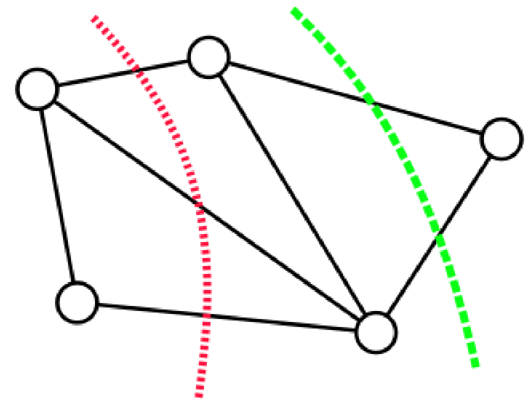
WIKIPEDIA

Karger's algorithm

In computer science and graph theory, **Karger's algorithm** is a randomized algorithm to compute a minimum cut of a connected graph. It was invented by David Karger and first published in 1993.[1]



A graph and two of its cuts. The dotted line in red is a cut with three crossing edges. The dashed line in green is a min-cut of this graph, crossing only two edges.

The idea of the algorithm is based on the concept of contraction of an edge (u, v) in an undirected graph $G = (V, E)$. Informally speaking, the contraction of an edge merges the nodes u and v into one, reducing the total number of nodes of the graph by one. All other edges connecting either u or v are "reattached" to the merged node, effectively producing a multigraph. Karger's basic algorithm iteratively contracts randomly chosen edges until only two nodes remain; those nodes represent a cut in the original graph. By iterating this basic algorithm a sufficient number of times, a minimum cut can be found with high probability.

Contents

References

The global minimum cut problem

A *cut* (S,T) in an undirected graph $G=(V,E)$ is a partition of the vertices V into two non-empty, disjoint sets $S\cup T=V$. The *cutset* of a cut consists of the edges $\{\,uv\in E\colon u\in S, v\in T\,\}$ between the two parts. The *size* (or *weight*) of a cut in an unweighted graph is the cardinality of the cutset, i.e., the number of edges between the two parts,

$$w(S,T)=|\{\,uv\in E\colon u\in S, v\in T\,\}|\,.$$

There are $2^{|V|}$ ways of choosing for each vertex whether it belongs to S or to T, but two of these choices make S or T empty and do not give rise to cuts. Among the remaining choices, swapping the roles of S and T does not change the cut, so each cut is counted twice; therefore, there are $2^{|V|-1}-1$ distinct cuts. The *minimum cut problem* is to find a cut of smallest size among these cuts.

For weighted graphs with positive edge weights $w\colon E\rightarrow\mathbf{R}^+$ the weight of the cut is the sum of the weights of edges between vertices in each part

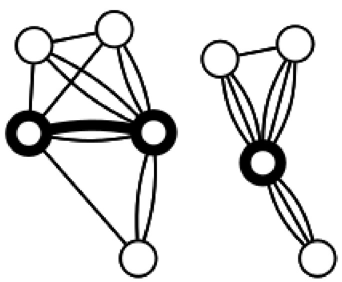
$$w(S,T)=\sum_{uv\in E\colon u\in S,v\in T} w(uv)\,,$$

which agrees with the unweighted definition for $w=1$.

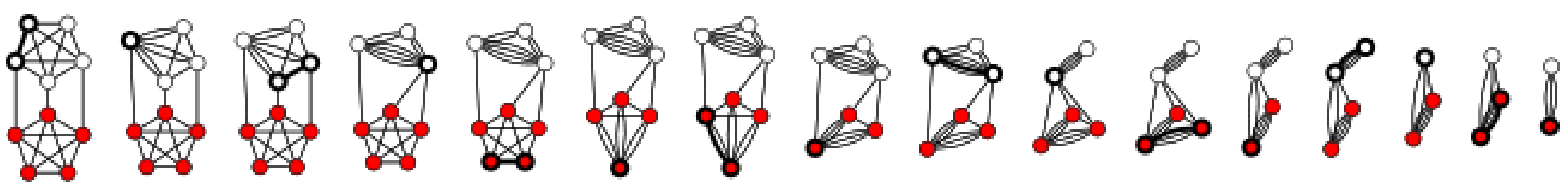
A cut is sometimes called a “global cut” to distinguish it from an “s-t cut” for a given pair of vertices, which has the additional requirement that $s\in S$ and $t\in T$. Every global cut is an s-t cut for some $s,t\in V$. Thus, the minimum cut problem can be solved in polynomial time by iterating over all choices of $s,t\in V$ and solving the resulting minimum s-t cut problem using the max-flow min-cut theorem and a polynomial time algorithm for maximum flow, such as the push-relabel algorithm, though this approach is not optimal. Better deterministic algorithms for the global minimum cut problem include the Stoer–Wagner algorithm, which has a running time of $O(mn + n^2 \log n)$.[2]

Contraction algorithm

The fundamental operation of Karger's algorithm is a form of edge contraction. The result of contracting the edge $e = \{u, v\}$ is new node uv. Every edge $\{w, u\}$ or $\{w, v\}$ for $w \notin \{u, v\}$ to the endpoints of the contracted edge is replaced by an edge $\{w, uv\}$ to the new node. Finally, the contracted nodes u and v with all their incident edges are removed. In particular, the resulting graph contains no self-loops. The result of contracting edge e is denoted G/e.



The contraction algorithm repeatedly contracts random edges in the graph, until only two nodes remain, at which point there is only a single cut.

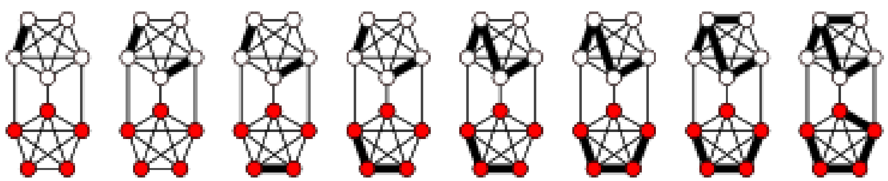


Successful run of Karger's algorithm on a 10-vertex graph. The minimum cut has size 3.

procedure contract($G = (V, E)$):
while $|V| > 2$

choose $e \in E$ uniformly at random
$G \leftarrow G/e$
return the only cut in G

When the graph is represented using adjacency lists or an adjacency matrix, a single edge contraction operation can be implemented with a linear number of updates to the data structure, for a total running time of $O(|V|^2)$. Alternatively, the procedure can be viewed as an execution of Kruskal's algorithm for constructing the minimum spanning tree in a graph where the edges have weights $w(e_i) = \pi(i)$ according to a random permutation π. Removing the heaviest edge of this tree results in two components that describe a cut. In this way, the contraction procedure can be implemented like Kruskal's algorithm in time $O(|E| \log |E|)$.



The random edge choices in Karger's algorithm correspond to an execution of Kruskal's algorithm on a graph with random edge ranks until only two components remain.

The best known implementations use $O(|E|)$ time and space, or $O(|E| \log |E|)$ time and $O(|V|)$ space, respectively.[1]

Success probability of the contraction algorithm

In a graph $G = (V, E)$ with $n = |V|$ vertices, the contraction algorithm returns a minimum cut with polynomially small probability $\binom{n}{2}^{-1}$. Every graph has $2^{n-1} - 1$ cuts,[3] among which at most $\binom{n}{2}$ can be minimum cuts. Therefore, the success probability for this algorithm is much better than the probability for picking a cut at random, which is at most $\binom{n}{2}/(2^{n-1}-1)$

For instance, the cycle graph on n vertices has exactly $\binom{n}{2}$ minimum cuts, given by every choice of 2 edges. The contraction procedure finds each of these with equal probability.

To establish the bound on the success probability in general, let C denote the edges of a specific minimum cut of size k. The contraction algorithm returns C if none of the random edges belongs to the cutset of C. In particular, the first edge contraction avoids C, which happens with probability $1-k/|E|$. The minimum degree of G is at least k (otherwise a minimum degree vertex would induce a smaller cut), so $|E| \geq nk/2$. Thus, the probability that the contraction algorithm picks an edge from C is

$$\frac{k}{|E|} \leq \frac{k}{nk/2} = \frac{2}{n}.$$

The probability p_n that the contraction algorithm on an n-vertex graph avoids C satisfies the recurrence $p_n \geq \left(1-\frac{2}{n}\right) p_{n-1}$, with $p_2 = 1$, which can be expanded as

$$p_n \geq \prod_{i=0}^{n-3} \left(1-\frac{2}{n-i}\right) = \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} = \binom{n}{2}^{-1}.$$

Repeating the contraction algorithm

By repeating the contraction algorithm $T = \binom{n}{2} \ln n$ times with independent random choices and returning the smallest cut, the probability of not finding a minimum cut is

$$\left[1 - \binom{n}{2}^{-1}\right]^T \leq \frac{1}{e^{\ln n}} = \frac{1}{n}\,.$$

The total running time for T repetitions for a graph with n vertices and m edges is $O(Tm) = O(n^2 m \log n)$.

10 repetitions of the contraction procedure. The 5th repetition finds the minimum cut of size 3.

Karger–Stein algorithm

An extension of Karger's algorithm due to David Karger and Clifford Stein achieves an order of magnitude improvement.[4]

The basic idea is to perform the contraction procedure until the graph reaches t vertices.

procedure contract($G = (V, E)$, t):
while $|V| > t$
  choose $e \in E$ uniformly at random
  $G \leftarrow G/e$
return G

The probability $p_{n,t}$ that this contraction procedure avoids a specific cut C in an n-vertex graph is

$$p_{n,t} \geq \prod_{i=0}^{n-t-1}\left(1 - \frac{2}{n-i}\right) = \binom{t}{2} \Big/ \binom{n}{2}\,.$$

This expression is approximately t^2/n^2 and becomes less than $\frac{1}{2}$ around $t = n/\sqrt{2}$. In particular, the probability that an edge from C is contracted grows towards the end. This motivates the idea of switching to a slower algorithm after a certain number of contraction steps.

```
procedure fastmincut(G = (V, E)):
if |V| ≤ 6:
   return mincut(V)
else:
   t ← ⌈1 + |V|/√2⌉
   G_1 ← contract(G, t)
   G_2 ← contract(G, t)
   return min {fastmincut(G_1), fastmincut(G_2)}
```

Analysis

The probability $P(n)$ the algorithm finds a specific cutset C is given by the recurrence relation

$$P(n) = 1 - \left(1 - \frac{1}{2} P\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil\right)\right)^2$$

with solution $P(n) = \Omega\left(\frac{1}{\log n}\right)$. The running time of fastmincut satisfies

$$T(n) = 2T\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil\right) + O(n^2)$$

with solution $T(n) = O(n^2 \log n)$. To achieve error probability $O(1/n)$, the algorithm can be repeated $O(\log n / P(n))$ times, for an overall running time of $T(n) \cdot \frac{\log n}{P(n)} = O(n^2 \log^3 n)$. This is an order of magnitude improvement over Karger's original algorithm.

Improvement bound

To determine a min-cut, one has to touch every edge in the graph at least once, which is $\Theta(n^2)$ time in a dense graph. The Karger–Stein's min-cut algorithm takes the running time of $O(n^2 \ln^{O(1)} n)$, which is very close to that.

References

1. Karger, David (1993). "Global Min-cuts in RNC and Other Ramifications of a Simple Mincut Algorithm" (http://people.csail.mit.edu/karger/Papers/mincut.ps). Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms.
2. Stoer, M.; Wagner, F. (1997). "A simple min-cut algorithm". Journal of the ACM. **44** (4): 585. doi:10.1145/263867.263872 (https://doi.org/10.1145%2F263867.263872).
3. Patrignani, Maurizio; Pizzonia, Maurizio (2001), "The complexity of the matching-cut problem", in Brandstädt, Andreas; Le, Van Bang (eds.), Graph-Theoretic Concepts in Computer Science: 27th International Workshop, WG 2001, Boltenhagen, Germany, June 14-16, 2001, Proceedings, Lecture Notes in Computer Science, **2204**, Berlin: Springer, pp. 284–295, doi:10.1007/3-540-45477-2_26 (https://doi.org/10.1007%2F3-540-45477-2_26), MR 1905640 (https://www.ams.org/mathscinet-getitem?mr=1905640).
4. Karger, David R.; Stein, Clifford (1996). "A new approach to the minimum cut problem" (http://www.columbia.edu/~cs2035/courses/ieor6614.S09/Contraction.pdf) (PDF). Journal of the ACM. **43** (4): 601. doi:10.1145/234533.234534 (https://doi.org/10.1145%2F234533.234534).

Retrieved from "https://en.wikipedia.org/w/index.php?title=Karger%27s_algorithm&oldid=956445953"

This page was last edited on 13 May 2020, at 11:41 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.