2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

# A New Infinity of Distance Oracles for Sparse Graphs

Mihai Pătraşcu
AT&T Labs—Research
Passed away June 5, 2012

Liam Roditty
Bar-Ilan University
liamr@macs.biu.ac.il

Mikkel Thorup
AT&T Labs—Research
and University of Copenhagen
mikkel2thorup@gmail.com

**Abstract**—Given a weighted undirected graph, our basic goal is to represent all pairwise distances using much less than quadratic space, such that we can estimate the distance between query vertices in constant time. We will study the inherent trade-off between space of the representation and the stretch (multiplicative approximation disallowing underestimates) of the estimates when the input graph is sparse with $m = \tilde{O}(n)$ edges.

In this paper, for any fixed positive integers $k$ and $\Delta$ we obtain stretches $\alpha = 2k + 1 \pm \frac{2}{\Delta} = 2k + 1 - \frac{2}{\Delta}, 2k + 1 + \frac{2}{\Delta}$, using space $S(\alpha, m) = \tilde{O}(m^{1+2/(\alpha+1)})$. The query time is $O(k + \Delta) = O(1)$. For integer stretches, this coincides with the previous bounds (odd stretches with $\Delta = 1$ and even stretches with $\Delta = 2$). The infinity of fractional stretches between consecutive integers are all new (even though $\Delta$ is fixed as a constant independent of the input, the number of integers $\Delta$ is still countably infinite). We will argue that the new fractional points are not just arbitrary, but that they, at least for fixed stretches below 3, provide a complete picture of the inherent trade-off between stretch and space in $m$. Consider any fixed stretch $\alpha < 3$. Based on the hardness of set intersection, we argue that if $\Delta$ is the largest integer such that $3 - 2/\Delta \leq \alpha$, then $\tilde{\Omega}(S(3 - \frac{2}{\Delta}, m))$ space is needed for stretch $\alpha$. In particular, for fixed stretch below $2\frac{2}{3}$, we improve Pătraşcu and Roditty's lower bound from $\tilde{\Omega}(m^{3/2})$ to $\tilde{\Omega}(m^{5/3})$, thus matching their upper bound for stretch 2. For space in terms of $m$, this is the first hardness matching the space of a non-trivial/sub-quadratic distance oracle.

**Keywords**-sparse graphs; distance oracles; shortest paths; distances;

## I. Introduction

A distance oracle for an undirected graph is a compact replacement for the all-pairs shortest paths matrix of a graph. We have a stretch parameter $\alpha$. Given any two vertices $v$ and $w$, a distance oracle with stretch $\alpha$ returns a distance estimate $\hat{\delta}(v, w)$ which is never less than the real distance $\delta(v, w)$ from $v$ to $w$ but at most $\alpha$ times larger, that is, $\delta(v, w) \leq \hat{\delta}(v, w) \leq \alpha\delta(v, w)$. We require that the estimate is returned in constant time. Such distance oracles compose nicely with multiplicative approximation algorithms that need access to a distance metric. We are interested in the inherent trade-offs between stretch and the space needed to store the distance oracles for sparse graphs with $m = \tilde{O}(n)$ edges. For the space we will typically not worry about logarithmic factors. Therefore we use the $\tilde{O}/\tilde{\Omega}$-notation which suppresses log-factors.

### A. State-of-the-art

The state of the art in distance oracles is best understood by reference to the results of Thorup and Zwick [TZ05] considering weighted undirected graphs with $n$ nodes and $m$ edges. For any fixed positive integer $k$, they describe a stretch $2k-1$ distance oracle of size $\tilde{O}(n^{1+1/k})$. The query time is $O(k) = O(1)$, as required.

Based on a girth conjecture of Erdős, Thorup and Zwick argue that for dense enough graphs, these are the only relevant bounds for fixed stretch $\alpha$, that is, if $k$ is the largest integer such that $2k-1 \leq \alpha$, then $\tilde{\Omega}(n^{1+1/k})$ space is needed for stretch $\alpha$. In particular, there are dense graphs such that stretch below 3 requires space near-quadratic in $n$.

Getting bounds in terms of $n$ is natural when we think of the shortest path metric as a general metric on $n$ points, but for graph algorithms, we often measure complexity in terms of $m$ which is equivalent to the input size (isolated vertices are trivial and can be handled implicitly by a dictionary over the connected vertices). We seek the best $\tilde{O}$-bounds in $m$. Such bounds are valid for all graphs, but essentially they are equivalent to bounds in terms of $n$ for sparse graphs with $m = \tilde{O}(n)$. The trivial point is that if a graph is not sparse, we can always add $m$ isolated edges with $2m$ new vertices.

Sparse graphs with $m = O(n)$ are important. This includes mathematically defined graph models like bounded degree graphs and planar graphs, but it is also common in practice in networks where links are expensive. We may, for example, have a few high degree hub vertices, but low average degree. We note that planar graphs have a special structure that admits much more efficient distance oracles (see, e.g., [Tho04]), but many sparse graphs do not have such special structure. For $m = \tilde{O}(n)$, the space of the Thorup-Zwick distance oracles is $\tilde{\Theta}(m^{1+1/k})$.

The lower bounds of Thorup and Zwick are incompressibility bounds and can never get higher than $m$. However, Sommer et al. [SVY09] proved in the cell-probe model that there are sparse graphs such that constant stretch and query time requires space $m^{1+\Omega(1)}$. With current cell-probe techniques we cannot hope for a more specific lower-bound trade-off, e.g., we do not have any asymptotic separation in query time between space $\Theta(m^{1.01})$ and $\Theta(m^{100})$ for any static data structure problem with instances of size $m$.

Pătraşcu and Roditty [PR10] showed that a stretch below

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.44

3 is possible with space that is substantially subquadratic in $m$. They obtained stretch 2 using space $\tilde{O}(m^{5/3})$. Based on a conjectured hardness of set intersection queries, they get that space $\tilde{\Omega}(m^2)$ is needed for stretch below 2 and that space $\tilde{\Omega}(m^{3/2})$ is needed for stretch below $2\frac{1}{3}$. This leaves open a polynomial gap for stretch 2. Abraham and Gavoille [AG11] implicitly (we shall shortly discuss what they did explicitly) generalized the stretch 2 oracle to all even stretches. With stretch $2k$, the space becomes $\tilde{O}(m^{1+1/(k+1/2)})$.

Other related work: The distance oracles of Thorup and Zwick have inspired a broad array of loosely related results, focusing on such issues like construction time [BS06], [BK06], [BGSU08], the query time [MN07] and space bounds for random graphs [KFY04], [CSTW09], [EWG08] and derandomization [RTZ05]. These efforts do not lead to better space / stretch tradeoffs.

The above mentioned paper [PR10] of Pătraşcu and Roditty contains several other results. For the stretch 2 distance oracle, they present the mixed bound $\tilde{O}(n^{4/3}m^{1/3}) = \tilde{O}(m^{5/3})$. For unweighted graphs they also show that if we add one to the distance estimate, that is, $\hat{\delta}(u,v) \leq 2\delta(u,v) + 1$, then space $\tilde{O}(n^{5/3})$ is possible. In fact, they start with this simpler multiplicative-additive version for unweighted graphs before they get the clean stretch 2 for weighted graphs. Abraham and Gavoille [AG11] continued with the multiplicative-additive version, showing how to generalize it to estimates $\hat{\delta}(u,v) \leq 2k\delta(u,v) + 1$, using space $\tilde{O}(n^{1+1/(k+1/2)})$, and using it for labeling and routing schemes. They say they will handle the weighted case in the full version. Indeed it is a simple translation of their generalization to get stretch $2k$ with space $\tilde{O}(m^{1+1/(k+1/2)})$, as claimed above.

## B. Our results

In this paper we focus on pure multiplicative stretches with no extra additive terms. This means that we are not restricted to unweighted graphs. A purely multiplicative stretch also means that our distance oracles can be used naturally in multiplicative approximation algorithms that need access to a distance matrix. Our focus is to understand the space bounds purely in terms of $m$.

First we observe that for all the integer stretches $\alpha$, the current space bounds fall on a smooth curve:

$$S(\alpha, m) = \tilde{O}(m^{1+2/(\alpha+1)})$$

This includes both the odd stretches from [TZ05] and the even stretches from [PR10], [AG11]. For arbitrary (non-integer stretches) $\alpha$, the best known space was $S(\lfloor\alpha\rfloor, m)$.

1) Upper bounds: In this paper, we add an infinity of fractional stretches between consecutive integers; namely stretches of the form $\alpha = 2k + 1 \pm \frac{2}{\Delta}$ ($\alpha = 2k + 1 - \frac{2}{\Delta}$ or $\alpha = 2k + 1 + \frac{2}{\Delta}$) for fixed integers $k$ and $\Delta$.

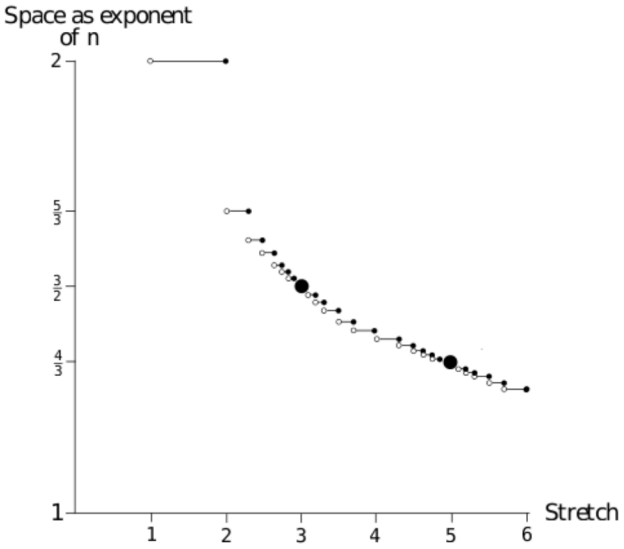


Figure 1. The step function. The big dots denote points of Thorup and Zwick on odd stretches to which other stretches converge.

**Theorem 1.** *Fix positive integers $k$ and $\Delta$. For any weighted graph with $n$ vertices and $m$ edges, there is a distance oracle with stretch $\alpha_- = 2k + 1 - \frac{2}{\Delta}$ using space*

$$\tilde{O}((m^{3+k}/n^{2+k})^{1/(k+2-1/\Delta)}n^2/m) = \tilde{O}(m^{1+1/(k-1/\Delta)}) = S(\alpha_-, m) \quad (1)$$

*and a distance oracle with stretch $\alpha_+ = 2k - 1 + \frac{2}{\Delta}$ using space*

$$\tilde{O}((m^{3+k}/n^{2+k})^{1/(k+2+1/\Delta)}n^2/m) = \tilde{O}(m^{1+1/(k+1/\Delta)}) = S(\alpha_+, m). \quad (2)$$

*The query time in both cases is $O(k + \Delta) = O(1)$.*

See Figure I-B for an illustration of the step function of space stretch curve. Concerning the mixed bounds involving both $m$ and $n$, we note that for dense enough graphs, we get better results with the Thorup-Zwick oracles. Our focus here are bounds purely in $m$.

Like in [PR10], for the case of unweighted graphs, we could replace $m$ with $n$ in the space bounds accepting an extra additive 1 in the estimates, but this is not our focus here.

For any $\alpha$ and any set $A$ of reals, let $\lfloor\alpha\rfloor_A = \max\{a \leq \alpha | a \in A\}$. For fixed stretch $\alpha$, the space offered by Theorem 1 follows the step function

$$S(\lfloor\alpha\rfloor_{\{2k+1\pm\frac{2}{\Delta}|k,\Delta\in\mathbb{N}_{>0}\}}, m) = \tilde{O}(m^{1+2/(\Delta\lfloor\alpha\rfloor_{\{2k+1\pm\frac{2}{\Delta}|k,\Delta\in\mathbb{N}_{>0}\}}+1)}). \quad (3)$$

Formally we note here that when $\alpha$ is fixed as a constant independent of the input, then so $k$ and $\Delta$ and therefore $O(k+\Delta) = O(1)$.

Some obvious questions remain:

- Is it a coincidence that all the best known distance oracles are on the space curve $S$, or is it possible to do polynomially better than $S$ for any stretch? The conditional lower bound from [PR10] for stretch 2 is only $\tilde{\Omega}(m^{3/2}) \blacklozenge S(2,m) = \tilde{O}(m^{5/3})$.
- If not, can we continuously match $S$ for all stretches?
- If not, is the step function behavior inherent, or are there distance oracles outside the curve that are not dominated by any distance oracles on the curve?

2) Hardness: Recall that the lower bounds of Thorup and Zwick [TZ05] are incompressability bounds that can never go higher than $m$. Also, concerning unconditional cell-probe lower bounds, recall that we do not have any asymptotic separation in query time between space $\Theta(m^{1.01})$ and $\Theta(m^{100})$ for any static data structure problem with instances of size $m$, so we cannot hope to get any remotely tight cell-probe understanding of the stretch-space trade-off.

As with NP-hardness for off-line problems, we resort to hardness with respect to the conjectured hardness of a core problem. Following the lead of [PR10] we relate to the presumed hardness of set intersection. We will argue that our step function (3) cannot be improved for any fixed stretch below 3. In particular, this implies that we improve the $\tilde{\Omega}(m^{3/2})$ lower bound from [PR10] for stretch 2 to match the upper bound of $S(2,m) = \tilde{O}(m^{5/3})$. For space in terms of $m$, this is the first hardness matching the space of a non-trivial/sub-quadratic distance oracle. The only tight hardness from [PR10] was the quadratic space for stretch strictly below 2.

Hardness by set intersection for stretch below 3: We first set up our hardness assumption about intersection queries. Consider the static data structure problem:

instance for preprocessing: The construction algorithm receives the $n$ sets $S_1, \ldots, S_n \subseteq [u]$. In a regular instance, for some set size parameter $s \leq u$, each set has size at most $s$, and each element appears in at most $ns/u$ sets.

query: for given $(i,j) \in [n]^2$, the boolean query is whether $S_i$ intersects $S_j$.

The two obvious solutions are to either store all the (positive) answers in the preprocessing phase, or to simply store the sets directly and intersect them during the query. A popular belief consistent with all current upper bound ideas is that in general there is no smooth trade-off between these two extreme types of solutions. Pătraşcu and Roditty [PR10, Conjecture 7] considered a polylogarithmic universe with no particular regularity constraint. The conjecture they used was that for some large enough constants $a$ and $b$, if we have a data structure that represents any family of $n$ input sets over $[u]$, $u = \bullet\log^a n\circ$ using $O(n^2/\log^b n)$ space, and answers any intersection query in constant time, then there is a family of input sets such that 1/12 of the all $\binom{n}{2}$ intersection queries are answered falsely.

For our stronger and more general lower bounds, we consider regular instances with an arbitrary universe size $u$ and a polylogarithmic set size $s = \bullet\log^a n\circ$. Each element appears in at most $ns/u$ sets, and is hence witnessing at less than $(ns/u)^2$ set intersections. It follows that we have at most $(ns)^2/u$ positive set intersections in total, and in fact, we expect $\Theta((ns)^2/u)$ intersection for a random instance. With a hash table we can store all positive answers in $O((ns)^2/u)$ space, and the basic conjecture is that this is best possible for constant query time. For $u \to s^2$, we get almost all queries right by answering "no". To get a plausible lower bound with larger universes, we therefore have to require that there are no false negatives.

Conjecture 2. Let $a$ and $b$ be sufficiently large constants. Consider regular set intersection instances with $n$ sets, universe size $u$, and set size $s = \bullet\log^a n\circ$. If a data structure with constant query time uses only $O(n^2/(u\log^b n))$ space and makes no false negatives, then for some set intersection instance, the fraction of false positives is $\Omega(1)$ over all $\binom{n}{2}$ possible queries.

Even though this conjecture does not have any explicit step function in it, we use it to prove that our step function (3) cannot be improved for any fixed stretch below 3

Theorem 3. Under Conjecture 2, for any fixed positive integer $\Delta$, that are graphs with $m$ edges such that a distance oracle with constant query time and stretch below $3-2/(\Delta+1)$ must use space $\tilde{\Omega}(S(3-2/\Delta, m)) = \tilde{\Omega}(m^{1+1/(2-1/\Delta)})$.

We note here that the point of a hardness assumption is to help us identify the limits of what can be achieved, leaving us with a single hard core problem capturing what has to be done for further progress to be made. The concept of NP-hardness has taken us far in this direction for off-line problems, but for the understanding of approximability, the stronger Unique Games conjecture has helped us to close the gap between what we can do, and what we believe to be hard. Here we suggest using the presumed hardness of set intersection in the case of larger universes and regular instances, formulating a more flexible conjecture than [PR10, Conjecture 7], and use it to get tight space bounds for all fixed stretches below 3.

To argue hardness of stretch below $3-2/(\Delta-1)$, we will use the following combinatorial reduction.

Theorem 4. Fix an integer $\Delta > 1$. Consider a regular set intersection instance such that $u = s^6 n^{1-1/(2-1/\Delta)}/f$ where $f = o(1)$. We can construct an unweighted graph $G$ with $m = O(sn\log n)$ edges and $n$ sources and sinks such that if sets $S_i$ and $S_j$ intersect, then $\delta(i,j) = \Delta+1$. Moreover, among all source-sink pairs $(i,j) \in [n]^2$, there is only a fraction $O(f)$ with $\delta(i,j) < 3\Delta+1 = (\Delta+1)(3-2/(\Delta+1))$.

Proof that Theorem 4 implies Theorem 3: Given a distance oracle with stretch below $3-2/(\Delta+1)$ for $G$,

we interpret $\hat{\delta}(i,j) < 3\Delta+1$ as $S_i \cap S_j \neq \emptyset$ yielding at most a fraction $O(f) = o(1)$ of false positives for the set intersection problem. By Conjecture 2, if the queries take constant time, the space needed is $\tilde{\Omega}(n^2/u) = \tilde{\Omega}(n^2/(s^6 n^{1-1/(2-1/\Delta)}/f) = \tilde{\Omega}(n^{1+1/(2-1/\Delta)})$. ∎

Mixed hardness for larger stretches: We note that the mixed bounds from (1) in Theorem 1 cannot be improved in general. Recall from [TZ05] that assuming Erdős' girth conjecture, for any integer $k$, there are families of graphs with $m = \Theta(n^{1+1/(k+1)})$ edges that require $\Omega(m)$ bits of representation for any stretch strictly below $2k+3$. With such $m$ and $k$, we will argue that our mixed bound from (1) cannot be improved polynomially for large $\Delta$. The point is simply that for $m = n^{1+1/(k+1)}$ and stretch $2k+3-2/\Delta$ when $\Delta \to \infty$, we get the bound

$$\tilde{\Omega}((m^{3+k}/n^{2+k})^{1/(k+2-1/\Delta)} n^2/m) =$$
$$\tilde{\Omega}(((n^{1+1/(k+1)})^{3+k}/n^{2+k})^{1/(k+2-1/\Delta)} n^2/n^{1+1/(k+1)}) =$$
$$\tilde{\Omega}((n^{(4+2k)/(k+1)})^{1/(k+2)} n^{1-1/(k+1)}) =$$
$$\tilde{\Omega}(n^{2/(k+1)} n^{1-1/(k+1)}) = \tilde{\Omega}(n^{1+1/(k+1)}).$$

which is best possible.

3) Techniques: In the area of distance oracles, many things are simple when viewed the right way. Using hardness of set intersection for larger universe sizes gives us not only a tight hardness for stretch 2, but also lower bounds matching the step function (3) for all fixed stretches below 3. Technically the construction is less ad-hoc than the one used for the weaker lower bound in [PR10]. Moreover, it was the steps in the lower bounds, that lead us to look for the fractional stretch distance oracles in Theorem 1. A moral difference between our oracles and the previous ones from [PR10], [TZ05] is that the previous oracles aim for a case of a small set $S$ of vertices such that $\delta(u,S)$ is small. Here, more generally, we operate with two vertex sets $R$ and $T$ where the product $|R||T|$ of the sizes is small and sums like $\delta(u,R)+\delta(v,T)$ and $\delta(u,R)+\delta(u,T)$ are small. The more appealing aspect of our techniques is that upper and lower bounds match in many places. We see this as an indication that (3) is the optimal trade-off between stretch and space in terms of $m$.

Notations: Let $G = (V,E)$ be an undirected graph. All our graphs are weighted and we assume weights and distances fit in machine words (all our algorithms use only additions and comparisons of weights). For every $v \in V$ and every set $X \subseteq V$ let $p(v,X)$ be the vertex that is closest to $v$ among the vertices of $X$. Let $u,w \in V$. We denote with $\delta(u,w)$ the length of the shortest path (distance) between $u$ and $w$. We denote with $\delta(v,X)$ the distance between $v$ and its closest vertex in $X$, that is, $\delta(v,X) = \delta(v,p(v,X))$.

## II. Distance oracles with $3+2/\Delta$ stretch

In this section we prove Theorem 1 for $k=1$. We start by describing a general technique for growing balls which is the backbone of the new distance oracles. Next, we show that for certain vertex pairs it is possible to store the exact distance. Then, we show how to maintain estimated distances for all other vertex pairs. We end by showing how to combine all these ideas in order to prove Theorem 1 for $k=1$.

### A. Growing balls

Algorithm 1: Ball($v, c(\cdot), s$)

```
H ← {v}; // H is a priority queue with tentative distances
B ← ∅
while H is not empty do
    u ← extract-min(H);
    foreach (u,w) ∈ E ∧ (u,w) ∉ B do
        if c((u,w)) ≤ s then
            B ← B ∪ {(u,w)};
            s ← s − c((u,w));
            relax(u,w);
        else
            return B;
return B;
```

The algorithm for growing a ball around a vertex is given a total budget and a cost function that assigns edge costs. Edges are added to the ball by scanning vertices in order of increasing distance breaking ties arbitrarily. When we scan the edges of a certain vertex we add its incident edges one by one as long as we have enough budget. We reduce from the budget the cost of each edge that we add. All the edges of the current vertex have to be added before we can move to the next vertex and scan its edges. If we reach to an edge that is too costly for our budget we stop and the current set of edges is the ball that we output. We denote the ball of $v$ with $B(v)$. The algorithm for computing a ball is given in Algorithm 1. It is important to note that edges have both a weight and a cost which are two different values.

The radius $\mathrm{rad}(B(v))$ of a ball is the distance to the last vertex for which the algorithm has started an edge scanning. A vertex $w$ is properly contained in $B(v)$ if $\delta(v,w) < \mathrm{rad}(B(v))$. In such a case $B(v)$ contains all edges incident to $w$. We say that two balls $B(u)$ and $B(w)$ intersect if and only if $\mathrm{rad}(B(u)) + \mathrm{rad}(B(w)) > \delta(u,w)$. With a slight abuse of notations we assume that $B(v)$ contains the vertex $v$ and distances to all properly contained vertices in $B(v)$.

We now turn to describe how we use this algorithm for growing balls with different cost functions to create a set of balls around each vertex. These balls are then used in order to maintain distances. Let $G = (V,E)$ be an weighted undirected graph, where $|V| = n$ and $|E| = m$. For technical reasons, we add $m/n$ loops to each vertex so we have $m' =$

$2m$ edges. Replacing $m$ with $m'$, we obtain that every vertex has at least $m/(2n)$ incident edges.

For $i = 0, \ldots, k$ we will define increasing balls $B_i(v)$ around each vertex $v$. We set $B_0(v) = v$ and $B_k(v) = Ball(v, \bar{1}, q^k)$, where $\bar{1}$ is a cost function that assigns each edge a cost of 1. We assume also that $B_k(v)$ stores the distance to all vertices $w$ that are properly contained in $B_k(v)$, that is, $\delta(v,w) < rad(B_k(v))$. Since the smallest vertex degree is $m/(2n)$ the number of distances stored with each vertex is at most $q^k 4n/m = O(q^k \frac{n}{m})$.

The balls for $i = 1, \ldots, k-1$, are created in two phases. In the first phase we set $A_i(u) = Ball(u, \bar{1}, q^i)$. We then use these balls to define a cost function as follows:

$$\forall e \in E,\ \varphi_i(e) = 1 + \#\{w : e \in A_{k-i}(w)\}\frac{m}{nq^{k-i}}.$$

In the second phase we use the cost function to create balls. We set $B_i(v) = Ball(v, \varphi_i, q^i)$. We denote with $\bar{B}_i(v)$ the ball with the edge that was scanned and not added to $B_i(v)$ due to lack of budget, that is, the edge in which we stopped growing $B_i(v)$. Assuming that for each ball grown from $v$ we break ties in the same way then for every $v \in V$ we have $B_i(v) \subseteq A_i(v)$ as the cost of every edge is at least 1.

Also, for consistency, we note

Lemma 5. For all $i \in \{0, \ldots, k-1\}$, $B_i(v) \subseteq B_{i+1}(v)$.

Proof: For $i = 0$ we have $B_0(v) = \{v\}$ and since $v \in B_i(v)$ for every $i \in \{1, \ldots k\}$ it follows that $B_0(v) \subseteq B_1(v)$. For $i = k-1$ we have $B_{k-1}(v) \subseteq A_{k-1}(v)$ and since $A_{k-1}(v) \subseteq B_k(v)$ we get that $B_{k-1}(v) \subseteq B_k(v)$.

Assume now that $i \in \{1, \ldots, k-2\}$. Recall that $B_i(v) = Ball(v, \varphi_i, q^i)$ and $B_{i+1}(v) = Ball(v, \varphi_{i+1}, q^{i+1})$, thus, if for every $e$ it holds that $\varphi_i(e) \geq \varphi_{i+1}(e)/q$ then any edge of $B_i(v)$ is also in $B_{i+1}(v)$ and the claim follows.

We now show that $\varphi_i(e) \geq \varphi_{i+1}(e)/q$. Let $\#(e, A_j) = \#\{w : e \in A_j(w)\}$

$$\varphi_i(e) \geq \varphi_{i+1}(e)/q \Leftarrow\Rightarrow$$
$$\#(e, A_{k-i})\frac{m}{nq^{k-i}} \geq \#(e, A_{k-(i+1)})\frac{m}{nq^{k-(i+1)}}/q$$
$$\Leftarrow\Rightarrow \#(e, A_{k-i}) \geq \#(e, A_{k-(i+1)})$$

The last inequality holds as $A_{k-i}(w) \supseteq A_{k-(i+1)}(w)$. ■

## B. Intersecting balls

We now analyze the space that is needed in order to maintain exact distances between every pair of vertices $v, w$ that for some $i \in \{0, \ldots, k\}$ their balls $B_i(v)$ and $B_{k-i}(w)$ intersect. We start by showing that any two balls with non-zero radius that intersect have a common edge.

Lemma 6. If two balls $B(u)$ and $B(w)$, both with non-zero radius, intersect then they must have a common edge.

Proof: As $rad(B(u)) + rad(B(w)) > \delta(u,w)$ there must be a value $x$ such that $x < rad(B(u))$ and $\delta(u,w) - x < rad(B(w))$. Let $(v, v')$ be an edge on the shortest path between $u$ and $w$, such that $\delta(u,v) \leq x$ and $\delta(u,v') > x$. It follows that $\delta(v', w) < \delta(u,w) - x$. We get that $\delta(u,v) < rad(B(u))$ and $\delta(v', w) < rad(B(w))$, hence, $v$ is properly contained in $B(u)$ and $v'$ is properly contained in $B(w)$ and the edge $(v, v')$ is in $B(u) \cap B(w)$. ■

We will store with $B_i(v)$ the distances to all vertices $w \neq v$ such that $B_{k-i}(w)$ intersects $B_i(v)$. The distance between such a pair of vertices $v$ and $w$ is stored in a hash table. We will, however, ignore the case where one of the balls have radius 0; for if $B_{k-i}(w)$ has radius 0 and intersects $B_i(v)$, then $w$ is properly contained in $B_i(v) \subseteq A_i(v) \subseteq B_k(v)$, and then the distance is already stored with $B_k(v)$. Thus we only save the distance if $B_i(v)$ and $B_{k-i}(w)$ intersect and have non zero radius.

Lemma 7. The total space for storing distances between every pair of balls $B_i(v)$ and $B_{k-i}(w)$ that intersect is $O(q^k \frac{n^2}{m})$.

Proof: Given a vertex $v \in V$ and a ball $B_i(v)$ with non-zero radius we save distances to all vertices $w$ whose ball $B_{k-i}(w)$ intersects $B_i(v)$ and has a non-zero radius. From Lemma 6 it follows that in such a case $B_i(v)$ and $B_{k-i}(w)$ share a common edge. Thus, the space that we charge with ball $B_i(v)$ for saving distances is $\sum_{e \in B_i(v)} \#\{w : e \in B_{k-i}(w)\}$. We now bound $\sum_{e \in B_i(v)} \#\{w : e \in B_{k-i}(w)\}$. Recall that $B_i(v)$ was grown with budget $q^i$ and cost function $\varphi_i$, hence,

$$q^i \geq \sum_{e \in B_i(v)} \varphi_i(e) \quad (4)$$
$$\geq \sum_{e \in B_i(v)} \left(1 + \#\{w : e \in A_{k-i}(w)\}\frac{m}{nq^{k-i}}\right) \quad (5)$$
$$\geq \sum_{e \in B_i(v)} \#\{w : e \in B_{k-i}(w)\}\frac{m}{nq^{k-i}} \quad (6)$$

It follows that: $\sum_{e \in B_i(v)} \#\{w : e \in B_{k-i}(w)\} \leq q^k \frac{n}{m}$. ■

## C. Separated balls

For every $u \in V$ we have the set of balls $B_0(u), B_1(u), \ldots, B_k(u)$ as before. We now bound the distance between certain balls of two vertices $u$ and $w$, such that $B_i(u)$ and $B_{k-i}(w)$ do not intersect for every $i \in \{0, \ldots, k\}$.

For simplicity, we normalize distances so that $\delta(u,w) = 1$ and let $a_i = rad(B_i(u))$ and $b_i = rad(B_i(w))$. Since $B_i(u)$ and $B_{k-i}(w)$ do not intersect, we have $a_i + b_{k-i} \leq 1$ for all $i$. We can find non-negative real values $0 = x_0, x_1 \ldots, x_k = 1$ such that $a_i \leq x_i$ and $b_i \leq 1 - x_{k-i}$.

Lemma 8. There exists $i \in \{0,\ldots,\Delta-1\}$ such that,

$$a_i + b_{\Delta-i-1} \leq 1 - 1/\Delta \tag{7}$$

Moreover, for some $c \in \{a,b\}$ and $i \in \{0,\ldots,\Delta-1\}$,

$$c_i + c_{\Delta-i-1} \leq 1 - 1/\Delta \tag{8}$$

Proof: The proof follows from a simple averaging argument. We have:

$$\sum_{i=0}^{\Delta-1}(a_i+b_{\Delta-i-1}) \leq \sum_{i=0}^{\Delta-1}(1-(x_{i+1}-x_i)) = \Delta-(x_\Delta-x_0) = \Delta-1.$$

As there are $\Delta$ values and their total sum is $\Delta-1$ there must be an $i$ such that $(a_i + b_{\Delta-i-1}) \leq 1 - 1/\Delta$ and Equation (7) follows.

We now turn to prove the second part of the Lemma. Notice that $\sum_{i=0}^{\Delta-1}(b_i + a_{\Delta-i-1}) = \sum_{i=0}^{\Delta-1}(a_i + b_{\Delta-i-1})$.

$$2(\Delta-1) = \sum_{i=0}^{\Delta-1}(a_i + b_{\Delta-i-1}) + \sum_{i=0}^{\Delta-1}(b_i + a_{\Delta-i-1}) =$$

$$\sum_{i=0}^{\Delta-1}(a_i + a_{\Delta-i-1}) + \sum_{i=0}^{\Delta-1}(b_i + b_{\Delta-i-1}).$$

Again it follows from averaging that for some $c \in \{a,b\}$ and $i \in \{0,\ldots,\Delta-1\}$, $c_i + c_{\Delta-i-1} \leq 1 - 1/\Delta$ and Equation (8) follows. ■

Notice that simply by looking at the values $a_i$ and $b_i$ for every $i \in \{0,\ldots,\Delta-1\}$, we can find the value of $i$ that minimizes $a_i + b_{\Delta-i-1}$. Then we know that $a_i + b_{\Delta-i-1} \leq 1 - 1/\Delta$.

Similarly to Lemma 8, we have

Lemma 9. There exists $i \in \{0,\ldots,\Delta-1\}$ such that,

$$a_{i+1} + b_{\Delta-i} \leq 1 + 1/\Delta \tag{9}$$

Moreover, for some $c \in \{a,b\}$ and $i \in \{0,\ldots,\Delta-1\}$,

$$c_{i+1} + c_{\Delta-i} \leq 1 + 1/\Delta \tag{10}$$

Proof: As before, the proof follows from a simple averaging argument. ■

We conclude that for every two vertices $u$ and $w$ for which $B_i(u)$ and $B_{\Delta-i}(w)$ do not intersect for every $i \in \{0,\ldots,\Delta\}$ the above two Lemmas provide a bound for some $i$ on the distance between any pair of vertices one from $B_i(u)$ and one from $B_{\Delta-i-1}(w)$, in the case of Lemma 8, and one from $B_{i+1}(u)$ and one from $B_{\Delta-i}(w)$, in the case of Lemma 9. Thus, to maintain a distance estimation we just need to store one vertex from each ball. For this we use standard hitting set techniques:

Lemma 10. For every $i \in \{0,\ldots,\Delta\}$, there is a set $S_i$ of size $O(m/q^i \log n)$ vertices so that for every vertex $v \in V$, $\delta(v,S_i) \leq \mathrm{rad}(B_i(v))$.

Proof: We will sample edges $(u,w)$ and include both $u$ and $w$ in $S_i$. Then our condition is satisfied if we sample any edge from $\bar{B}_i$. The total weight of edges in $\bar{B}_i$ is at least $q^i$. The cases of $i = 0, j$ are trivial. For $i = 1,\ldots,j-1$, our condition is satisfied with high probability for every $v \in V$ if we sample each edge with probability $\min\{1, O((\varphi_i(e)/q^i)\log n)\}$. Now note that

$$\begin{aligned}\sum_e \varphi_i(e) &= m + \sum_e \#\{w : e \in A_{\Delta-i}(w)\}\frac{m}{nq^{\Delta-i}} \\ &= m + \sum_w \sum_{e} \#\{e \in A_{\Delta-i}(w)\}\frac{m}{nq^{\Delta-i}} \\ &\leq m + nq^{\Delta-i}\frac{m}{nq^{\Delta-i}} = 2m\end{aligned}$$

The expected size of $S_i$ is thus $O((m/q^i)\log n)$. We can easily derandomize the construction to find each $S_i$ deterministically. ■

D. Base case of Theorem 1

We will now prove the base case of Theorem 1 with $k = 1$, that is, stretch $3 \pm 2/\Delta$.

Stretch $3 - 2/\Delta$. For every vertex $v \in V$ we construct the balls $B_0(v),\ldots,B_\Delta(v)$ using the ball growing technique described in Section II-A. For every $i \in \{0,\ldots,\Delta-1\}$, we store all distances from $S_i \times S_{\Delta-i-1}$ as defined in Lemma 10. The number of distances stored is:

$$O(m/q^i \log n) \times O(m/q^{\Delta-i-1}\log n) = \tilde{O}(m^2/q^{\Delta-1}).$$

For each vertex $v$ and each $i$, let $c_i(v) = p(v,S_i)$. We also store the nearest vertex $c_i(v) \in S_i$ and $\delta(v,c_i(v))$.

The query works as follows. Given $u,w \in V$ we first check in the hash table if there is $i \in \{0,\ldots,\Delta\}$ such that $B_i(u)$ and $B_{\Delta-i}(w)$ intersect. In such a case the exact distance is stored and we output it. Otherwise, there is not any $i \in \{0,\ldots,\Delta\}$ such that $B_i(u)$ and $B_{\Delta-i}(w)$ intersect and we are in the case of separated balls as in Section II-C, where we do not stored the exact distance.

We now look for $i \in \{0,\ldots,\Delta-1\}$ that minimizes $\mathrm{rad}(B_i(u)) + \mathrm{rad}(B_{\Delta-i-1}(w))$. As an estimation, we use

$$\delta(u,c_i(u)) + \delta(c_i(u),c_{\Delta-i-1}(w)) + \delta(w,c_{\Delta-i-1}(w)).$$

This estimation was produced in $O(\Delta)$ time. From the triangle inequality it follows that:

$$\delta(c_i(u),c_{\Delta-i-1}(w)) \leq \delta(u,c_i(u)) + \delta(u,w) + \delta(w,c_{\Delta-i-1}(w)).$$

Hence, our estimation is bounded by $2(\delta(u,c_i(u)) + \delta(w,c_{\Delta-i-1}(w))) + \delta(u,w)$.

From Lemma 8 we have $\mathrm{rad}(B_i(u)) + \mathrm{rad}(B_{\Delta-i-1}(w)) \leq (1-1/\Delta)\delta(u,w)$. Moreover, $\delta(u,c_i(u)) \leq \mathrm{rad}(B_i(u))$ and $\delta(w,c_{\Delta-i-1}(w)) \leq \mathrm{rad}(B_{\Delta-i-1}(w))$. We get that

$$2((\delta(u,c_i(u)) + \delta(w,c_{\Delta-i-1}(w))) + \delta(u,w) \leq (3-2/\Delta)\delta(u,w).$$

To minimize the space we need to consider also the $O(q\Delta\frac{n^2}{m})$ space needed in order to store distances in the hash table for vertices with intersecting balls. To balance, we set

$$m^2/q^{\Delta-1} = q^{\Delta}\frac{n^2}{m} \iff q = \left(\frac{m^3}{n^2}\right)^{1/(2\Delta-1)},$$

and then the total space is

$$O\left(\frac{n^2}{m}\left(\frac{m^3}{n^2}\right)^{1/(2-1/\Delta)}\right) = O\left(m^{1+1/(2-1/\Delta)}\right).$$

Stretch $3+2/\Delta$. For every $i \in \{0,\ldots,\Delta-1\}$, we store all distances from $S_{i+1} \times S_{\Delta-i}$ as defined in Lemma 10. The number of distances stored is:

$$O(m/q^{i+1}\log n) \times O(m/q^{\Delta-i}\log n) = \tilde{O}(m^2/q^{\Delta+1}).$$

As before given two vertices $u$ and $w$ we check in the hash table to find $i$ such that $B_i(u)$ and $B_{\Delta-i}(w)$ intersect and we have an exact distance. If there is no such $i$ we search for $i$ such that $\delta(u,c_{i+1}(u)) + \delta(w,c_{\Delta-i}(w)) \le 1 + 1/\Delta$ which we know that exists from Lemma 9 and use it to estimate the distance with stretch $3+2/\Delta$.

Setting $q$ to balance the space with $O(q\Delta\frac{n^2}{m})$, we end up with a total space of

$$O\left(\frac{n^2}{m}\left(\frac{m^3}{n^2}\right)^{1/(2+1/\Delta)}\right) = O\left(m^{1+1/(2+1/\Delta)}\right).$$

## III. Distance oracle with stretch $2k+1\pm 2/\Delta$

In this section we prove Theorem 1 for $k \ge 1$. For our construction it is more convenient to subtract one from $k$, that is, for integers $k \ge 0$ and $\Delta \ge 1$, we will get distance oracles with stretch $2k+3\pm 2/\Delta$ and space $\tilde{O}(m^{1+1/(k+2\pm 1/\Delta)})$. We start by presenting a generalized implementation of Thorup—Zwick distance oracle.

### A. Generalizing the Thorup-Zwick-construction

We will now present a generalization of the Thorup-Zwick construction [TZ05] that is needed in order to obtain our general distance oracle. We start by a quick overview of Thorup and Zwick distance oracle. In the construction phase they create a sequence of vertex sets $A_0 \supseteq A_1 \supseteq \cdots \supseteq A_{k-1}$, where $A_0 = V$ and every $A_i$, for $1 \le i < k$, is obtained by placing each element of $A_{i-1}$ in $A_i$, independently, with probability $n^{-1/k}$. The expected size of $A_i$ is $n^{1-i/k}$. For every vertex $u$ and every $0 \le i \le k-1$ they define a ball $B_i(u)$ (in their terminology they called it a bunch) to be all vertices $w \in A_i$ such that $\delta(u,w) < \delta(u,p(u,A_{i+1}))$ (for $k$ we have $\delta(u,p(u,A_k)) = \infty$). Each vertex saves the distance to all the vertices in its balls. The expected size of each ball $B_i(v)$ for every $0 \le i < k-1$ is $O(n^{1/k})$. The ball $B_{k-1}(v) = A_{k-1}$ and hence its expected size is $O(n^{1/k})$ as well. The total size of the distance oracle is $O(n^{1+1/k})$.

For $u,w \in V$ an estimated distance is computed as follows. We run a loop for $i = 0,\ldots,k-1$. We have $v_i = u$ for even $i$ and $v_i = w$ for odd $i$. Let $\delta(u,w) = \Delta$ and let $p_i = p(v_i,A_i)$. We search for the smallest $i$ for which $p_i \in B_i(v_{i+1})$. In such a case we return $\delta(v_i,p_i) + \delta(p_i,v_{i+1})$ as these distances are saved in $B_i(v_{i+1})$. Such an $i$ must exists as $B_{k-1}(v) = A_{k-1}$ for every $v$. Thorup and Zwick showed that $\delta(v_i,p_i) \le i\Delta$ whenever $p_{i-1} \in B_{i-1}(v_i)$. Thus, for the smallest $i$ for which $p_i \in B_i(v_{i+1})$ we have $\delta(p_i,v_{i+1}) \le \delta(v_i,p_i) + \Delta \le (i+1)\Delta$ by the triangle inequality. We get $\delta(v_i,p_i) + \delta(p_i,v_{i+1}) \le (2i+1)\Delta$ and as $i \le k-1$ this gives $2k-1$ stretch.

The generalization of the Thorup-Zwick distance oracle by Abraham and Gavoille [AG11] was to let $A_0$ start as subset of $V$ with a good bound on $\delta(u,A_0)$, but here we need a stronger generalization. The input to the constriction algorithm is an integer parameter $k$ as before and three additional parameters. Two sets of vertices $R$ and $T$ and a size parameter $r$. The distance oracle will use $O(nr + |R|\cdot|T|/r^k)$ space and for two vertices $u,w \in V$ it will report an estimation bounded by

$$2(\delta(u,R) + \delta(v,T)) + (2k+1)\delta(u,w),$$

where $v = w$ for even $k$ and $v = u$ for odd $k$.

The construction works as follows. As in Thorup-Zwick distance oracle we create a sequence of vertex sets but with a base set $R$ and not $V$, that is, $R = R_0$ and $R_i$ is obtained from $R_{i-1}$ by placing each element of $R_{i-1}$ in $R_i$, independently, with probability $r^{-1}$. The sequence is composed of $k+1$ sets, $R = R_0, R_1, \ldots, R_k$. The expected size of $R_i$ is $|R|/r^i$. For every $0 \le i < k$ and for every $u \in V$ we have a ball $B_i(u)$ that saves distance to vertices $w \in R_i$ such that $\delta(u,w) < \delta(u,p(u,R_{i+1}))$. For $i = k$ we have a ball only for vertices that are in $T$, moreover, as $\delta(u,p(u,R_{k+1})) = \infty$ all these balls are the same and equals to $R_k$. In other words, we keep the distance from each vertex of $R_k$ to each vertex of $T$. The total size is composed of the space needed for the $k$ first balls around each vertex, which is $O(knr)$ and the space needed for saving the distance between every vertex pair $(r,t)$, where $r \in R_k$ and $t \in T$, which is $O(|T||R|/r^k)$.

For $u,w \in V$ an estimated distance is computed as follows. We run a loop for $i = 0,\ldots,k-1$. We have $v_i = u$ for even $i$ and $v_i = w$ for odd $i$. Let $\delta(u,w) = \Delta$ and let $p_i = p(v_i,R_i)$. We search for the first $i$ for which $p_i \in B_i(v_{i+1})$. Notice that as opposed to Thorup — Zwick distance oracle it might be that such an $i$ does not exist. Assume first that there is such an $i$. It follows by a similar arguments to those used by Thorup and Zwick that $\delta(v_i,p_i) \le i\Delta + \delta(v_0,p_0)$ whenever $p_{i-1} \in B_{i-1}(v_i)$. Thus, the estimation $\delta(v_i,p_i) + \delta(p_i,v_{i+1})$ that we use is bounded by

$$2(i\Delta + \delta(v_0,p_0)) + \Delta = (2i+1)\delta(u,w) + 2\delta(u,R).$$

In case that there is not such an $i$ we return $\delta(v_k,p_k) + \delta(p_k,t) + \delta(t,v_{k+1})$, where $t = p(v_{k+1},T)$. As before we

have $\delta(v_k, p_k) \leq k\Delta + \delta(v_0, p_0)$. By the triangle inequality we have $\delta(p_k, t) \leq \delta(v_k, p_k) + \delta(u, w) + \delta(t, v_{k+1})$. Hence, we have:

$$\delta(v_k, p_k) + \delta(p_k, t) + \delta(t, v_{k+1}) \leq$$
$$2(\delta(v_k, p_k) + \delta(t, v_{k+1})) + \delta(u, w \leq$$
$$2(\delta(v_0, p_0) + \delta(t, v_{k+1})) + (2k+1)\delta(u, w).$$

There are two possibilities. Recall that $v_0 = u$, now if $k$ is even then $v_{k+1} = w$ and we get that:

$$2(\delta(v_0, p_0) + \delta(t, v_{k+1})) + (2k+1)\delta(u, w) =$$
$$2(\delta(u, R) + \delta(w, T)) + (2k+1)\delta(u, w).$$

If $k$ is odd then $v_{k+1} = u$ and we get:

$$2(\delta(v_0, p_0) + \delta(t, v_{k+1})) + (2k+1)\delta(u, w) =$$
$$2(\delta(u, R) + \delta(u, T)) + (2k+1)\delta(u, w).$$

Summing up, we have proved:

Lemma 11. Given arbitrary vertex sets $R$ and $T$, parameter $r > 2$ and integer parameter $k$, there is a distance oracle $DO(k, R, T, r)$ that uses $\tilde{O}(rn + |R||T|/r^k)$ space and return estimation on the distance between $u$ and $w$ which is at most $2(\delta(u, R) + \delta(v, T)) + (2k+1)\delta(u, w)$, where $v = w$ for even $k$ and $v = u$ for odd $k$. The query time is $O(k)$.

### B. Proof of Theorem 1

We now combine the constructions presented above to obtain distance oracles with stretch $2k + 3 \pm 2/\Delta$ for $k \geq 0$. We have four parameters. The parameters $\Delta$ and $k$ control the stretch. The parameters $q$ and $r$ control the space.

For every vertex $v \in V$ we construct the balls $B_0(v), \ldots, B_\Delta(v)$ using the ball growing technique described in §II-A. For every $i \in \{0, \ldots, \Delta\}$ we have the sets $S_i$ as defined in Lemma 10. However, rather than saving all distances between vertices of $S_i$ and $S_{\Delta-i-1}$ as we have done in the case of $3 - 2/\Delta$ stretch distance oracle, we use the parameterized generalization of Thorup and Zwick distance oracle from §III-A. For every $i \in \{0, \ldots, \Delta\}$ and every pair of sets $S_i$ and $S_{\Delta-i-1}$ we have a distance oracle $DO(k, S_i, S_{\Delta-i-1}, r)$.

Given $u, w \in V$ if there is an $i \in \{0, \ldots, \Delta\}$ such that $B_i(u)$ and $B_{\Delta-i}(w)$ intersect we have the exact distance which we output. Otherwise, there is not any $i \in \{0, \ldots, \Delta\}$ such that $B_i(u)$ and $B_{\Delta-i}(w)$ intersect and we are in the scenario of separated balls as in §II-C. To perform the query in this scenario we have to consider two cases.

Case 1: $k$ is even. We search for $i \in \{0, \ldots, \Delta - 1\}$ that minimizes $rad(B_i(u)) + rad(B_{\Delta-i-1}(w))$. For this $i$ we query $DO(k, S_i, S_{\Delta-i-1}, r)$ for an estimation on $\delta(u, w)$. From Lemma 11 we know that this estimation is at most $2(\delta(u, S_i) + \delta(w, S_{\Delta-i-1})) + (2k+1)\delta(u, w)$.

From Equation (7) of Lemma 8 it follows that $rad(B_i(u)) + rad(B_{\Delta-i-1}(w)) \leq (1 - 1/\Delta)\delta(u, w)$. From Lemma 10 it follows that $\delta(u, S_i) \leq rad(B_i(u))$ and $\delta(w, S_{\Delta-i-1}) \leq rad(B_{\Delta-i-1}(w))$.

Thus we get that $\delta(u, S_i) + \delta(w, S_{\Delta-i-1}) \leq (1 - 1/\Delta)\delta(u, w)$ and the estimation is at most $(2k + 3 - 2/\Delta)\delta(u, w)$ for $k \geq 1$.

Case 2: $k$ is odd. We search for $i \in \{0, \ldots, \Delta - 1\}$ and $v \in \{u, w\}$ that minimizes $rad(B_i(v)) + rad(B_{\Delta-i-1}(v))$. Without loss of generality, assume that $v = u$. For this $i$ we query $DO(k, S_i, S_{\Delta-i-1}, r)$ for an estimation on $\delta(u, w)$. From Lemma 11 we know that this estimation is at most $2(\delta(u, S_i) + \delta(u, S_{\Delta-i-1})) + (2k+1)\delta(u, w)$.

From Equation (8) of Lemma 8 it follows that $rad(B_i(u)) + rad(B_{\Delta-i-1}(u)) \leq (1 - 1/\Delta)\delta(u, w)$. From Lemma 10 it follows that $\delta(u, S_i) \leq rad(B_i(u))$ and $\delta(u, S_{\Delta-i-1}) \leq rad(B_{\Delta-i-1}(u))$.

Thus we get that $\delta(u, S_i) + \delta(u, S_{\Delta-i-1}) \leq (1 - 1/\Delta)\delta(u, w)$ and the estimation is at most $(2k + 3 - 2/\Delta)\delta(u, w)$ for $k \geq 1$.

Next, we turn to analyze the space requirements of the construction for every $i \in \{0, \ldots, \Delta\}$. From Lemma 7 it follows that the space needed to store the ball intersections is $\tilde{O}(q^\Delta n^2/m)$. From Lemma 10 it follows that $S_i = \tilde{O}(m/q^i)$, thus, $|S_i| \cdot |S_{\Delta-i-1}| = \tilde{O}(m^2/q^{\Delta-1})$. By Lemma 11 $DO(k, S_i, S_{\Delta-i-1}, r)$ requires $\tilde{O}(rn + |S_i| \cdot |S_{\Delta-i-1}|/r^k)$ space. Thus, the total space is:

$$\tilde{O}(q^\Delta n^2/m + rn + m^2/(q^{\Delta-1} r^k)).$$

To minimize the space we chose $q$ and $r$ to balance the terms. We set $r = q^\Delta n/m$, then we just have to solve $q^\Delta n^2/m = m^2/(q^{\Delta-1}(q^\Delta n/m)^k)$. Reordering terms, we get $q^{2\Delta-1+k\Delta} = m^{3+k}/n^{2+k}$, and conclude that the total space is

$$\tilde{O}((m^{3+k}/n^{2+k})^{\Delta/(2\Delta-1+k\Delta)} n^2/m) =$$
$$\tilde{O}((m^{3+k}/n^{2+k})^{1/(k+2-1/\Delta)} n^2/m) =$$
$$\tilde{O}(m^{1+1/(k+2-1/\Delta)}) = S(2k + 3 - 2/\Delta, m).$$

As in §II-D we have an almost identical construction for stretch $2k + 3 + 2/\Delta$. The difference is that for every $i \in \{0, \ldots, \Delta - 1\}$, we use the parameterized version of Thorup and Zwick with $R = S_{i+1}$ and $T = S_{\Delta-i}$. Using Lemma 9 for the analysis, we get the claimed stretch, with space $\tilde{O}(m^{3+k}/n^{2+k})^{1/(k+2+1/\Delta)} n^2/m) = \tilde{O}(m^{1+1/(k+2+1/\Delta)}) = S(2k + 3 + 2/\Delta, m)$.

For the query time, we spend $O(\Delta)$ time finding the right $i \in \{0, ..., \Delta - 1\}$ and, for $k$ odd, $v \in \{u, w\}$. Next we spend $O(k)$ in Lemma 11, so the total query time is $O(k + \Delta)$.

## IV. Hardness reduction

We will now present our hardness reduction as stated in Theorem 4:

Fix an integer $\Delta > 1$. Consider a regular set intersection instance such that $u = s^6 n^{1-1/(2-1/\Delta)}/f$ where $f = o(1)$. We can construct an unweighted graph $G$ with $m = O(sn \log n)$ edges and $n$ sources and sinks such that if sets $S_i$ and $S_j$ intersect, then $\delta(i,j) = \Delta + 1$. Moreover, among all source-sink pairs $(i,j) \in [n]^2$, there is only a fraction $O(f)$ with $\delta(i,j) < 3\Delta + 1 = (\Delta+1)(3 - 2/(\Delta+1))$.

Proof of Theorem 4: In our reduction described below, we generally do not worry about rounding problems. Adding dummy vertices and elements, we can always increase $s$ and $u$ by a constant factor. These constants will be absorbed by the $O(f)$.

The graph will be a layered graph with $n$ vertices on each layer. At the beginning we have a layer of $n$ source vertices and at the end a layer of $n$ sink vertices. The hard instance only queries distances between vertices of the source layer and vertices of the sink layer. In the middle we have $u$ disjoint "components". To fit the overall structure, each component is also a layered graph with $w = n/u$ vertices in each layer. More precisely, each component is a binary butterfly graph of width $w = n/u$. In each layer of the butterfly, the vertices are indexed by $[w] = \{0, \ldots, w-1\}$ and the $2w$ edges from layer $i$ to layer $i+1$ connect a vertex to two vertices: one with the same index, and one which has the same index except that the $i$th bit is flipped. There are $\log_2 w$ bits to be switched, hence this many edge layers and one more vertex layer.

For each set $S_i$, $i \in [n]$, and each $a \in S_i$, we add a source edge from source $i$ to some vertex $v$ in the first layer of butterfly component $a$. By regularity, we know that $a$ is in at most $ns/u = sw$ sets $i$, so we can distribute the source edges to component $a$ in such a way that a vertex in the first butterfly layer has at most $s$ incident source edges. Corresponding sink edges are placed between the sinks and the last layer of the butterfly graphs. If $a \in S_i \cap S_j$, then there is a path from source $i$ via component $a$ to sink $j$. At this point we have $O(n \log n)$ vertices and $O(n(s + \log n))$ edges.

We will place $\Delta + 1$ layers of heavy edges with weight 1. All other edges will have weight 0. At the very end of our construction, we will contract all 0 weight edges. The result is an unweighted graph with exactly the same distances between sources and sinks. A direct path from a source to a sink has weight $\Delta + 1$. All source and sink edges will be heavy, and we will have $\Delta - 1$ almost evenly spread heavy layers in the butterfly graphs.

Our basic aim is to limit the number of source-sink paths of weight strictly less than $3\Delta + 1$, and hence the number of source-sink paths of weight at most $3\Delta$. We note that such a path cannot go zig-zag from a source to a sink to a source to a sink, for such a path would need at least $3(\Delta + 1)$ heavy edges. The path could first visit multiple sources, then cross over to a sink, and then visit multiple sinks. The cross-over part from the last source to the first sink uses $\Delta + 1$ edges, so the other parts which either go between sources, or go between sinks, can use at most $2\Delta - 1$ heavy edges in total. Note now that every source-source or sink-sink path segment $Q$ must cross any layer an even number of times, so together they must use an even number of heavy edges. Out of the $2\Delta - 1$ heavy edges available, they can therefore only use $2\Delta - 2$ heavy edges.

The source-source and the sink-sink cases are symmetric, so below we focus on the source-source case. Consider a source-source segment $Q$ that goes from a source $i$ to a source $j$ with no other sources between. Then $Q$ goes from source $i$ via a single butterfly component $a$ to a source $j$ and then $a \in S_i \cap S_j$ (recall that it is only source-sink paths that we use to code set intersections). Suppose $Q$ only moves within the first $q+1$ butterfly layers. How many possible sources $j$ can we reach from $i$ with such a path? From $i$ we can reach at most $s$ different vertices $v$ in the first butterfly layer. Fix $v$ and consider the segment $Q^\blacklozenge$ to the last vertex $w$ of $Q$ in the butterfly. Since $Q^\blacklozenge$ only uses $q$ edge layers, it can only change the corresponding $q$ bits, so from $v$, we can reach at most $2^q$ different vertices $w$. Finally, from each such vertex $w$, we can reach at most $s$ sources $j$. Thus, in total, from source $i$, we can reach at most $s^2 2^q$ different sources $j$.

We are now ready to make a strategic placement of the heavy layers. Recall that all source and sink edges are heavy. Technically, in the butterflies, the heavy layers will be vertex layers where we insert a heavy edge between the incoming and outgoing edges. We place the first heavy butterfly layer after the first $\log_2((s^4 w)^{1/\Delta}/s^2)$ original bit-switching edge layers. Symmetrically we have a heavy vertex layer in front of the last $\log_2((s^4 w)^{1/\Delta}/s^2)$ bit-switching layers. Otherwise we have $\log_2((s^4 w)^{1/\Delta})$ bit-switching layers between the heavy vertex layers. In total we end up with $\Delta - 1$ heavy vertex layers spread between the $\log_2 w$ bit-switching layers.

Return now to a path segment $Q$ from some source $i$ going through a single component to another source. If $Q$ passes only one heavy layer, namely that of the source edges, it can reach at most $s^2((s^4 w)^{1/\Delta}/s^2) = (s^4 w)^{1/\Delta}$ sources. For each additional heavy layer passed, it can reach another factor of $(s^4 w)^{1/\Delta}$ sources. Thus, if the segment passes $h$ heavy layers, it can reach $(s^4 w)^{h/\Delta}$ sources, and this is using at least $2h$ heavy edges. By composition this bound also holds for a segment passing multiple sources: using $2h$ heavy edges, we can reach $(s^4 w)^{h/\Delta}$ sources. The same statement holds for sink-sink paths.

The cross-over part is a direct source-sink paths through a single component. It can take us from a given source to one of at most $s^2 w$ distinct sinks using at least $\Delta + 1$ heavy edges. The source-source and sink-sink segments have $2\Delta - 1$ heavy edges to share, but can use only an even number, so all in all, using paths with at most $3\Delta$ heavy edges, a source

can reach at most

$$s^2 w(s^4 w)^{(\Delta-1)/\Delta} \le s^6 w^{2-1/\Delta}$$

possible sinks. We want this number to be bounded by $O(fn)$, so we require $w = O((fn/s^6)^{1/(2-1/\Delta)})$. This is implied by $w = O(n^{1/(2-1/\Delta)} f/s^6)$, or equivalently,

$$u = n/w = \Omega(n^{1-1/(2-1/\Delta)} s^6/f).$$

In Theorem 4 we defined $u = n^{1-1/(2-1/\Delta)} s^6/f$, but we ignored rounding problems in all the above calculations. Since $\Delta = O(1)$, a correct rounding is subsumed by the $\Omega$-notation. To finish the construction, we just contract all the original butterfly edges which all have weight 0. This does not affect any distance, and now the coding is with an unweighted graph. This completes the proof of Theorem 4. ■

## REFERENCES


[AG11] Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In Proc. 25th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pages 404–415, 2011.

[BGSU08] Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP), pages 609–621, 2008.

[BK06] Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS), pages 591–602, 2006.

[BS06] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. ACM Transactions on Algorithms, 2(4):557–577, 2006. See also SODA'04.

[CSTW09] Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. Compact routing in power-law graphs. In Proc. 23rd ACM Symposium on Parallel Algorithms and Architectures (SPAA), pages 379–391, 2009.

[EWG08] Mihaela Enăchescu, Mei Wang, and Ashish Goel. Reducing maximum stretch in compact routing. In Proc. IEEE INFOCOM, pages 336–340, 2008.

[KFY04] Dmitri V. Krioukov, Kevin R. Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In Proc. IEEE INFOCOM, 2004.

[MN07] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. Journal of the European Mathematical Society, 9(2):253–275, 2007. See also FOCS'06.

[PR10] Mihai Pătraşcu and Liam Roditty. Distance oracles beyond the Thorup–Zwick bound. In Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS), pages 815–823, 2010.

[RTZ05] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP), pages 261–272, 2005.

[SVY09] Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In Proc. 50th IEEE Symposium on Foundations of Computer Science (FOCS), pages 703–712, 2009.

[Tho04] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. Journal of the ACM, 51(6):993–1024, 2004. Announced at FOCS'01.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. Journal of the ACM, 52(1):1–24, 2005. See also STOC'01.