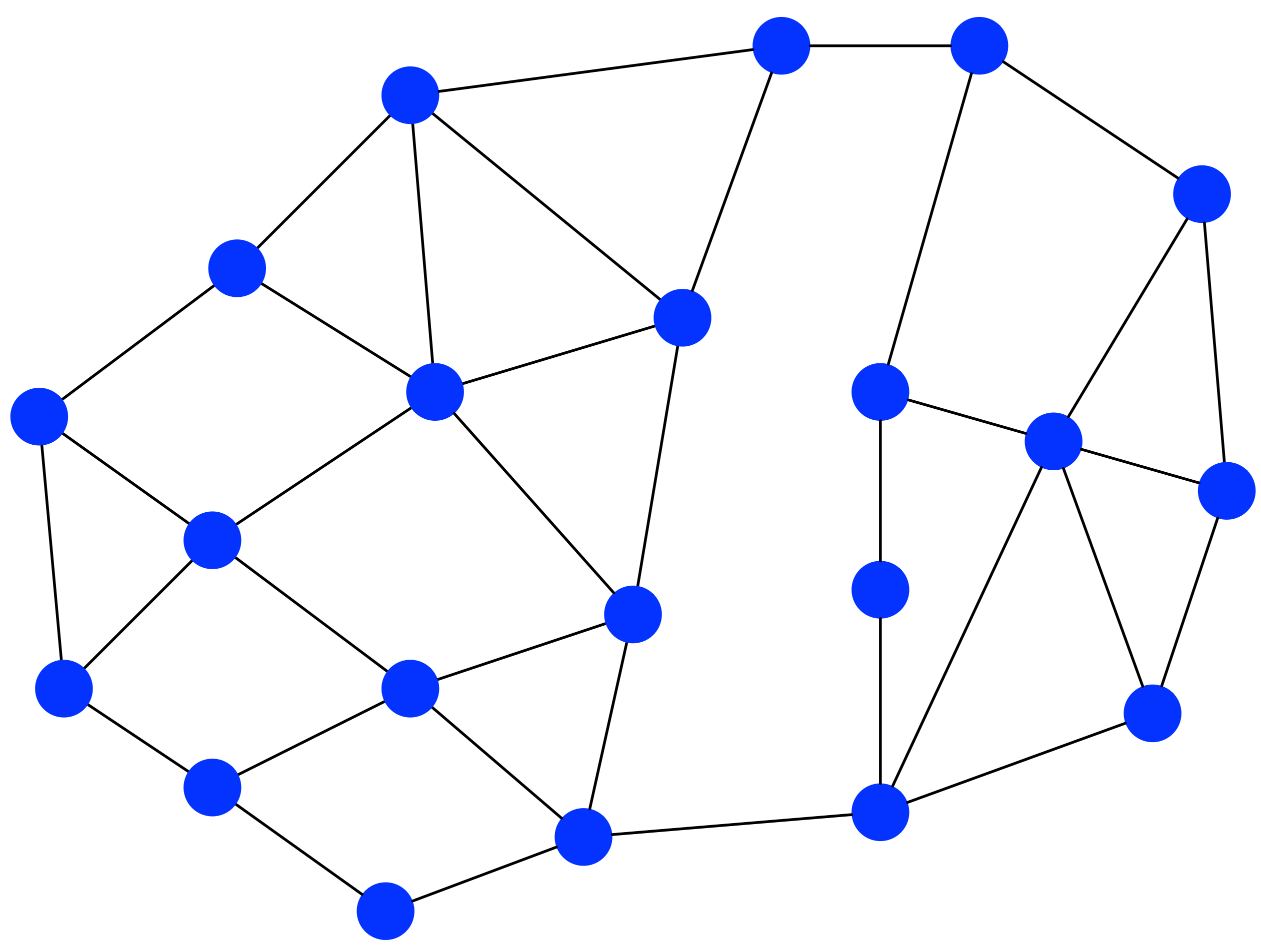
- **Lemma:** For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$
  - Each vertex is incident to at least $|C|$ edges

Contractions Preserve Cuts (Probably)

- **Lemma:** For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$
  - Each vertex is incident to at least $|C|$ edges
  - So there are at least $|C| (n / 2)$ edges

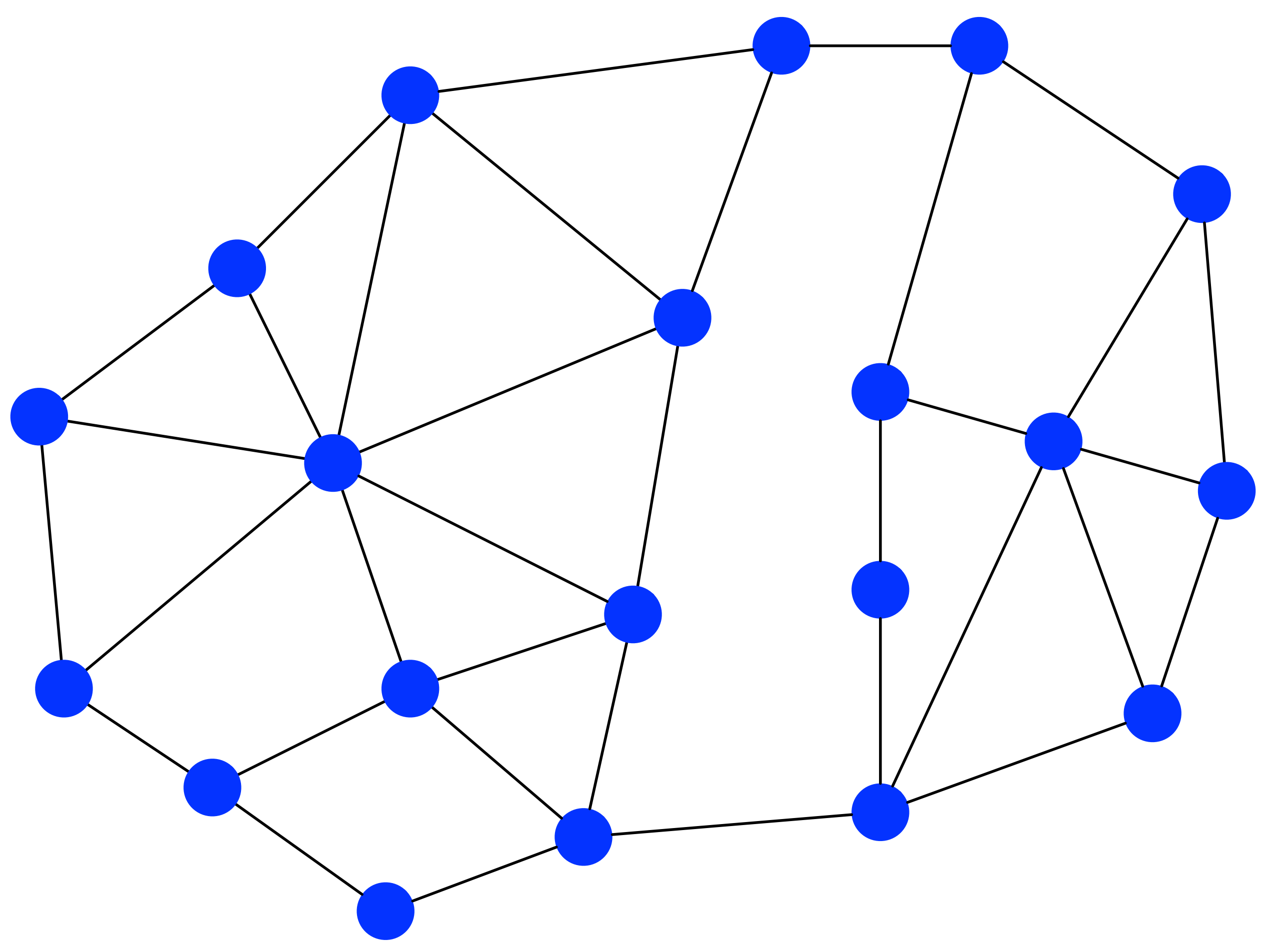
Repeating Contractions [Karger '93]

- Repeatedly contract until two vertices remain



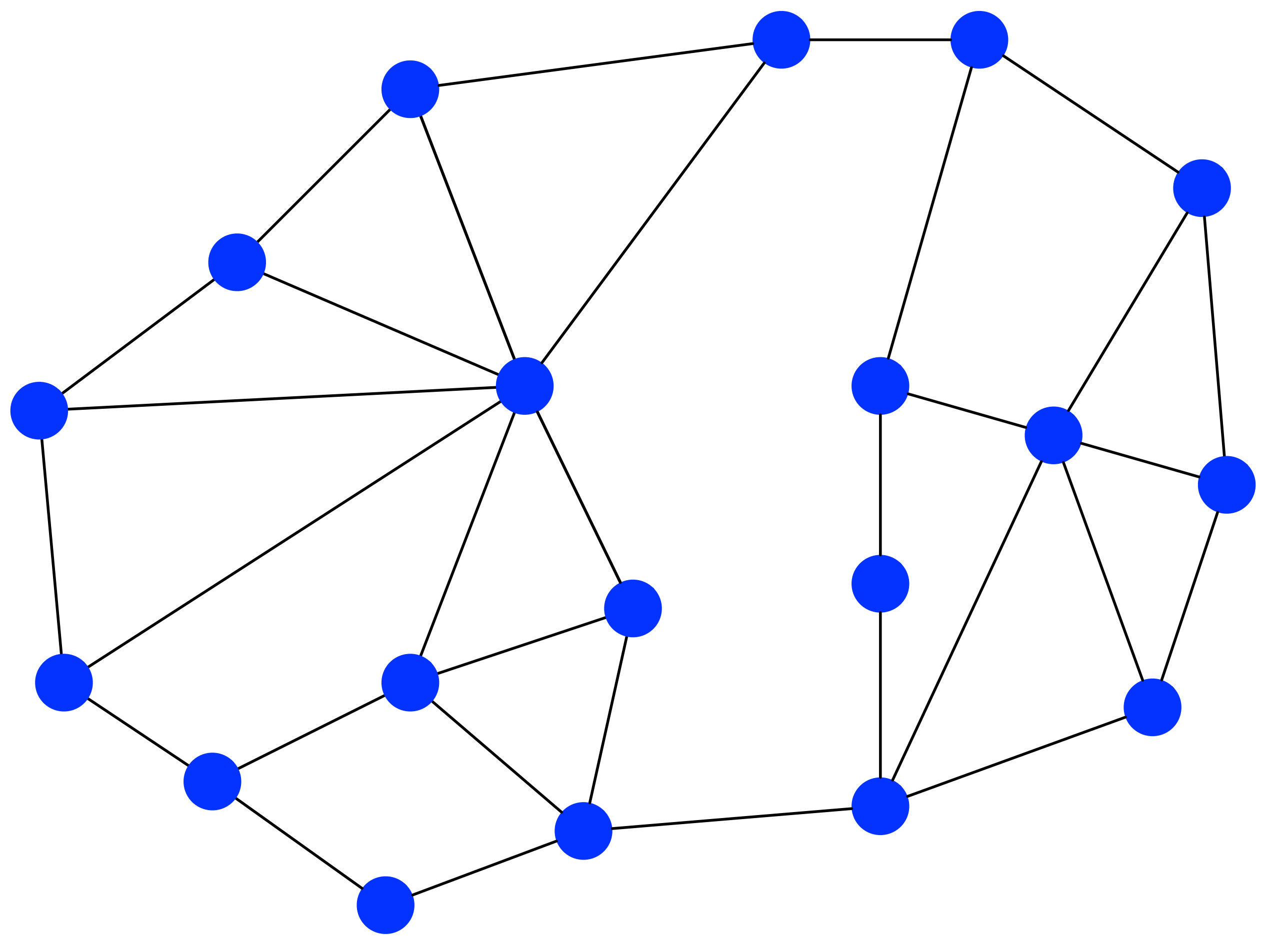
Repeating Contractions [Karger '93]

- Repeatedly contract until two vertices remain



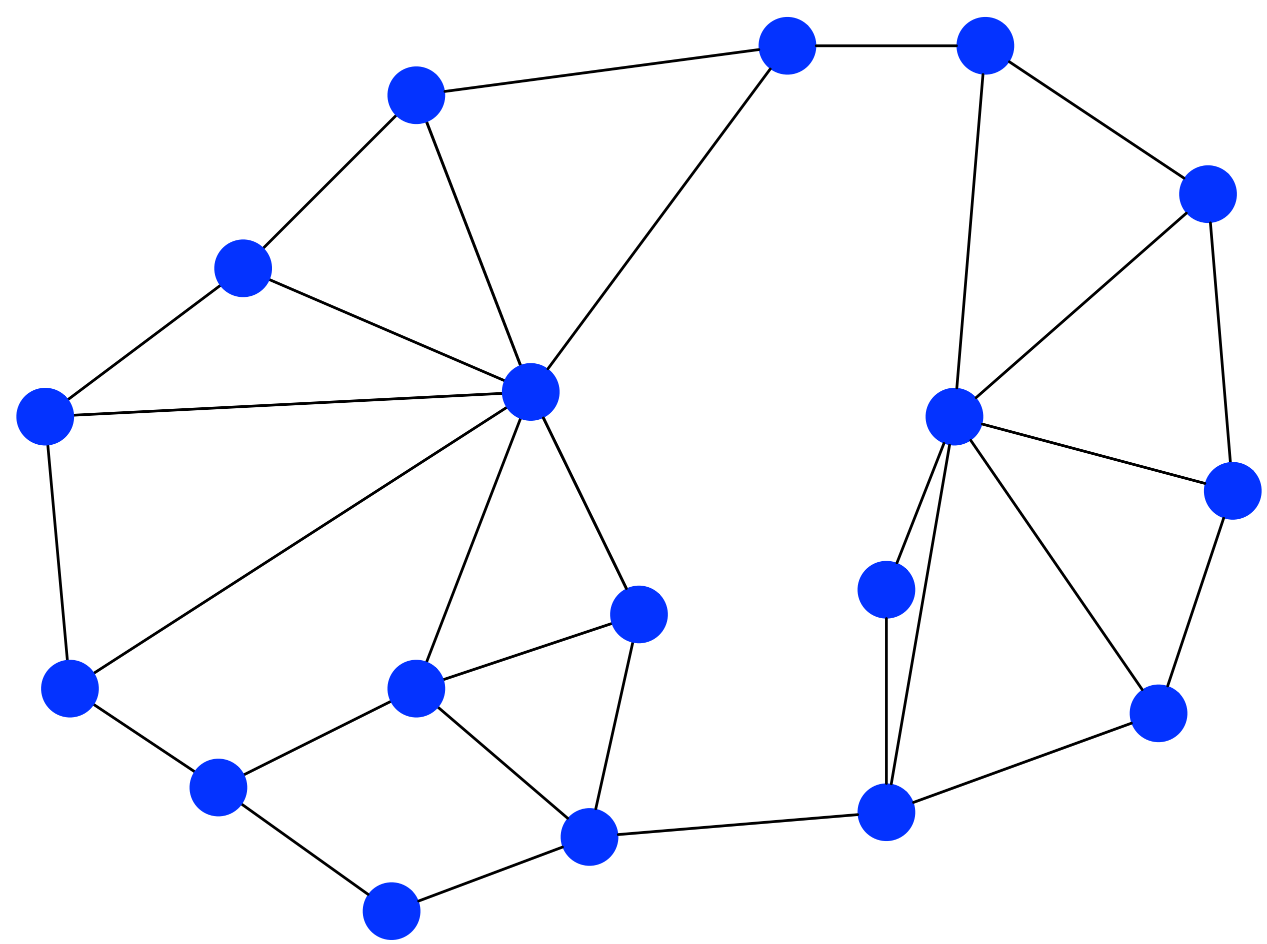
Repeating Contractions [Karger ’93]

- Repeatedly contract until two vertices remain



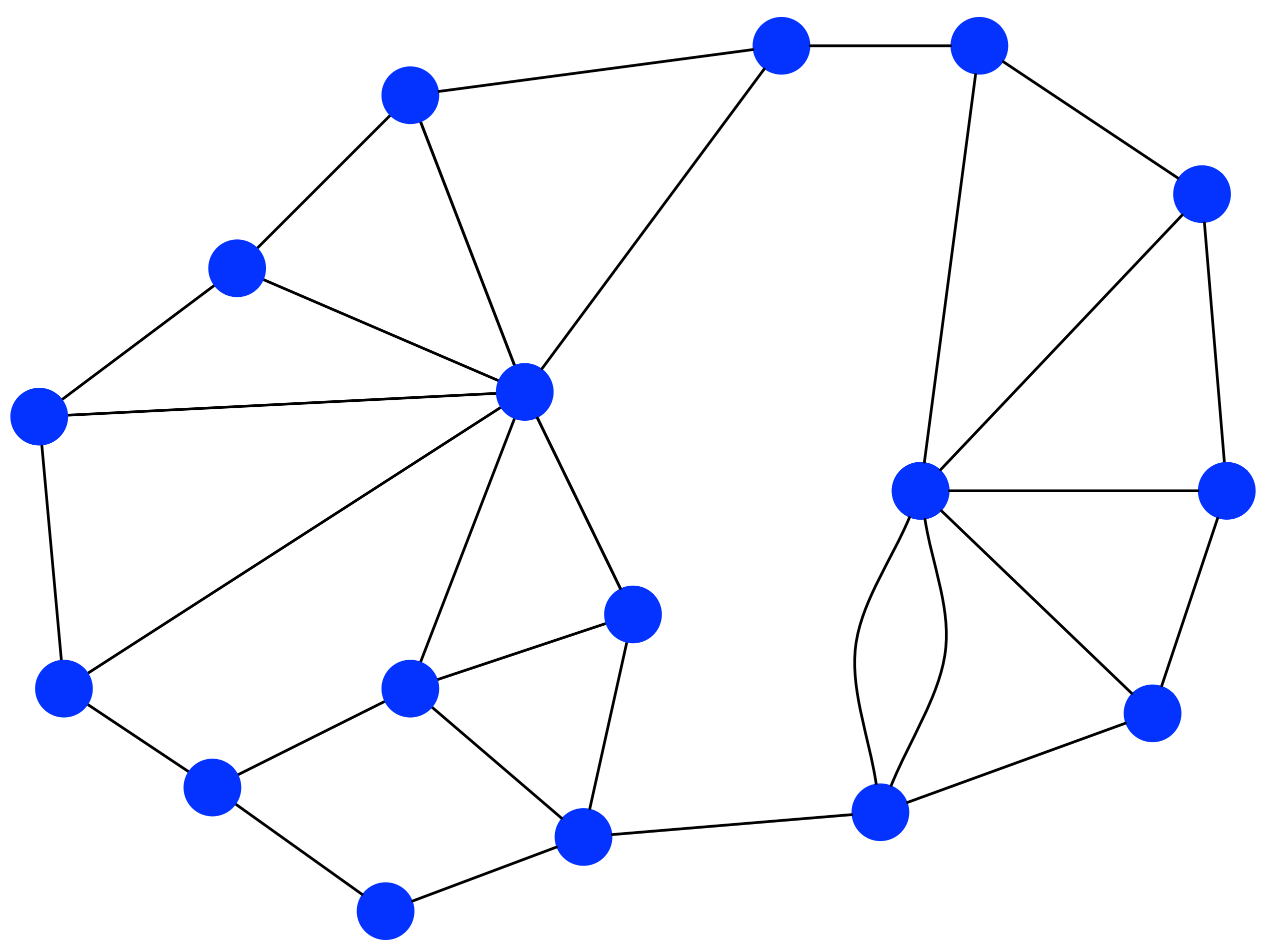
Repeating Contractions [Karger '93]

- Repeatedly contract until two vertices remain



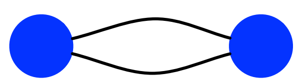
Repeating Contractions [Karger '93]

- Repeatedly contract until two vertices remain



Repeating Contractions [Karger '93]

- Repeatedly contract until two vertices remain
- Surviving edges form a cut



It Works!

- For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$

It Works!

- For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$

induction…

It Works!

- For *any* minimum cut C, we contract an edge of C with probability $\leq 2 / n$

induction…

- Minimum cut C survives the entire process with probability $\Omega(1 / n^2)$

Try, Try Again

- One run of the repeated contraction process finds a minimum cut with probability $\Omega(1 / n^2)$

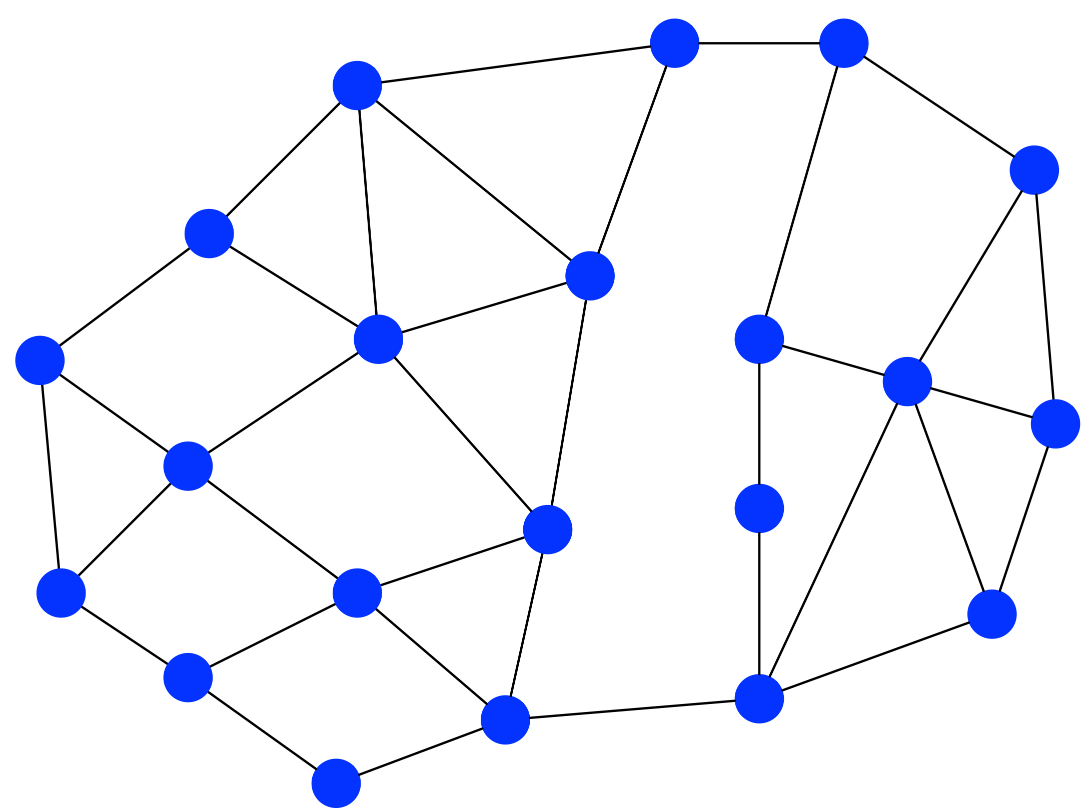
- Run it $O(n^2)$ times to succeed with constant probability

- Or $O(n^2 \log n)$ times to succeed with ***high probability*** (probability $\geq 1 - 1 / n$)

- Takes $O(n^2)$ time to fully contract once, so $O(n^4 \log n)$ time total

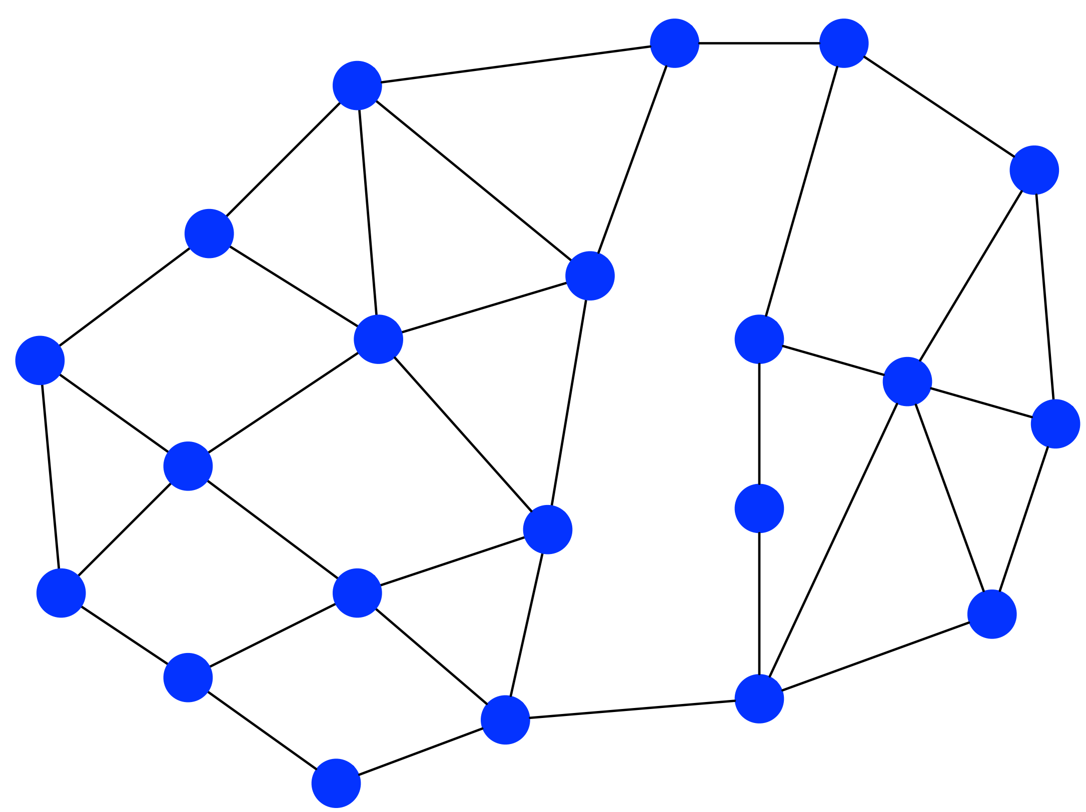
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!



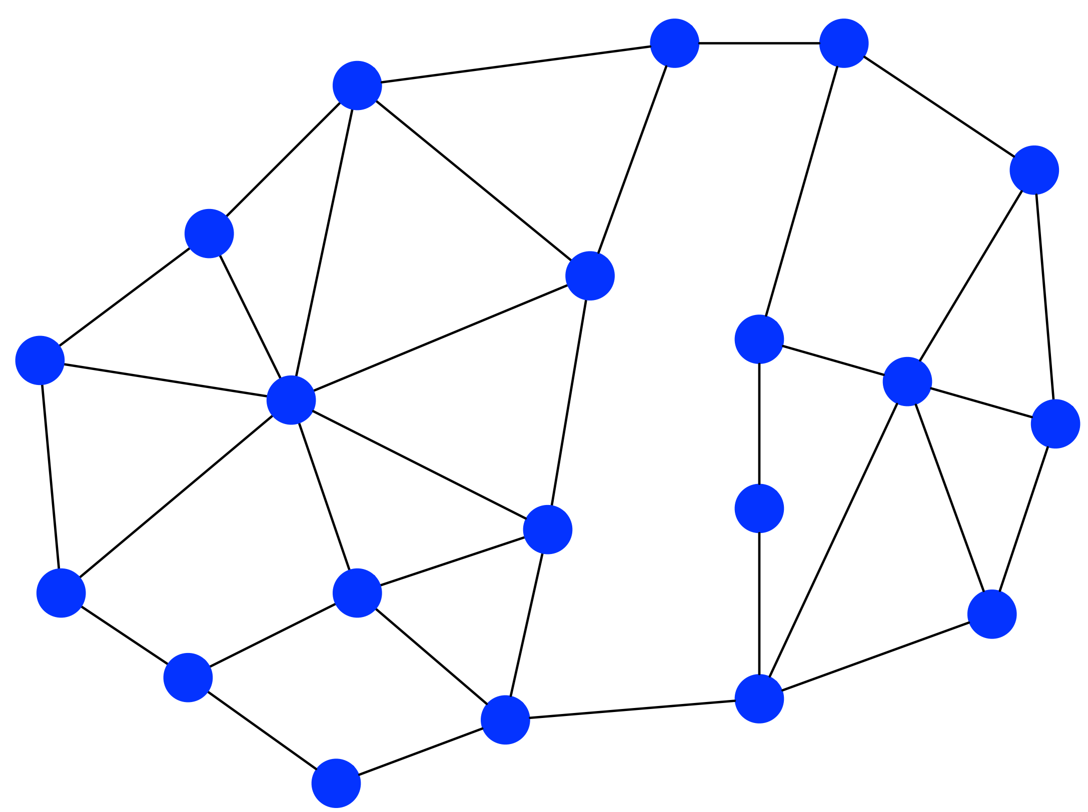
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain



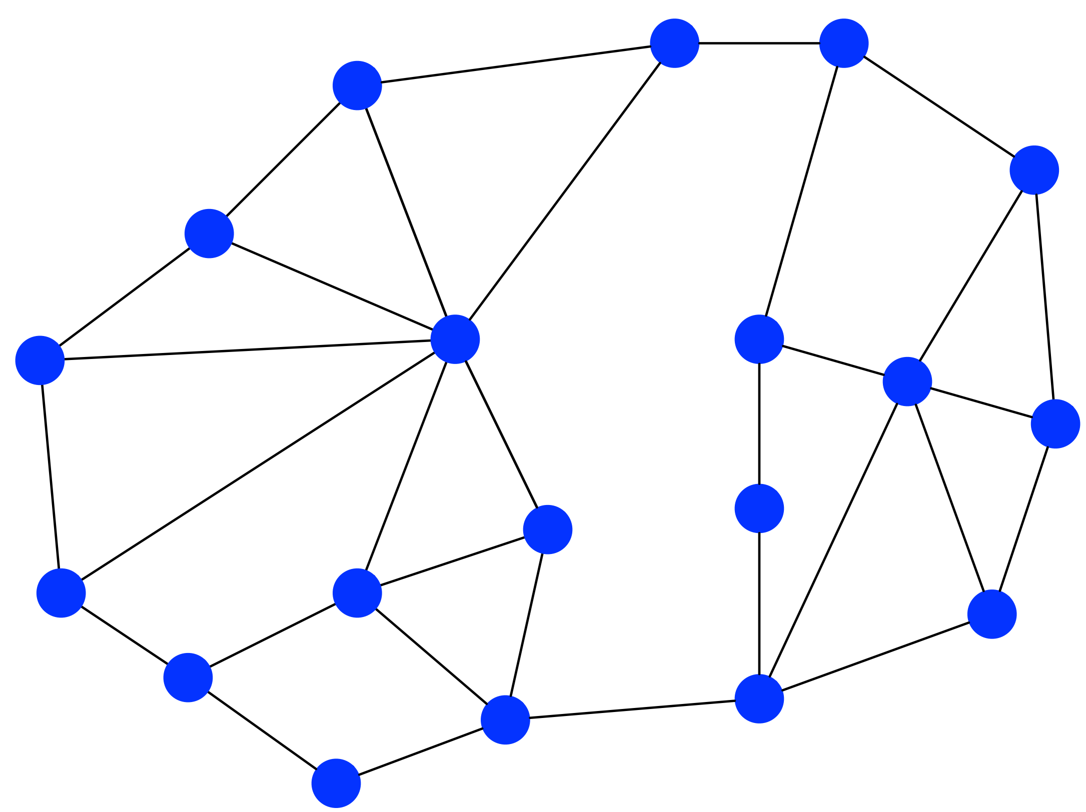
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain



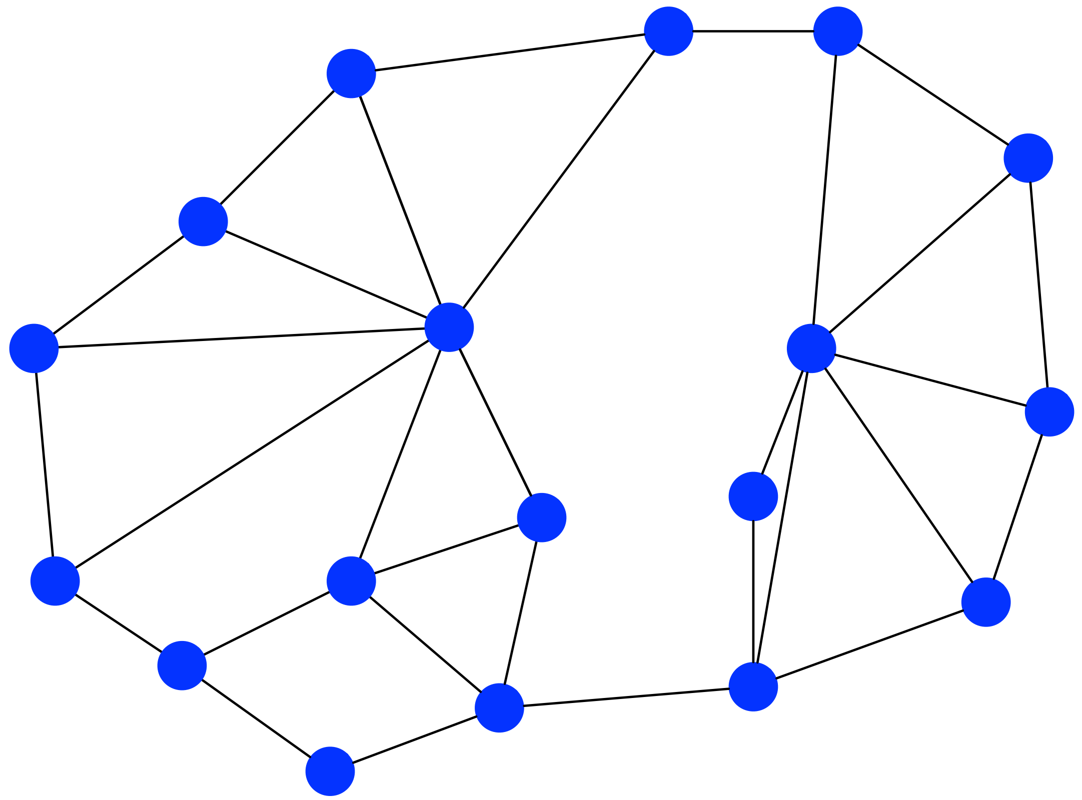
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain



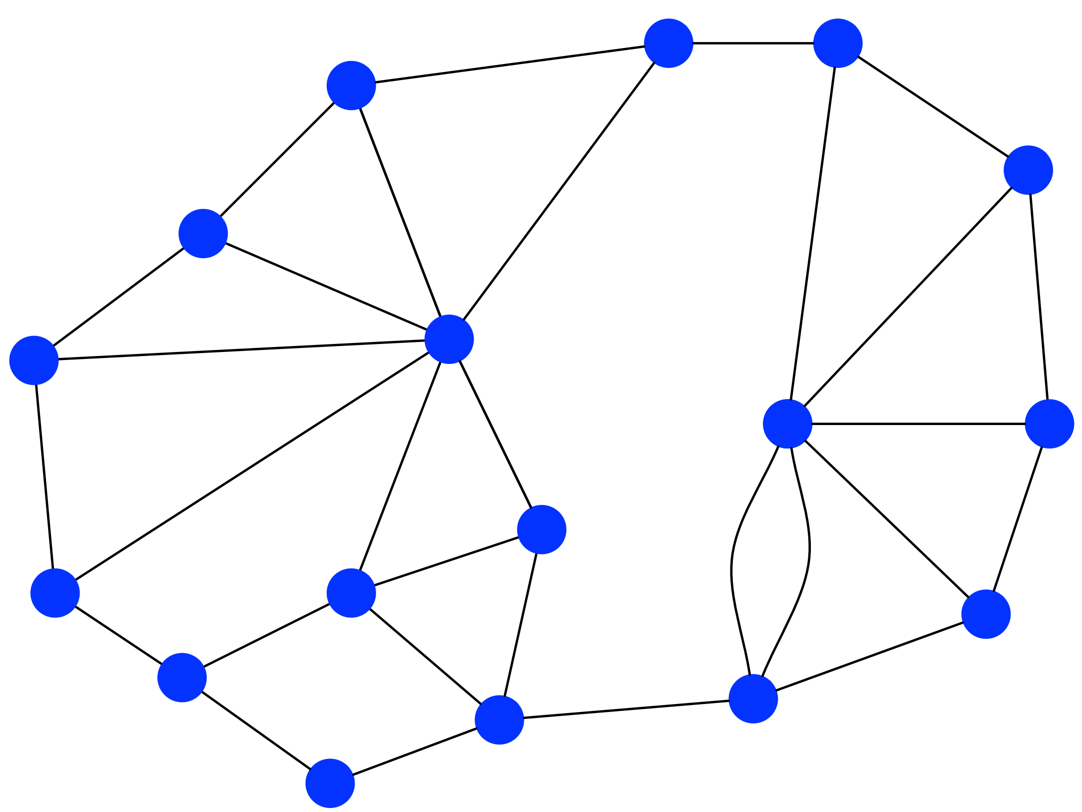
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain



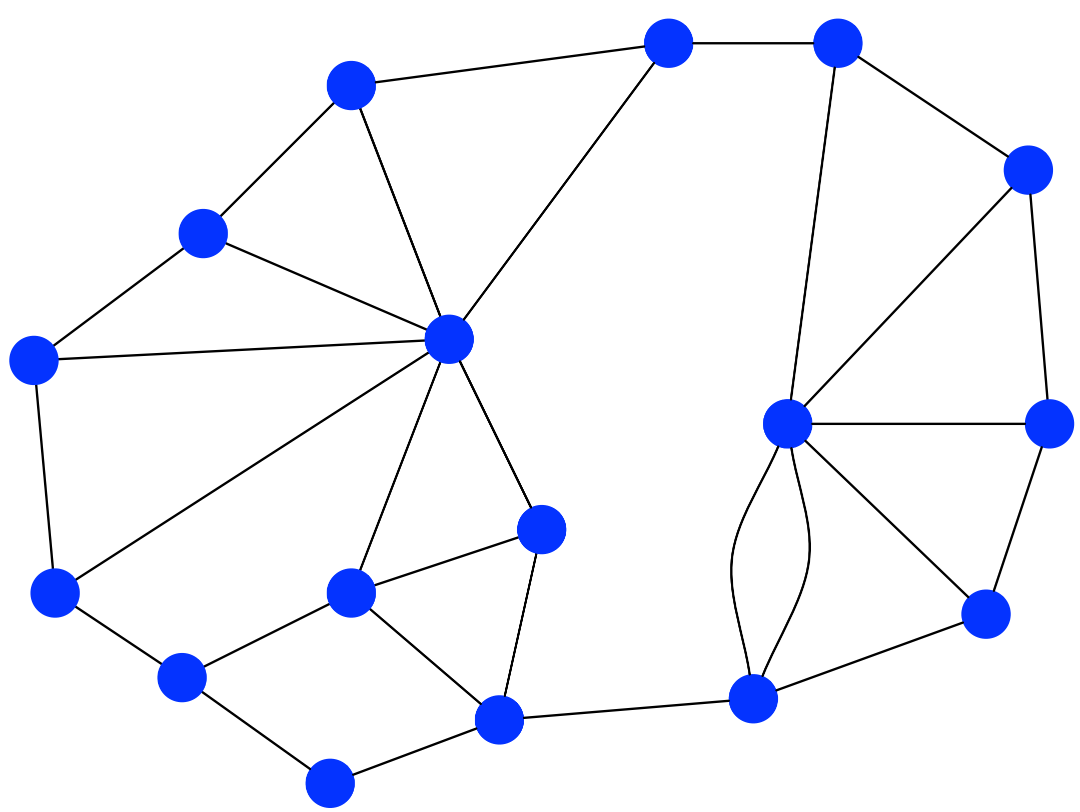
Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain



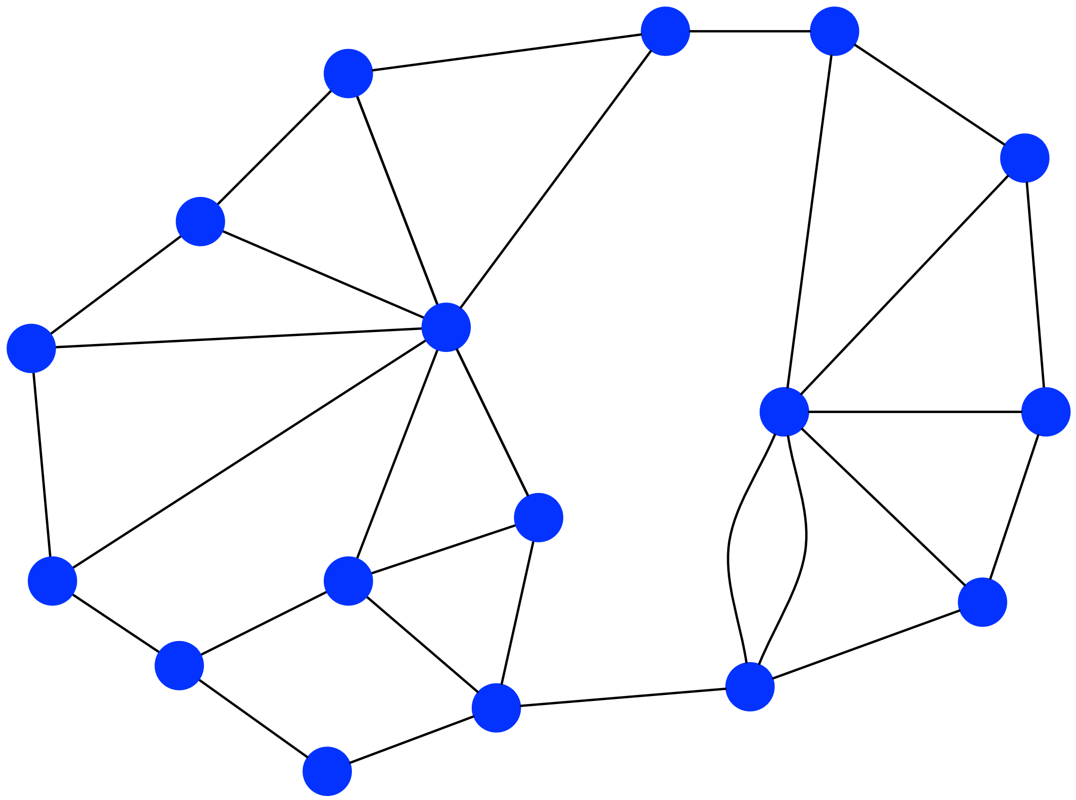
Branching Contractions [Karger, Stein '96]

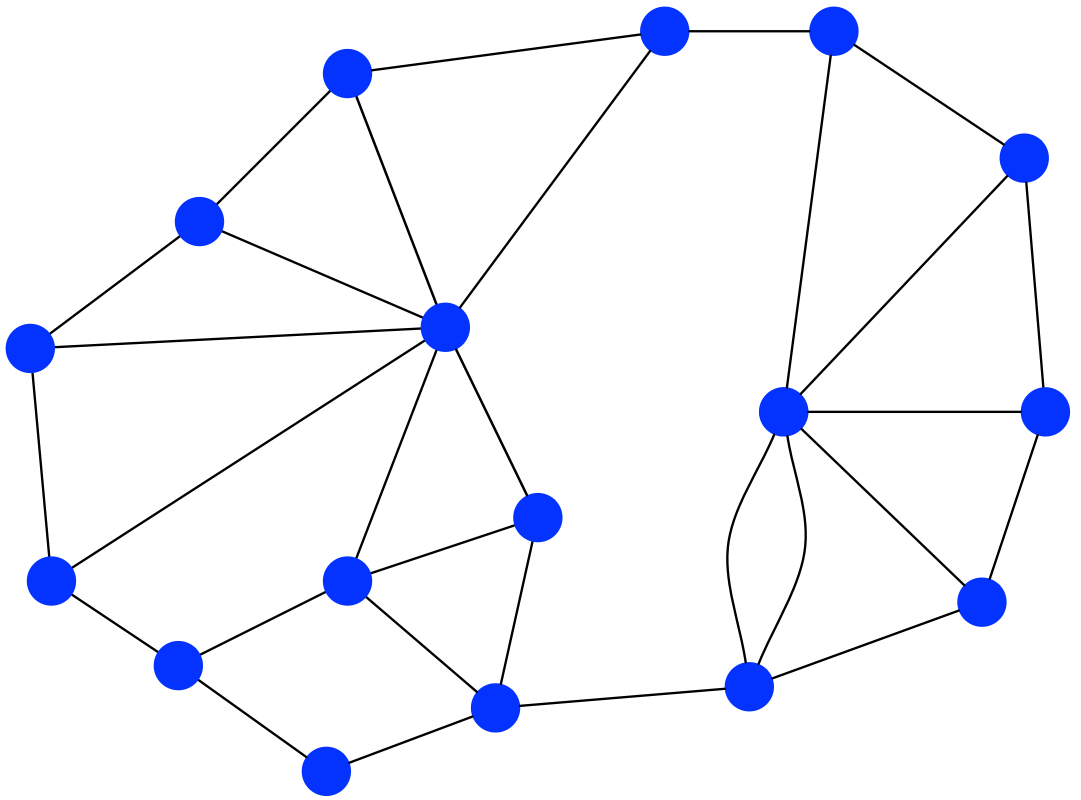
- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain
- Make two copies of contracted graph, recurse, and return the smaller cut



Branching Contractions [Karger, Stein '96]

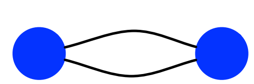
- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain
- Make two copies of contracted graph, recurse, and return the smaller cut

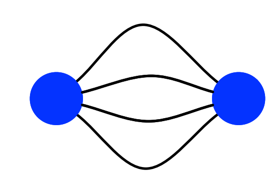




Branching Contractions [Karger, Stein '96]

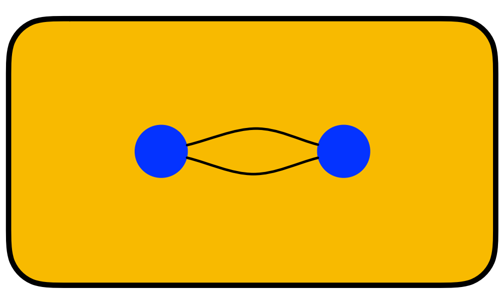
- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain
- Make two copies of contracted graph, recurse, and return the smaller cut

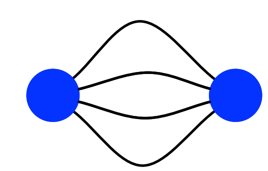




Branching Contractions [Karger, Stein '96]

- Probably for one run to succeed is too low!
- To boost probability, contract until $\lceil n/\sqrt{2} \rceil$ vertices remain
- Make two copies of contracted graph, recurse, and return the smaller cut





Still Fast

- A single try has the runtime recurrence $T(n) = 2T(n/\sqrt{2}) + O(n^2)$
- So $O(n^2 \log n)$ time per run

But More Accurate

- For *any* minimum cut *C*, probability we contract any edge in *C* *before recursion* is $\leq 1 / 2$

But More Accurate

- For *any* minimum cut *C*, probability we contract any edge in *C* *before recursion* is $\leq 1 / 2$

nasty induction…

But More Accurate

- For *any* minimum cut C, probability we contract any edge in C *before recursion* is $\leq 1 / 2$

nasty induction…

- We return a minimum cut with probability $\Omega(1 / \log n)$

Final Result for Graphs

- One run of the branching contraction process finds a minimum cut with probability $\Omega(1 / \log n)$

Final Result for Graphs

- One run of the branching contraction process finds a minimum cut with probability $\Omega(1 / \log n)$

- Run it $O(\log^2 n)$ times to succeed with high probability

Final Result for Graphs

- One run of the branching contraction process finds a minimum cut with probability $\Omega(1 / \log n)$
- Run it $O(\log^2 n)$ times to succeed with high probability
- $O(n^2 \log^3 n)$ time total!

Final Result for Graphs

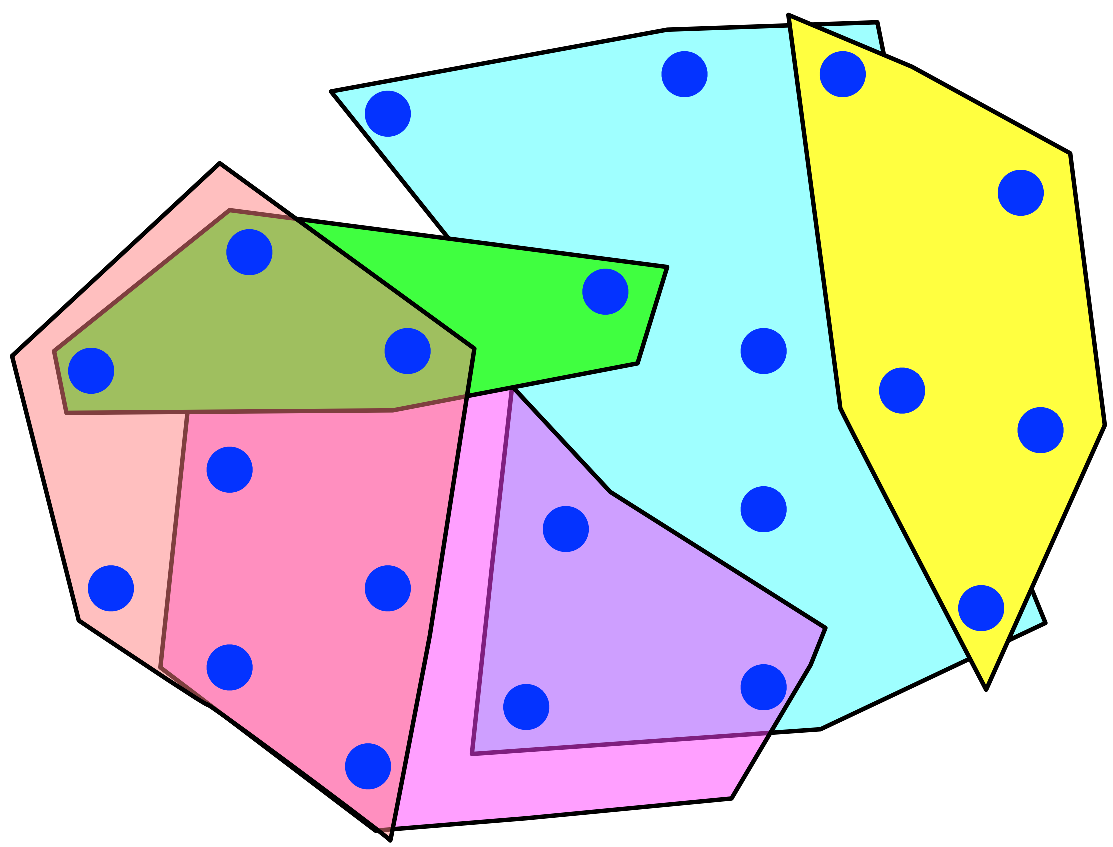
- One run of the branching contraction process finds a minimum cut with probability $\Omega(1 / \log n)$
- Run it $O(\log^2 n)$ times to succeed with high probability
- $O(n^2 \log^3 n)$ time total!
- Faster than known deterministic algorithms when the graph is dense

Final Result for Graphs

- One run of the branching contraction process finds a minimum cut with probability $\Omega(1 / \log n)$
- Run it $O(\log^2 n)$ times to succeed with high probability
- $O(n^2 \log^3 n)$ time total!
- Faster than known deterministic algorithms when the graph is dense
- Can easily modify algorithm to find minimum k-cuts in $O(n^{2k - 2} \log^2 n)$ time (best known algorithm)

Today's Real Topic: Hypergraphs

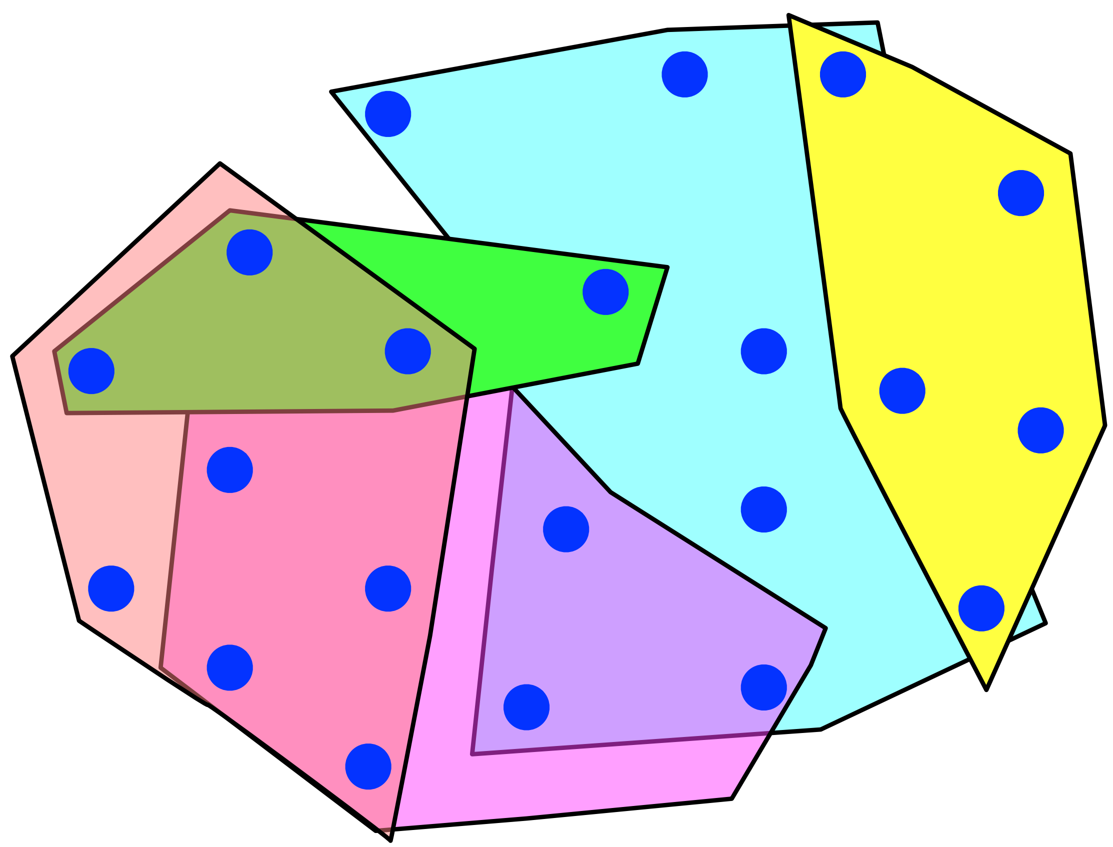
- A ***hypergraph*** is a collection of vertices and arbitrary subsets of vertices called ***hyperedges***



Today's Real Topic: Hypergraphs

- A ***hypergraph*** is a collection of vertices and arbitrary subsets of vertices called ***hyperedges***

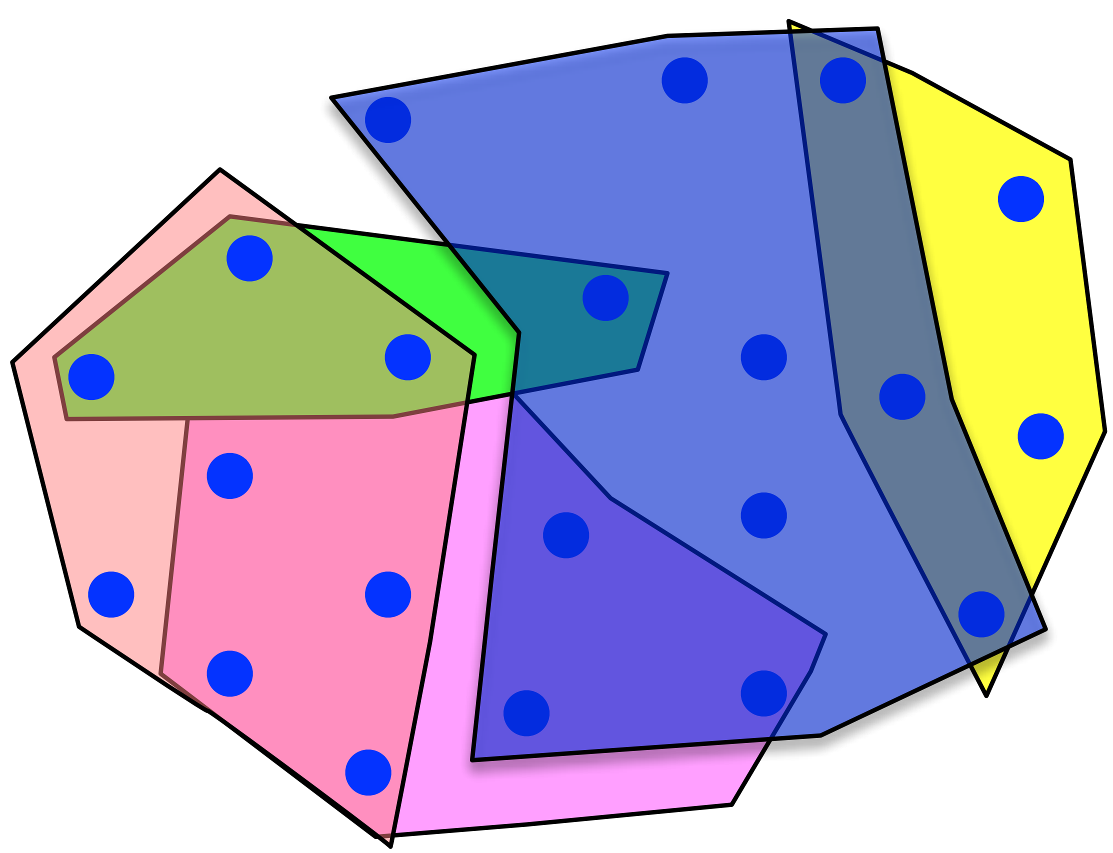
- A ***minimum (k-)cut*** is still a minimum cardinality set of hyperedges whose removal creates at least *k* components



Today's Real Topic: Hypergraphs

- A ***hypergraph*** is a collection of vertices and arbitrary subsets of vertices called ***hyperedges***

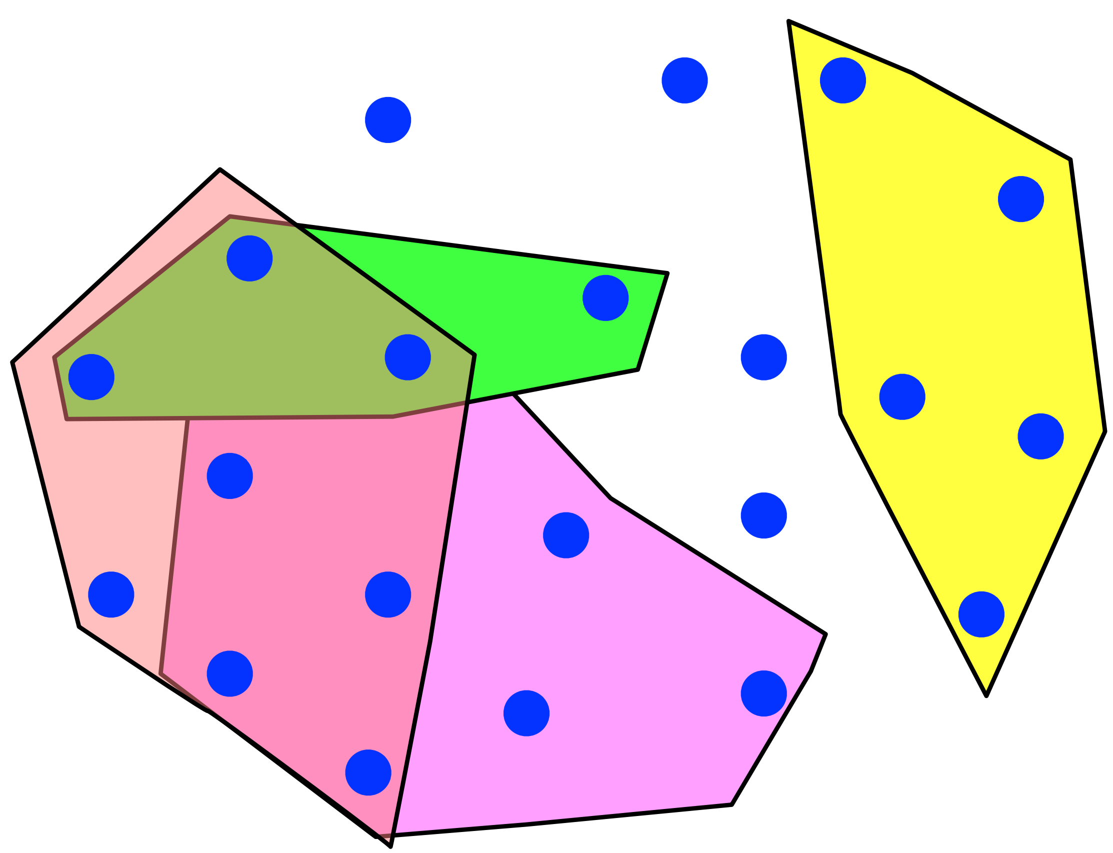
- A ***minimum (k-)cut*** is still a minimum cardinality set of hyperedges whose removal creates at least *k* components



Today's Real Topic: Hypergraphs

- A ***hypergraph*** is a collection of vertices and arbitrary subsets of vertices called ***hyperedges***

- A ***minimum (k-)cut*** is still a minimum cardinality set of hyperedges whose removal creates at least k components



Hyperstrategies

- Again, focus on finding minimum (2-)cuts

Hyperstrategies

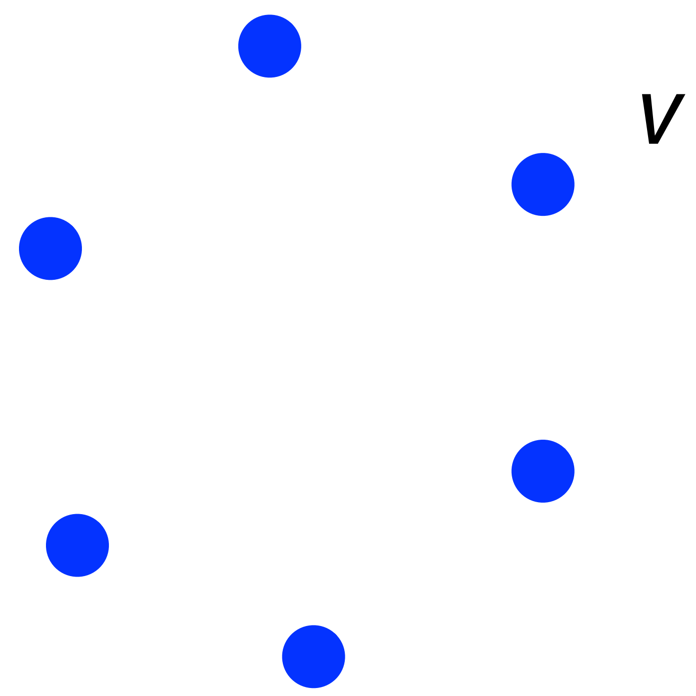
- Again, focus on finding minimum (2-)cuts
- Given a hypergraph with n vertices, m hyperedges, and total hyperedge size p

Hyperstrategies

- Again, focus on finding minimum (2-)cuts
- Given a hypergraph with n vertices, m hyperedges, and total hyperedge size p
  - Guess a hyperedge outside a minimum cut and ***contract***…?

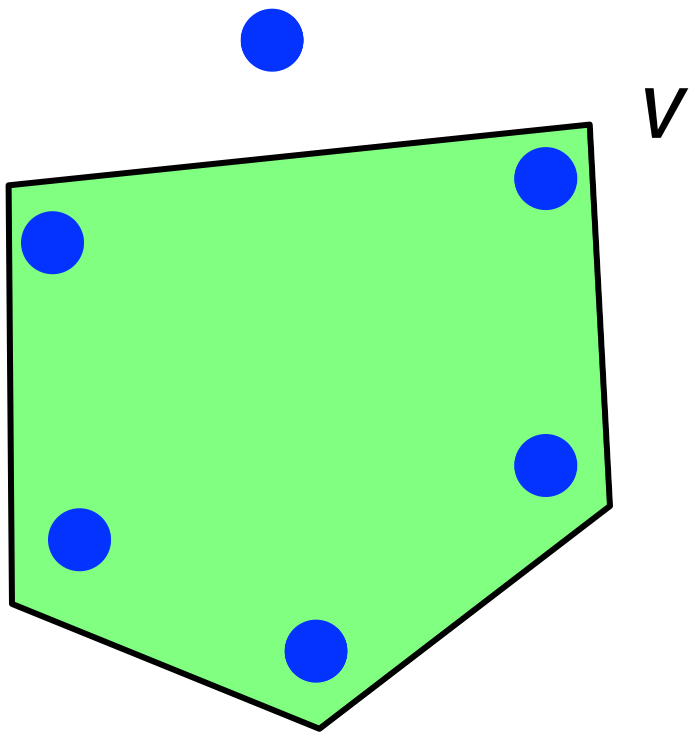
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



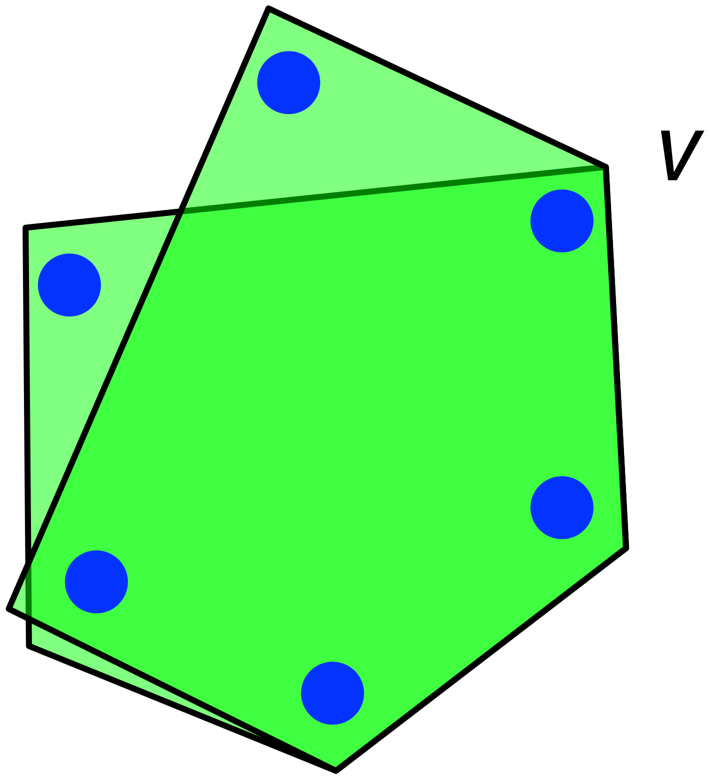
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



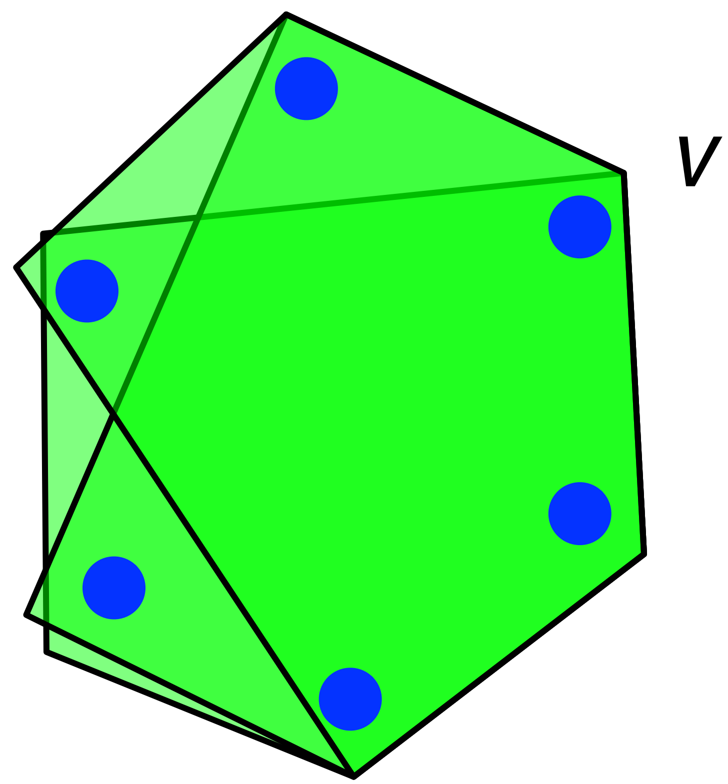
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



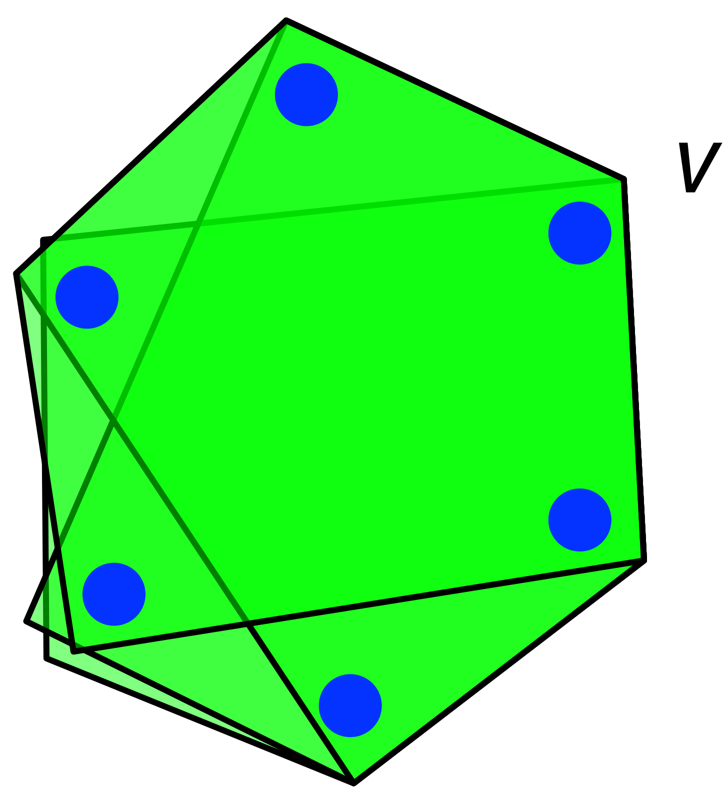
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



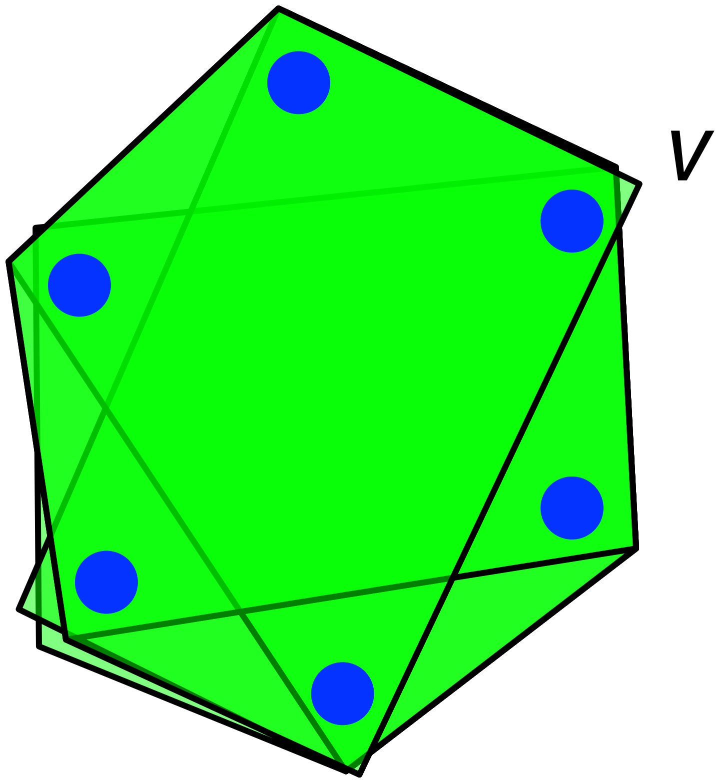
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



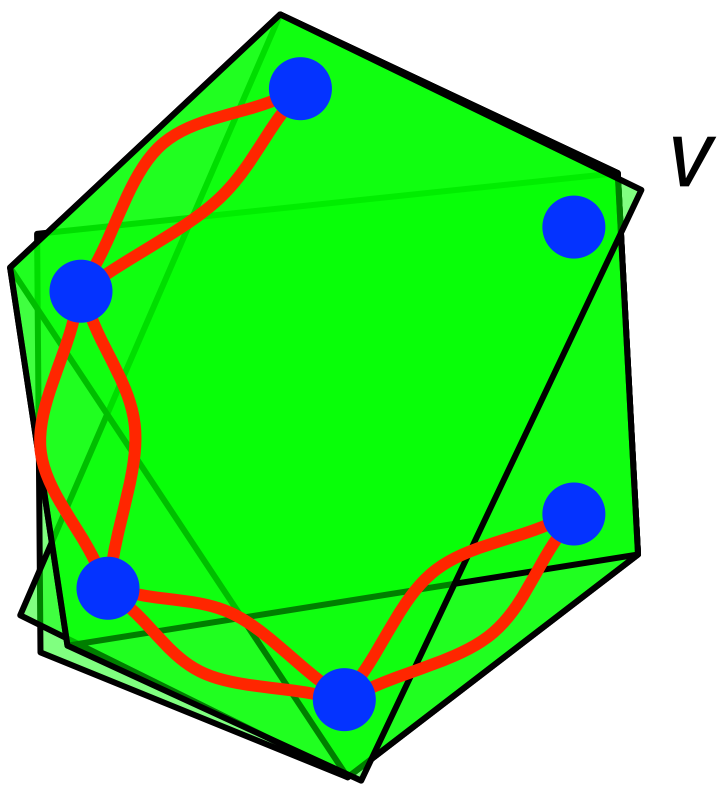
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v



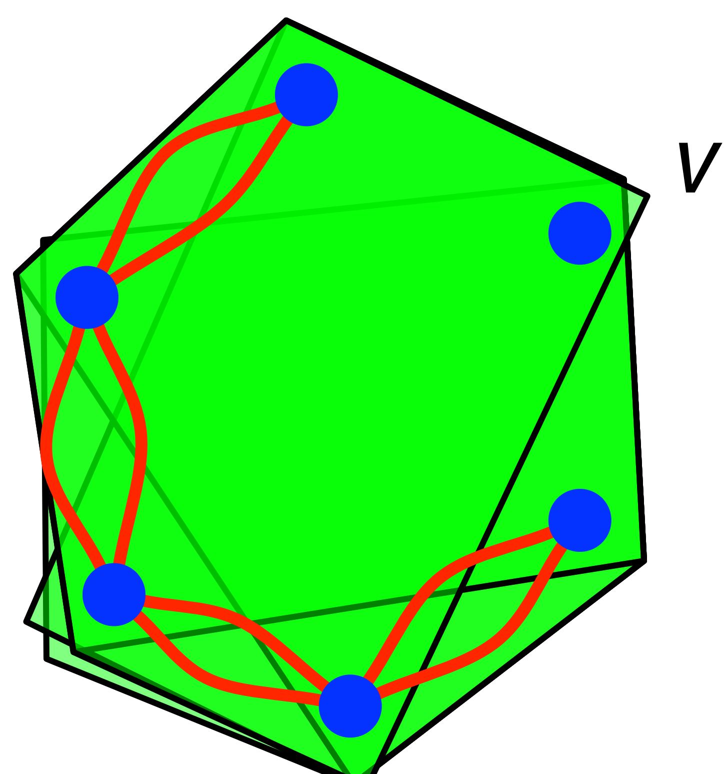
Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v
- Two copies of a path spanning over every vertex except v



Uniform Contracts are Bad

- n - 1 distinct edges of size n - 1, all containing vertex v
- Two copies of a path spanning over every vertex except v
- Minimum cut contains every hyperedge, but there is a ≥ 1/3 probability of contracting a hyperedge each round



Really Bad

- Minimum cut survives all contractions with probability ***exponentially*** low in maximum hyperedge size [Kogan, Krauthgamer '15]

- Need to bias selection away from large hyperedges

Biased Selection

- Should contract each hyperedge e with probability proportional to $(1 - |e| / n)$ [Chandrasekaran, Xu, Yu '18]

Biased Selection

- Should contract each hyperedge e with probability proportional to $(1 - |e| / n)$ [Chandrasekaran, Xu, Yu '18]

- Minimum cut survives all contractions with probability $\Omega(1 / n^2)$!

Biased Selection

- Should contract each hyperedge e with probability proportional to $(1 - |e| / n)$ [Chandrasekaran, Xu, Yu '18]

- Minimum cut survives all contractions with probability $\Omega(1 / n^2)$!

- Can perform all contractions in $O(p \log n)$ time total [Ghaffari, Karger, Panigrahi '17]

Biased Selection

- Should contract each hyperedge e with probability proportional to $(1 - |e| / n)$ [Chandrasekaran, Xu, Yu ’18]
- Minimum cut survives all contractions with probability $\Omega(1 / n^2)$!
- Can perform all contractions in $O(p \log n)$ time total [Ghaffari, Karger, Panigrahi ’17]
- So compute a minimum cut with high probability in $O(p\, n^2 \log^2 n) = O(mn^3 \log^2 n)$ time

Biased Selection

- Should contract each hyperedge e with probability proportional to $(1 - |e| / n)$ [Chandrasekaran, Xu, Yu ’18]
- Minimum cut survives all contractions with probability $\Omega(1 / n^2)$!
- Can perform all contractions in $O(p \log n)$ time total [Ghaffari, Karger, Panigrahi ’17]
- So compute a minimum cut with high probability in $O(p\, n^2 \log^2 n) = O(mn^3 \log^2 n)$ time
- Minimum k-cut ($k \geq 3$) with high probability in $O(p\, n^{2k-1} \log n) = O(mn^{2k} \log n)$ time

Branching Contractions?

- Karger and Stein branch at carefully chosen graph orders

Branching Contractions?

- Karger and Stein branch at carefully chosen graph orders
- Branch too early, and the number of recursive subproblems becomes **too large**

Branching Contractions?

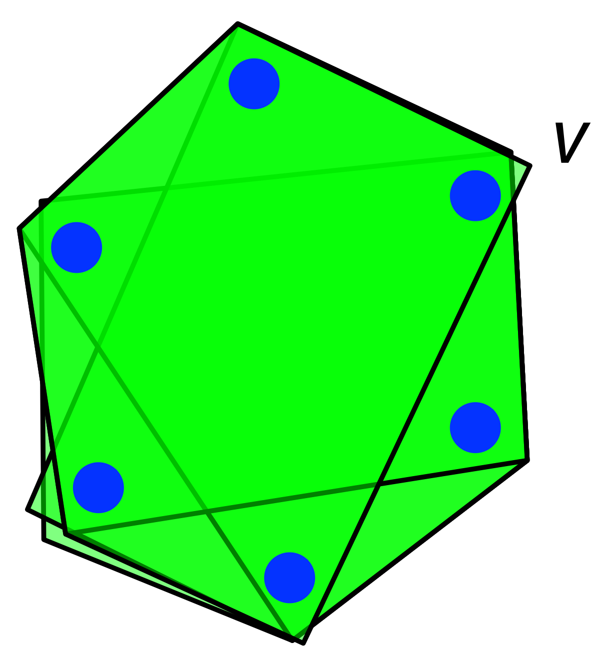
- Karger and Stein branch at carefully chosen graph orders
- Branch too early, and the number of recursive subproblems becomes **too large**
- Branch too late, and the probability of success becomes **too small**

Branching Contractions?

- Karger and Stein branch at carefully chosen graph orders
- Branch too early, and the number of recursive subproblems becomes **too large**
- Branch too late, and the probability of success becomes **too small**
- Large hyperedges make it impossible to branch at the same time they do

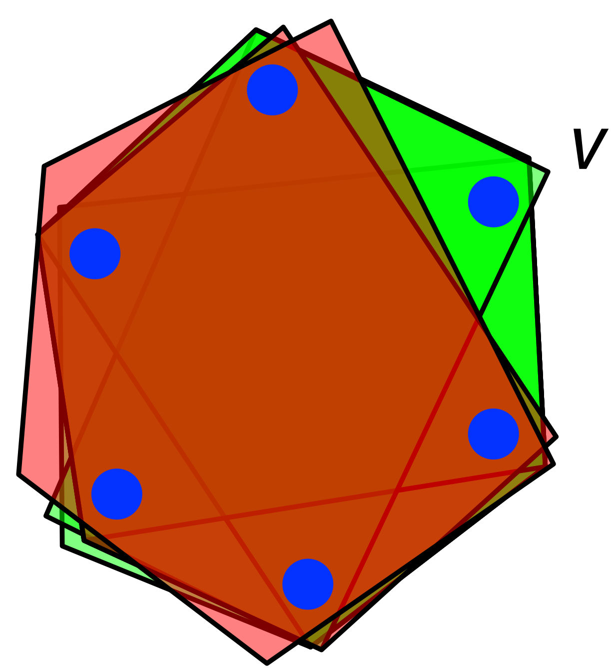
Many Branches Needed

- Same example as before, but replace the paths with two hyperedges containing every vertex except *v*



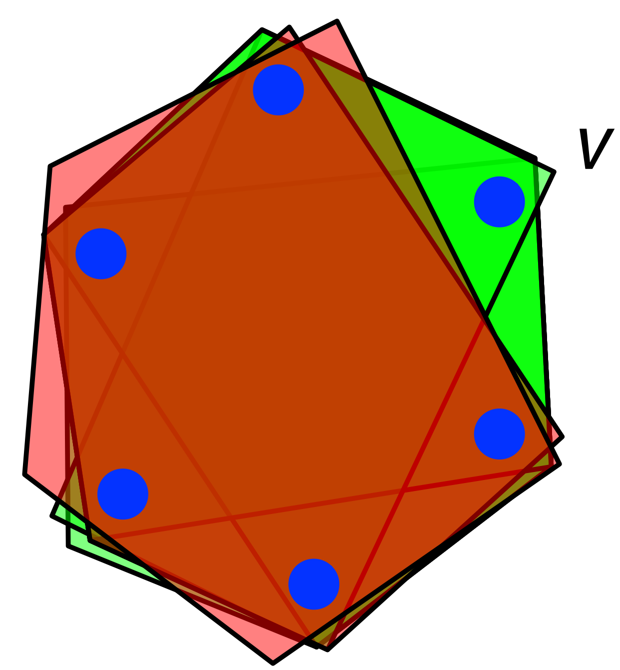
Many Branches Needed

- Same example as before, but replace the paths with two hyperedges containing every vertex except *v*



Many Branches Needed

- Same example as before, but replace the paths with two hyperedges containing every vertex except *v*
- Minimum cut contains every green hyperedge; contraction preserves cut with probability $\leq 2 / (n + 1)$

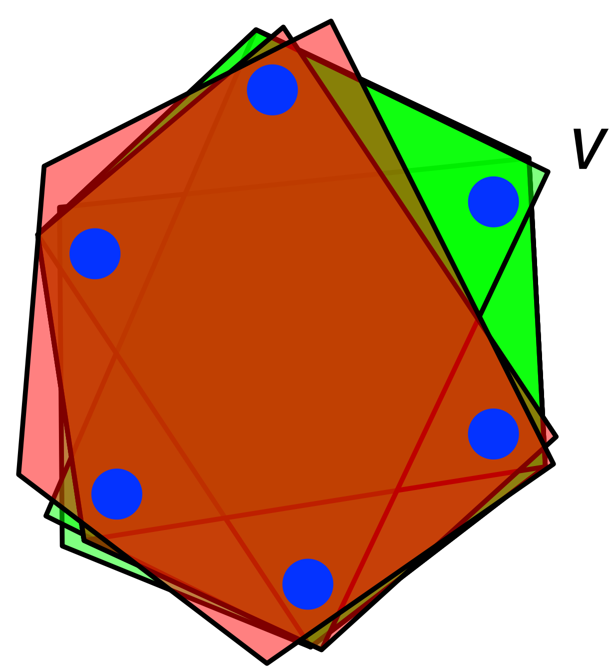


Many Branches Needed

- Same example as before, but replace the paths with two hyperedges containing every vertex except *v*

- Minimum cut contains every green hyperedge; contraction preserves cut with probability $\leq 2 / (n + 1)$

- Need $\Omega(n)$ independent contractions



A Smoother Procedure

- Larger hyperedges means we need more branches, but how many?

A Smoother Procedure

- Larger hyperedges means we need more branches, but how many?
- Math works out easiest if we could do a fractional number of branches before each contraction

A Smoother Procedure

- Larger hyperedges means we need more branches, but how many?
- Math works out easiest if we could do a fractional number of branches before each contraction
- Can achieve this ideal *in expectation* by branching with a probability dependent on the size of present hyperedges

Probabilistic Branching [FPZ]

- Before *every* contraction, we flip coins to decide whether or not to copy the current hypergraph and recursively call our algorithm

Probabilistic Branching [FPZ]

- Before *every* contraction, we flip coins to decide whether or not to copy the current hypergraph and recursively call our algorithm

- The larger the hyperedges, the more likely we are to branch

Bias Against Large Hyperedges

- Previous work biased hyperedge selection away from large hyperedges; meanwhile, we want large hyperedges to induce more branching

Bias Against Large Hyperedges

- Previous work biased hyperedge selection away from large hyperedges; meanwhile, we want large hyperedges to induce more branching

- Our algorithm moves all bias against large hyperedges into deciding whether or not to branch

Bias Against Large Hyperedges

- Previous work biased hyperedge selection away from large hyperedges; meanwhile, we want large hyperedges to induce more branching

- Our algorithm moves all bias against large hyperedges into deciding whether or not to branch

- First, we select a hyperedge uniformly at random and commit to contracting it

Bias Against Large Hyperedges

- Previous work biased hyperedge selection away from large hyperedges; meanwhile, we want large hyperedges to induce more branching
- Our algorithm moves all bias against large hyperedges into deciding whether or not to branch
- First, we select a hyperedge uniformly at random and commit to contracting it
- But before contraction, we copy the hypergraph with probability based on the selected hyperedge's size

Probability of Branching

- Intuitively, we want to balance the probability of contracting a minimum cut’s hyperedge with the probability of branching

Probability of Branching

- Intuitively, we want to balance the probability of contracting a minimum cut's hyperedge with the probability of branching

- Then, *in expectation*, we preserve one copy of the minimum cut in either the contracted hypergraph or the copy we create before contraction

Probability of Branching

- **Lemma**: For *any* minimum cut C, an edge chosen uniformly at random is in C with probability $\leq \frac{1}{m}\sum_{e}\frac{|e|}{n}$

Probability of Branching

- **Lemma**: For *any* minimum cut C, an edge chosen uniformly at random is in C with probability $\leq \frac{1}{m}\sum_{e}\frac{|e|}{n}$
- So if we select hyperedge e, we branch with probability $|e| / n$

That Looks Familiar…

- We branch with probability $|e| / n$; Chandrasekaran *et al*. would have *selected* hyperedge e with probability proportional to $(1 - |e| / n)$

That Looks Familiar…

- We branch with probability $|e| / n$; Chandrasekaran *et al*. would have *selected* hyperedge e with probability proportional to $(1 - |e| / n)$

- Their algorithm is the same as selecting a hyperedge e uniformly at random, and then ***redoing*** the selection with probability $|e| / n$

The Algorithm

- Maintain a set S of hyperedges we believe belong to the a minimum cut in hypergraph H

The Algorithm

- Maintain a set S of hyperedges we believe belong to the a minimum cut in hypergraph H

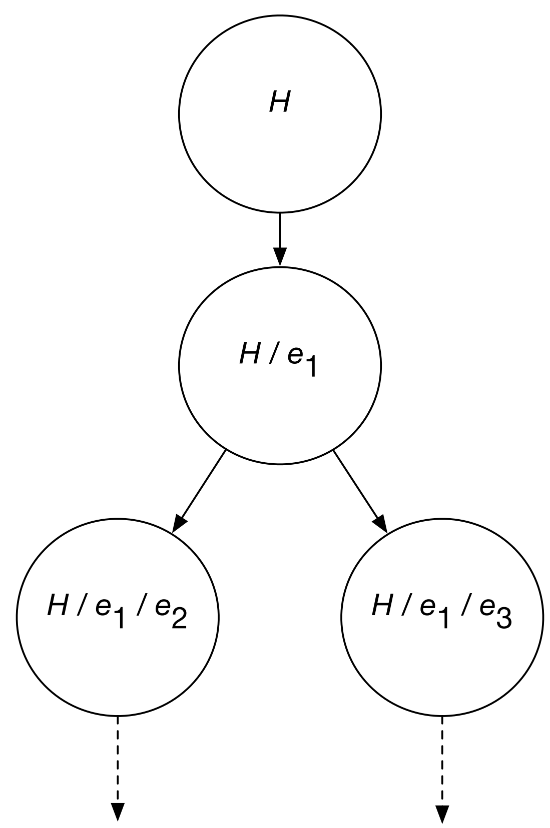
BranchingContract(H, S):

Add each spanning hyperedge to S and remove it from H
If H has no edges, return S

Select hyperedge e uniformly at random
With probability $|e| / n$, return the smaller of the cuts
BranchingContract(H / e, S) and
BranchingContract(H, S)
Otherwise, return BranchingContract(H / e, S)

Computation Tree

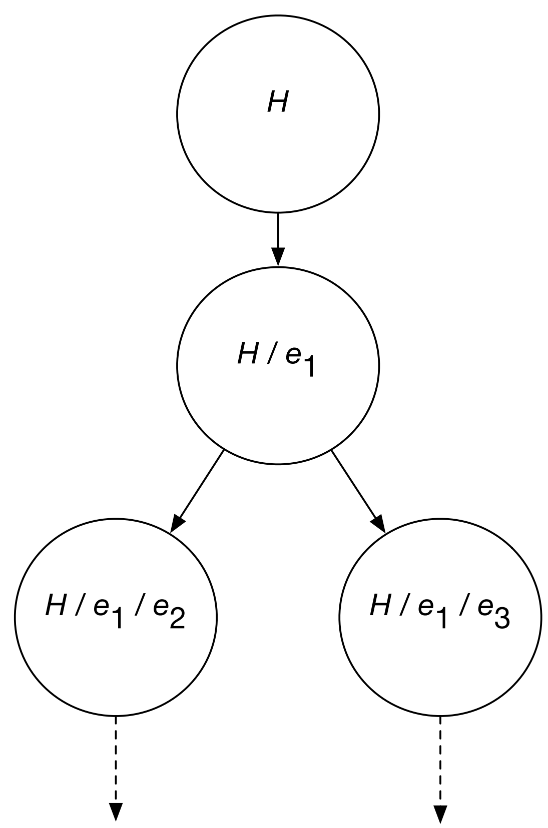
- Visualize the algorithm's execution as a rooted tree over hypergraphs; input hypergraph *H* is the root, each time we perform a contraction, that node gets a child



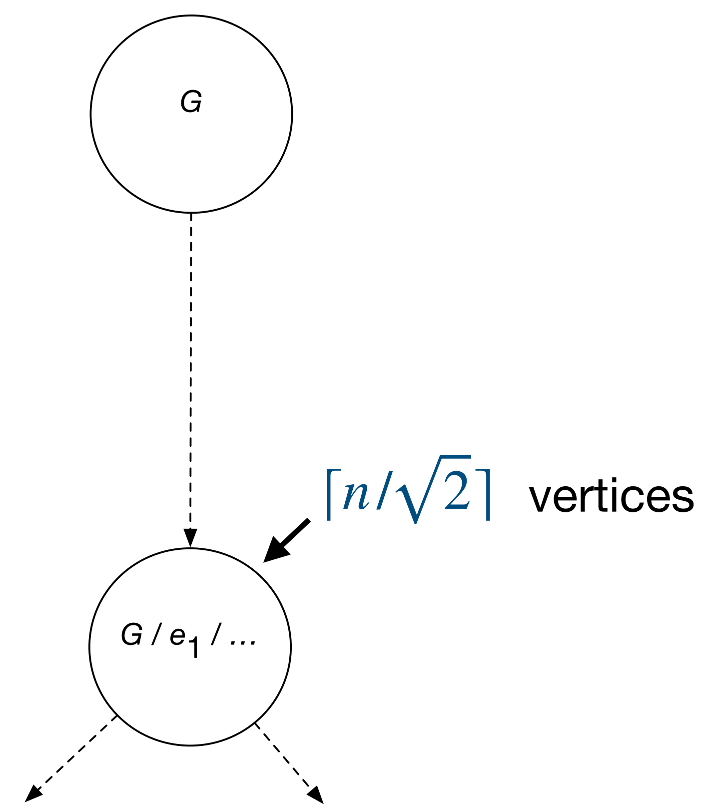
our algorithm

Computation Tree

- Our computation tree is probabilistic while Karger-Stein's is deterministic



our algorithm

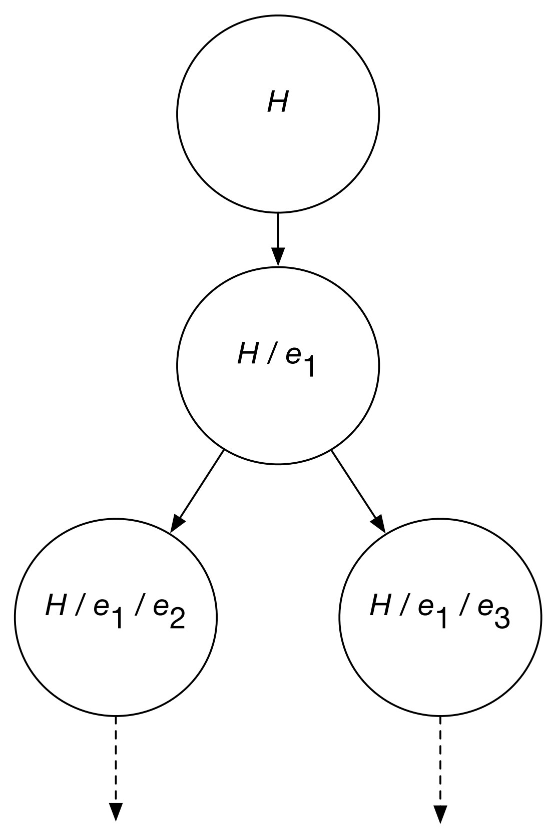


Karger-Stein

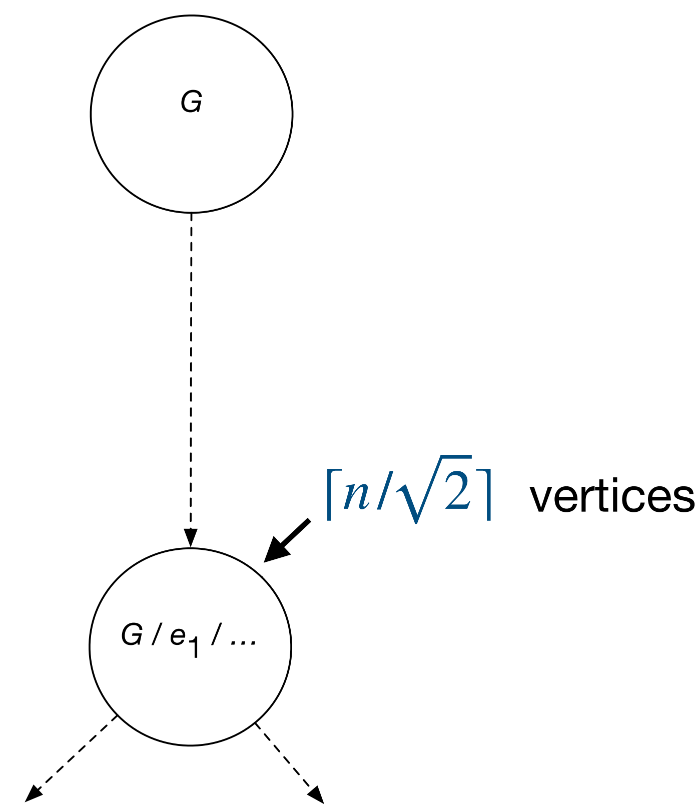
Computation Tree

- Our computation tree is probabilistic while Karger-Stein's is deterministic

- They have more branches per level *in exception*



our algorithm



Karger-Stein

Just As Accurate

- We randomly select a hyperedge to contract and branch probabilistically

Just As Accurate

- We randomly select a hyperedge to contract and branch probabilistically

tedious induction…

Just As Accurate

- We randomly select a hyperedge to contract and branch probabilistically

tedious induction…

- We return a minimum cut with probability $\geq 1 / (2H_n - 2)$ where $H_n = 1 + 1/2 + \ldots + 1 / n = \Theta(\log n)$

And Nearly as Fast

- **Lemma**: Given a hypergraph of order n, the computation tree contains expected $O((n / n_0)^2)$ hypergraphs of order n_0

And Nearly as Fast

- **Lemma**: Given a hypergraph of order n, the computation tree contains expected $O((n / n_0)^2)$ hypergraphs of order n_0

- We give a method to contract any hyperedge e in $O(m(n - |e| + 1))$ time

And Nearly as Fast

- **Lemma**: Given a hypergraph of order n, the computation tree contains expected $O((n / n_0)^2)$ hypergraphs of order n_0
- We give a method to contract any hyperedge e in $O(m(n - |e| + 1))$ time
- Sum over all n_0 to get a total running time of $O(mn^2 \log n)$ for BranchingContract

And Nearly as Fast

- **Lemma**: Given a hypergraph of order n, the computation tree contains expected $O((n / n_0)^2)$ hypergraphs of order n_0

- We give a method to contract any hyperedge e in $O(m(n - |e| + 1))$ time

- Sum over all n_0 to get a total running time of $O(mn^2 \log n)$ for BranchingContract

- Can compute a minimum cut with high probability in $O(mn^2 \log^3 n)$ time, a nearly $\Omega(n)$ improvement

Low Rank

- ***Rank*** *r* of a hypergraph is the maximum hyperedge size

Low Rank

- ***Rank*** *r* of a hypergraph is the maximum hyperedge size
- We give a method to contract any hyperedge in $O(n^{r-1})$ time when r is a constant

Low Rank

- ***Rank*** *r* of a hypergraph is the maximum hyperedge size
- We give a method to contract any hyperedge in $O(n^{r-1})$ time when r is a constant
- Can compute a minimum cut with high probability in $O(n^r \log^2 n)$ time when $r \geq 3$

Low Rank

- ***Rank*** r of a hypergraph is the maximum hyperedge size
- We give a method to contract any hyperedge in $O(n^{r-1})$ time when r is a constant
- Can compute a minimum cut with high probability in $O(n^r \log^2 n)$ time when $r \geq 3$
- Nearly optimal for dense hypergraphs (and we match Karger-Stein when $r = 2$)

Minimum *k*-cut

- Branch more often when $k \geq 3$

Minimum *k*-cut

- Branch more often when $k \geq 3$

- Minimum k-cut with high probability in $O(mn^{2k-2} \log^2 n)$ time for any $k \geq 3$

Minimum k-cut

- Branch more often when $k \geq 3$
- Minimum k-cut with high probability in $O(mn^{2k-2} \log^2 n)$ time for any $k \geq 3$
- Minimum k-cut with high probability in $O(n^{2k-2} \log^3 n)$ time if $r = 2k - 2$ and $O(n^{\max\{r,2k-2\}} \log^2 n)$ time otherwise for any constant k, $r \geq 2$

Thanks!