Randomized algorithms 4
Distributed computing

CS240 Spring 2020

Rui Fan

Distributed computing

- Distributed system
  - Set of autonomous nodes, working independently of each other.
  - Nodes may be able to communicate, at a cost.
  - Ex Internet, computer cluster, sensor network.
- Nodes need to coordinate to solve some problem.
- Coordination can be done using communication. But communication is expensive.
- By making nodes randomized, they can coordinate with minimal communication.
- Randomization also simplifies symmetry breaking between nodes.
- Today we'll look at randomized contention resolution and maximal independent set.

Contention resolution

- Set of n nodes (e.g. cellphones) want to send each other messages.
- Only one node can send at a time.
  - If two nodes send at same time, their signals interfere and both transmissions fail.
- Nodes can’t communicate.
  - Communicating requires sending messages, which is the problem we’re trying to solve!
  - Nodes can’t coordinate to work out a schedule. They have to randomize.

Contention resolution

- Assume system is synchronous.
  - Nodes work in rounds.
  - Each node can try to send once per round. It succeeds if and only if it's the only node to try to send in that round.

- Algorithm Each node tries to send with probability 1/n in every round.

Analysis

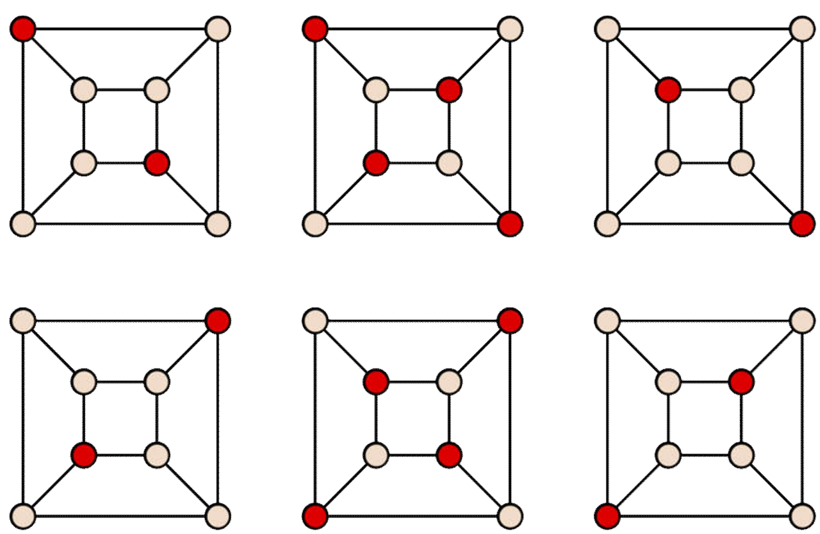
- How many rounds before all the nodes can send?
- Let $S_{i,t}$ be the event that node i successfully sends in t'th round.
  - $S_{i,t}$ occurs iff i tries to send in t'th round and all other nodes do not.
- $Pr[S_{i,t}]=1/n*(1-1/n)^{n-1}$.
  - i tries to send with prob. 1/n, and each of i's n-1 neighbors don't send with prob. 1-1/n.
- Fact For all $n\geq 2$, $1/e \leq (1-1/n)^{n-1} \leq ½$, and $¼ \leq (1-1/n)^n \leq 1/e$.
- So $Pr[S_{i,t}] \geq 1/en$.
- Pr[i fails to send in t'th round] = $1- Pr[S_{i,t}] \leq 1-1/en$.

Analysis

- Thm After $2e*n \ln(n)$ rounds, all nodes succeed sending with probability $\geq 1-1/n$.
- Proof Let F_i denote event that node i fails to send after $2e*n \ln(n)$ rounds, and let F denote event that any node fails to send after $2e*n \ln(n)$ rounds.
  - $Pr[F_i] \leq (1-1/en)^{2e*n \ln(n)} \leq (1/e)^{2 \ln(n)} \leq 1/n^2$.
    - In each round i fails independently with prob. $\leq (1-1/en)$.
  - $Pr[F] \leq \Sigma_i Pr[F_i] \leq n*1/n^2=1/n$, by the union bound.
  - So all nodes succeed with prob. $\geq 1-1/n$.

Maximal independent set

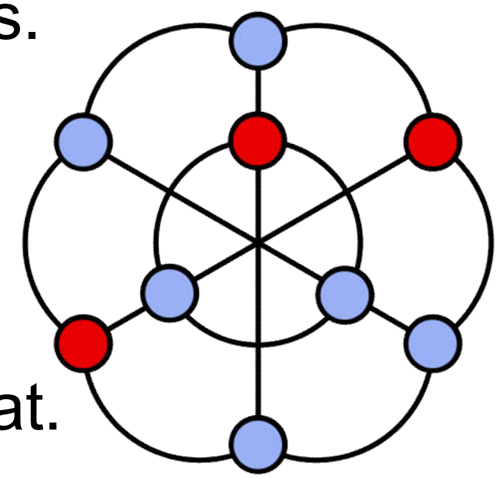
- Given a graph, an independent set is a set of vertices, none of which are connected to each other.
- A maximal independent set (MIS) is an independent set such that if we add any other vertex, it would be connected to some vertex in the independent set.
  - I.e. An MIS can't be made any larger.
- A maximum independent set (MaxIS) is an independent set of maximum cardinality in the graph.
- Note that an MIS might not be a MaxIS. An MIS is a "local" max, while a MaxIS is the "global" max.



All 6 MIS's of the cube graph. Note only the two center MIS's are MaxIS.

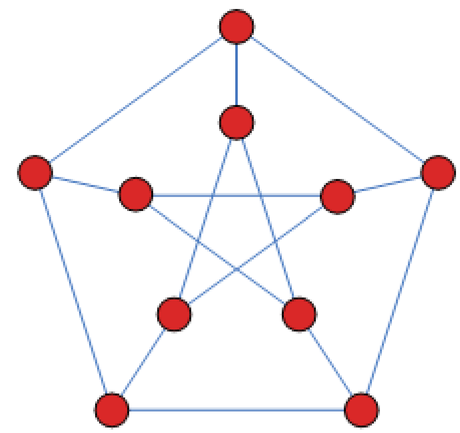
Distributed MIS

- Compute an MIS on a network of n nodes.
  - The MIS nodes can be "leaders", used to coordinate the other nodes in some distributed computation.
- A simple algorithm is to continually add a node to the MIS, then remove its neighboring nodes and edges, then repeat.
- This algorithm takes O(n) time.
- It's also sequential. We have to remove all the neighbors of a selected node before we select the next node.
  - Otherwise we can add two neighboring nodes both to the MIS.
- We want a fast, distributed MIS algorithm.



Distributed MIS

- Again consider synchronous model where all nodes work in rounds.
- Each node can broadcast a message to its neighbors in each round.



- Each node v chooses a random number $r(v) \in [0,1]$ and sends it to its neighbors.
- If $r(v)<r(w)$ for all neighbors w of v, then v adds itself to the MIS and informs its neighbors.
- If v or one of its neighbors entered the MIS, v terminates. Remove all of v's edges.
- Otherwise go back to first step, until graph is empty.

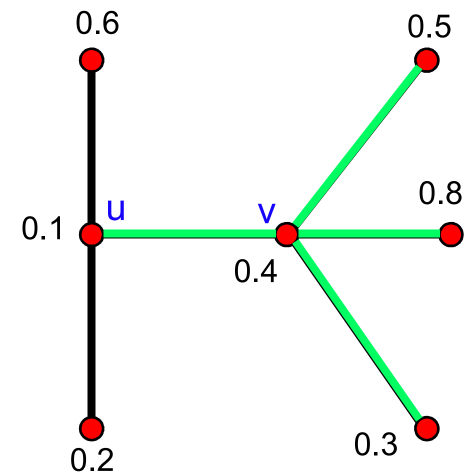
- Call these three steps a phase.
- Assume no ties, i.e. for any u,v, either $r(u)<r(v)$ or vice versa.

Analysis

- We show this algorithm outputs an MIS, and terminates quickly.
- The output is an independent set.
  - For every two neighbors, only the one with smaller r value can join MIS.
  - When a node joins the MIS, all its neighbors are removed and can’t join the MIS.
- It’s a maximal IS because we only ever take away a node if its neighbor is in the MIS.

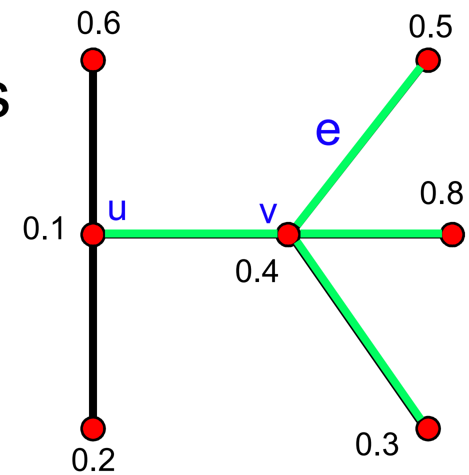
Analysis

- How many rounds does it take to terminate?
- Lemma In each phase, at least half the edges are removed in expectation.
  - We'll prove this after proving Claims 1-4.
- Def Let u, v be two nodes. Say u preemptively removes v if $u \in N(v)$, and $r(u)<r(u')$ for all $u' \in N(u) \cup N(v)$.
  - Denote as $u<<v$.
  - Given an edge $e=(v,w)$, we say e is preemptively removed by u if $u<<v$.
- Claim 1 For any v, there's at most one u s.t. $u<<v$.
  - Proof If $u<<v$, then u is the neighbor of v with min r value.



Analysis

- Let P = {all preemptively removed edges in phase}, R = {all edges removed in phase}.
- Claim 2 $P \subseteq R$
  - Let $e=(v,w) \in P$. Then v has nbr u s.t. $r(u)<r(u')$ for all $u' \in N(u) \cup N(v)$.
  - So u will get removed.
  - So v is also removed. All edges incident to v, including e, are also removed. So $e \in R$.
- Let $X_{u<<v}=1$ if $u<<v$ and 0 otherwise.
- If $X_{u<<v}=1$, all edges incident to v are removed.
  - So if $X_{u<<v}=1$, d(v) edges get removed, where d(v) is degree v.



Analysis

- Claim 3 $\Sigma_u \Sigma_{v \in N(u)} X_{u<<v} * d(v) \leq 2*|P|$.
  - Given any edge e=(v,w), the sum counts e once each time e is preemptively removed by some other node u.
  - How many such u's are there?
    - u preemptively removes e only if u<<v or u<<w.
    - By claim 1, there's only at most one u that << v, and at most one that << w.
    - So e is preemptively removed by at most 2 other nodes.
  - So any e in the sum is a preemptively removed edge that's counted at most twice.
  - Since P is set of all preemptively removed edges, then the sum $\leq 2*|P|$.
- Cor 1 $\Sigma_u \Sigma_{v \in N(u)} X_{u<<v} * d(v) \leq 2*|R|$.
  - Because $P \subseteq R$ by Claim 2.

Analysis

- Claim 4 $E[\sum_u \sum_{v\in N(u)} X_{u<<v} * d(v)] \geq |H|$, where $H=\{edges\}$.
  - For any u and $v \in N(u)$, $E[X_{u<<v} * d(v)] = Pr[u<<v] * d(v)$.
  - $u<<v$ only if $r(u)<r(u')$ for all $u' \in N(u) \cup N(v)$.
  - There are at most $d(u)+d(v)$ nodes in $N(u) \cup N(v)$.
  - Each node picks a random value r. Probability it's min among $\leq d(u)+d(v)$ random values is $\geq 1/(d(u)+d(v))$.
  - So $Pr[u<<v] \geq 1/(d(u)+d(v))$.
  - So $E[X_{u<<v} * d(v)] \geq d(v)/(d(u)+d(v))$.
  - $E[\sum_u \sum_{v\in N(u)} X_{u<<v} * d(v)] =$

    $\sum_{e=(u,v)\in H} (E[X_{u<<v} * d(v)] + E[X_{v<<u} * d(u)]) \geq$

    $\sum_{e=(u,v)\in H} d(v)/(d(u)+d(v)) + d(u)/(d(u)+d(v)) =$

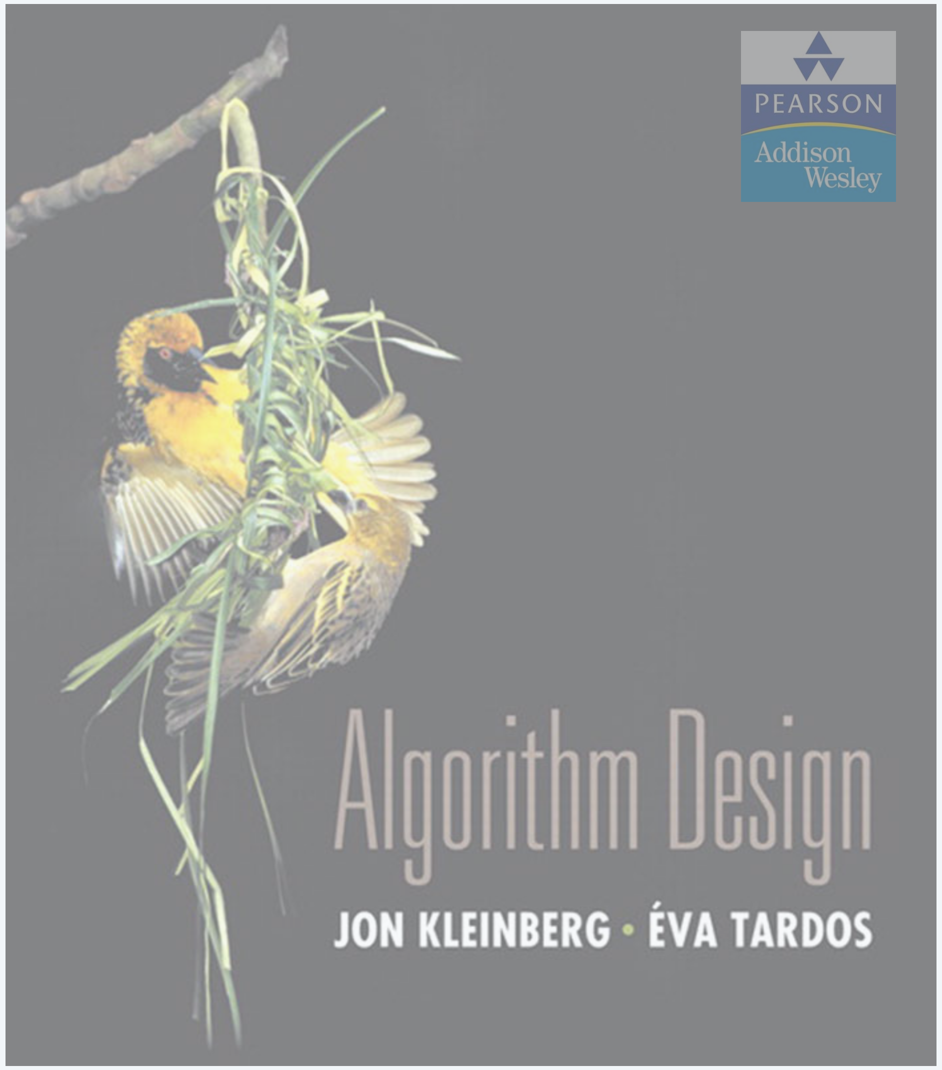
    $\sum_{e=(u,v)\in H} 1 = |H|$.

Analysis

- Proof of Lemma
  - By Claim 4 and Cor. 1, $|H| \leq 2*E[|P|] \leq 2*E[|R|]$.
  - So $E[|R|] \geq |H|/2$, i.e. half the edges get removed in expectation every phase.
- Cor 2 With probability $\geq 1/3$, at least 1/4 the edges get removed in every phase.
  - Otherwise, the probability less than 1/4 edges get removed every phase is greater than 2/3.
  - So expected number of edges removed in the phase is $< 2/3*|E|/4+1/3*|E| = |E|/2$, contradicting the lemma.
- Thm The algorithm computes an MIS in $42*\ln(n)$ phases with probability $\geq 1-1/n$.

Analysis

- Proof Say a phase is good if $\geq$ 1/4 the edges get removed.
  - So Pr[phase is good] $\geq$ 1/3 by Cor 2. Also, these probabilities are independent.
  - In 42*ln(n) phases, we expect $\geq \mu$=14*ln(n) good phases .
  - Pr[< 7*ln(n) good phases in 42*ln(n) rounds] = Pr[number good rounds < ½ expectation] $\leq$ $e^{-14*\ln(n)/8}$ <1/n, by Chernoff bounds.
  - If we get 7*ln(n) good phases, then fraction of remaining edges is $\leq (3/4)^{7*\ln(n)} = n^{7*\ln(3/4)} \approx n^{-2.01}$.
  - Since there are $O(n^2)$ edges, all the edges get removed after 7*ln(n) good phases.
  - We get 7*ln(n) good phases in 42*ln(n) phases with probability > 1-1/n.



13. Randomized Algorithms

- *contention resolution*
- *global min cut*
- *linearity of expectation*
- *max 3-satisfiability*
- *universal hashing*
- *Chernoff bounds*
- *load balancing*

Global minimum cut

Global min cut. Given a connected, undirected graph $G = (V, E)$, find a cut (A, B) of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

- Replace every edge (u, v) with two antiparallel edges (u, v) and (v, u).
- Pick some vertex s and compute min s–v cut separating s from each other node $v \in V$.

False intuition. Global min-cut is harder than min s-t cut.

Contraction algorithm

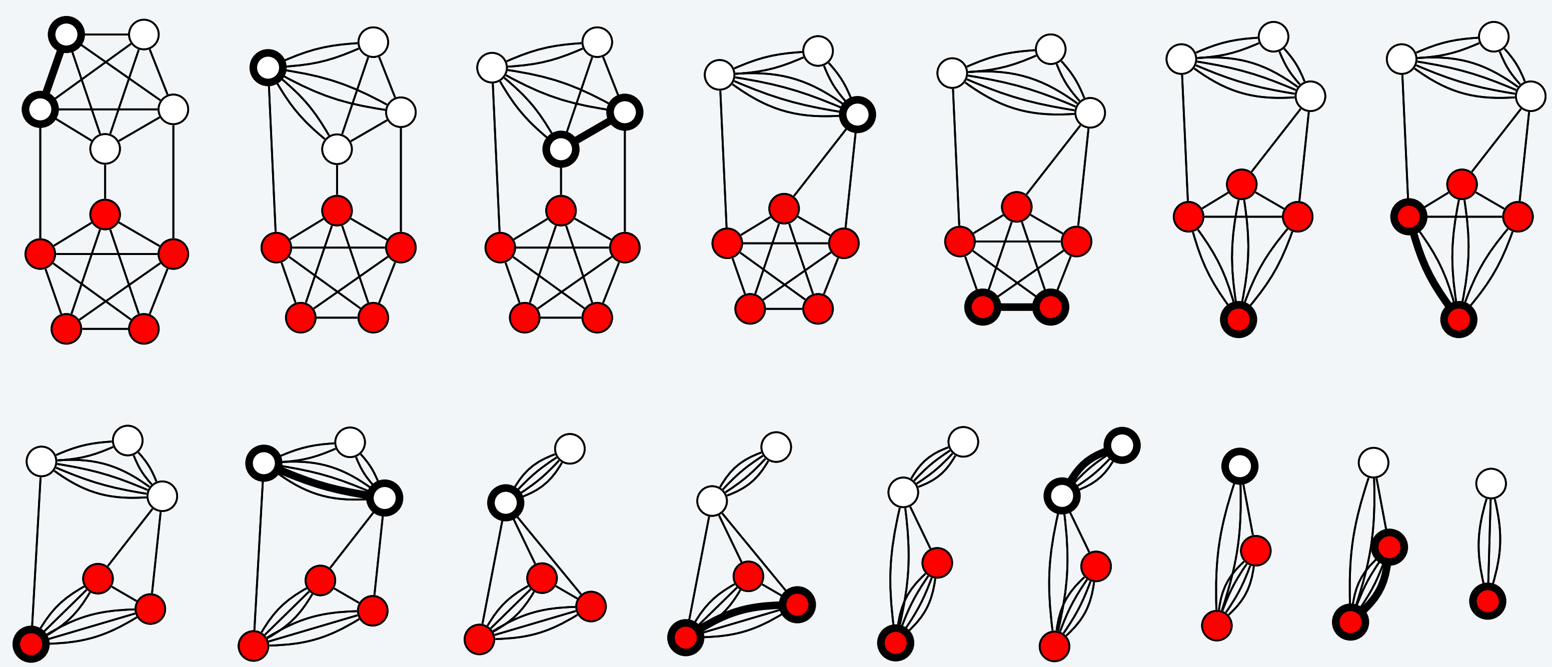
Contraction algorithm. [Karger 1995]

- Pick an edge $e = (u, v)$ uniformly at random.
- Contract edge e.
  - replace u and v by single new super-node w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes u_1 and v_1.
- Return the cut (all nodes that were contracted to form v_1).

Contraction algorithm

Contraction algorithm. [Karger 1995]

- Pick an edge $e = (u, v)$ uniformly at random.
- Contract edge e.
  - replace u and v by single new super-node w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes u_1 and v_1.
- Return the cut (all nodes that were contracted to form v_1).



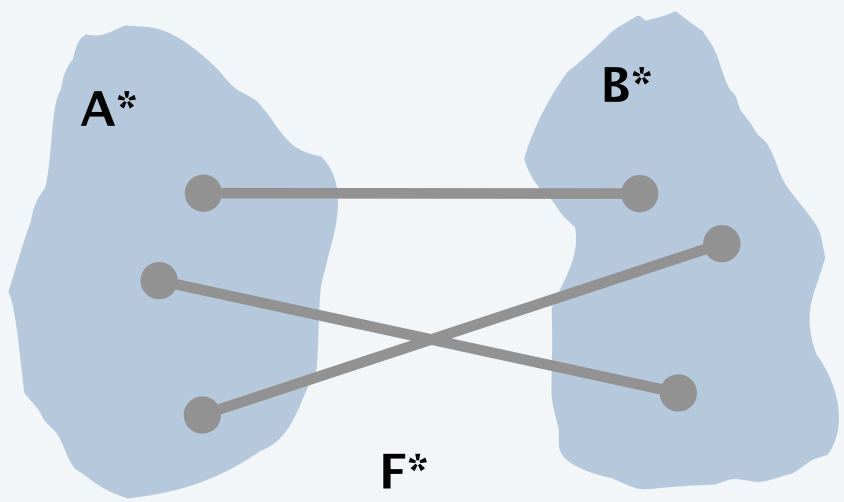
Reference: Thore Husfeldt

Contraction algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2 / n^2$.

Pf. Consider a global min-cut (A^*, B^*) of G.

- Let F^* be edges with one endpoint in A^* and the other in B^*.
- Let $k = |F^*|$ = size of min cut.
- In first step, algorithm contracts an edge in F^* probability $k / |E|$.
- Every node has degree $\geq k$ since otherwise (A^*, B^*) would not be a min-cut $\Rightarrow |E| \geq \frac{1}{2} k n \Leftrightarrow k / |E| \leq 2 / n$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n$.



Contraction algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2 / n^2$.

Pf. Consider a global min-cut (A^*, B^*) of G.

- Let F^* be edges with one endpoint in A^* and the other in B^*.
- Let $k = |F^*|$ = size of min cut.
- Let G' be graph after j iterations. There are $n' = n - j$ supernodes.
- Suppose no edge in F^* has been contracted. The min-cut in G' is still k.
- Since value of min-cut is k, $|E'| \geq \frac{1}{2} k n' \;\Leftrightarrow\; k / |E'| \leq 2 / n'$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n'$.
- Let E_j = event that an edge in F^* is not contracted in iteration j.

$$
\begin{aligned}
\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] &= \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}] \\
&\geq \left(1 - \tfrac{2}{n}\right)\left(1 - \tfrac{2}{n-1}\right) \cdots \left(1 - \tfrac{2}{4}\right)\left(1 - \tfrac{2}{3}\right) \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right) \cdots \left(\frac{2}{4}\right)\left(\frac{1}{3}\right) \\
&= \frac{2}{n(n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}
$$

Contraction algorithm

Amplification. To amplify the probability of success, run the contraction algorithm many times.

with independent random choices,

Claim. If we repeat the contraction algorithm $n^2 \ln n$ times, then the probability of failing to find the global min-cut is $\leq 1 / n^2$.

Pf. By independence, the probability of failure is at most

$$\left(1 - \frac{2}{n^2}\right)^{n^2 \ln n} = \left[\left(1 - \frac{2}{n^2}\right)^{\frac{1}{2} n^2}\right]^{2 \ln n} \leq \left(e^{-1}\right)^{2 \ln n} = \frac{1}{n^2}$$

$(1 - 1/x)^x \leq 1/e$

Contraction algorithm: example execution

trial 1

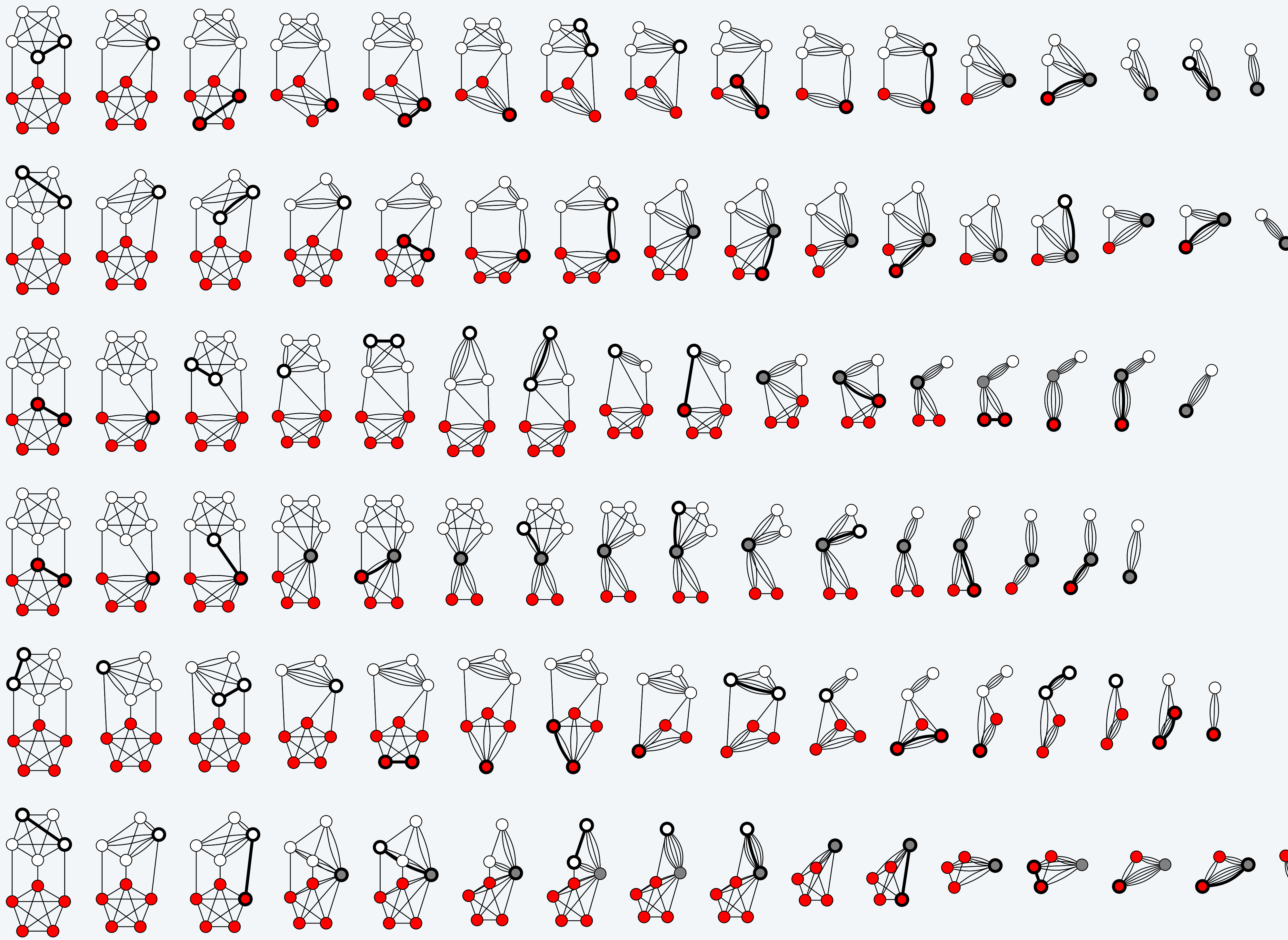
trial 2

trial 3

trial 4

trial 5
(finds min cut)

trial 6



...

Reference: Thore Husfeldt

Global min cut: context

Remark. Overall running time is slow since we perform $\Theta(n^2 \log n)$ iterations and each takes $\Omega(m)$ time.

Improvement. [Karger–Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm until $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph and return best of two cuts.

Extensions. Naturally generalizes to handle positive weights.

Best known. [Karger 2000] $O(m \log^3 n)$.

faster than best known max flow algorithm or
deterministic global min cut algorithm