Brief Announcement: A Simple Stretch 2 Distance Oracle*

Rachit Agarwal P. Brighten Godfrey

University of Illinois at Urbana-Champaign, IL, USA
{agarwa16, pbg}@illinois.edu

ABSTRACT

We present a distance oracle that, for weighted graphs with n vertices and m edges, is of size $8n^{4/3}m^{1/3}\log^{2/3} n$ and returns stretch-2 distances in constant time. Our oracle achieves bounds identical to the constant-time stretch-2 oracle of Pătraşcu and Roditty, but admits significantly simpler construction and proofs.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and Networks; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*

General Terms

ALgorithms, Theory

Keywords

Approximate distance oracles, distance queries

1. INTRODUCTION

A distance oracle is a compact representation of the all-pairs shortest path matrix of a graph. To achieve a compact (that is, subquadratic in number of vertices) representation, we allow approximation measured in terms of *stretch*. A stretch-c oracle returns, for any pair of vertices at distance d, a distance estimate of at most $c \cdot d$; corresponding path can be retrieved in constant time per hop. Distance oracles have a wide range of applications including compact routing [1, 5, 9] and quickly computing paths on large networks [1, 3, 10]. For general weighted graphs, Thorup and Zwick [10] designed an oracle of size $O(n^{3/2})$ that returns distances of stretch 3 in constant time. Furthermore, they showed that this oracle is optimal in the worst case — there exist graphs for which any oracle that returns distances of stretch 3 requires space $\Omega(n^{3/2})$ and that returns distances of stretch less than 3 requires space $\Omega(n^2)$.

*This work was supported by National Science Foundation grant CNS 1017069.

PODC'13, July 22–24, 2013, Montréal, Québec, Canada.
ACM 978-1-4503-2065-8/13/07.

However, the graphs that constitute the hard cases for stretch less than 3 are extremely dense, while essentially all real-world graphs are sparse. For oracles that improve upon the Thorup-Zwick oracle by exploiting graph sparsity, new upper bounds [1, 2, 4–7] and lower bounds [8] have recently been derived. In particular, Pătraşcu and Roditty [6] designed a constant-time stretch-2 oracle of size $O(n^{4/3}m^{1/3})$ for weighted graphs; their construction was extended for larger stretch values for unweighted [1] and for weighted graphs [7]. In fact, a more general space-stretch-time trade-off can be achieved [2, 4, 5] by exploiting graph sparsity; this further reduces the space requirements for stretch 2 and larger [5] and even allows computing distances of stretch less than 2 [2, 4].

A particularly interesting result among the aforementioned is that of Pătraşcu and Roditty [6] — a stretch-2 constant-time oracle of size $O(n^{4/3}m^{1/3})$. However, their construction uses substantially more complex techniques than oracles for dense graphs and oracles with super-constant query time. For weighted graphs, their algorithm for constructing the oracle is particularly complex — it first samples a set of edges A and a set of vertices B (each with a different probability); it then constructs partial shortest path trees around each vertex in B with a stopping criteria that depends on edges in set A. Finally, the algorithm constructs partial shortest path trees around each remaining vertex with a new stopping criteria that depends on edges in set A, vertices in set B and the edges explored while constructing partial shortest path trees around vertices in set B.

We present a new constant-time stretch-2 oracle for *weighted* graphs that admits significantly simpler construction and proofs. Our algorithm requires sampling a set A of vertices and constructing partial shortest path trees around each vertex using a single stopping criteria that depends only on vertices in set A:

THEOREM 1. *Given a weighted undirected graph with n vertices and m edges with non-negative edge weights, there exists a distance oracle of expected size* $8n^{4/3}m^{1/3}\log^{2/3} n$ *that returns a stretch-*2 *distance in constant time.*

Our construction uses the notion of balls used in [10] and of vicinities used in [2, 4, 5]. We say that a pair of vertices have a ball-vicinity intersection if the ball of one vertex has a non-empty intersection with the vicinity of the other vertex. To bound the space requirements, we exploit graph sparsity to prove a non-trivial upper bound on the number of vertex pairs with ball-vicinity intersection; this requires a special ball construction algorithm previously used in design of compact routing schemes [9]. Furthermore, to bound the stretch, we show that for any pair of vertices with non-intersecting ball-vicinity, a stretch-2 distance can be computed by storing a small amount of information per vertex in the graph.

2. PRELIMINARIES

We assume that $G = (V, E)$ is a weighted undirected graph with n vertices and m edges with non-negative edge weights. Let $d(s,t)$ denote the shortest distance between a pair of vertices $s, t \in V$. For any subset of vertices $V' \subset V$, we denote by $N(V')$ the set of neighbors of vertices in V'. Given a vertex v and a subset of vertices $L \subset V$, we let the **nearest vertex in set** L, denoted by $\ell(v)$, be the vertex $a \in L$ that minimizes $d(v,a)$, ties broken arbitrarily. The **ball radius** of v, denoted by r_v, is the distance between v and $\ell(v)$.

Balls and Vicinities, Inverse-balls and Inverse-Vicinities. We will also need the following definitions:

- **Ball of a vertex $B(v)$:** the set of vertices $w \in V$ for which $d(v,w) < r_v$;
- **Inverse-Ball of a vertex $\bar{B}(v)$:** the set of vertices that contain v in their ball;
- **Vicinity of a vertex $B^+(v)$:** the set of vertices in $B(v) \cup N(B(v))$;
- **Inverse-vicinity of a vertex $\bar{B}^+(v)$:** the set of vertices that contain v in their vicinity.

Our construction of balls, vicinities, inverse-balls and inverse-vicinities will use the following result:

LEMMA 2. [2, 9] *For any weighted undirected graph and for any $1 \le \alpha \le n$, there exists a subset of vertices L of expected size $8n \log n/\alpha$ such that $|\bar{B}(v)| \le \alpha$ and $|\bar{B}^+(v)| \le \alpha \deg(v)$ for each vertex v in the graph.*

The first part of the lemma that shows the existence of a set L to bound the size of the inverse-ball of each vertex is due to Thorup and Zwick [9]; for sake of completeness, the algorithm for constructing such a set L is informally described in Appendix A. It is easy to verify that the set of vertices in the inverse-vicinity of any vertex v is given by $\bar{B}^+(v) = \bigcup_{w \in N(v)} \bar{B}(w)$; this leads to the bound on the size of the inverse-vicinity of each vertex (using the same set L). We emphasize that the above lemma bounds the size of set L in expectation, while the size of inverse-ball and inverse-vicinity for any vertex is bounded deterministically.

3. DISTANCE ORACLE

Our construction of the oracle begins by creating a set L of vertices using the result of Lemma 2 (the value of α will be specified later). The oracle stores, for each $v \in V$:

- a hash table storing the exact distance to each vertex in L;
- the nearest vertex $\ell(v)$ and the ball radius r_v; and
- a hash table storing the exact distance to each vertex in the set $S_v = \{w \ : \ B(v) \cap B^+(w) \neq \emptyset\}$, that is, to each vertex w whose vicinity intersects with the ball of v.

Query algorithm. When queried for the distance between vertices $s, t \in V$, the algorithm returns the exact distance if $s \in S_t$ or if $t \in S_s$. Else, the algorithm returns $d(s,\ell(s)) + d(t,\ell(s))$ if $r_s \le r_t$ and $d(t,\ell(t)) + d(s,\ell(t))$ otherwise.

3.1 Proof of Theorem 1

The proof borrows two ideas from [2]. The first is used to bound the size of the oracle — intuitively, if each vertex has a small size inverse-ball (or equivalently, is contained in a few balls) as guaranteed by Lemma 2, then the number of vertex pairs with ball-vicinity intersection is also small, thereby bounding $\sum_v |S_v|$. The second is used to bound the stretch — any pair of vertices s, t with non-intersecting ball-vicinity must be rather far away and either the path $s \rightsquigarrow \ell(s) \rightsquigarrow t$ or the path $t \rightsquigarrow \ell(t) \rightsquigarrow s$ must be a stretch-2 path.

LEMMA 3. *Let $G = (V, E)$ be a weighted undirected graph with n vertices and m edges. For any fixed $1 \le \alpha \le n$, if the oracle is constructed as above, then: $\sum_{v \in V} |S_v| \le 2\alpha^2 m$.*

Proof: For any vertex $w \in V$, let $\gamma(w)$ be the number of vertex pairs whose ball-vicinity intersection contains w; that is, $\gamma(w) = |\{(u,v) \ : \ w \in B(u) \cap B^+(v)\}|$. Then, by definition, we get that $\sum_{v \in V} |S_v| \le \sum_{w \in V} \gamma(w)$. Recall, using Lemma 2, each vertex w (deterministically) belongs to at most α balls and at most $\alpha \deg(w)$ vicinities. Hence, the number of ball-vicinity intersections that can occur at w is bounded by $\gamma(w) \le \alpha^2 \deg(w)$. Hence, $\sum_{v \in V} |S_v| \le \sum_{w \in V} \gamma(w) \le 2\alpha^2 m$. □

LEMMA 4. [2] *Let $G = (V, E)$ be a weighted undirected graph. For any pair of vertices $s, t \in V$, if $B(s) \cap B^+(t) = \emptyset$, then the shortest distance is lower bounded as $d(s,t) \ge r_s + r_t$.*

Proof: Let $P = (s, x_1, x_2, \ldots, t)$ be the shortest path between s and t. Let $i_0 = \max\{i | x_i \in P \cap B(s)\}$, $w = x_{i_0}$ and $w' = x_{i_0+1}$. Since $w' \notin B(s)$, we get that $d(s,w') \ge r_s$. Since $B(s) \cap B^+(t) = \emptyset$, we have that $w \notin B^+(t)$ and hence, $w' \notin B(t)$ leading to the fact that $d(t,w') \ge r_t$. Finally, w' being on the shortest path between s and t, we have that $d(s,t) = d(s,w') + d(t,w') \ge r_s + r_t$. □

Proof of Theorem 1. We first bound the size of the oracle. Using Lemma 2, the expected size of set L is $8n \log n/\alpha$; and, using Lemma 3, the size of set $\sum_{v \in V} |S_v|$ is bounded by $2\alpha^2 m$. Hence, the oracle's size is bounded by $8n^2 \log n/\alpha + 2\alpha^2 m$; this expression is minimized for $\alpha = 2n^{2/3} m^{-1/3} \log^{1/3}(n)$, leading to the desired bound.

Next, we show that the query algorithm returns a distance of at most $2d(s,t)$. If $B(s) \cap B^+(t) \neq \emptyset$, the algorithm returns the exact distance. For the case when $B(s) \cap B^+(t) = \emptyset$, assume, without loss of generality, that $r_s \le r_t$. Then, using Lemma 4, $d(s,t) \ge 2r_s$; or equivalently, $2r_s \le d(s,t)$. The distance returned by the query algorithm is $d(s,\ell(s)) + d(t,\ell(s))$, which using triangle inequality, is at most $2d(s,\ell(s)) + d(s,t) = 2r_s + d(s,t) \le 2d(s,t)$, as claimed. □

For the special case of unweighted graphs, it is possible to reduce the space requirements at the cost of a small additive stretch. Pătraşcu and Roditty [6] designed a constant time oracle of size $O(n^{5/3})$ for unweighted graphs that, for any pair of vertices at distance d, returns a path of length at most $2d + 1$. Using ideas similar to above, we get a simplified construction for the case of unweighted graphs as well (see Appendix B).

4. REFERENCES

[1] I. Abraham and C. Gavoille. On approximate distance labels and routing schemes with affine stretch. In *International Symposium on Distributed Computing (DISC)*, pages 404–415, 2011.

[2] R. Agarwal. Distance oracles with super-constant query time, Technical report, 2013.

[3] R. Agarwal, M. Caesar, P. B. Godfrey, and B. Y. Zhao. Shortest paths in less than a millisecond. In *ACM SIGCOMM Workshop on Online Social Networks (WOSN)*, 2012.

[4] R. Agarwal and P. B. Godfrey. Distance oracles for stretch less than 2. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[5] R. Agarwal, P. B. Godfrey, and S. Har-Peled. Approximate distance queries and compact routing in sparse graphs. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 1754–1762, 2011.

[6] M. Pătraşcu and L. Roditty. Distance oracles beyond the Thorup-Zwick bound. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 815–823, 2010.

[7] M. Pătraşcu, L. Roditty, and M. Thorup. A new infinity of distance oracles for sparse graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012.

[8] C. Sommer, E. Verbin, and W. Yu. Distance oracles for sparse graphs. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 2009.

[9] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

[10] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, January 2005.

APPENDIX

A. INFORMAL PROOF OF LEMMA 2

Fix some $1 \leq \alpha \leq n$. The algorithm maintains two set of vertices — a set L that constitutes the final output of the algorithm and another set W that contains all vertices that have inverse-ball of size more than α. The set L is initialized to an empty set and W is initialized to the vertex set V. The algorithm runs in multiple iterations; in each iteration, it uniform randomly samples $4n/\alpha$ vertices from W, inserts them to set L; re-computes the inverse-ball of each vertex and updates W to all vertices that still contains more than α vertices in their inverse-ball. The algorithm terminates when W contains $4n/\alpha$ or fewer vertices; in this case, all vertices in W are inserted in set L.

The main idea behind the proof of correctness is as follows. Clearly, by construction, each vertex has inverse-ball of size at most α. The main challenge is to bound the size of set L. It is shown in [9] that the expected number of iterations performed by the algorithm before termination is at most $2\log n$; since $4n/\alpha$ vertices are added to L in each iteration, the size of the set L output by the algorithm is at most $8n\log n/\alpha$.

B. UNWEIGHTED GRAPHS

A stretch-(c, c') oracle for unweighted graphs returns, for any pair of vertices at distance d, a path of length at most $c \cdot d + c'$. Pătraşcu and Roditty [6] designed a constant-time stretch-$(2,1)$ oracle of size $O(n^{5/3})$ for general unweighted graphs. Using ideas similar to those for weighted graphs, we get a simpler construction for the case of unweighted graphs as well:

THEOREM 5. *Given a unweighted undirected graph with n vertices and m edges, there exists a distance oracle of expected size $4n^{5/3}\log^{2/3} n$ that returns a stretch-$(2,1)$ distance in constant time.*

Abraham and Gavoille [1] presented a similar construction and further generalized it for larger stretch values. Due to the focus on small stretch values, our exposition is slightly simpler than their. The construction and proofs for the following oracle is similar to that for weighted graphs with the only difference that it now suffices to consider ball-ball intersections rather than ball-vicinity intersections.

B.1 Distance oracle

Our construction of the oracle begins by creating a set L of vertices using the result of Lemma 2 (the value of α will be specified later). The oracle stores, for each $v \in V$:

- a hash table storing the exact distance to each vertex in L;
- the nearest vertex $\ell(v)$ and the ball radius r_v; and
- a hash table storing the exact distance to each vertex in the set $S_v = \{w \ : \ B(v) \cap B(w) \neq \emptyset\}$, that is, to each vertex w whose ball intersects with the ball of v.

Query algorithm. When queried for the distance between vertices $s, t \in V$, the algorithm returns the exact distance if $s \in S_t$ or if $t \in S_s$. Else, the algorithm returns $d(s,\ell(s)) + d(t,\ell(s))$ if $r_s \leq r_t$ and $d(t,\ell(t)) + d(s,\ell(t))$ otherwise.

B.2 Proof of Theorem 5

As with the proof of Theorem 1, this proof uses two ideas. The first is used to bound the oracle's size — we show that if each vertex has a small size inverse-ball (or equivalently, is contained in a few balls) as guaranteed by Lemma 2, then the number of vertex pairs with ball-ball intersection is also small, thereby bounding $\sum_v |S_v|$. Second, we show that any pair of vertices s, t with non-intersecting ball-ball must be rather far away and either the path $s \rightsquigarrow \ell(s) \rightsquigarrow t$ or the path $t \rightsquigarrow \ell(t) \rightsquigarrow s$ must be a stretch-$(2,1)$ path.

LEMMA 6. *Let $G = (V, E)$ be a unweighted undirected graph with n vertices. For any fixed $1 \leq \alpha \leq n$, if the oracle is constructed as above, then: $\sum_{v \in V} |S_v| \leq \alpha^2 n$.*

Proof: For any vertex $w \in V$, let $\gamma(w)$ be the number of vertex pairs whose ball-ball intersection contains w; that is, $\gamma(w) = |\{(u,v) \ : \ w \in B(u) \cap B(v)\}|$. Then, by definition, we get that $\sum_{v \in V} |S_v| \leq \sum_{w \in V} \gamma(w)$. Recall, using Lemma 2, each vertex w (deterministically) belongs to at most α balls. Hence, the number of ball-ball intersections that can occur at w is bounded by α^2; consequently, we have that for any vertex $w \in V$, $\gamma(w) \leq \alpha^2$. Hence, $\sum_{v \in V} |S_v| \leq \sum_{w \in V} \gamma(w) \leq \alpha^2 n$. □

Proof of Theorem 5. We first bound the size of the oracle. Using Lemma 2, the expected size of set L is $8n\log n/\alpha$; and, using Lemma 6, the size of set $\sum_{v \in V} |S_v|$ is bounded by $\alpha^2 n$. Hence, the size of the oracle is bounded by $8n^2 \log n/\alpha + \alpha^2 n$; this expression is minimized for $\alpha = 2n^{1/3}\log^{1/3}(n)$, leading to the desired bound.

Next, we show that the query algorithm returns a distance of at most $2d(s,t) + 1$. If $B(s) \cap B(t) \neq \emptyset$, the algorithm returns the exact distance. For the case when $B(s) \cap B(t) = \emptyset$, assume, without loss of generality, that $r_s \leq r_t$. Let $P = (s, x_1, x_2, \ldots, t)$ be the shortest path between s and t. Let $i_0 = \max\{i | x_i \in P \cap B(s)\}$, $w = x_{i_0}$ and $w' = x_{i_0+1}$. Since $w' \notin B(s)$, we get that $d(s,w') \geq r_s$. Since $B(s) \cap B(t) = \emptyset$, we have that $w \notin B(t)$ and hence, $d(t,w) \geq r_t$. Finally, w' being on the shortest path between s and t, we have that $d(s,t) = d(s,w') + d(t,w') = d(s,w') + d(t,w) - 1 \geq r_s + r_t - 1 \geq 2r_s - 1$; or equivalently, $2r_s \leq d(s,t)+1$. The distance returned by the query algorithm is $d(s,\ell(s)) + d(t,\ell(s))$, which using triangle inequality, is at most $2d(s,\ell(s)) + d(s,t) = 2r_s + d(s,t) \leq 2d(s,t) + 1$, as claimed. □