A Note on the Nearest Neighbor in Growth-Restricted Metrics

Kirsten Hildrum John Kubiatowicz Sean Ma Satish Rao
University of California, Berkeley
{hildrum@cs,kubitron@cs,seanma@cal,satishr@cs}.berkeley.edu

Abstract

In this paper, we give results relevant to sequential and distributed dynamic data structures for finding nearest neighbors in growth-restricted metrics. Our sequential data structure uses linear space, and requires $O(\log n)$ queries in expecation and $O(\log n)$ queries for lookups with high probability. This improves the results of Karger and Ruhl [4], whose data structure uses $O(n \log n)$ space with comparable expected time bounds. This also improves on the time bound of a load-balanced version of algorithm (for dynamic networks) presented in [3].

Our algorithm was inspired by the object location data structure developed by Plaxton, Rajaraman and Richa [6], and is similar in structure to the algorithm of Krauthgamer and Lee [5]. It is significantly different that of Karger and Ruhl [4].

A distributed version of the algorithm presented here is in use as a part of Tapestry [3, 8], a peer-to-peer object location system based on [6].

1 Introduction

Finding the nearest neighbor is hard in general; this paper looks at a specific class of metric spaces, called by Karger and Ruhl [4] *growth restricted*. Intuitively, a growth-restricted metric space looks like a d-dimensional grid for some some dimension d. The algorithm of this paper (and the algorithm of Karger and Ruhl) accesses the metric space only through queries to a distance oracle. The goal is to find the nearest neighbor of a query point x with the fewest of queries to the distance oracle. In addition to giving an algorithm for finding the nearest neighbor, Karger and Ruhl [4] describe the following general technique. Given a starting point x and a query point q, find a point about halfway between q and x. (In some metric spaces, no such halfway point can be found.) Repeated $O(\log n)$ times, this finds the nearest neighbor. Though our algorithm also uses this technique, it is substantially different than theirs.

Krauthgamer and Lee [5] use an approach similar to the one presented here, but do so deterministically. Their solution has applications in a broader class of metric spaces. Also related is the approach of Clarkson in [1] and the sampling technique used by Thorup and Zwick [7] for approximate distance oracles. Our algorithm is based on ideas used by Plaxton, Rajaraman and Richa [6] for object location.

The algorithm of [3] used $O(\log^2 n)$ queries to a distance oracle to find the nearest neighbor with high probability. In constrast, the algorithm here always finds the nearest neighbor, though the number of queries is a random variable with expectation $O(\log n)$. This matches the bound given by Karger and Ruhl. In the sequential case, our algorithm can be implemented in linear space, whereas that of Karger and Ruhl uses $O(n \log n)$.

2 Our Algorithm

Let us formally define growth-restricted. Let the ball around x of radius r be all nodes of distance less than or equal to r from x. The volume of this ball is the number of nodes it contains. A metric is *growth-restricted* with constant c if, for any x and r, when the ball around x of radius r has volume s, the ball around x of radius $2r$ has volume no more than cs.

The algorithm uses random sampling. We say all nodes are at level 0. Given the set of level-i nodes (a node is a point in the metric space) each of them is independently chosen with probability $1/b$ to also be a level-$(i+1)$ node. That is, for $i \in [0, \log_b n - 1]$, we produce a random sample of the network, with level $i+1$ being a sampling of level i. A node is in the ith sample with probability $1/b^i$. For level $\log_b n$, pick exactly one node to be the root. A node in a level-i sample picks the closest node in the level-$(i+1)$ sample to be its parent. (A node may be its own parent.) This produces a tree.

Given the single level-$(\log_b n)$ root node and the query point x, we can find the nearest neighbor as follows. First, query the root for its children, and keep all the children "close enough" to x. Then, query their children, and keep the children that are "close enough" to x and so on. Let q_i be this "close enough" distance for level-i. (We find the q_i's via a guess-and-check method; for details, see [2].)

2.1 A Certificate To analyze the the algorithm we introduce the notion of a certificate. The certificate is the union, over all i, of the level-i nodes within q_i. This certificate can be used to verify that y is the nearest neighbor of x. If we can show that the size of the certificate is $O(\log n)$, then an algorithm that touches only nodes in the certificate takes time $O(\log n)$. (This is a simplification; the queries to the distance oracle are actually bounded by the number of children of the nodes in the certificate, but this difference only affects the constant.)

For an index i, let d_i be the distance from the query point x to the closest level-i node. Then, let $q_0 = d_0$, and for $i > 0$, $q_i = \max(3d_i, 3q_{i-1})$. With this definition, all level-$(i-1)$ nodes within q_{i-1} have parents within q_i of the query node. Since the certificate contains, for all i, the level-i nodes within distance q_i of the query node, this ensures that if a node is in the certificate, its parent is also in the certificate. This is formalized in the following lemma.

LEMMA 2.1. ([3]) *The parent of every level-$(i-1)$ node within q_{i-1} (of x) is within q_i (of x).*

2.2 Bounding the Certificate Size The difficulty in bounding the certificate size is that any given level may contain $O(\log n)$ nodes, so bounding one level and multiplying by the number of levels gives $O(\log^2 n)$. Instead, we view the certificate as being divided up in pieces, called subcertificates. Two adjacent levels i and $i-1$ are in the same subcertificate if $q_i = 3q_{i-1}$. The lowest level in a subcertificate is called a base level. By definition, level 0 is always a base level. Next, we show that the certificate has $O(\log n)$ nodes in expectation if the metric space is growth restricted. We first bound the size of a subcertificate.

LEMMA 2.2. *Suppose i is a base level (i.e., the lowest level in some subcertificate), and the number of nodes within d_i of x is s. Then the expected size of that subcertificate is $O(s/b^{i-1})$, provided that $c^2 < b$.*

Proof. For a given j, we must find all the level-$(i+j)$ nodes within a factor 3^{j+1} times the base radius (d_i). If the original ball had volume s, then each factor of 3 increase in radius increases the volume of the ball by no more than a factor of c^2. So the ball of radius $3^{j+1}d_i$ has volume bounded by $s(c^2)^{j+1}$, where d_i and s are the base radius and base volume, respectively. For a given j, we only need to store the level-$(i+j)$ nodes. The probability that a node is an level-$(i+j)$ node is $b^{-(i+j)}$. Combining these two facts with a little algebra, we expect to have no more than $s/b^{i-1}(c^2/b)^{j+1}$ level-$(j+i)$ nodes in the certificate. Summing over all possible j, this gives an upper bound of $O(s/b^{i-1})$.

Now we can prove the main size lemma.

LEMMA 2.3. *The total expected size of the certificate is $O(\log n)$ if b is larger than c^2, where c is the expansion constant of the network.*

Proof. We bound the total size of a subcertificate at level i by considering the expected size of the subcertificate when there is no base level larger than i. This is an overcount since some levels may be charged to more than one base level.

Let s_i be the number of nodes within d_i of x. If $s_i = s$, that means the first s nodes were not part of the ith sample. Using this fact, we get $Pr[s_i > cb^i] \leq (1 - 1/b^i)^{cb^i} \leq e^{-c}$. We use this to show that $E[s_i/b^{i-1}]$ is a constant.

High probability versions of both Lemma 2.2 and Lemma 2.3 are proved in [2].

References

[1] K. L. Clarkson. Nearest neighbor queries in metric spaces. In *Proc. of the 29th Annual ACM Symp. on Theory of Comp.*, pages 609–617, May 1997.

[2] K. Hildrum, J. Kubiatowicz, and S. Rao. Another way to find the nearest neighbor in growth-restricted metrics. Technical Report UCB/CSD-03-1267, UC Berkeley, Computer Science Division, Aug. 2003.

[3] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed object location in a dynamic network. In *Proceedings of the Fourteenth ACM Symposium on Parallel Algorithms and Architectures*, pages 41–52, Aug. 2002.

[4] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proc. of the 34th Annual ACM Symp. on Theory of Comp.*, pages 741–750, May 2002.

[5] R. Krauthgamer and J. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. of the 15th Annual ACM-SIAM Symp. on Discrete Algorithms*, Jan. 2004.

[6] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of the 9th Annual Symp. on Parallel Algorithms and Architectures*, pages 311–320, June 1997.

[7] M. Thorup and U. Zwick. Approximate distance oracles. In *Proc. of the 33th Annual ACM Symp. on Theory of Comp.*, pages 183–192, July 2001.

[8] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 2003. Special Issue on Service Overlay Networks, to appear.