Lecture 7 — 3rd June

Lecturer: Christian Sommer *Scribe: Tatiana V. Martsinkevich*

Recall that a graph $G = (V, E)$ is a discrete object that consists of a set of vertices V and a set of edges E. Each edge connects two vertices, $E \subseteq \binom{V}{2}$. Let $n = |V|$ denote the number of vertices. Then, $|E| \leq \binom{n}{2} = \mathcal{O}(n^2)$.

In this lecture we also consider *multigraphs*, wherein there can be multiple edges between any two vertices. The number of edges of a multigraph is not necessarily bounded by $\mathcal{O}(n^2)$.

7.1 The MinCut Problem and a Solution

The MinCut problem is to partition the vertex set of a graph $G = (V, E)$ into two sets $S \subseteq V$ and $\bar{S} := V \setminus S$ such that the number of edges between S and $\bar{S}$ is minimized, more formally,

$$\min_S |E(S, \bar{S})|, \text{ where } E(S, T) := \{(u, v) \in E : u \in S \wedge v \in T\}.$$

There is a simple randomized algorithm that *contracts* an edge at each step until only two nodes are left (see Algorithm 1 or the lecture notes of the first lecture, see also Figure 7.1 for an example; the algorithm is by Karger and Stein [KS96, Section 2]). Recall that contracting an edge (u, v) amounts to "identifying" its endpoints u and v in a new vertex uv that is adjacent to all the neighbors of both u and v.

Algorithm 1 MinCutRand($G = (V, E)$)

```
while |V| ≥ 2 do
  pick an edge e uniformly at random
  contract e: combine nodes, keep multi-edges, remove loops
end while
return cut
```

The main loop of Algorithm 1 is executed $\mathcal{O}(n)$ times. Choosing an edge and contracting it requires time $\mathcal{O}(n)$. Algorithm 1 thus runs in time $\mathcal{O}(n^2)$ [KS96, Corollary 3.2.1].

7.2 Success Probability of MinCutRand

Let $\mathsf{OPT}(G) := \min_S |E(S, \bar{S})|$. If $\mathsf{OPT}(G) = k$, then the degree of each node $v \in V$ is at least $\deg(v) \geq k$ and thus the total number of edges is at least $|E| \geq \frac{kn}{2}$. In the following,

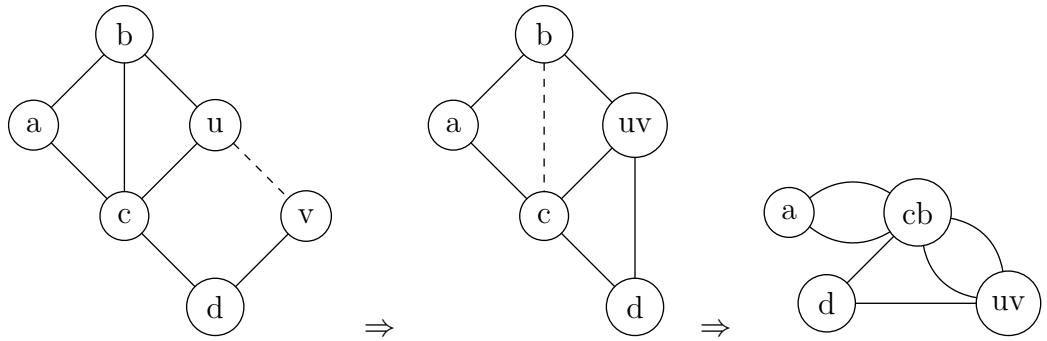

Figure 7.1. An example of contracting edges of a 6–nodes graph. The edge that is to be contracted in the next step is indicated by a dashed line.

we derive an estimate of the probability that the algorithm contracts an edge that belongs to a particular minimum cut. For the analysis, we first fix an arbitrary minimum cut. We call the edges outside this cut "good" and we call the edges that belong to this cut "bad" (we hope to avoid "bad" edges for contraction).

Notation: For a graph $G = (V, E)$ and an edge $e \in E$, we write G/e to denote the graph obtained from G by contracting e.

Lemma 7.1. $\mathsf{OPT}(G) \leq \mathsf{OPT}(G/e)$. □

After t iterations of the algorithm, $n - t + 1$ nodes are left. Since each node has degree at least k, at least $\frac{k(n-t+1)}{2}$ edges are left. The probability that a "bad" edge is chosen next for contraction is

$$\mathsf{Pr}[\text{"bad" edge}] = \frac{\not{k}2}{\not{k}(n-t+1)}.$$

The probability that a "good" edge is chosen is

$$\mathsf{Pr}[\text{"good" edge}] = 1 - \frac{2}{(n-t+1)}.$$

The probability to obtain the minimum cut we fixed at the beginning is thus

$$\prod_{t=1}^{n-2}\left(1-\frac{2}{n-t+1}\right) = \prod_{t=1}^{n-2}\frac{n-t+1-2}{n-t+1} = \frac{\not{n}\not{-}\not{2}}{n}\cdot\frac{\not{n}\not{-}\not{3}}{n-1}\cdot\frac{\not{n}\not{-}\not{4}}{\not{n}\not{-}\not{2}}\cdot\ldots\cdot\frac{2}{\not{4}}\cdot\frac{1}{\not{3}} = \frac{1}{\binom{n}{2}} = \Omega\left(\frac{1}{n^2}\right).$$

In order to obtain a minimum cut with constant probability, we need to repeat Algorithm 1 roughly $\binom{n}{2}$ times. The resulting time complexity is thus $\mathcal{O}(n^4 \log n)$ (log factor to increase the success probability to $1 - 1/n$). In the following, we discuss how to improve upon this complexity.

7.3 Improvements upon MinCutRand

Note that during the first couple of executions of the main loop of Algorithm 1, the probability of choosing a "bad" edge is quite low; the probability of choosing a "bad" edge is rather high during the final stages of the execution.

7.3.1 Switch to a Deterministic Strategy

Idea: choose edges at random and contract them for the first $n-r$ rounds only; thereafter, instead of executing another r rounds, run a deterministic algorithm (returning the optimal result for this contracted graph on r nodes)

Let r denote the number of remaining nodes. The probability that none of the edges of a particular minimum cut (the "bad" edges) are contracted until r nodes are left is

$$\frac{\binom{r}{2}}{\binom{n}{2}} = \Omega\left(\frac{r^2}{n^2}\right).$$

Repeating this new algorithm n^2/r^2 times yields constant success probability. Executing the first $n-r$ rounds requires time $\mathcal{O}(n^2)$. For graphs on r nodes, there is a deterministic algorithm that runs in time $\tilde{\mathcal{O}}(r^3)$ [NI92]. The running time is thus roughly (up to constant and logarithmic factors)

$$\frac{n^2}{r^2}\left(n^2 + r^3\right).$$

By setting $r := n^{2/3}$, we obtain a running time of $\tilde{\mathcal{O}}(n^{8/3})$ — beating the cubic-time deterministic algorithm. The idea described in the next section helps to improve the running time even further.

7.3.2 "Re-use" Computations

Idea: instead of repeating the whole algorithm (Algorithm 1) for n^2 times, "re-use" the results achieved by earlier rounds (these results are correct with rather high probability) and repeat later rounds only (see also [KS96, Section 4])

Similar to the idea of Section 7.3.1, we contract the graph until r nodes are left. The probability of not contracting a single "bad" edge is r^2/n^2. Instead of running this contraction step only once, the idea is to run these $n-r$ contraction steps on approximately n^2/r^2 independent copies of the graph and then to recurse separately on each of these (see Algorithm 2). We set $r := |V|/2$ and we run the contraction steps (the WHILE part) on four independent copies.

Lemma 7.2 ([KS96, Lemma 4.1]). *For a graph with n nodes, the running time of Algorithm 2 is bounded by $\mathcal{O}(n^2 \log n)$.*

Algorithm 2 MinCutRand'$(G = (V, E))$

$r := \frac{|V|}{2}$
if $|V| \leq 8$ **then**
  return optimal minimum cut of G
end if
for four independent copies of G, say H_1, H_2, H_3, and H_4, **do**
  while $|V_i| \geq r$ **do**
    pick an edge e uniformly at random
    contract e: combine nodes, keep multi-edges, remove loops
  end while
end for
recurse with these four contracted graphs H_1, H_2, H_3, and H_4
return $\min_i$ MinCutRand'(H_i)

Proof: We give a recursive formula for the running time $T(n)$:

$$T(n) = \mathcal{O}(n^2) + \underbrace{4 \cdot T\left(\frac{n}{2}\right)}_{\text{recursive part}} = \mathcal{O}(n^2 \log n).$$

□

Lemma 7.3 ([KS96, Lemma 4.3]). *For a graph with n nodes, the success probability $P(n)$ of Algorithm 2 is $P(n) = \Omega\left(\frac{1}{\log n}\right)$.*

Proof: We give a recursive formula for the success probability:

$$P(n) \geq 1 - \underbrace{\left(1 - \frac{1}{4}P(n/2)\right)^4}_{\text{failure probability}} \geq P(n/2)\left(1 - \frac{3}{8}P(n/2)\right).$$

□

In order to reach a success probability of $1 - 1/n$, the algorithm can be repeated $O(\log^2 n)$ times. The overall running time of this new recursive algorithm is $\tilde{\mathcal{O}}(n^2)$ (as opposed to $\tilde{\mathcal{O}}(n^4)$ of Algorithm 1).

Bibliography

[KS96] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996.

[NI92] Hiroshi Nagamochi and Toshihide Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Discret. Math.*, 5(1):54–66, 1992.