The Space-Stretch-Time Tradeoff in Distance Oracles

Rachit Agarwal

University of California at Berkeley, CA, USA
ragarwal@berkeley.edu

Abstract. We present new distance oracles for computing distances of stretch less than 2 on general weighted undirected graphs. For the realistic case of sparse graphs and for any integer k, the new oracles return paths of stretch $1+1/k$ and exhibit a smooth three-way tradeoff of $S \times T^{1/k} = O(n^2)$ between space S, stretch and query time T. This significantly improves the state-of-the-art *for each point in the space-stretch-time tradeoff space*, and matches the known space-time curve for stretch 2 and larger. We also present new oracles for stretch $1 + 1/(k + 0.5)$. A particularly interesting case is of stretch $5/3$, where improving the query time of our oracles from T to $T^{1-\varepsilon}$ for any $\varepsilon > 0$ would lead to the first purely $o(mn)$-time combinatorial algorithm for Boolean Matrix Multiplication, a longstanding open problem.

1 Introduction

A distance oracle is a compact representation of all-pair shortest path matrix of a graph. A stretch-c oracle for a weighted undirected graph $G = (V, E)$ returns, for any pair of vertices $s, t \in V$ at distance $d(s, t)$, a distance estimate $\delta(s, t)$ that satisfies $d(s, t) \leq \delta(s, t) \leq c \cdot d(s, t)$. Let $n = |V|$ and $m = |E|$. For general graphs, Thorup and Zwick [31] showed a fundamental space-stretch tradeoff — for any integer $k \geq 2$, they designed an oracle of size $O(kn^{1+1/k})$ that returned distances of stretch $(2k - 1)$ in $O(k)$ time; the construction time of their oracle was $\widetilde{O}(kmn^{1/k})$, in expectation. The Thorup-Zwick (TZ) oracle was a significant improvement over previous constructions that had much higher stretch and/or query time [8, 15, 21].

Improvements in Construction and Query Time. Much of the early research following the TZ result focused on improving the construction time. Roditty, Thorup and Zwick [27] derandomized the TZ construction. Baswana and Sen [11] improved the construction time to $O(n^2)$ for unweighted graphs. Their result was extended to weighted graphs by Baswana and Kavitha [10]. Finally, Wulff-Nilsen [33] achieved subquadratic construction time for weighted graphs with $m = o(n^2)$ edges.

The query time of the TZ oracle is not constant for super-constant stretch. Wulff-Nilsen [32] reduced the query time of the TZ oracle to $O(\log k)$ using a new query algorithm that incorporates binary search within the TZ oracle. Mendel and Naor [22] reduced the query time to $O(1)$ at the expense of increasing the stretch to $O(k)$ and the construction time to $\widetilde{O}(n^{2+1/k})$. Interestingly, Chechik [13] showed that it is possible to reduce the query time of TZ oracle to an absolute constant, without increasing the stretch or space of the original TZ construction.

A. Schulz and D. Wagner (Eds.): ESA 2014, LNCS 8737, pp. 49–60, 2014.

Improvements in Space-Stretch Tradeoff. Thorup and Zwick showed that, assuming a girth conjecture of Erdős, any oracle that returns distances of stretch less than $(2k+1)$ must have size $\Omega(n^{1+1/k})$. However, the hard instances used to prove this lower bound are extremely dense graphs; for instance, their construction uses a graph with $m = \Omega(n^2)$ edges to prove the space lower bound for stretch less than 3. For graphs with $m = o(n^2)$ edges, it may in fact be possible to get a better space-stretch tradeoff — their result merely implies a trivial space lower bound of $\Omega(m)$, that is, compression is impossible.

However, improving the space-stretch tradeoff turned out to be a much harder problem. Until 2010, a better tradeoff was known only for special graph classes such as planar graphs [20, 30], bounded-genus and minor-free graphs [19], power-law graphs [14] and random graphs [17]. Pătraşcu and Roditty [23] achieved the first breakthrough, constructing a stretch-2 constant-time oracle of size $\widetilde{O}(n^{4/3}m^{1/3})$. Their result was generalized for larger stretch values by Abraham and Gavoille [1] and by Pătraşcu, Roditty and Thorup [24]. The original construction of Pătraşcu and Roditty was rather complex; simpler construction and analysis are now known [3].

Lower Bounds. Sommer *et al.* [28] proved in the cell-probe model that the size of stretch-s time-t distance oracles is lower bounded by $n^{1+\Omega(1/st)}$. That is, for graphs with $m = \widetilde{O}(n)$ edges, computing distances of constant stretch in constant time requires super-linear space. Conditioned on hardness of set intersection, Pătraşcu and Roditty [23] strengthened their result by proving an $\Omega(n^2)$ space lower bound for constant-time stretch-less-than-2 oracles. Pătraşcu, Roditty and Thorup [24] proved, among other results, a conditional space lower bound of $\Omega(m^{5/3})$ for constant-time stretch-2 oracles. Due to several upper bounds matching these lower bounds [23, 24], these results are believed to provide a complete understanding of the space-stretch tradeoff *for constant-time oracles*.

Distance Oracles with Super-Constant Query Time. The problem of improving the space-stretch tradeoff of the TZ oracle is wide open if one allows super-constant query time. No non-trivial lower bounds are known for this regime and it is possible that there exist constant-stretch oracles of size $\widetilde{O}(m)$ with polylog(n) query time!

Agarwal, Godfrey and Har-Peled [5, 6] constructed oracles with super-constant query time for stretch 2 and larger. Their stretch-2 and stretch-3 oracles achieve a space-time tradeoff of $S \times T = O(n^2)$ and $S \times T^2 = O(n^2)$, respectively, for sparse graphs. For instance, stretch-2 and stretch-3 distances can be computed using $\widetilde{O}(n^{3/2})$ and $\widetilde{O}(n)$ space, respectively, if one allows $O(\sqrt{n})$ query time. Even on graphs with millions of nodes and edges, a query time of $O(\sqrt{n})$ time can be engineered to return results in less than a millisecond [2], an extremely acceptable latency for most real-world applications [2, 5, 18, 26].

Porat and Roditty [25] showed existence of $o(n^2)$-size stretch-less-than-2 oracles for unweighted graphs given super-constant query time. Agarwal and Godfrey [4] explored a general space-stretch-time tradeoff for oracles with stretch less than 2 — their oracle is for general weighted graphs and significantly reduces both the space and query time of the Porat-Roditty oracle for any fixed stretch. For space, stretch, query time and construction time bounds for these oracles, see Table 1.

Table 1. Summary of results and comparison with oracles in [25] and in [4]

Stretch	Space	Query time	Construction time	Remarks	Ref.
$1+\frac{1}{k}$	$\widetilde{O}\left(\frac{nm}{m^{1/(4k+2)}}\right)$	$O\left(\frac{m}{m^{1/(4k+2)}}\right)$	$\widetilde{O}(mn)$	$1 \le \alpha \le n$	[25]
	$\widetilde{O}(m+n^2/\alpha)$	$O((\alpha\mu)^{2k-1})$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le n$	[4]
	$\widetilde{O}(m+n^2/\alpha)$	$O((\alpha\mu)^{k})$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le n$	§3
$1+\frac{1}{k+0.5}$	$\widetilde{O}(m+n^2/\alpha)$	$O((\alpha\mu)^{2k})$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le n$	[4]
	$\widetilde{O}(m+n^2/\alpha)$	$O(\alpha(\alpha\mu)^{k})$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le n$	§4
$1+\frac{2}{3}$	$\widetilde{O}(m+n^2/\alpha)$	$O((\alpha\mu)^{2})$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le (n^2/m)^{1/3}$	[4]
	$\widetilde{O}(m+n^2/\alpha)$	$O(\alpha\mu)$	$\widetilde{O}(mn/\alpha)$	$1 \le \alpha \le (n^2/m)^{1/3}$	§5

1.1 Our Contributions

This paper makes two contributions. Our first contribution is a new space-stretch-time tradeoff for distance oracles for stretch less than 2:

Theorem 1. *Let G be a non-negatively weighted undirected graph with n vertices, m edges and average degree $\mu = 2m/n$. Then, for any fixed $1 \le \alpha \le n$ and for any integer $k \ge 1$, there exist distance oracles of size $\widetilde{O}(m+n^2/\alpha)$ that return distances of stretch $(1+\frac{1}{k})$ in $O((\alpha\mu)^k)$ time and of stretch $(1+\frac{1}{k+0.5})$ in $O(\alpha(\alpha\mu)^k)$ time. For $1 \le \alpha \le n^{2/3}m^{-1/3}$, there also exist oracles of size $\widetilde{O}(m+n^2/\alpha)$ that return distances of stretch $5/3$ in $O(\alpha\mu)$ time. All these oracles can be constructed in time $\widetilde{O}(mn/\alpha)$.*

The first oracle of Theorem 1, for sparse graphs, achieves a space-stretch-time tradeoff of $S \times T^{1/k} = O(n^2)$ for stretch $(1+1/k)$. For any fixed space and stretch, the oracle reduces the query time in [4] from T^{2k-1} to T^k (or alternatively, reduces space for any fixed stretch and query time). Interestingly, the space-stretch-time tradeoff achieved by this oracle matches the known space-time tradeoff space for stretch 2 and larger. For instance, setting $k=1$, we get $S \times T = O(n^2)$ for stretch 2 and setting $1/k = 2$, we get $S \times T^2 = O(n^2)$ for stretch 3, precisely as in [5].

The second oracle reduces the query time from T^{2k-2} in [4] to T^k for any fixed stretch and space. Note that both the first and the second construction also enable non-trivial constructions that were not possible using the results in [4]. For instance, the new results make it possible to compute stretch-1.5 distances using oracles of size $\widetilde{O}(n^{5/3})$ in sub-linear time.

The third oracle of Theorem 1 is particularly interesting. For any space $S = \Omega(n^{5/3})$, this oracle reduces the query time of the stretch-5/3 oracle of [4] from T to $\sqrt{T}$. In particular, this oracle achieves a space-time tradeoff of $S \times T = O(n^2)$, which is same as the stretch-2 oracle of [5]. Essentially, compared to the stretch-2 oracle of [5], this oracle reduces the stretch from 2 to 5/3 without any increase in space or query time for the regime of $S = \Omega(n^{5/3})$.

We argue that the query time of the second and the third oracles may be close to optimal. Specifically, the problem of computing all-pair stretch-less-than-2 distances in undirected graphs is equivalent to combinatorial[1] boolean matrix multiplication (BMM) over the (OR, AND) semiring [16]. Hence, for $k = 1$, if the query time of the second oracle of Theorem 1 can be reduced to $O((\alpha^2\mu)^{1-\varepsilon})$ for any $\varepsilon > 0$, it would be possible to multiply two boolean matrices in time $\widetilde{O}(mn/\alpha + n^2(\alpha^2\mu)^{1-\varepsilon})$. By setting $\alpha = o((m/n)^{\beta})$ for $\beta = \frac{\varepsilon}{2(1-\varepsilon)}$, we get that the time would be $o(mn)$. Hence, improving the query time from T to $T^{1-\varepsilon}$ for any $\varepsilon > 0$ would lead to a purely $o(mn)$ time combinatorial algorithm for BMM, a long standing open problem [9, 12].

New Query Algorithms. In contrast to the elegant and compact data structures used in constant-time oracles [1, 23, 24, 31], the data structures for super-constant time oracles [4,5] are usually relatively simpler — in addition to the graph, distance from a few sampled vertices to each vertex in the graph is stored. The main technique used in super-constant time oracles is more sophisticated query algorithms that allow exploring a tradeoff between space, stretch and query time (cf. [4]). Our second contribution is, indeed, such new query algorithms.

Our query algorithms perform a *bidirectional recursion* to compute (not necessarily shortest) distances to vertices in carefully defined neighborhoods of both the source s and the destination t. Specifically, the algorithm explores recursively larger neighborhoods of both s and t in each step, and computes distances from s and t to vertices in the respective neighborhoods. The neighborhoods are defined in a manner that once the recursion depth is reached, the explored neighborhoods either intersect along the shortest path or we are able to prove a non-trivial lower bound on the exact distance between s and t. Intuitively, these neighborhood definitions ensure that two new "subpaths" of the shortest path between s and t are explored in each recursive step (one closer to s and one closer to t). When neighborhoods do not intersect, the length of the shortest of these subpaths times twice the recursion depth is a lower bound on the exact distance between s and t. Moreover, the neighborhood definitions also ensure that the neighborhoods explored in each recursive step also contain at least one of the "landmark" vertices that store distances to each vertex in the graph (computed and stored during graph preprocessing). The path via the landmark vertex in the neighborhood containing the shortest of the subpaths gives us a path with desired stretch.

Our new query algorithms are simpler, faster and compute paths of smaller stretch than the ones in [4]. In contrast to the algorithm in [4] that explores the neighborhood of only either the source or the destination in each recursive step, our new definition of neighborhoods allow us to perform bidirectional recursion. This, in turn, leads to significantly stronger lower bound on the exact distance between the source and the destination without any asymptotic increase in the query time.

[1] Although not defined precisely, we say that an algorithm is "combinatorial" in nature if it does not use algebraic techniques of fast matrix multiplication.

2 Preliminaries

This section sets up the notation and basic results [4, 5, 31] used throughout the paper. We assume that the graph $G = (V, E)$ is a weighted undirected graph with n vertices and m edges with non-negative edge weights.

2.1 Reducing the Problem to Degree-Bounded Graphs

The following lemma shows that the problem of designing oracles and algorithms for computing low stretch distances on weighted graphs with n vertices and m edges is no harder than designing oracles for $O(m/n)$-degree bounded graphs.

Claim 1 ([4–6]). *Let $G = (V, E)$ be a weighted undirected graph with n vertices, m edges with non-negative edge weights, and average degree $\mu = 2m/n$. Then, it is possible to construct an equivalent graph with maximum degree $\Delta = \lceil \mu + 2 \rceil$, such that the new graph has $2n$ vertices, $m + n$ edges, and has the same distances between any pair of vertices as the distance in the original graph between the corresponding vertices. The new graph can be computed in $O(n + m)$ time.*

2.2 Balls and Vicinities, Shortest Distances and Candidate Distances

Let $d(s, t)$ denote the exact distance between any vertex pair $s, t \in V$. For any $V' \subset V$, we denote by $N(V')$ the set of neighbors of vertices in V'. Given G, a vertex v and a subset of vertices $L \subset V$, we use the following definitions:

- **Nearest vertex in set L — $\ell(v)$:** the vertex $a \in L$ that minimizes $d(v, a)$, ties broken arbitrarily.
- **Ball radius r_v:** the distance from v to its nearest neighbor in L, that is, $d(v, \ell(v))$.
- **Ball of a vertex $B(v)$:** the set of vertices $w \in V$ for which $d(v, w) < d(v, \ell(v))$.
- **Vicinity of a vertex $B^*(v)$:** the set of vertices in $B(v) \cup N(B(v))$.
- **Candidate distance from v to w — $d'_v(w)$:** cost of the least-cost path from v to w such that all intermediate vertices on this path are contained in $B(v)$; that is:

$$d'_v(w) = \min_{x \in N(w) \cap B(v)} \{d(v, x) + \text{weight of edge}(x, w)\}$$

If $N(w) \cap B(v) = \emptyset$, we let $d'_v(w) = \infty$.

The following lemma gives an efficient way of sampling vertices for set L such that the ball of each vertex is of bounded size (for degree-bounded graphs, we also get a bound on the size of the vicinity of each vertex):

Lemma 1 ([7, 31]). *Let $G = (V, E)$ be a weighted undirected graph with n vertices, m edges with non-negative weights and maximum degree $\mu = O(m/n)$. For any fixed $1 \leq \alpha \leq n$, there exists a subset of vertices L of size $\widetilde{O}(n/\alpha)$ such that for each vertex $v \in V$, we have that $|B(v)| = O(\alpha)$ and $|B^*(v)| = O(\alpha\mu)$ with high probability. Moreover, such a set L can be computed in time $\widetilde{O}(m)$.*

For a $\mu = O(m/n)$-degree bounded graph, it is not very hard to construct a set L in time $\widetilde{O}(m\alpha)$ that deterministically guarantees the above bound. The following claim, which settles a sufficient condition for the candidate distance to be equal to the exact shortest distance, will play a crucial role in our proofs:

Claim 2. *Let s, t be a vertex pair such that $t \notin B(s)$. Let $P = (s, x_1, x_2, \ldots, t)$ be a shortest path between s and t. Let x_{i_0} be the first vertex from s along P that does not lie in $B(s)$; that is, let $i_0 = \max\{i \ : \ x_j \in B(s) \cap P, \forall j < i\}$. Then, $d'_s(x_{i_0}) = d(s, x_{i_0})$.*

3 Stretch $\left(1 + \frac{1}{k}\right)$ Oracle

In this section, we prove the first part of Theorem 1: for a weighted undirected graph with n vertices, m edges with non-negative weights, and for any $1 \leq \alpha \leq n$, there exists an oracle of size $\widetilde{O}(m + n^2/\alpha)$ that returns distances of stretch $1 + 1/k$ in time $O((\alpha\mu)^k)$. We need some notation to succinctly describe the construction.

3.1 i-Balls and i-Vicinities

We will generalize the idea of balls and vicinities from §2.2. In particular, we define the i-vicinity of a vertex $v \in V$, denoted as $\Gamma^\star_i(v)$ as follows:

$$\Gamma^\star_0(v) = \{v\}; \qquad \text{and} \qquad \Gamma^\star_i(v) = \bigcup_{w \in \Gamma^\star_{i-1}(v)} B^\star(w) \tag{1}$$

For instance, the 1-vicinity of any vertex includes all the vertices in its vicinity and the 2-vicinity of any vertex v is the union of all the vicinities of vertices in $B^\star(v)$. Given the definition of i-vicinities, we can now define the i-ball of a vertex v:

$$\Gamma_0(v) = \emptyset; \qquad \text{and} \qquad \Gamma_i(v) = \bigcup_{w \in \Gamma^\star_{i-1}(v)} B(w) \tag{2}$$

Note that $\Gamma_i(v) \subseteq \Gamma^\star_i(v)$ for any vertex v. We will also need a generalization for the definition of the *candidate* distance. Given a vertex v and a vertex w in the i-vicinity of v, the candidate distance from v to w is given by the cost of the least-cost path from v to w such that all intermediate vertices are contained in the i-ball of v. We will slightly abuse the notation and use $d'_v(w)$ to denote this candidate distance.

3.2 Oracle and Query Algorithm

Our oracle is similar to the one used in [4]. Fix some $1 \leq \alpha \leq n$. The preprocessing algorithm first replaces the original graph with a degree-bounded graph using Claim 1. The algorithm then samples a set L of vertices of size $\widetilde{O}(n/\alpha)$ using the result of Lemma 1. The oracle stores, for each $v \in V$: (1) a hash table storing the shortest distance to each vertex in L; and (2) the nearest neighbor $\ell(v)$ and the ball radius r_v. In addition, the oracle also stores the degree-bounded graph computed in the first step of the preprocessing algorithm.

We now describe our query algorithm (see Algorithm 1). In the first two steps, the query algorithm computes candidate distance from s and from t to each vertex in their respective k-vicinities; these distances are temporarily stored in a hash table. Then, the algorithm computes three sets of paths between s and t. The first set of paths are of the form $s \rightsquigarrow w \rightsquigarrow t$ via vertices w in $\Gamma_k^\star(s) \cap \Gamma_k^\star(t)$. The second set of paths are of the form $s \rightsquigarrow w \rightsquigarrow \ell(w) \rightsquigarrow t$ via vertices $w \in \Gamma_k^\star(s)$. The third set of paths are of the form $t \rightsquigarrow w \rightsquigarrow \ell(w) \rightsquigarrow s$ via vertices $w \in \Gamma_k^\star(t)$. Finally, the least-cost path among all the above three sets of paths is returned.

Algorithm 1. Query algorithm for the stretch-$(1+1/k)$ oracle

1: Compute candidate distance from s to each vertex in $\Gamma_k^\star(s)$
2: Compute candidate distance from t to each vertex in $\Gamma_k^\star(t)$
3: $\gamma_1 \leftarrow \infty$, $\gamma_2 \leftarrow \infty$, $\gamma_3 \leftarrow \infty$
4: $\gamma_1 \leftarrow \min_{w \in \Gamma_k^\star(s) \cap \Gamma_k^\star(t)} \left\{ d'_s(w) + d'_t(w) \right\}$
5: $\gamma_2 \leftarrow \min_{w \in \Gamma_k^\star(s)} \left\{ d'_s(w) + d(w, \ell(w)) + d(\ell(w), t) \right\}$
6: $\gamma_3 \leftarrow \min_{w \in \Gamma_k^\star(t)} \left\{ d'_t(w) + d(w, \ell(w)) + d(\ell(w), s) \right\}$
7: return $\min\{\gamma_1, \gamma_2, \gamma_3\}$

3.3 Analysis

For any pair of vertices $s, t \in V$, let $P(s,t) = (s, x_1, x_2, \ldots, t)$ denote the shortest path between s and t. Let

$$w_i^s(t) = x_{i_0}, \text{ where } i_0 = \max\{i \ : \ x_j \in B(w_{i-1}^s(t)) \cap P(s,t), \forall j < i\}; \qquad w_0^s(t) = t$$

Intuitively, $w_i^s(t)$ is the first vertex from $w_{i-1}^s(t)$ along $P(s,t)$ that is not contained in the ball of $w_{i-1}^s(t)$. Let

$$r_i^s(t) = \min_{j \le i} \{ d(w_j^s(t), \ell(w_j^s(t))) \}$$

that is, $r_i^s(t)$ is the smallest ball radius among all vertices $w_j^s(t)$ for $j \le i$. When the context is clear, we will denote $w_i^s(t)$ and $r_i^s(t)$ simply as w_i^s and r_i^s. We will need the following claims to prove our main result:

Claim 3. *Let $P(s,t) = (s, x_1, x_2, \ldots, t)$ be the shortest path between a pair of vertices s and t. Let i_0 and j_0 be such that $w_k^s = x_{i_0}$ and $w_k^t = x_{j_0}$. Then, for all $i \le i_0$, $d'_s(x_i) = d(s, x_i)$ and for all $j \ge j_0$, $d'_t(x_j) = d(t, x_j)$.*

Claim 4. *For any vertex pair s, t, we have that $d(s, w_i^s) \ge i \cdot r_{i-1}^s$ and $d(t, w_i^t) \ge i \cdot r_{i-1}^t$.*

Claim 5. *For any pair of vertices $s, t \in V$, if $w_k^s \notin \Gamma_k^\star(t)$, then we have that $d(s,t) \ge 2k \min\{r_{k-1}^s, r_{k-1}^t\}$.*

Claim 6. *For any pair of vertices $s, t \in V$, the query algorithm returns a distance estimate of at most $d(s,t) + 2\min\{r_{k-1}^s, r_{k-1}^t\}$.*

Proof of First Oracle of Theorem 1. The oracle stores the input graph and the distance from each vertex in the graph to each vertex in a set L of size $\widetilde{O}(n/\alpha)$; hence, the size of the oracle is $\widetilde{O}(m+n^2/\alpha)$. Constructing the oracle requires computing a shortest path tree from each vertex in set L, and hence, requires time $\widetilde{O}(mn/\alpha)$.

Next, we bound the query time of the query algorithm. We first claim that the size of the k-vicinity of each vertex is bounded by $O((\alpha\mu)^k)$. This follows from the definition of the i-vicinity and from the fact that the size of the vicinity of each vertex is bounded by $O(\alpha\mu)$. Furthermore, the candidate distance from any vertex v to vertices in $B^\star(v)$ can be computed in $O(\alpha\mu)$ time. Hence, by definition of i-vicinity, it takes time $O((\alpha\mu)^k)$ to compute the candidate distance from s to vertices in $\Gamma_k^\star(s)$. Finally, lines (4), (5) and (6) of Algorithm 1 take time linear in the size of the i-vicinities of s and t, leading to the desired bound of $O((\alpha\mu)^k)$ on query time.

Finally, we prove a bound on stretch. If $w_k^s \in \Gamma_k^\star(t)$, then $\gamma_1 \leq d'_s(w_k^s) + d'_t(w_k^s) = d(s,w_k^s)+d(t,w_k^s) = d(s,t)$; hence, the exact distance is returned. Consider the case when $w_k^s \notin \Gamma_k^\star(t)$. Then, by Claim 5, we have that the distance between s and t is lower bounded by $d(s,t) \geq 2k\min\{r_{k-1}^s, r_{k-1}^t\}$. On the other hand, from Claim 6, the distance returned by the query algorithm is at most $d(s,t)+2\min\{r_{k-1}^s, r_{k-1}^t\} \leq d(s,t)+2d(s,t)/(2k)$, leading to the desired bound on stretch. □

4 Stretch $\left(1+\frac{1}{k+0.5}\right)$ Oracle

We now prove the second part of Theorem 1: for a weighted undirected graph with n vertices, m edges with non-negative weights, and for any $1 \leq \alpha \leq n$, there exists an oracle of size $\widetilde{O}(m+n^2/\alpha)$ that returns distances of stretch $1+1/(k+0.5)$ in time $O(\alpha(\alpha\mu)^k)$. See notation in §2.2 and §3.1.

4.1 Oracle and Query Algorithm

We will use the oracle of §3.2 with the addition that the exact distance from each vertex v to each vertex in $B(v)$ will be stored within the oracle. The query algorithm for this oracle (see Algorithm 2) is similar to that of Algorithm 1 with the only difference that the k-vicinities $\Gamma_k^\star(s)$ and $\Gamma_k^\star(t)$ are now replaced by $(k+1)$-balls $\Gamma_{k+1}(s)$ and $\Gamma_{k+1}(t)$, respectively (and γ_1, γ_2 and γ_3 modified accordingly).

4.2 Analysis

The proof is facilitated by the following two claims that are used to bound the stretch of the oracle:

Claim 7. *For any vertex pair s,t, if $w_k^s \notin \Gamma_{k+1}(t)$ then $d(s,t) \geq (2k+1)\min\{r_{k-1}^s, r_k^t\}$.*

Claim 8. *For any pair of vertices s,t, the query algorithm returns a distance estimate of at most $d(s,t)+2\min\{r_{k-1}^s, r_k^t\}$.*

The above two claims directly lead to the stretch bound claimed in Theorem 1. The proof on the size, construction time, and query time follow using straightforward changes in the proof for the first oracle.

Algorithm 2. The query algorithm for stretch-$(1+1/(k+0.5))$ oracle

1: Compute candidate distance from s to each vertex in $\Gamma_{k+1}(s)$
2: Compute candidate distance from t to each vertex in $\Gamma_{k+1}(t)$
3: $\gamma_1 \leftarrow \infty$, $\gamma_2 \leftarrow \infty$, $\gamma_3 \leftarrow \infty$
4: $\gamma_1 \leftarrow \min_{w \in \Gamma_{k+1}(s) \cap \Gamma_{k+1}(t)} \left\{ d'_s(w) + d'_t(w) \right\}$
5: $\gamma_2 \leftarrow \min_{w \in \Gamma_{k+1}(s)} \left\{ d'_s(w) + d(w, \ell(w)) + d(\ell(w), t) \right\}$
6: $\gamma_3 \leftarrow \min_{w \in \Gamma_{k+1}(t)} \left\{ d'_t(w) + d(w, \ell(w)) + d(\ell(w), s) \right\}$
7: return $\min\{\gamma_1, \gamma_2, \gamma_3\}$

5 Stretch $\left(1 + \frac{2}{3}\right)$ Oracle

Finally, we prove the third part of Theorem 1: for a weighted undirected graph with n vertices, m edges with non-negative weights and for any $1 \leq \alpha \leq n$, an oracle of size $\widetilde{O}(m + n^2/\alpha)$ that returns distances of stretch $5/3$ in time $O(\alpha\mu)$.

5.1 Inverse-Ball and Inverse-Vicinities

The **inverse-ball of a vertex, denoted by** $\bar{B}(v)$**,** is the set of vertices w that contain v in their ball. Similar, the **inverse-vicinity of a vertex, denoted by** $\bar{B}^\star(v)$**,** is the set of vertices w for which $v \in B^\star(w)$. For constructing this oracle, we will use a different sampling technique given by the following lemma:

Lemma 2 ([29,31]). *Let $G = (V, E)$ be a weighted undirected graph with n vertices, m edges and maximum degree $\mu = 2m/n$. For any fixed $1 \leq \alpha \leq n$, there exists a subset of vertices L of expected size $\widetilde{O}(n/\alpha)$ such that for each vertex $v \in V$, we have that $|B(v)| = O(\alpha)$, $|\bar{B}(v)| = O(\alpha)$, $|B^\star(v)| = O(\alpha\mu)$ and $|\bar{B}^\star(v)| = O(\alpha\mu)$. Moreover, such a set L can be computed in expected time $\widetilde{O}(m\alpha)$.*

5.2 Oracle and Query Algorithm

Fix some $1 \leq \alpha \leq n$. The preprocessing algorithm first replaces the original graph with a degree-bounded graph using the result of Corollary 1. The algorithm then samples a set L of vertices of size $\widetilde{O}(n/\alpha)$ using the result of Lemma 2. The algorithm then constructs a data structure that stores, for each $v \in V$:

- a hash table storing the shortest distance to each vertex in L;
- the nearest neighbor $\ell(v)$ and the ball radius r_v;
- a hash table storing the distance $d'_s(w) = \min_{x \in B^\star(v) \cap B(w)} d'_s(x) + d(x, w)$ to each vertex w in the set $S_v = \{w \ : \ B^\star(v) \cap B(w) \neq \emptyset\}$, that is, to all vertices w whose ball intersects with the *vicinity* of v.

The oracle also stores the degree-bounded graph computed in the first step of the preprocessing algorithm.

We now describe our query algorithm (see Algorithm 3). In the first and the second step, the query algorithm computes candidate distances from s and t to vertices in their respective vicinities; these distances are temporarily stored in a hash table. The algorithm then computes three set of paths. The first set of paths is of the form $s \rightsquigarrow w \rightsquigarrow w' \rightsquigarrow t$ for some $w \in B^\star(s)$ and $w' \in S_s \cap B^\star(t)$. The second set of paths are of the form $s \rightsquigarrow w \rightsquigarrow \ell(w) \rightsquigarrow t$ for vertices $w \in B^\star(s)$ and the final set of paths are of the form $t \rightsquigarrow w \rightsquigarrow \ell(w) \rightsquigarrow s$ for vertices $w \in B^\star(t)$. The least-cost path among these paths is returned by the algorithm.

Algorithm 3. The query algorithm for the third oracle of Theorem 1

1: Compute candidate distance from s to each vertex in $B^\star(s)$
2: Compute candidate distance from t to each vertex in $B^\star(t)$
3: $\gamma_1 \leftarrow \infty$, $\gamma_2 \leftarrow \infty$, $\gamma_3 \leftarrow \infty$
4: $\gamma_1 \leftarrow \min_{w \in S_s \cap B^\star(t)} \left\{ d(s,w) + d'_t(w) \right\}$
5: $\gamma_2 \leftarrow \min_{w \in B^\star(s)} \left\{ d'_s(w) + d(w, \ell(w)) + d(\ell(w), t) \right\}$
6: $\gamma_3 \leftarrow \min_{w \in B^\star(t)} \left\{ d'_t(w) + d(w, \ell(w)) + d(\ell(w), s) \right\}$
7: return $\min\{\gamma_1, \gamma_2, \gamma_3\}$

5.3 Analysis

Claim 9. *Let $P = (s, x_1, x_2, \ldots, t)$ be the shortest path between any pair of vertices s and t. Let $i_0 = \max\{i | x_i \notin P \cap B^\star(t)\}$ and $w = x_{i_0+1}$. If $w \notin B(s)$, then $d(s,t) \geq r_s + r_t$.*

Lemma 3. *Let $G = (V, E)$ be a weighted undirected graph with n vertices, m edges and maximum degree $\mu = O(m/n)$. For any fixed $1 \leq \alpha \leq n$, let L be the set of vertices sampled using the algorithm of Lemma 2. Then, $\sum_{v \in V} |S_v| \leq O(m\alpha^2)$.*

Claim 10. *Let $G = (V, E)$ be a weighted undirected graph with n vertices, m edges and maximum degree $\mu = O(m/n)$. For any fixed $1 \leq \alpha \leq n$, let L be the set of vertices sampled using Lemma 2. Then, constructing a hash table that contains, for each vertex $v \in V$, distance to each vertex in S_v can be constructed in time $O(m\alpha^2)$.*

Proof of the Third Oracle of Theorem 1. The oracle stores, in addition to the oracle of [4], a distance from each vertex v to vertices in set S_v. Using Lemma 3, it follows that the size of the oracle if $\widetilde{O}(m\alpha^2 + m + n^2/\alpha)$; for $1 \leq \alpha \leq n^{2/3}m^{-1/3}$, the size is $\widetilde{O}(m + n^2/\alpha)$ as desired. The construction of the oracle requires running a shortest path algorithm from each vertex in L and computing distances to vertices in set S_v for each vertex v. Using Lemma 2 and Claim 10, it follows that the oracle can be constructed in time $\widetilde{O}(m\alpha^2 + n^2/\alpha)$. Finally, to bound the query time, recall that the size of the vicinity of each vertex is bounded by $O(\alpha\mu)$ and a candidate distance to each vertex in the vicinity can be computed in time $O(\alpha\mu)$; the bound follows.

Let $P = (s, x_1, x_2, \ldots, t)$ be the shortest path between s and t. Let $i_0 = \max\{i | x_i \notin P \cap B^\star(t)\}$ and $w = x_{i_0+1}$; note that $x_{i_0} \notin B^\star(t)$ and hence, $w \in B^\star(t) \setminus B(t)$. If $w \in S_s$, we get that $\gamma_1 \leq d(s,w) + d'_t(w) = d(s,w) + d(t,w) = d(s,t)$, since w

lies along P; hence, the algorithm returns the exact distance. Consider the case when $w \notin S_s$. In this case, using Lemma 9, we get that $d(s,w) \geq 2\min\{r_s, r_w\}$; also $d(t,w) \geq r_t$. Since w lies along the shortest path between s and t, we get that $d(s,t) \geq 2\min\{r_s, r_w\} + r_t \geq 3\min\{r_s, r_w, r_t\}$. We now give an upper bound on the distance returned by the query algorithm. Note that $s \in B^\star(s)$ and $t \in B^\star(t)$; it follows that $\gamma_2 \leq d(s,\ell(s)) + d(\ell(s),t) \leq 2d(s,\ell(s)) + d(s,t) = 2r_s + d(s,t)$. Similarly, we get that $\gamma_3 \leq 2r_t + d(s,t)$. Finally, since $w \in B^\star(t)$, we get that $\gamma_3 \leq d'_t(w) + d(w,\ell(w)) + d(\ell(w),s)$. Since w lies along the shortest path between s and t, we get that $d'_t(w) = d(t,w)$; using this along with triangle inequality, we get that $\gamma_3 \leq d(t,w) + 2d(w,\ell(w)) + d(w,s) = 2r_w + d(s,t)$. Hence, $\gamma_3 \leq 2\min\{r_w, r_t\} + d(s,t)$. Since the algorithm returns $\min\{\gamma_2, \gamma_3\}$, the returned distance is at most $2\min\{r_s, r_t, r_w\} + d(s,t)$. The proof follows using the upper bound established above, which says that $\min\{r_s, r_t, r_w\} \leq d(s,t)/3$. □

6 Open Problems

We close the discussion with some of the most interesting open problems:

- Is it possible to prove or disprove a separation between oracles with stretch-k and stretch-less-than-k for $1 < k < 2$? In particular, do stretch-4/3 oracles require more space or time compared to stretch-3/2 oracles?
- There is an interesting problem related to improving the lower order terms in our results. Specifically, if one can reduce the query time of our algorithm of Theorem 1 (for $k = 1$, stretch-$(1 + 1/(k + 0.5))$) by $\log^c(n)$ for some large enough c, we would get a combinatorial algorithm for BMM that is asymptotically faster than the state-of-the-art [12]. Is it possible?

Finally, the most interesting open problem is to prove or disprove the existence of near-linear size oracles that compute distances of $O(1)$ stretch in polylog(n) time.

Acknowledgments. The author would like to thank Philip Brighten Godfrey and Mikkel Thorup for many helpful discussions.

References

1. Abraham, I., Gavoille, C.: On approximate distance labels and routing schemes with affine stretch. In: Peleg, D. (ed.) Distributed Computing. LNCS, vol. 6950, pp. 404–415. Springer, Heidelberg (2011)
2. Agarwal, R., Caesar, M., Godfrey, P.B., Zhao, B.Y.: Shortest paths in less than a millisecond. In: SIGCOMM WOSN (2012)
3. Agarwal, R., Godfrey, P.B.: Brief announcement: A simple stretch 2 distance oracle. In: PODC (2013)
4. Agarwal, R., Godfrey, P.B.: Distance oracles for stretch less than 2. In: SODA (2013)
5. Agarwal, R., Godfrey, P.B., Har-Peled, S.: Approximate distance queries and compact routing in sparse graphs. In: INFOCOM (2011)
6. Agarwal, R., Godfrey, P.B., Har-Peled, S.: Faster approximate distance queries and compact routing in sparse graphs (2012)
7. Alon, N., Spencer, J.H.: The probabilistic method, vol. 57. Wiley Interscience (1992)
8. Awerbuch, B., Berger, B., Cowen, L., Peleg, D.: Near-linear time construction of sparse neighborhood covers. SIAM Journal on Computing 28(1), 263–277 (1998)

9. Bansal, N., Williams, R.: Regularity lemmas and combinatorial algorithms. In: FOCS (2009)
10. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pair small stretch paths. In: FOCS (2006)
11. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. ACM Transactions on Algorithms 2(4), 557–577 (2006)
12. Blelloch, G.E., Vassilevska, V., Williams, R.: A new combinatorial approach for sparse graph problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 108–120. Springer, Heidelberg (2008)
13. Chechik, S.: Approximate distance oracles with constant query time. In: STOC (2014)
14. Chen, W., Sommer, C., Teng, S.-H., Wang, Y.: A compact routing scheme and approximate distance oracle for power-law graphs. ACM Transactions on Algorithms 9(1), 4:1–4:26 (2012)
15. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch t. SIAM Journal on Computing 28(1), 210–236 (1998)
16. Dor, D., Halperin, S., Zwick, U.: All pairs almost shortest paths. In: FOCS (1996)
17. Enachescu, M., Wang, M., Goel, A.: Reducing maximum stretch in compact routing. In: INFOCOM (2008)
18. Gubichev, A., Bedathur, S., Seufert, S., Weikum, G.: Fast and accurate estimation of shortest paths in large graphs. In: CIKM (2010)
19. Kawarabayashi, K.-I., Klein, P.N., Sommer, C.: Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 135–146. Springer, Heidelberg (2011)
20. Kawarabayashi, K.-I., Sommer, C., Thorup, M.: More compact oracles for approximate distances in undirected planar graphs. In: SODA (2013)
21. Matoušek, J.: On the distortion required for embedding finite metric spaces into normed spaces. Israel Journal of Mathematics 93(1), 333–344 (1996)
22. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. Journal of European Mathematical Society 2(9), 253–275 (2007)
23. Pătraşcu, M., Roditty, L.: Distance oracles beyond the Thorup-Zwick bound. In: FOCS (2010)
24. Pătraşcu, M., Roditty, L., Thorup, M.: A new infinity of distance oracles for sparse graphs. In: FOCS (2012)
25. Porat, E., Roditty, L.: Preprocess, set, *query!*. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 603–614. Springer, Heidelberg (2011)
26. Potamias, M., Bonchi, F., Castillo, C., Gionis, A.: Fast shortest path distance estimation in large networks. In: CIKM (2009)
27. Roditty, L., Zwick, U.: Replacement paths and k simple shortest paths in unweighted directed graphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 249–260. Springer, Heidelberg (2005)
28. Sommer, C., Verbin, E., Yu, W.: Distance oracles for sparse graphs. In: FOCS (2009)
29. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA (2001)
30. Thorup, M.: Compact oracles for reachability and approximate distances in planar digraphs. Journal of the ACM 51(6), 993–1024 (2004)
31. Thorup, M., Zwick, U.: Approximate distance oracles. Journal of the ACM 52(1), 1–24 (2005)
32. Wulff-Nilsen, C.: Approximate distance oracles with improved query time. In: SODA (2013)
33. Wulff-Nilsen, C.: Approximate distance oracles with improved preprocessing time. In: SODA (2012)