COMP251: Randomized Algorithms

Jérôme Waldispühl & Roman Sarrazin-Gendron
School of Computer Science
McGill University

Based on (Kleinberg & *Tardos*, 2006)

Algorithm Design Techniques

- Greedy Algorithms
- Dynamic Programming
- Divide-and-Conquer
- Network Flows
- Randomization

Randomization

Principle: Allow fait coin flip in unit time.

Why? Can lead to simplest, fastest, or only known algorithm for a particular problem.

Examples:

- Quicksort
- Graph Algorithms
- Hashing
- Monte-Carlo integration
- Cryptography

Global Min Cut

Definition: Given a connected, undirected graph G=(V,E), find a cut with minimum cardinality.

Applications:

- Partitioning items in database
- Identify clusters of related documents
- Network reliability
- TSP solver

Network solution:

- Replace every edge (u,v) with 2 antiparallel edges (u,v) & (v,u)
- Pick some vertex s, and compute min s-v cut for each other vertex v.

False Intuition: Global min-cut is harder that min s-t cut!

Contraction algorithm

Contraction algorithm. [Karger 1995]

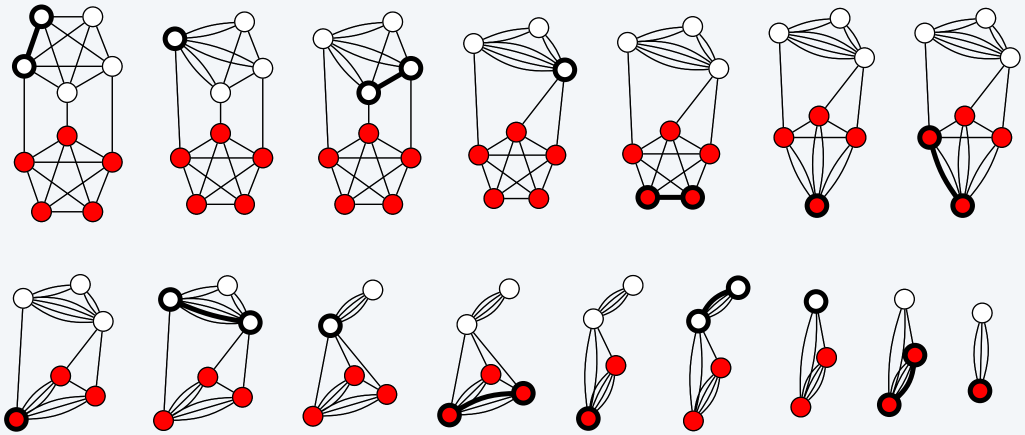
- Pick an edge $e = (u, v)$ uniformly at random.
- Contract edge e.
  - replace u and v by single new super-node w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes u_1 and v_1.
- Return the cut (all nodes that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

Merging nodes is equivalent to build disjoint sets of increasing sizes



Reference: Thore Husfeldt

Contraction Algorithm

Randomization

Contraction(V,E):

While $|V| > 2$ do

Choose $e \in E$ uniformly at random

$G \leftarrow G - \{e\}$ // contract G

return { the only cut in G }

Why are we doing that?
What is the likelihood to get a global min cut at the end?

Contraction algorithm

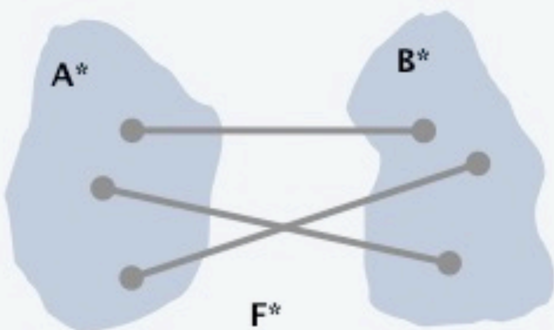
Claim. The contraction algorithm returns a min cut with prob $\geq 2 / n^2$.
$(n = |V|)$

That's the probability to pick a wrong edge…

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 $\Rightarrow |E| \geq \frac{1}{2} k n$. $\Leftrightarrow \frac{k}{|E|} \leq \frac{2}{n}$
- Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n$.

If a node u has a degree $d_u < k$, we can make a partition $\{\{u\}, V-\{u\}\}$ whose cut is $d_u < k$ (and F^* cannot be a global min-cut).



Contraction algorithm

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

We are just repeating the same observations

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, $|E'| \geq \frac{1}{2} k n'$.
- 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.

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

$2/n^2$ is not much but it is still something…

Contraction algorithm

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

with independent random choices,

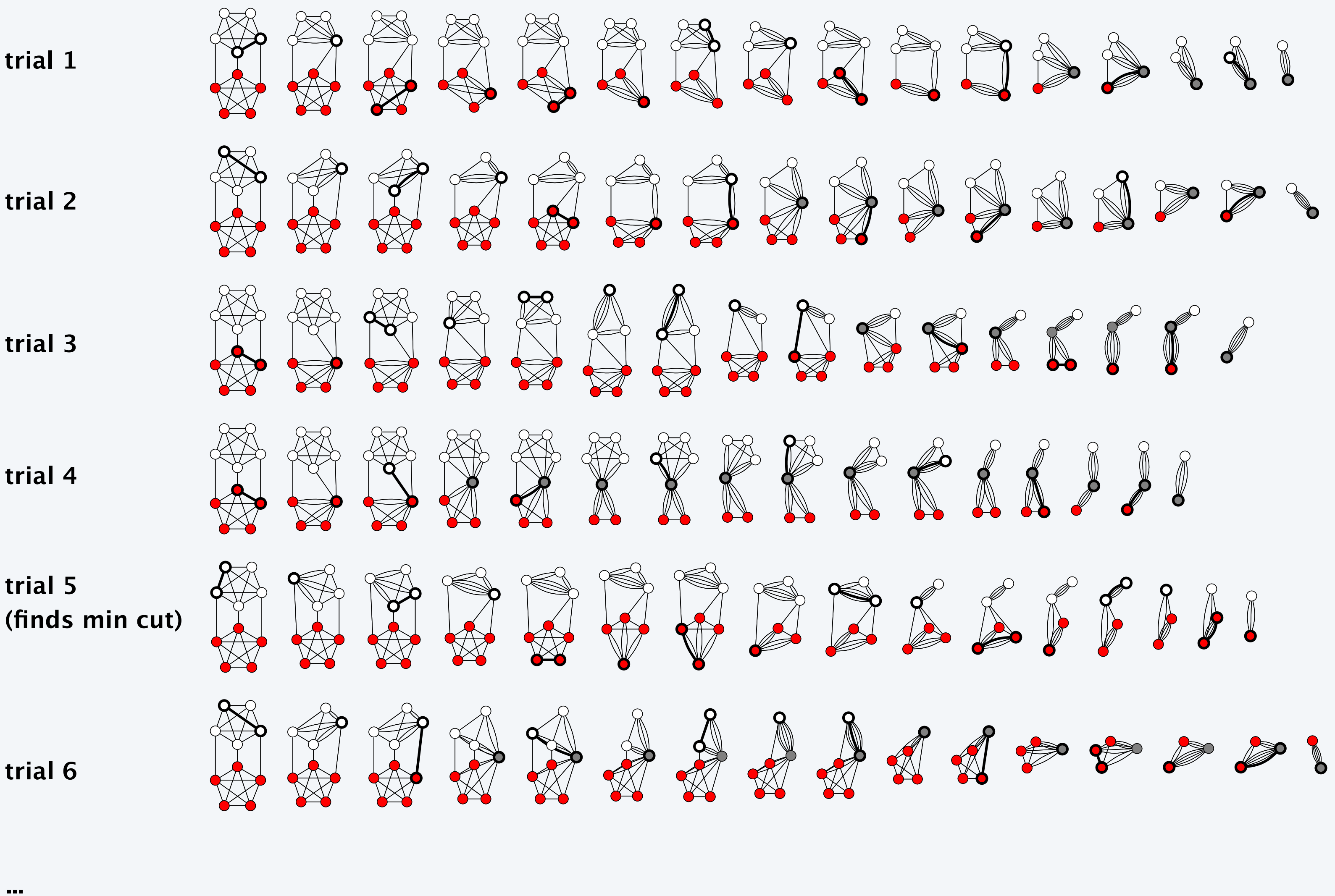
Claim. If we repeat the contraction algorithm $n^2 \ln n$ times, then the probability of failing to find the global min-cut is $\le 1 / n^2$.

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

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

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

Contraction algorithm: example execution



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. Where $m = |E|$. Overall complexity $O(n^2 m \log n)$

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

SAT formula

Boolean variables: $x_i$or $\overline{x_i}$

Clause: $C_i = x_{i_1} \vee x_{i_2} \vee x_{i_3} \vee \cdots \vee x_{i_k}$

A clause is true if one of his variable is true

Size of a clause = number of variables in the clause

SAT formula: $\bigwedge_i C_i$ (a formula is satisfied if all clauses are true)

Example (3 SAT formula):

$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_4)$

x_1=1; x_2=1; x_3=1; x_4=1 ⇒ True

x_1=1; x_2=0; x_3=1; x_4=0 ⇒ False

Maximum 3-satisfiability

exactly 3 distinct literals per clause

Maximum 3-satisfiability. Given a 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

$$\begin{aligned}
C_1 &= x_2 \vee \overline{x_3} \vee \overline{x_4} \\
C_2 &= x_2 \vee x_3 \vee \overline{x_4} \\
C_3 &= \overline{x_1} \vee x_2 \vee x_4 \\
C_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
C_5 &= x_1 \vee \overline{x_2} \vee \overline{x_4}
\end{aligned}$$

Remark. **NP**-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability ½, independently for each variable.

Is it a good idea?

Maximum 3-satisfiability: analysis

Claim. Given a 3-SAT formula with k clauses, the expected number of clauses satisfied by a random assignment is $7k / 8$.

Pf. Consider random variable $Z_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$

- Let Z = weight of clauses satisfied by assignment Z_j.

$$\begin{aligned} E[Z] &= \sum_{j=1}^{k} E[Z_j] \\ &= \sum_{j=1}^{k} \Pr[\text{clause } C_j \text{ is satisfied}] \\ &= \tfrac{7}{8} k \end{aligned}$$

linearity of expectation

There is 2^3 possible assignments for a clause of 3-SAT. Only one makes the clause false (all variable are false). Thus, 7/8 return true.

The Probabilistic Method

Corollary. For any instance of 3-SAT, there exists a truth assignment that satisfies at least a $7/8$ fraction of all clauses.

Pf. Random variable is at least its expectation some of the time. ▪

Probabilistic method. [Paul Erdös] Prove the existence of a non-obvious property by showing that a random construction produces it with positive probability!

Although the proof uses probability, the conclusion is determined for *certain*, without any possible error.

Not at the final!

Maximum 3-satisfiability: analysis

Q. Can we turn this idea into a $7/8$-approximation algorithm?
A. Yes (but a random variable can almost always be below its mean).

Lemma. The probability that a random assignment satisfies $\geq 7k / 8$ clauses is at least $1 / (8k)$.

Pf. Let p_j be probability that exactly j clauses are satisfied; let p be probability that $\geq 7k / 8$ clauses are satisfied.

$$\begin{aligned} \tfrac{7}{8}k \;=\; E[Z] &= \sum_{j \geq 0} j\, p_j \\ &= \sum_{j < 7k/8} j\, p_j \;+\; \sum_{j \geq 7k/8} j\, p_j \\ &\leq \left(\tfrac{7k}{8} - \tfrac{1}{8}\right) \sum_{j < 7k/8} p_j \;+\; k \sum_{j \geq 7k/8} p_j \\ &\leq \left(\tfrac{7}{8}k - \tfrac{1}{8}\right) \cdot 1 \;+\; k\, p \end{aligned}$$

Split around the mean

Find an upper bound for each

Rearranging terms yields $p \geq 1 / (8k)$. ▪

Maximum 3-satisfiability: analysis

Johnson's algorithm. Repeatedly generate random truth assignments until one of them satisfies $\geq 7k / 8$ clauses.

Theorem. Johnson's algorithm is a $7/8$-approximation algorithm.

Pf. By previous lemma, each iteration succeeds with probability $\geq 1 / (8k)$. By the waiting-time bound, the expected number of trials to find the satisfying assignment is at most $8k$. ▪

An approximation algorithm returns a solution provably close to the optimal.

Monte Carlo vs. Las Vegas algorithms

Monte Carlo. Guaranteed to run in poly-time, likely to find correct answer.
Ex: Contraction algorithm for global min cut.

Las Vegas. Guaranteed to find correct answer, likely to run in poly-time.
Ex: Randomized quicksort, Johnson's MAX-3-SAT algorithm.

stop algorithm
after a certain point

Remark. Can always convert a Las Vegas algorithm into Monte Carlo, but no known method (in general) to convert the other way.