Weekplan: Approximate Distance Oracles

Philip Bille   Inge Li Grtz   Eva Rotenberg*

References and Reading

[1] Approximate distance oracles, M. Thorup, U. Zwick, Journal of the ACM, 2005.

[2] Undirected single-source shortest paths with positive integer weights in linear time, M. Thorup, Journal of the ACM, 1999.

[3] Approximate Distance Oracles with Improved Preprocessing Time, C. Wulff-Nilsen, SODA, 2012

[4] Extremal problems in graph theory, P. Erdős, Proc. Symposium on Graph theory, Smolenice, 1963

[5] Faster Algorithms for All-Pairs Approximate Shortest Paths in Undirected Graphs, S. Baswana and T. Kavitha, SIAM J. Computing, 2010

This weekplan/these lecture notes contain exercises that support an understanding of [1] excluding sections 3.5, 3.6, 4.4, and 5.

Sections 1, 2, and 4 go through the selected material from [1], and Section 3 contains conventional exercises.

1 Approximate Distance Oracle

Given a weighted undirected graph, G, we want a data structure that answers $d(u, v) = \texttt{distance}(u, v)$ for vertices u, v. Here, $d(u, v)$ is the length of the shortest path connecting u and v. (Throughout the text, weighted graphs have strictly positive edge weights.)

Assume that all edge-weights are strictly positive.

1 Exercise. Consider trivial solutions.

- With no preprocessing of the graph. What is the query-time?
- Can this be done in $O(n^2)$ space and $O(1)$ query time? What is your preprocessing time?

*We would like to thank Christian Wulff-Nilsen for his suggestions to exercises.

2 Exercise. Convince yourself that the shortest-path-distance indeed is a metric on the set of vertices. Remember that edge-weights are strictly positive.

- $d(u, v) = 0 \Leftrightarrow u = v$
- $d(u, v) = d(v, u)$
- $d(u, w) \leq d(u, v) + d(v, w)$

Lower bound Follows from Erdős' Girth Conjecture: $\Omega(n^2)$ space is *necessary* in order to output exact distances in $O(1)$ time.

Approximation A data structure that outputs an estimated distance between $d(u, v)$ and $S \cdot d(u, v)$ has a *stretch* of S.

Devastating lower bound Follows from Erdős' Girth Conjecture: $\Omega(n^2)$ space is *necessary* for any stretch < 3 for $O(1)$ query-time.

Goal Let's just aim for any constant stretch and $o(n^2)$ space.

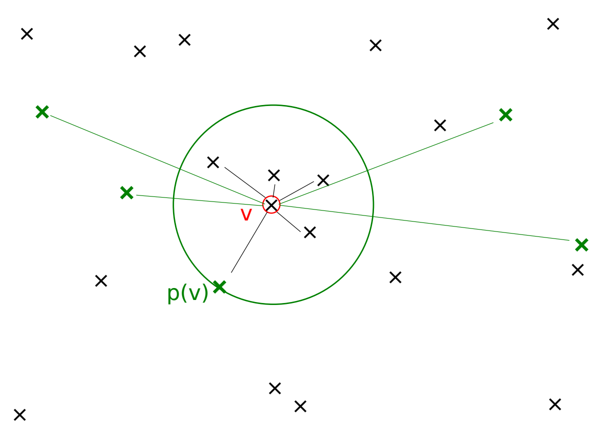

Figure 1: The vertex v knows its distance to all sampled vertices, and to the nonsampled vertices that are closer than $p(v)$.

Idea Sample each point independently with some probability p to be determined later. Store the following: (See Figure 1)

1. All distances involving at least one sampled point
2. For each vertex v, store an identifier of the closest sampled vertex $p(v)$. (If v is sampled, $p(v) = v$.)

3. For each vertex v, store distances to all vertices u that are closer than $p(v)$. That is, we store $d(v,u)$ for $u \in B_0(v)$ where $B_0(v) := \{u \in V \mid d(v,u) < d(v,p(v))\}$.

3 Exexercise Recall that for each v, the distances involving v can be stored using linear space and with constant look-up time. (Hint: Perfect hashing.)

4 Exercise Given the information above, devise any algorithm that takes a pair of vertices u, v and outputs an estimate of the distance.

- What is the query-time of your algorithm? What is its stretch?
- Can you make an algorithm with $O(1)$ query-time and a constant stretch? What is its stretch?

5 Exercise What is the space consumption?

1. What is the expected total space consumption of storing all distances involving sampled vertices?
   - How many sampled vertices are there? In expectation. As a function of p.
   - How many distances do we store for each sampled vertex?
2. What is the space consumption of storing $p(v)$ for each vertex v?
3. What is the expected total space consumption for storing, for each vertex v, distances to all vertices u that are closer than $p(v)$?

   (Hint: For each vertex v, we want to calculate the expected size of $B_0(v)$.

   Consider the vertices in decreasing order starting form v. That is, $w_0, w_1, w_2, \ldots$ with $w_0 = v$. Now, the next vertex only belongs to $B_0(v)$ if all the previous were not sampled.

   What is the probability of $w_1 \in B_0(v)$? Of $w_2 \in B_0(v)$? Of $w_i \in B_0(v)$?

   Can you give an upper bound on $\sum_{i=1}^{n-1} \Pr[w_i \in B_0(v)]$?)

- Which value for p gives the best trade-off?

Theorem 1. *[1] Given any weighted undirected graph with n vertices, there is a data structure for answering approximate distance-queries with*

- *Space:*
- *Query-time:*
- *Stretch:*

2 Generalising to a higher stretch

Idea Sample in k levels.

- $A_0 = V$ contains all vertices.
- Let A_1 samples vertices of $A_0 = V$ independently with probability p.
- A_2 samples vertices of A_1 independently with probability p.
- ...
- A_{k-1} samples each vertex of A_{k-2} independently with probability p.

p is a probability to be determined later, but it is the same probability throughout.

Generalising $p(v)$ For a vertex v, for each $i = 1, \ldots, k-1$, let $p_i(v)$ denote vertex in A_i closest to v.

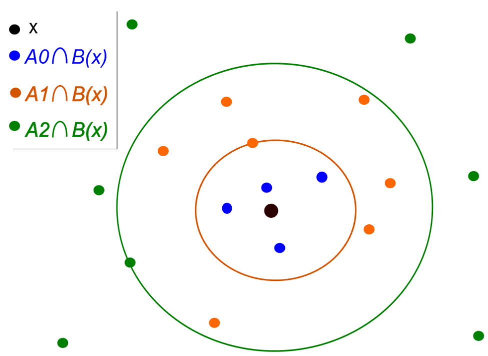

Figure 2: The bunch of x. The blue elements outside the brown circle are not included. The blue or brown elements outside the green circle are not included either. All green elements belong to the bunch.

Generalising $B_0(v)$ For a vertex v, for each $i = 0, \ldots, k-2$, let $B_i(v)$ denote the vertices of A_i that are closer to v than $p_{i+1}(v)$. (See Figure 2)

Let $B_{k-1}(v)$ denote A_{k-1}.

For a vertex v, let $B(v) = \bigcup_{i=0}^{k-1} B_i(v)$. We call this *the bunch* of v.

What to store

1. For each vertex v, store all identifiers $p_1(v), p_2(v), \ldots, p_{k-1}(v)$.
2. For each vertex v, store the distance to the vertices $u \in B(v)$.

6 Exercise Generalise your solution to Exercise 3 to see that each bunch (together with all its distances) can be stored in linear space with constant lookup-time.

7 Exercise What is the space consumption? For each vertex ...

1. How many identifiers do we store?
2. How large is $B(v)$ in expectation?
   - Can you bound the expected size of $B_i(v)$? (Hint: See exercise 5.3)
   - What is the expected size of $B_{k-1} = A_{k-1}$?
     (as a function of p)
   - Find a value for p such that the expected space consumption of $B_{k-1}(v)$ is the same as the other $B_i(v)$.
   - Can you bound the expected size of the union $\bigcup_{i=0}^{k-1} B_i(v)$?
     (Use the value of p that you just found.)

Algorithm 2 (Distance(u,v))**.**

$w = u$; $i = 0$;
while $w \notin B(v)$ ***do***
  i++;
  $(u, v) = (v, u)$;
  $w = p_i(u)$;
end while
return $d(w, u) + d(w, v)$

8 Exercise (hard but important)

- Assume $d(u, p_{i-1}(u)) \leq (i-1) \cdot d(u, v)$.
  Show that $p_{i-1}(u) \in B(v) \vee d(v, p_i(v)) \leq i \cdot d(u, v)$
  (Hint: use the triangle inequality.)
- Show that when the algorithm returns, it returns at most $(2i+1) \cdot d(u, v)$.

Note that upon return, $i < k$.

Theorem 3. *[1] Given any weighted undirected graph with n vertices, for any value k, there is a data structure for answering approximate distance-queries with*

- *Space:*
- *Query-time:*
- *Stretch:*

(Fill in the blanks as functions of k.)
What space, time and stretch do you get when you set $k = \log n$?

3 Conventional exercises

9 A modelling exercise In [1], the authors state that "the US road network is a planar graph".

Indeed, a planar, undirected weighted graph models a large road network.

Can you think of different ways of modelling a large road network as a graph? What are their advantages and disadvantages?

10 Exercise The approximate distance oracle of [1] presented in this lecture does not work If G is a weighted *directed* graph. Point to where the argument breaks down.

11 Exercise Let $S \subset V$ be a subset of vertices. Assume we only want to answer approximate distance-queries for $s_1, s_2 \in S$.

- For which vertices $v \in V$ is the bunch $B(v)$ needed in order to answer such queries?
- What is the space consumption for this "restricted" oracle?
- Can you remove the any dependency on $n = |V|$ from th the space consumption, so there is no dependency on n?

Theorem Let $G = (V, E)$ be an undirected graph with non-negative edge weights and let $\delta \geq 1$. A δ-spanner of G is a subgraph $S = (V, E')$ of G spanning all vertices such that for all $u, v \in V$, the shortest path distance between u and v in S is at most a factor of δ longer than the shortest path distance between u and v in G.

It has been shown that for any integer $k \geq 1$ and any graph G with m edges and n vertices, G contains a $(2k-1)$-spanner S with $O(kn^{1+1/k})$ edges, and S can be found in $O(km + n)$ time.

12 Exercise Let $\varepsilon > 0$ be a given constant. Combine the above Theorem with that of Thorup and Zwick to obtain an approximate distance oracle with $O(1)$ stretch, $O(1)$ query time, $O(n^{1+\varepsilon})$ space, and $O(m + n^{1+\varepsilon})$ construction time.

4 Fast construction time

13 Exercise Assume there is an $O(m)$ time algorithm for calculating the single-source-shortest path tree of a graph (Thorup [2]).

- Given any $i < k$, show how to determine $p_i(v)$ for all $v \in V$ in $O(m)$ time.

For any vertex $u \in A_i \setminus A_{i+1}$, define the *co-bunch* $C(u)$ as $\{v \in V | d(u, v) < d(v, p_{i+1}(v))\}$. That is, $v \in C(u) \iff u \in B(v)$.

14 Exercise Show that for any $i < k$, for any vertex $w \in A_i$, if v' lies on a shortest path from w to v and $v \in C(w)$, then $v' \in C(w)$.

15 Exercise[$*$] Show that for each w, all the distances $d(u,w)$ for $u \in C(w)$ can be calculated in time proportional to $\log n \cdot \sum_{v \in C(w)} \deg(v)$.
(Hint: use a priority queue.)

Lemma: For each w, all the distances $d(u,w)$ for $u \in C(w)$ can be calculated in $O(\sum_{v \in C(w)} \deg(v))$ time (using methods from [2]).

16 Exercise Use the above Lemma to show that the total time of obtaining all the distances in the bunches of all vertices is at most

$$O\left(\sum_{v \in V} |B(v)| \deg(v)\right)$$

17 Exercise Analyse the construction time of the data structures given in Sections 1 and 2. (Hint: Use the result of Exercise 16.)