CS630 Graduate Algorithms

October 22, 2024

by Dora Erdos and Jeffrey Considine

- min-cut KT ch. 13.2

min st-cut in directed graphs

Max Flow Min Cut theorem: given a directed graph G(V,E) with source s, sink t and edge capacities the value of the max flow is = the capacity of the min st-cut

⇒ find the min-cut through the max-flow.

min st-cut in undirected unweighted graph

Find the min st-cut in the undirected, unweighted graph G(V,E) with source s and sink t.

capacity/weight of a cut = number of edges between S and V-S.

How?

Global min-cut

Global min-cut: Given an undirected graph *G(V,E)*, find a cut that partitions the nodes into two sets *A* and *V-A* with minimum weight.

- no designated source or sink
- $weight(C) = \sum_{edge\ e\ in\ C} w(e)$
- works for unweighted and weighted graphs (today: unweighted)

Deterministic algorithm (using st-cuts):

Global min-cut

Global min-cut: Given an undirected graph *G(V,E)*, find a cut that partitions the nodes into two sets *A* and *V-A* with minimum weight.

- no designated source or sink
- $$weight(C) = \sum_{edge\ e\ in\ C} w(e)$$
- works for unweighted and weighted graphs (today: unweighted)

Deterministic algorithm:

For each pair of vertices *u, v* assign *u* as source *v* as sink, and run the min st-cut algorithm

- since this is an undirected graph, it doesn't matter which of u and v is the source

running time:

- there are $O(n^2)$ iterations
- each requires a run of FF. We know that C is at most the max degree D.
- in total $O(n^2mD)$

Global min-cut

Global min-cut: Given an undirected graph *G(V,E)*, find a cut that partitions the nodes into two sets *A* and *V-A* with minimum weight.

Deterministic algorithm — speed up:

claim: it's enough to fix one source s, and compute the min-cut between s and every other node.

runtime:

Global min-cut

Global min-cut: Given an undirected graph *G(V,E)*, find a cut that partitions the nodes into two sets *A* and *V-A* with minimum weight.

Deterministic algorithm — speed up:

claim: it's enough to fix one source s, and compute the min-cut between s and every other node.

proof:

- in any min-cut A and V-A, the node s is assigned to one of the two. wlog we can assume that s is assigned to A.
- let v be some other node in A
- since s and v are both in A, this means that any cut separating the two is larger than the min-cut
- This implies that if v were the source instead of s, the same set A would have been found.

Randomized algorithm for min-cut

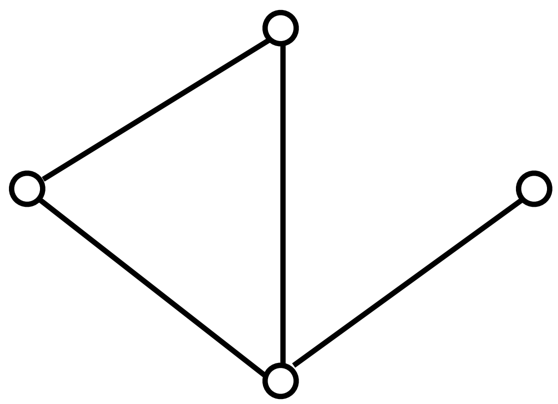
- O(nmD) is polynomial, but slow
    - if $n = 10^6$, $m = O(n^2)$, $D = 10$, then we get $O(10^{19})$

- idea: make random choices
  - give up on finding an exact solution, i.e. the true min-cut
      - do we really give up…?
  - make it very fast
      - we can get O(n)

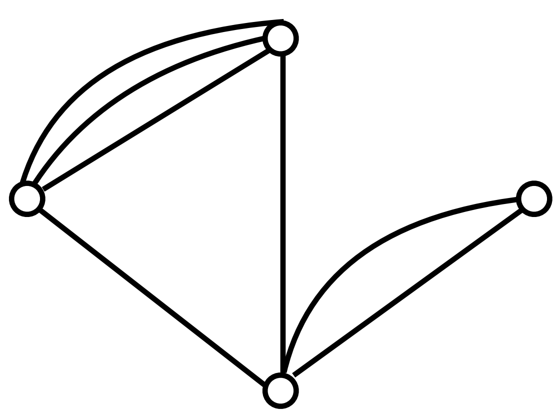
  - result: the algorithm will find the min-cut with high(ish) probability
      - trick: repeat the algorithm multiple times and return the best of the outputs

multi graphs

simple graph: each pair of vertices are connected by at most one edge



multi graph: vertices may have multiple edges between them

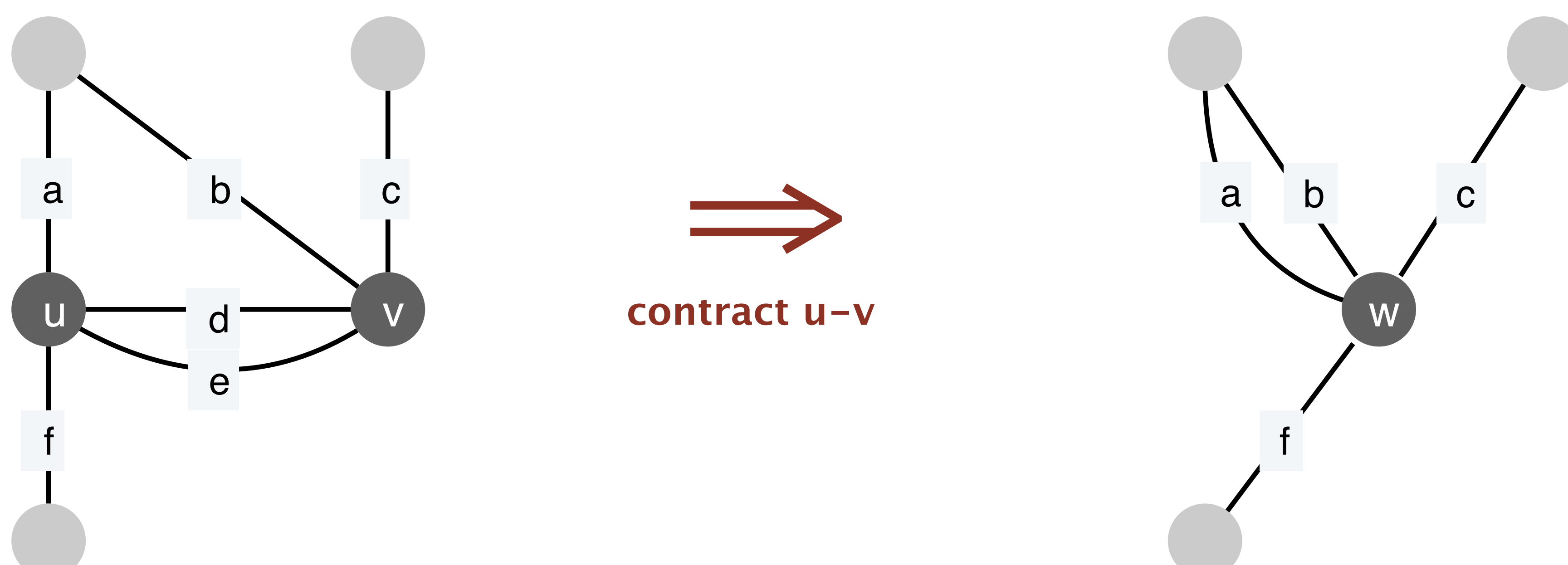


Contraction algorithm

Contraction algorithm. [Karger 1995]

contraction operation:

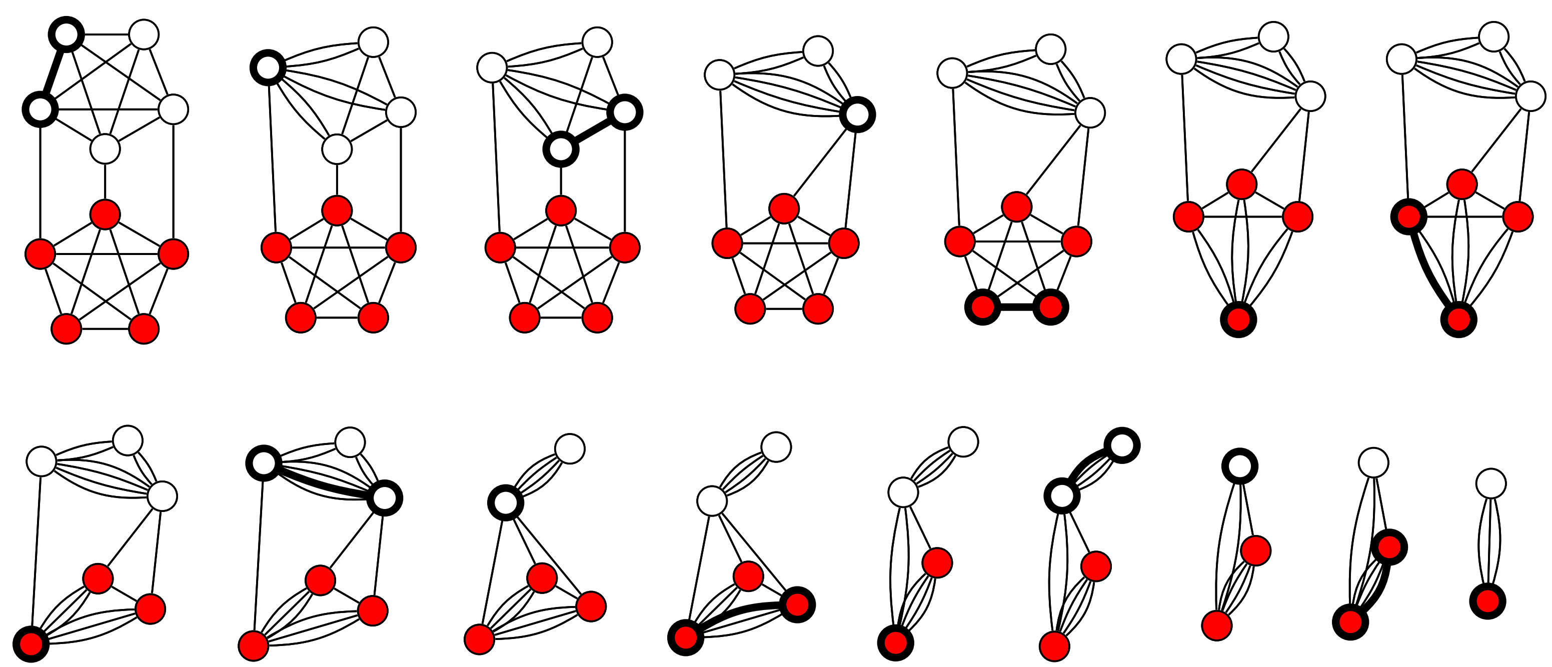
- select an edge (u,v) at random
- replace u and v by a single super-node w
  - hereby deleting all edges between (u,v)
  - all edges previously adjacent to either u or v are now connected to w
    - keep parallel edges but delete self loops



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 v_1 and v_2.
- Return the cut (all nodes that were contracted to form v_1).



Reference: Thore Husfeldt

Contraction algorithm – probability analysis

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

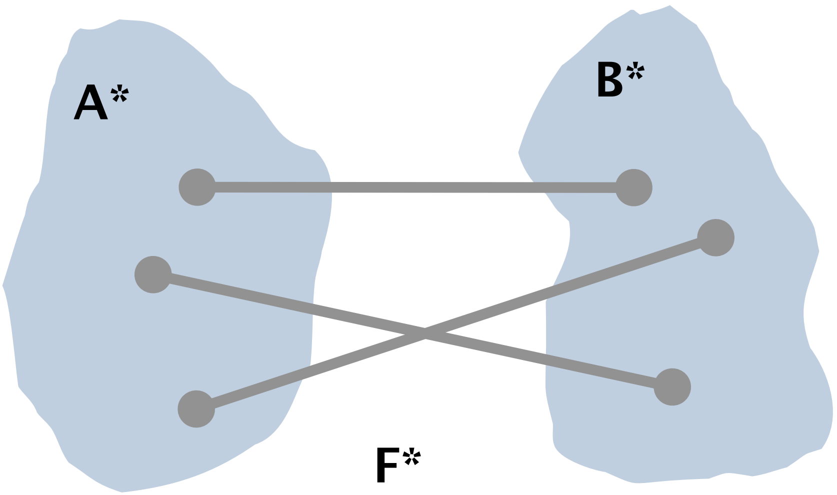
the contraction algorithm finds a min-cut, iff none of the edges in the min-cut get contracted.

goal: compute the probability of the event that the min-cut edges are not selected for contraction

Intuition:

vertices connected by multi-edges are more likely to be contracted

implies that larger cuts are more likely to be contracted

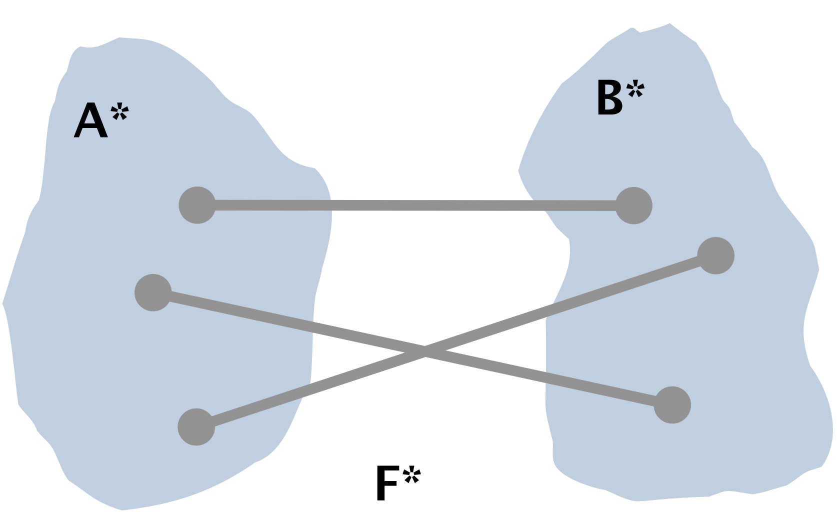


Contraction algorithm – probability analysis

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.
- pick a random edge to contract —> what is the probability that this edge is in F*?

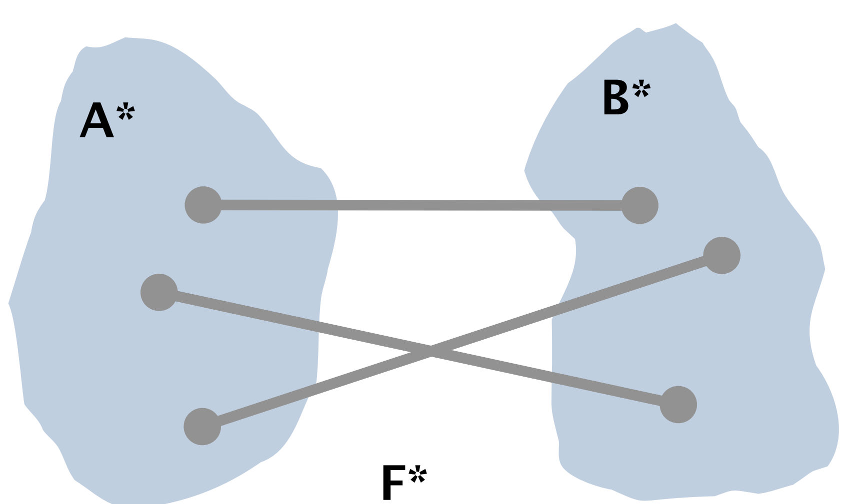


Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.
- observation: each node has degree at least k
- number of edges |E| =



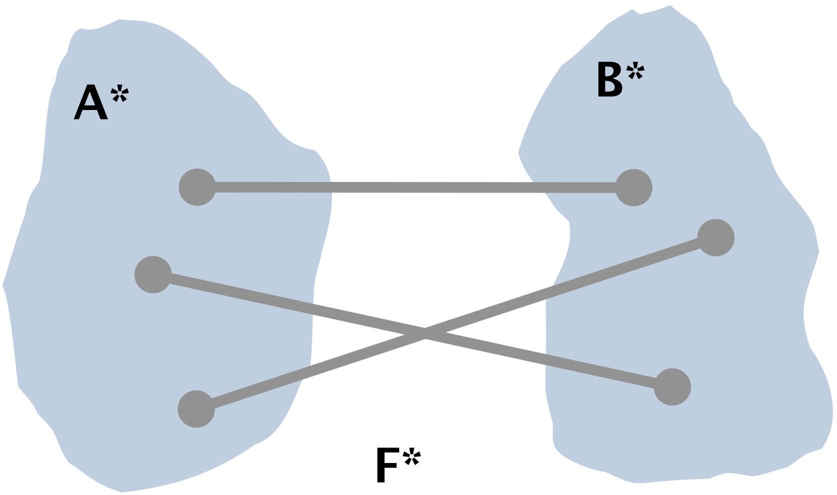
Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.

- observation: each node has degree at least k
  - if some node v had degree less than k, then separating v from the rest of the graph creates a cut smaller than k
- number of edges $|E| = \frac{1}{2}kn$
  - each vertex has degree $\geq k$, there are n vertices. This way each edge is counted twice.



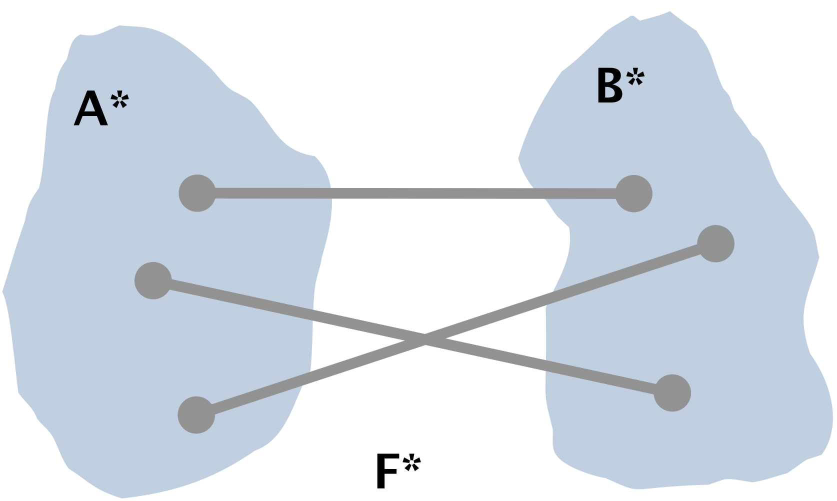
Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.

- In first step, algorithm contracts an edge in F^* probability $k / |E|$.
  - size of |E|?
    - Every node has degree $\geq k \Rightarrow |E| \geq \frac{1}{2}kn$
- Thus, algorithm contracts an edge in F^* with probability $\leq$



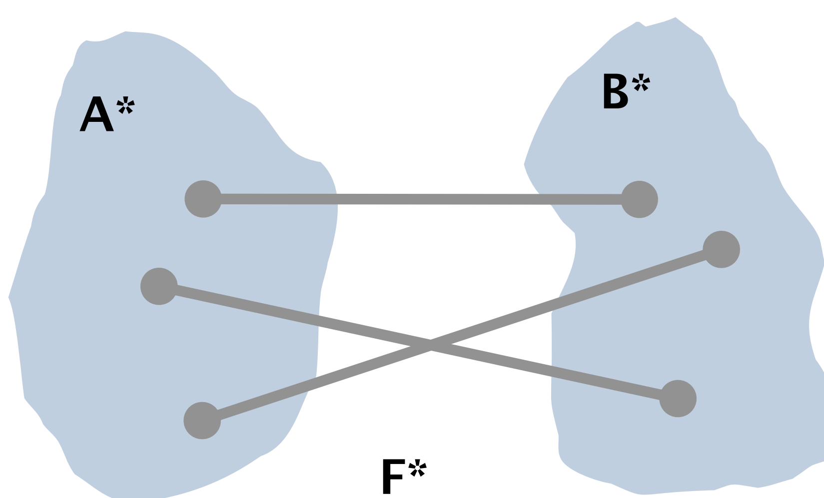
Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.

- After j iterations, we have (contracted) graph G' with n' = n-j nodes
- if none of the edges in F* have been contracted:
  - the min-cut in G' still has size k
  - number of edges $|E'| \geq \frac{1}{2}kn'$
  - the algorithm contracts an edge in F* with probability $\frac{2}{n'}$



Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.

- After j iterations, we have (contracted) graph G' with n' = n-j nodes
  - the algorithm contracts an edge in F* with probability $\frac{2}{n'}$
  - E_j = event that an edge in F* is *not* contracted in iteration j

$$P(E_1 \cap E_2 \cap \ldots E_{n-2}) =$$

Contraction algorithm – probability analysis

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

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

- F* = edges in the min cut, $k = |F^*|$ = size of min cut.

- After j iterations, we have (contracted) graph G' with n' = n-j nodes
  - the algorithm contracts an edge in F* with probability $\dfrac{2}{n'}$
  - E_j = event that an edge in F* is *not* contracted in iteration j

$$P(E_1 \cap E_2 \cap \ldots E_{n-2}) = P(E_1) \cdot P(E_2 \,|\, E1) \cdot \ldots \cdot P(E_{n-2} \,|\, E_1 \cap E_2 \cap \ldots \cap E_{n-3})$$

chain rule

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\ldots\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right)$$

substitute 2/n'

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\ldots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

$$= \frac{2}{n(n-1)} \geq \frac{2}{n^2}$$

Contraction algorithm: example execution

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

trial 1

trial 2

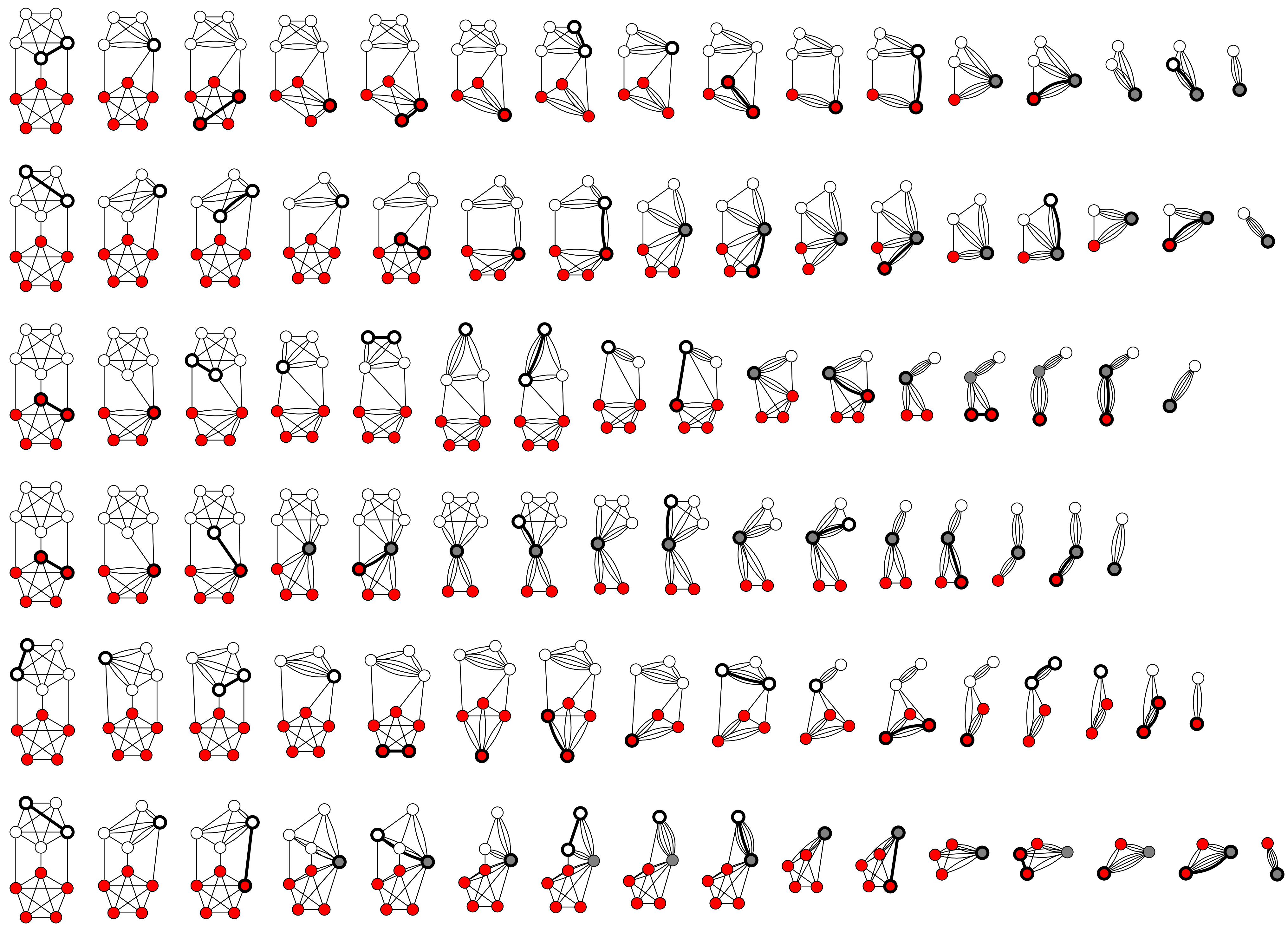
trial 3

trial 4

trial 5
(finds min cut)

trial 6

...



Reference: Thore Husfeldt

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 $\leq 1 / n^2$.

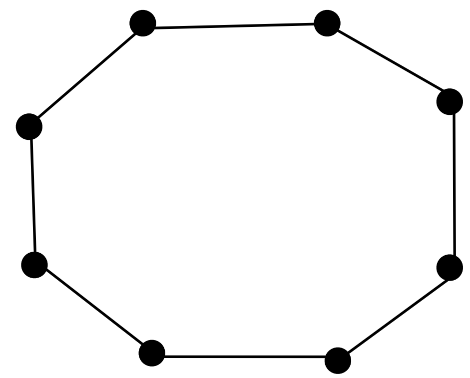
Pf. By independence, the probability of failure is at most

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

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

max number of global min-cuts in undirected graphs

How many global min-cuts on a cycle of length-n?



How many global min-cuts (in worst case) are there in an undirected unweighted graph?

Union bound

Event space (universe) Ω of all possible outcomes of a random trial

event $E_i \subseteq \Omega$ one (set of) possible outcome

example of events:

- roll 2 with a dice
- roll an even number with a dice
- process p_i is unsuccessful at accessing the database after t attempts
- the

intuitively: union of sets is less than the sum of individual sets

Union bound: given events $E_1, E_2, \ldots E_r$ we have $P\left(\cup_{i=1}^{r} E_i\right) \leq \sum_{i=1}^{r} P(E_i)$

Number of global min-cuts

Claim: An undirected graph on n nodes has at most $\binom{n}{2}$ global min-cuts.

Number of global min-cuts

Claim: An undirected graph on n nodes has at most $\binom{n}{2}$ global min-cuts.

proof:

- $C_1, C_2, \ldots C_r$ are the global min-cuts
- Let E_i be the event that C_i is returned by the Contraction Algorithm
- And $E = \cup_{i=1}^{r} E_i$ the event that the contraction algorithm returns *a* min-cut

- we know $P(E_i) \geq \frac{2}{n^2}$ from previous proof
- $P(E) = P(\cup_{i=1}^{r} E_i) = \sum_{i=1}^{r} P(E_i) \geq \frac{r}{n^2}$
- finally $1 \geq P(E) = \frac{r}{n^2} \rightarrow n^2 \geq r$

Union bound suggests $\leq$, however the events E_i are independent (as the contraction algorithm can only return one of C_i in one specific run)

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

Best known deterministic. [Nagamochi-Ibaraki 1992] $O(mn + n^2 \log n)$