095946- ADVANCED ALGORITHMS AND PARALLEL PROGRAMMING

Fabrizio Ferrandi

a.a. 2021-2022

KARGER'S MIN-CUT ALGORITHM

Karger's Min-Cut Algorithm

KARGER'S MIN-CUT ALGORITHM

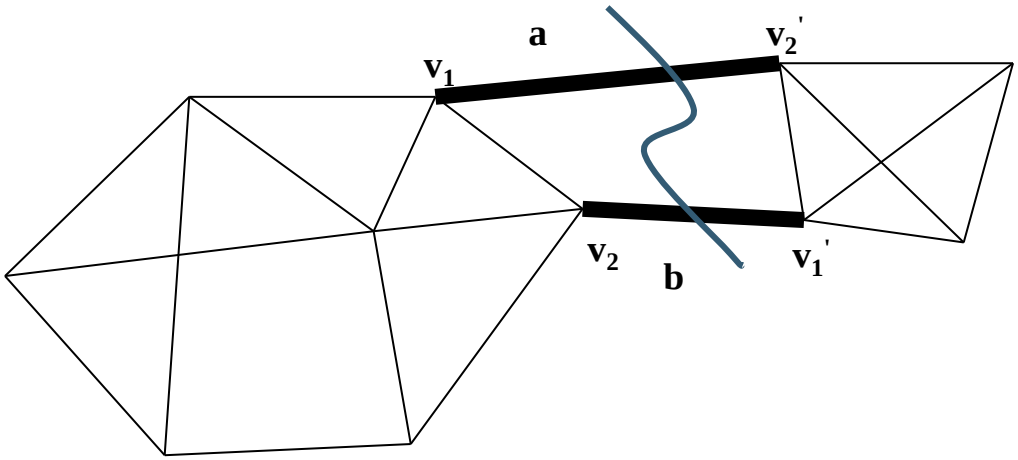
problem of the minimum cut of a graph

vertices and edges

Let G = (V, E) be a connected undirected graph. Let n = |V |, m = |E|.

For S ⊂ V , the set δ(S) = {(u, v) ∈ E : u ∈ S, v ∈ S' } is a cut since their removal from G disconnects G into more than one component.

Goal: Find the cut of minimum size. (minimum number of edges for the cut)



$S=\{v_1, v_2, \ldots\}$

$\delta(S) = \{(v_1, v_2'), (v_2, v_1')\}$

a b

MINIMUM ST-CUT PROBLEM

Closely related is the minimum st-cut problem.

In this problem, for specified vertices s and t we restrict attention to cuts $\delta(S)$ where $s \in S$, $t \notin S$.

Traditionally, the min-cut problem was solved by solving

- $n - 1$ min-st-cut problems.

In the min-st-cut problem we are given as input two vertices s and t, our aim is to find the set S where $s \in S$ and $t \notin S$ which minimizes the size of the cut (S, S'), i.e., $|\delta(S)|$.

The size of the min-st-cut is equal to

- the value of the max-st-flow (equivalent by linear programming duality).

The fastest algorithm for solving max-st-flow runs

- in $O(nm \log(n^2/m))$ time [1]
  - all $n - 1$ max-st-flow computations can be done simultaneously with the same time bounds [2].

KARGER'S APPROACH

randomized algorithm for solving the problem.

A clever algorithm to solve the min-cut problem (without the st-condition) without using any max-flow computations is due to Karger [3] based on a randomized approach.

A refinement of Karger's algorithm, due to Karger and Stein [4] is also faster than the approach based on max-st-flow computations. We will present Karger's algorithm, followed by the refinement.

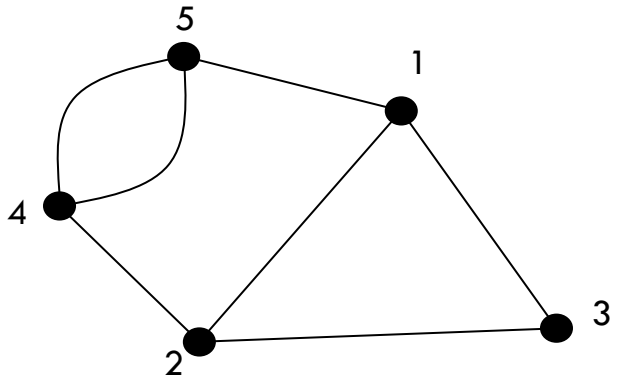
MULTIEDGES

The algorithm will start initially with a simple graph as input, and then it will need to consider multigraphs

In a multigraph there are possibly multiple edges between a pair of vertices

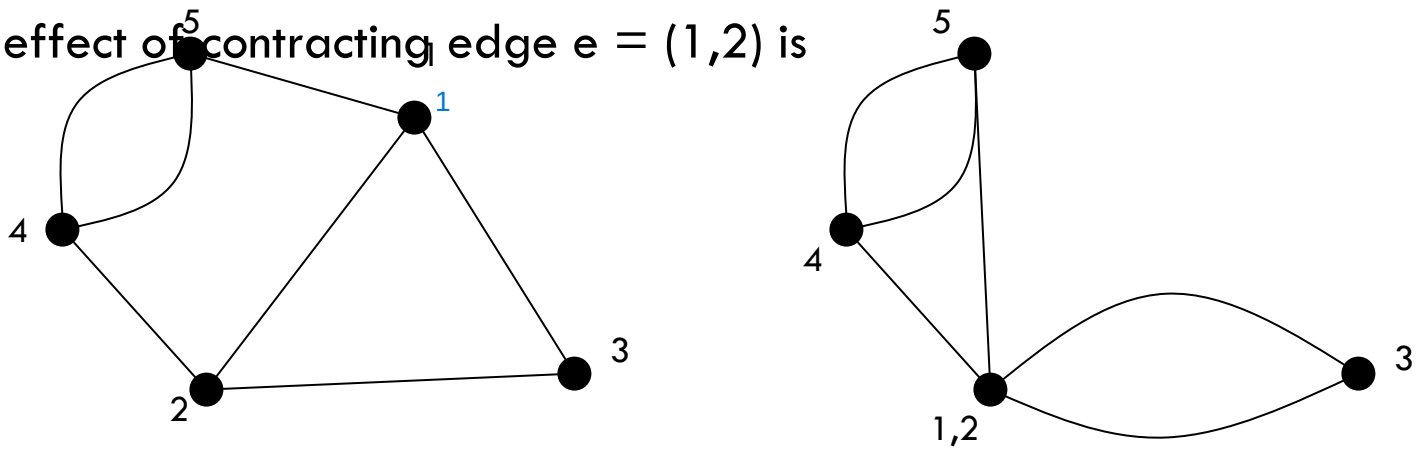
We do not have edges of the form (v, v) in our multigraphs

- we refer to these types of edges as "self-loops"



CONTRACT EDGES

the effect of contracting edge e = (1,2) is



stiamo contraendo i vertici 1 e 2 ottenendo la figura di dx

CONTRACT EDGES DEFINITION

Let $G = (V, E)$ be a multigraph without self loops. For $e = \{u, v\} \in E$, the contraction with respect to e, denoted G/e, is formed by:

- Replace vertices u and v with a new vertex, w
- Replace all edges (u, x) or (v, x) with an edge (w, x)
- Remove self-loops to w. G/e is a multigraph

The key observation is that if we contract an edge (u, v) then we preserve those cuts where u and v are both in S or both in S'. Observation 1:

- Let $e = (u, v) \in E$. There is a one-to-one correspondence between cuts in G which do not contain any edge (u, v), and cuts in G/e. In fact, for $S \subset V$ such that $u, v \in S$, $\delta_G(S) = \delta_{G/e}(S)$ (with w substituted for u and v).

SIMPLE ALGORITHM FOR FINDING A MIN-CUT OF A GRAPH

pick an edge uniformly at random and merge the two vertices at its end-points

the algorithm continues the contraction process until only two vertices remain (n − 2 edges contracted)

These two vertices correspond to a partition (S, S') of the original graph, and the edges remaining in the two vertex graph correspond to $\delta(S)$ in the original input graph.

DOES THIS ALGORITHM ALWAYS FIND A MIN-CUT?

- Let k be the min-cut size. We fix our attention on a particular min-cut with k edges: $|\delta(S)| = k$.
- Lemma 1: Let $\delta(S)$ be a cut of minimum size of the graph $G = (V, E)$
  - Pr (Karger's algorithm ends with the cut $\delta(S)$) $\geq \frac{1}{\binom{n}{2}}$

queste proof a lezione sono state sorvolate un bel po'. Ha parlato direttamente della proof 3

PROOF (1)

Denote the edges we contract in the algorithm as $\{e_1, e_2, \ldots, e_{n-2}\}$

The algorithms succeeds if none of the contracted edges are in $\delta(S)$. To upper bound the probability that the algorithm succeeds in the first contraction, i.e., that $e_1 \notin \delta(S)$ we need to lower bound the number of edges in the input graph G in terms of k.

- Note, the minimum degree is at least k in G, otherwise we have a cut of size smaller than k since we can disconnect the vertex from the rest of the graph by removing all edges incident to it.

This implies that the original input graph G has at least $nk/2$ edges.

sto considerando il fatto che per ogni nodo ho al minimo k edges. Adesso dato che gli edges collegano due nodi avrò k*(n/2) edges, con n=numero di nodi del grafo

PROOF (2)

By Observation 1, since every cut in an intermediate multigraph corresponds to a cut of the original graph, we have

Observation 2

si intende forse il minimo numero di edges

- The minimum degree in all of the intermediate multigraphs is at least k. Otherwise, the edges incident the (meta)vertex with degree smaller than k would correspond to a cut of size $< k$ in the original graph

After j contractions, the multigraph, denote as G_j , contains $n - j$ vertices since we lose one vertex per contraction. Then, Observation 2 implies that G_j has at least $(n - j)k/2$ edges.

PROOF (3)

- We can now compute the probability the algorithm successfully finds our specific minimum cut δ(S).
- To do so, all of the contracted edges must not be in δ(S):

con e indichiamo tutti gli edges che sono stati rimossi dopo una contrazione

- $\Pr(\text{final graph} = \delta(S)) = \Pr(e_1, e_2, \dots, e_{n-2} \notin \delta(S))$
- $= \Pr(e_1 \notin \delta(S)) \sum_{j=1}^{n-3} Pr(ej_{+1} \notin \delta(S) | e_1, \dots, ej \in \delta(S))$
- $\geq \sum_{j=1}^{n-3} 1 - \frac{k}{(n-j)k/2} = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$

probability of their intersection

probabilità che hai ad ogni iterazione dell'algoritmo -> alla jesima iterazione avrai che hai ((n-j)/2)*k edges, perchè hai k edges per ogni nodo rimanente (cioè n-j), ma diviso due perchè un edge collega due nodi.

NB: (n-j) perchè hai fatto j volte la contrazione

REPEATING THE ALGORITHM

- In order to boost the probability of success, we simply run the algorithm $l\binom{n}{2}$
- The probability that at least one run succeeds is at least
- $\left(1 - \frac{1}{\binom{n}{2}}\right)^{l\binom{n}{2}} \geq 1 - e^{-l}$
- Setting l = c ln n we have an error probability ≤ $1/n^c$
- It's easy to implement Karger's algorithm so that one run takes $O(n^2)$ time.
  - we have an $O(n^4 \log n)$ time randomized algorithm with error probability $1/\text{poly}(n)$.

FASTER VERSION BY KARGER AND STEIN [4]

- The key idea comes from looking at the telescoping product.
  - In the initial contractions it's very unlikely we contracted an edge in the minimum cut. Towards the end of the algorithm, our probability of contracting an edge in the minimum cut grows.
- For a fixed minimum cut δ(S), the probability that this cut survives down to vertices is at least $\binom{l}{2}/\binom{n}{2}$
- for $l = \frac{n}{\sqrt{2}}$ we have probability ≥ 1/2 of succeeding
  - in expectation two trails should suffice

IMPROVED ALGORITHM

- From a multigraph G, if G has at least 6 vertices, repeat twice:
  - run the original algorithm down to $\frac{n}{\sqrt{2}} + 1$ vertices
  - recurse on the resulting graph
- Return the minimum of the cuts found in the two recursive calls leggi bene il codice dopo e lo capisci

The choice of 6 as opposed to some other constant will only affect the running time by a constant factor

PSEUDO CODE

```
procedure contract(G=(V,E), t):
   while |V| > t
       choose e\in E uniformly at random
       G = G/e
   return G
procedure fastmincut(G= (V,E)):
   if |V| < 6:
       return mincut(V)
   else:
       t= ceil( 1 + |V|/√2)
       G_1 =  contract(G, t)
       G_2 =  contract(G, t)
       return min {fastmincut(G_1), fastmincut(G_2)}
```

indico la contrazione in questo modo

leggilo che si capisce abbastanza bene

COMPUTE THE RUNNING TIME

straightforward to solve, e.g., the Master theorem applies

- Recurrence equation
  - $T(n) = 2n^2 + T(\frac{n}{\sqrt{2}}) = O(n^2 \log n)$
- Since we succeed down to $\frac{n}{\sqrt{2}}$ with probability ≥ 1/2,
  - we have the following recurrence for the probability of success, denote by P(n):
  - $P(n) \geq 1 - (1 - \frac{1}{2}P(\frac{n}{\sqrt{2}} + 1))^2$
  - $P(n) = \Omega(\frac{1}{\log n})$
- Hence, similar to the earlier argument for the original algorithm, with $O(\log^2 n)$ runs of the algorithm, the probability of success is ≥ 1 − 1/poly(n).

COROLLARY OF KARGER'S ORIGINAL ALGORITHM

Any graph has at most $O(n^2)$ minimum cuts

This follows from Lemma 1 since that holds for any specified minimum cut

Note, we can also enumerate all of these cuts by the above algorithm.

REFERENCES

[1] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. J. Assoc. Comput. Mach., 35(4):921–940, 1988.

[2] J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a graph. In Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992), pages 165–174, New York, 1992. ACM.

[3] D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993), pages 21–30, New York, 1993. ACM.

[4] D. R. Karger and C. Stein. A new approach to the minimum cut problem. J. ACM, 43(4):601–640, 1996.

ACKNOWLEDGE

Based on scribe notes first prepared by Tom Hayes at the University of Chicago in the winter quarter, 2003 and then elaborated by Eric Vigoda - Georgia Institute of Technology Last updated for 7530 - Randomized Algorithms, Spring 2010.

and on

Book: Randomized Algorithms, by R. Motwani and P. Raghavan Cambridge University Press, 1995. 492 pages. ISBN: 0-521-47465-5.