Randomized Algorithms 2013A
Lecture 8 – Edge Sparsification (cont'd) and Distance Oracles*

Robert Krauthgamer

We continue our plan from the previous class to prove the following theorem.

Theorem 7 [Benczur-Karger, 1996]:

For every weighted graph $G = (V, E)$ on n vertices and error parameter $\varepsilon > 0$, there is a weighted subgraph $G' = (V, E')$ with $O(\varepsilon^{-2} n \log n)$ edges such that $G' \in (1 \pm \varepsilon)G$. Moreover, G' can be constructed in $O(|E| \log^2 n)$ time.

We will actually prove a slightly weaker version, for unweighted graphs, with another $\log^2 n$ factor, and without the near-linear time algorithm.

Main idea: Sample edges non-uniformly, each edge e with probability p_e that is inversely proportional to its "connectivity" c_e. So "dense" regions will be sampled with smaller probability, thereby reducing the number of edges there more aggresively.

Definitions of Connectivity:

A graph is *k-connected* if every cut in it has capacity $\geq k$.

A *k-strong component* is a maximal vertex-induced subgraph that is k-connected.

Example: Consider 3 cliques, connected by one cycle (triangle).

Exer: Prove there is a unique partitioning of the vertices into k-strong components. (Hint: If V_1 and V_2 are k-connected and have non-empty intersection, then also $V_1 \cup V_2$ is k-connected.)

It follows that the k-strong components partition the vertices of the graph, obtained by repeatedly removing every cut of capacity $< k$. Moreover, a $(k+1)$-strong components is a refinement of that partition.

The *strong connectivity of an edge* $e \in E$, denoted c_e, is the maximum value k such that e is contained in a k-strong component. An edge is called *k-strong* if its strong connectivity is at least k; otherwise *k-weak*.

Note that strong connectivity differs from the usual definition of connectivity. (Example: n parallel paths between s, t.)

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Construction of sparsifier G':

Set $q = q_\varepsilon := 4(d+2)\varepsilon^{-2} \ln n$, and sample every $e \in E$ with probability $p_e = \min\{q/c_e, 1\}$, in which case it is given weight $1/p_e$.

Lemma 8: With probability $\geq 1 - O(1/n^d)$, the resulting graph G' has $O(qn)$ edges.

The proof for the expected number of edges was seen in class, using the following claim: A graph with total edge weight $\geq k(n-1)$ has a k-strong component (which may be the graph itself).

Exer: Complete the high-probability proof using Chernoff bound.

Lemma 9: With high probability $G' \in (1 \pm \varepsilon \log |E|)G$.

Lemmas 8,9 together indeed prove (a weaker version of) Theorem 7, by simply using a smaller value $\varepsilon_1 = \varepsilon / \log |E|$.

Idea of Proof: The proof ws seen in class using uniform sampling (Theorem 6). The idea is to describe the same algorithm in a different way, as if we divide the sampling process into phases, and phase $i = 0, 1, \ldots$ flips the coins only for edges e with $2^i \leq c_e < 2^{i+1}$. The analysis then applies Theorem 6 separately on each 2^i-strong component, which means that we basically "remove" edges with $c_e < 2^i$. Inside each such component, the edges from level i are chosen at random, but all edges with $c_e \geq 2^{i+1}$ are kept (deterministically).

Exer: It is sometimes easier/faster to compute an approximation to c_e. So suppose we use in p_e an approximation to c_e, say within factor 3, i.e., values $c'_e \in [c_e, 3c_e]$. Explain how the theorem and analysis shown in class would extend.

Distance Oracles

Goal: Preprocess a graph $G = (V, E)$ with edge lengths $l : E \to \mathbb{R}_+$ into a (small) data structure that can answer in time $O(1)$ queries about the distance $d = d_G$ (between any two vertices $u, v \in V$).

We denote $n = |V|$ and $m = |E|$.

Naive solution: Store all $\binom{n}{2}$ distances in a matrix/array, with direct access in time $O(1)$.

Can one "compress" the information, perhaps at the expense of accuracy, i.e., the distances are only approximated?

Theorem 10 [Thorup-Zwick, 2001]: Let $k > 1$ be an integer. There is an algorithm that preprocesses the graph G in expected time $O(kmn^{1/k})$, and produces a data structure that can answer a distance query in time $O(k)$ and with approximation factor $2k - 1$.

Remark: We will ignore the preprocessing time, and focus on storage (space). In particular, we assume the shortest-path between every two vertices is computed, and essentially use only the fact that distances satisfy the triangle inequality.

Algorithm Prep(G,k):

1. $A_0 = V$; $A_k = \emptyset$.

2. for $i = 1, \dots, k-1$

3. Construct A_i by including each $u \in A_{i-1}$ with probability $1/n^{1/k}$.

4. for every $v \in V$

5. for $i = 0, \dots, k-1$

6. store $d(v, A_i) = \min\{d(v,w) : \ w \in A_i\}$ and the minimizer w as $p_i(v)$

7. set $d(v, A_k) = \infty$.

8. store $B(v) = \cup_{i=0}^{k-1} \{w \in A_i \setminus A_{i-1} : \ d(v,w) < d(v, A_{i+1})\}$ in a hash table that answers whether $w \overset{?}{\in} B(v)$ and if so, what is its distance to v, in $O(1)$ worst-case time.

Remark: We can use a two-level hash table of size $O(|B(v)|)$.

Intuition of preprocessing:

The sets A_i are subsamples of V at different "levels", and provide "landmarks".

Each $p_i(v)$ is just the level i landmark closest to v.

What is a set $B(v)$? sort V by distance from v, and partition it into k levels (rings) at positions $n^{1/k}, n^{2/k}, \dots$; store $n^{1/k}$ random vertices from each ring.

Analysis of preprocessing storage: The only concern is the $\sum_v |B(v)|$, and this was sketched in class.

Exer: Prove that for every $v \in V$ and $i \in \{0, \dots, k-1\}$,

$$\mathbb{E}[|B(v) \cap A_i|] \leq n^{1/k}.$$

Algorithm Query(u,v):

1. $w = u$; $i = 0$

2. while $w \notin B(v)$

3. $i = i + 1$

4. $(u, v) = (v, u)$ //swap

5. $w = p_i(u)$

6. return $d(u, w) + d(w, v)$

The runtime is obviously $O(k)$.

Analysis of query algorithm: The entire $A_{k-1} \subseteq B(v)$, hence some answer is always returned, and the number of $u - v$ swaps (the final i) is at most $k - 1$.

Let $\Delta = d(u, v)$. We claim that each swap of u, v increases $d(w, u)$ by at most Δ; denoting by u_i, w_i etc. the values at the end of iteration i, we claim that $d(w_i, u_i) \leq d(w_{i-1}, u_{i-1}) + \Delta$. This will imply the approximation factor (strech bound), since we start with $d(w_0, u_0) = 0$, at the final i we have $d(w_i, u_i) \leq i \cdot \Delta \leq (k-1)\Delta$, and thus also $d(w_i, v_i) \leq d(w_i, u_i) + d(u_i, v_i) \leq k\Delta$.

Suppose iteration i passes the while loop's condition. Then $w_{i-1} \notin B(v_{i-1}) = B(u_i)$. By the construction of $B(u_i)$, there must be some vertex in A_i that is even closer to u_i than w_{i-1}, i.e., $d(u_i, A_i) \leq d(u_i, w_{i-1})$ hence

$$d(u_i, w_i) = d(u_i, A_i) \leq d(u_i, w_{i-1}) \leq \Delta + d(u_{i-1}, w_{i-1}).$$