# Distance Oracles Beyond the Thorup–Zwick Bound

Mihai Pătraşcu
AT&T
mip@alum.mit.edu

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

**Abstract**

We give the first improvement to the space/approximation trade-off of distance oracles since the seminal result of Thorup and Zwick [STOC'01].

For unweighted graphs, our distance oracle has size $O(n^{5/3}) = O(n^{1.66\cdots})$ and, when queried about vertices at distance $d$, returns a path of length $2d + 1$.

For weighted graphs with $m = n^2/\alpha$ edges, our distance oracle has size $O(n^2/\sqrt[3]{\alpha})$ and returns a factor 2 approximation.

Based on a plausible conjecture about the hardness of set intersection queries, we show that a 2-approximate distance oracle requires space $\widetilde{\Omega}(n^2/\sqrt{\alpha})$. For unweighted graphs, this implies a $\widetilde{\Omega}(n^{1.5})$ space lower bound to achieve approximation $2d + 1$.

## 1 Introduction

A distance oracle is a replacement for the all-pairs shortest paths matrix of a graph. Given a graph with $n$ vertices and $m$ edges, the goal is to construct a data structure of small (subquadratic) space, such that the distance between any two vertices can be computed efficiently (say, in constant time). Since subquadratic space is believed to be unattainable in the exact case, we allow approximation. Say the query asks about the distance between vertices $s$ and $t$, and let $d = d(s, t)$. Then, a $k$-approximate distance oracle might return a distance $\hat{d}$ of $d \leq \hat{d} \leq k \cdot d$. The oracle should also be able to list a path of length $\hat{d}$ in constant time per hop.

The state of the art in distance oracles can be understood by reference to the landmark result of Thorup and Zwick from STOC'01 [TZ05]. They describe a distance oracle of size $O(kn^{1+1/k})$ which achieves a factor $2k - 1$ approximation, for any integer $k \geq 2$. Thus, for approximation 3, the space is $O(n^{1.5})$; for approximation below 3, no subquadratic bound is known.

The original construction time was $O(mn^{1/k})$. In SODA'04, Baswana and Sen [BS06] improved this to $O(n^2)$ for unweighted graphs. In FOCS'06, Baswana and Kavitha [BK06] extended the $O(n^2)$ running time to weighted graphs. Subsequently, Baswana, Gaur, Sen, and Upadhyay [BGSU08] obtained subquadratic construction time in unweighted graphs, at the price of increasing the approximation from $k \cdot d$ to $k \cdot d + O(1)$. For instance, their algorithm runs in time $O(m + n^{23/12})$ for $3d + 8$ approximation.

The original query time was $O(k)$, which is not constant for super-constant approximation. In FOCS'06, Mendel and Naor [MN07] gave an oracle with space $O(n^{1+1/k})$ and $O(1)$ query time, at the price of increasing the approximation factor to $O(k)$.

Motivated by experimental evidence from Krioukov et al. [KFY04], Chen, Sommer, Teng, and Wang [CSTW09] show that the Thorup–Zwick scheme has smaller space on random power-law

graphs. Enăchescu, Wang, and Goel [EWG08] demonstrate better space for Erdős-Rényi random graphs.

**Space versus approximation** Despite considerable follow-up work, there has been no progress on what appears to be the main consideration: the trade-off between approximation and space.

Thorup and Zwick [TZ05] show that their space bound is "optimal" based on the girth conjecture of Erdős (which is partially proved, and widely believed). However, one must be careful about the notion of optimality here: this lower bound only shows that there exist dense enough graphs, which cannot be "compressed" to less than $n^{1+1/k}$ edges without introducing approximation above $2k-1$.

There are two scenarios in which better distance oracles could be possible:

1. The incompressibility argument says nothing about sparse graphs (the most realistic network scenario). If the graph has less than $n^{1+1/k}$ edges, the problem is not one of compression, but of data representation: how to store shortest paths information for quick retrieval.

2. For unweighted graphs, it makes sense to consider additive approximation in addition to multiplicative. The lower bound is based on distances $d = 1$, and thus only implies that an approximation of $d + 2k - 1$ requires space $\Omega(n^{1+1/k})$. Needless to say, an additive approximation of $2k - 1$ is much more desirable than the same multiplicative factor.

A reason to be skeptical about the first scenario is that, even for sparse graphs, the *exact* problem is thought to require quadratic space. This can be argued quite persuasively under a popular conjecture about the hardness of set intersection (see below).

In FOCS'09, Sommer, Verbin, and Yu [SVY09] provided a further negative result: they showed that $c$-approximate distance oracles with constant query time require space $n^{1+\Omega(1/c)}$. Obtaining a sharp constant in the exponent is beyond the state of the art in lower bounds, so this is the best argument one can currently bring for the optimality of the Thorup–Zwick oracle.

Regarding the second scenario, we note that much progress has been made in the related area of graph theory that deals with spanners. Given a graph $G$, a $(\alpha, \beta)$-spanner is a subgraph $H$ which guarantees that, for any pair of vertices $u, v$, $d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$. In other words, distances increase at most by a factor of $\alpha$ and an additive $\beta$ if we throw out the edges in $G \setminus H$. In the spirit of distance oracles, a classic result [ADD+93] constructs a $(2k-1, 0)$-spanner. Among purely multiplicative results, this is optimal assuming the girth conjecture.

In STOC'01, Elkin and Peleg [EP04] constructed spanners with $1 + \varepsilon$ multiplicative stretch and $n^{1+\varepsilon}$ edges, but with an additive overhead (that depends on $\varepsilon$). In SODA'06, Thorup and Zwick [TZ06] showed that a spanner with this type of guarantee also follows from the techniques of their distance oracle. Specifically, they obtain $(1+\epsilon, \beta)$-spanners of size $O(n^{1+1/k})$, for any constant $k$, where $\beta = (\frac{O(1)}{\varepsilon})^k$.

One can also construct purely additive spanners. Dor, Helperin, and Zwick [DHZ00] present a simple $(1, 2)$-spanner with $\widetilde{O}(n^{3/2})$ edges. Elkin and Peleg [EP04] obtain $(1, 2)$-spanner with $O(n^{3/2})$ edges using a different construction. In SODA'05, Baswana, Kavitha, Mehlhorn, and Pettie [BKMP05] present a spanner with $O(n^{4/3})$ edges and additive 6 approximation. Sparser spanners are not known for any additive guarantee, though it is conceivable that a spanner with $O(n^{1+1/k})$ edges and additive approximation $2k - 1$ could exist. This would match the lower bound of the girth conjecture; in fact, in FOCS'06, Woodruff [Woo06] showed this lower bound unconditionally for additive spanners.

**Additive guarantees in distance oracles** None of the improvements in spanners translated into an improved distance oracle with additive guarantees. There is, in fact, a strong barrier to additive guarantees in distance oracles. Consider breaking every edge into $c$ pieces, introducing vertices along the edge. This increases the number of vertices to $n + cm$ and multiplies each distance by $c$. Since any pairwise distance is a multiple of $c$, any additive guarantee of less than $c$ is meaningless.

If we started with a sparse graph, $m = O(n)$, the size has not changed asymptotically. Thus, an $(\alpha, \beta)$-distance oracle is at least as hard to construct as an $\alpha$-approximate distance oracle on sparse graphs. In other words, we cannot hope for scenario 2 without also solving scenario 1 (building better distance oracles for sparse graphs).

This reveals a large gap between spanners (a graph theoretic problem) and distance oracles (an algorithmic problem). If the graph has $O(n)$ edges, a spanner is trivial: just include all the edges. For a distance oracle, sparse graphs can be hard.

## 1.1 New Upper Bounds

In this paper, we break the long-standing bounds of Thorup and Zwick, and give the first nontrivial distance oracles for approximation below 3.

For unweighted graphs, we realize scenario 2 from above: we show that it is possible to gain from additive approximation! Specifically, we obtain an oracle of size $O(n^{1.66\cdots})$, which returns a distance of at most $2d + 1$:

**Theorem 1.** *For any unweighted graph, there exists a distance oracle of size $O(n^{5/3})$ that, given any nodes $u$ and $v$ at distance $d$, returns a distance of at most $2d+1$ in constant time. The distance oracle can be constructed in expected time $O(m \cdot n^{2/3})$.*

Notice that the size of the oracle is independent of the graph size, i.e. our oracle can also be used to compress the graph. If compression were the goal, one could use a $(1, 2)$-spanner, and achieve a representation of size $O(n^{1.5})$ (which is optimal). For distance oracles, however, no $o(n^2)$ space bound was known even for graphs with $m = O(n)$ edges.

By breaking up the edges, this immediately implies a 2-approximate distance oracle of space $O(n^{5/3})$ for graphs with $m = O(n)$ edges. The following generalizes this result for weighted graphs and for any graph density:

**Theorem 2.** *Given a weighted graph with $m = n^2/\alpha$ edges, we can construct a distance oracle of size $O(n^2/\alpha^{1/3})$, which returns a 2-approximation to any distance in constant time.*

This realizes scenario 1 from above: we obtain the first distance oracles that take advantage of graph sparsity. Our space bound is "in between" $m$ and $n^2$. For any $m < n^{2-\varepsilon}$, the space is superlinear, but subquadratic.

## 1.2 Lower Bounds

Since our results achieve unexpected space bounds, understanding the limitations of distance oracles becomes a very intriguing question. A priori, it is conceivable that $2d + 1$ approximation in sparse graphs can be achieved by an oracle taking space $O(n^{1+\varepsilon})$. For 2-approximate distance oracles, the space needs to be $\Omega(m)$ [TZ05], i.e. compression is impossible. But it is unclear that it needs to be

superlinear in $m$, or have our "between $m$ and $n^2$" flavor. For instance, could one achieve a space of $O(m + n^{5/3})$?

The state of the art in lower bounds does not allow us to address these questions unconditionally: current lower bounds cannot differentiate between space $O(n^{1.01})$ and space $O(n^{1.99})$. To achieve some guidance in understanding distance oracles, we will prove a conditional lower bound.

**Conjecture 3.** *Consider a data structure that preprocesses sets $S_1, \ldots, S_n \subseteq [X]$, and answers queries of the form "does $S_i$ intersect $S_j$?" Let $X = \lg^c n$ for a large enough constant $c$.*

*If the query takes constant time, the space must be $\widetilde{\Omega}(n^2)$.*

This conjecture is folklore, and appears rather widely believed. The problem is of fundamental importance in information retrieval. For instance, $S_i$ could be the set of documents containing word $i$. Then, $S_i \cap S_j$ represents the documents that contain two search terms. An algorithm that could retrieve these documents significantly faster than scanning sets $S_i$ or $S_j$ would be a major breakthrough.

An important implication of this conjecture is a strong lower bound on exact distance oracles; see [CP10]:

**Corollary 4.** *A distance oracle for unweighted graphs with $m = \widetilde{O}(n)$ edges, which can distinguish between distances of 2 and 4 in constant time requires $\widetilde{\Omega}(n^2)$ space, assuming Conjecture 3.*

*Proof.* Build a bipartite graph, with $n$ vertices on the left and $X$ on the right; the number of vertices is $n + X = O(n)$. Connect vertex $i$ to the elements of $S_i$; the number of edges is $nX = \widetilde{O}(n)$. Two left vertices are at distance 2 if the corresponding sets intersect, and distance at least 4 otherwise. Thus, the distance oracle can solve set intersection queries. □

It follows that distance oracles with approximation better than 2 require nearly quadratic space. Since the lower bound applies to sparse graphs, no additive approximation can help (by breaking edges into pieces). Thus, approximation $(2 - \varepsilon)d + O(1)$ is impossible with subquadratic space. In other words, our distance oracles give the best possible approximation with nontrivial space.

We will take this idea further, using the hardness of set intersection to argue a lower bound for 2-approximate oracles. Unfortunately, we need a distributional version of the conjecture, which will be stated shortly as Conjecture 7. Before dealing with these technicalities, we discuss the implications for distance oracles.

**Theorem 5.** *Consider a distance oracle for unweighted graphs with $m = n^2/\alpha$ edges. If the query algorithm can distinguish between distances 3 and 7 in constant time, the oracle needs $\widetilde{\Omega}(n^2/\sqrt{\alpha})$ space, assuming Conjecture 7.*

This should be contrasted to our upper bound, which gives 2-approximate distances using space $O(n^2/\sqrt[3]{\alpha})$. The flavor of our upper bound is found to be correct: superlinear space is needed for any subquadratic $m$.

In the regime $m = O(n)$, a $2d + 1$ distance oracle is equivalent to a 2-approximate distance oracle. Thus, our $O(n^{1.66\cdots})$ upper bound is complemented by the following:

**Corollary 6.** *Under Conjecture 7, a $(2, 1)$-distance oracle for unweighted graphs with $m = O(n)$ edges requires space $\widetilde{\Omega}(n^{1.5})$.*

**Technical details** We will use the following amended version of Conjecture 3:

**Conjecture 7.** *Consider a data structure in the cell-probe model that preprocesses sets $S_1, \ldots, S_n \subseteq [X]$, and answers queries of the form "does $S_i$ intersect $S_j$?" Let $\alpha, \beta$ be large enough constants. Let $X = O(\lg^\alpha n)$, and assume that the data structure uses constant query time and space $O(n^2/\lg^\beta n)$.*

*There exists a distribution $\mathcal{D}$ such that, when $(S_1, \ldots, S_n)$ is drawn from $\mathcal{D}$ and $(i, j) \in [n]^2$ uniformly at random, the data structure has error at least $\frac{1}{10}$.*

A reasonable way to state our simple Conjecture 3 is that a *randomized* data structure with constant query time must use $\widetilde{\Omega}(n^2)$ space. If we had hardness for randomized algorithms, by Yao's minimax principle [Yao77], there would exist a distribution $\mathcal{D}'$ that is hard for any algorithm. For technical reasons, we cannot deal with an arbitrary distribution $\mathcal{D}'$, but only one of the form $\mathcal{D}' = \mathcal{D} \times [n]^2$ (the query $(i, j)$ is uniform). Intuitively, this assumption is quite plausible: if $(i, j)$ has too biased a distribution, one can tabulate the answers for the frequent $(i, j)$ values, so the space is certainly subquadratic.

One can imagine strengthening the conjecture to super-constant query time (even a lower bound of $\Omega(X/\lg n)$ seems plausible). In this case, we would obtain space lower bounds against distance oracles with polylogarithmic query time. Going the other way, one may assume a weaker space lower bound of $\Omega(n^{2-\varepsilon})$, with creates an $\varepsilon$ loss in the exponent of the lower bound.

Even if one chooses to disbelieve Conjecture 7, a "lower bound" based on it reveals the limit of current techniques. Indeed, if intersection queries were miraculously easy, it is quite likely that such a data structure could be used to give better distance oracles. (The upper bounds of this paper need to jump several hoops to avoid naturally occurring set intersection queries.)

## 2 Upper Bound for Unweighted Graphs

In this section, we prove Theorem 1: we construct a distance oracle of size $O(n^{5/3})$ that approximates the distance between any vertices $s, t$ with a distance of at most $2 \cdot d(s, t) + 1$.

Our construction will be randomized, and will yield an oracle of expected size $O(n^{5/3})$; the query algorithm is deterministic (and never makes an error). Given this guarantee, a distance oracle of *worst-case* size $O(n^{5/3})$ can be constructed by a Las Vegas algorithm.

Our construction begins by sampling vertices at two different rates:

**set $A$:** sample every vertex independently at random with probability $n^{-1/3}$. Then $\mathbf{E}[|A|] = n^{2/3}$.

**set $B$:** sample every vertex independently at random with probability $n^{-2/3}$. Then $\mathbf{E}[|B|] = n^{1/3}$.

Given a vertex $u \in V$ and a set of vertices $S \subseteq V$, we make the following convenient definitions:

$\mathrm{NN}_S(u) =$ the nearest neighbor of $u$ in $S$ (i.e. the node $v \in S$ that minimizes $d(u, v)$, where ties are broken arbitrarily).

$R_S(u) =$ the distance from $u$ to its nearest neighbor $\mathrm{NN}_S(u)$.

$\mathcal{B}_S(u) =$ the set of vertices strictly closer to $u$ than $\mathrm{NN}_S(u)$. In other words, $\mathcal{B}_S(u)$ is the ball around $u$ of radius $R_S(u) - 1$.

Our construction first considers the balls $\mathcal{B}_A(u)$ for all $u \in B$. Intuitively speaking, we grow balls around vertices of $B$, stopping whenever a vertex of $A$ is reached. Define $C$ to contain all vertices reached by this process (some may be reached multiple times, for various $u \in B$):

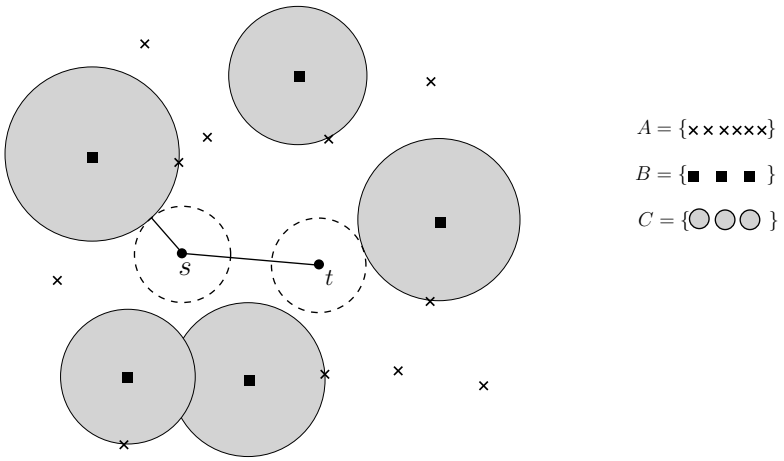


Figure 1: The dashed balls represent $\mathcal{B}_C(s)$ and $\mathcal{B}_C(t)$.

**set** $C$**:** $A \cup \left( \bigcup_{u \in B} \mathcal{B}_A(u) \right)$.

We store the distance from each $u \in C$ to all other nodes of $G$.

**Claim 8.** *The expected size of $C$ is at most $n^{2/3}$.*

*Proof.* By linearity of expectation, we have:

$$\mathbf{E}[|C|] \;\leq\; \mathbf{E}[|A|] + \sum_u \Pr[u \in B] \cdot \mathbf{E}[|\mathcal{B}_A(u)|] \;=$$

$$n^{2/3} + \sum_u \mathbf{E}[|\mathcal{B}_A(u)|]/n^{2/3}$$

The size $|\mathcal{B}_A(u)|$ is a random variable depending on the samples $A$. If we imagine all nodes sorted by distance to $u$ (breaking ties arbitrarily), an upper bound on $|\mathcal{B}_A(u)|$ is the location of the first element from $A$ in this list. In other words, the size of $\mathcal{B}_A(u)$ is bounded from above by a geometric random variable with rate $n^{-1/3}$. Hence, $\mathbf{E}[|\mathcal{B}_A(u)|] \leq n^{1/3}$ for all $u$, and we obtain

$$\mathbf{E}[|C|] \leq n^{2/3} + n \cdot n^{1/3}/n^{2/3} = 2n^{2/3}.$$

□

We now turn our attention to $\mathcal{B}_C(u)$, for all vertices $u \in V \setminus C$. Intuitively, we grow a ball around each vertex, stopping whenever we reach a vertex from $C$. See Figure 1 for a geometrical depiction.

With this build-up, the strategy of our oracle is simple to describe. Consider the balls around our source and destination, $\mathcal{B}_C(s)$ and $\mathcal{B}_C(t)$. We have two possibilities:

$\mathcal{B}_C(s) \cap \mathcal{B}_C(t) = \emptyset$**.** By definition, $\mathcal{B}_C(s)$ contains *all* nodes at distance $R_C(s) - 1$ from $s$, and similarly, $\mathcal{B}_C(t)$ contains *all* nodes at distance $R_C(t) - 1$ from $t$. If the balls are disjoint, then necessarily:

$$(R_C(s) - 1) + (R_C(t) - 1) < d(s,t) \;\Rightarrow$$

$$R_C(s) + R_C(t) \le d(s,t) + 1 \;\Rightarrow$$
$$\min\{R_C(s), R_C(t)\} \le \tfrac{d(s,t)+1}{2}$$

Recall that we have stored the distance from each $u \in C$ to all other nodes. If $R_C(s) \le R_C(t)$, we will approximate $d(s,t)$ by the length of the path $s \rightsquigarrow \mathrm{NN}_C(s) \rightsquigarrow t$. Applying the triangle inequality, the length of this path is:

$$d\big(s, \mathrm{NN}_C(s)\big) + d\big(\mathrm{NN}_C(s), t\big) \;\le$$
$$2R_C(s) + d(s,t) \le 2 \cdot d(s,t) + 1$$

The case $R_C(s) > R_C(t)$ is symmetric, where we take the path $s \rightsquigarrow \mathrm{NN}_C(t) \rightsquigarrow t$.

**$\mathcal{B}_C(s) \cap \mathcal{B}_C(t) \neq \emptyset$.** For all pairs $(s,t)$ such that $\mathcal{B}_C(s)$ and $\mathcal{B}_C(t)$ intersect, we store their exact distances in a hash table.

To summarize, our query algorithm is the following:

1. Look up $(s,t)$ in the hash table. If found, return the exact distance.
2. If $R_C(s) \le R_C(t)$, return the length of the path $s \rightsquigarrow \mathrm{NN}_C(s) \rightsquigarrow t$.
3. Otherwise, return the length of the path $s \rightsquigarrow \mathrm{NN}_C(t) \rightsquigarrow t$.

It remains to understand the space requirement. Since $\mathbf{E}[|C|] = O(n^{2/3})$, the first part of the distance oracle (the distance from each vertex in $C$ to all other vertices) takes $O(n^{5/3})$ space in expectation. The remaining challenge is to analyze the size of our hash table from the second part.

**Claim 9.** *For any $u \in V$, the number of vertices $v$ with $\mathcal{B}_C(u) \cap \mathcal{B}_C(v) \neq \emptyset$ is at most $n^{2/3}$ in expectation.*

*Proof.* Let $S$ be the set of vertices $v$ whose ball intersects the ball of $u$. For ease of counting $|S|$, we can consider the following algorithmic definition. Sort $w \in V$ by increasing distance from $u$. In this order, keep adding to $S$ all vertices $v$ whose ball contains $w$, i.e. $w \in \mathcal{B}_C(v)$. Stop when you reach some $w \in C$.

To ease analysis, we will overcount $|S|$: imagine that the algorithm adds all $v$ such that $w \in \mathcal{B}_A(v)$. Since $A \subseteq C$, $\mathcal{B}_C(v)$ is subset of $\mathcal{B}_A(v)$, and this new algorithm outputs a superset of $S$.

We now claim that the algorithm stops before adding any $v \in B$. Suppose for contradiction that $v \in B$, and $w \in \mathcal{B}_A(v)$. But $C$ contains all balls $\mathcal{B}_A(v)$ for $v \in B$, so $w \in C$. This is a contradiction, since the algorithm stops upon seeing $w \in C$.

We can finally bound $|S|$: if we sort $v$'s in the order in which they are added by the algorithm, $|S|$ is bounded by the first position on which an element $v \in B$ appears. Observe that the order only depends on $B$ (the algorithm looks at $\mathcal{B}_A(v)$), but is independent of $A$. Thus, $|S|$ is bounded by a geometric random variable with rate $n^{-2/3}$. We have shown $\mathbf{E}[|S|] \le n^{2/3}$.

□

**Construction time** The cost for computing the distance from each $u \in C$ to all other nodes of $G$ is $\widetilde{O}(mn^{2/3})$. Also, the cost for computing $\mathcal{B}_A(u)$ for every $u \in B$ is $O(n)$ since $\mathbf{E}[|B|] = n^{1/3}$ and $\mathbf{E}[|\mathcal{B}_A(u)|] \leq n^{1/3}$.

As $\mathcal{B}_C(u) \subseteq \mathcal{B}_A(u)$, the cost of growing $\mathcal{B}_C(u)$ is bounded by the cost of growing $\mathcal{B}_A(u)$, for all vertices $u \in V \setminus C$. Since $\mathbf{E}[|\mathcal{B}_A(u)|] \leq n^{1/3}$ the cost is bounded by $O(n^{5/3})$.

We also need to compute the intersection between $\mathcal{B}_C(s) \cap \mathcal{B}_C(t)$ for every pair of vertices $s, t \in V$. Since $\mathbf{E}[|\mathcal{B}_A(u)|] \leq n^{1/3}$ for every $u \in V$ we can compute all the intersections in $O(n^{7/3})$ time. We conclude that the construction time is $\widetilde{O}(mn^{2/3}) + O(n^{7/3})$.

**Reporting a path** It is not too hard to extend our data structure to return also actual paths in constant time per hop. In the case that $\mathcal{B}_C(s) \cap \mathcal{B}_C(t) = \emptyset$ and we use either the shortest paths tree of $\mathrm{NN}_C(s)$ or of $\mathrm{NN}_C(t)$ to report the distance we can also use it to report a path. In the case that $\mathcal{B}_C(s) \cap \mathcal{B}_C(t) \neq \emptyset$ we can save a vertex from the intersection. Then a shortest path from $s$ to this vertex is obtained using $\mathcal{B}_C(s)$ and a shortest path from this vertex to $t$ is obtained using $\mathcal{B}_C(t)$.

# 3 Upper Bound for Weighted Graphs

We now show how to obtain 2-approximate distance oracles for weighted graphs. Since an additive term does not make sense for weighted graphs, we must implicitly eliminate the $+1$ term from the previous approximation. This term comes from the case when $\mathcal{B}_C(s)$ and $\mathcal{B}_C(t)$ almost touch, i.e. they are connected by an edge. In the weighted case, that edge could have arbitrary weight, destroying the approximation guarantee.

To address this problem, we change our focus from *vertices* to *edges*. With $S \subseteq E$ being a set of edges, we make the following convenient definitions:

$N(S) =$ the set of vertices incident to the edges of $S$.

$\mathcal{B}^\star_S(u) =$ the set of edges incident to the vertices of $\mathcal{B}_{N(S)}(u)$.

A convenient mental picture of $\mathcal{B}^\star_S(u)$ is in terms of growing balls: consider a Dijkstra exploration around $u$, which stops when it "touches" an edge from $S$ (when it reaches the first vertex incident to an edge from $S$). The edges incident to the explored vertices form $\mathcal{B}^\star_S(u)$. Note that some of these edges may go far outside the ball.

It is crucial to observe the new meaning of $\mathcal{B}^\star_S(u) \cap \mathcal{B}^\star_S(v) \neq \emptyset$: the balls of vertices around $u$ and $v$ have a connecting edge between them. This addresses the reason for the additive one in the previous algorithm.

From now on, we will assume perfect tie-breaking (all $n^2$ possible distances are distinct). The construction algorithm will build:

**set** $A$**:** sample each *edge* with probability $\frac{n}{m}/\alpha^{1/3}$. We have $\mathbf{E}[|A|] = m \cdot \frac{n}{m}/\alpha^{1/3} = n/\alpha^{1/3}$. (Compare to the previous definition, where $A$ sampled $n^{2/3}$ *vertices*.)

**set** $B$**:** sample each *vertex* with probability $\alpha^{1/3}/n$. We have $\mathbf{E}[|B|] = \alpha^{1/3}$. (In the previous definition, $B$ samples $n^{1/3}$ vertices.)

**set** $C$**:** $A \cup \left( \bigcup_{u \in B} \mathcal{B}^\star_A(u) \right)$.

**Claim 10.** *The expected size of $C$ is at most $n/\alpha^{1/3}$.*

*Proof.* We claim that, for any fixed $u$, the expected size of $\mathcal{B}^\star_A(u)$ is at most $\frac{m}{n}\alpha^{1/3}$. Remember the interpretation of $\mathcal{B}^\star_A(u)$ in terms of growing balls. Whenever a node $v$ is added to the ball by Dijkstra's algorithm, all edges incident to $v$ are added to $\mathcal{B}^\star_A(u)$. But if an edge was sampled in $A$, the algorithm stops. Thus, the size of $\mathcal{B}^\star_A(u)$ is bounded by the first occurrence of an $A$ sample, which is a geometric random variable with rate $\frac{n}{m}/\alpha^{1/3}$.

Since every $u$ is placed into $B$ with probability $\alpha^{1/3}/n$, the expected size of $C$ is:

$$n \cdot \left(\alpha^{1/3}/n\right) \cdot \mathbf{E}[|\mathcal{B}^\star_A(u)|] \;\leq\; \alpha^{1/3} \cdot \tfrac{m}{n}\alpha^{1/3} \;=$$

$$\tfrac{n^2/\alpha}{n} \cdot \alpha^{2/3} \;=\; n/\alpha^{1/3}$$

We will now consider balls $\mathcal{B}^\star_C(u)$ around every vertex $u$. The distance oracle will store:

1. the distance from each vertex in $N(C)$ to each other vertex.
2. the distance between any $u$ and $v$ with $\mathcal{B}^\star_C(u) \cap \mathcal{B}^\star_C(v) \neq \emptyset$.

The first component has expected size $O(n^2/\alpha^{1/3})$. Whenever $\mathcal{B}^\star_C(s)$ is disjoint from $\mathcal{B}^\star_C(t)$, this component will provide a 2-approximate answer. Indeed, consider a shortest path from $s$ to $t$. By definition of $\mathcal{B}^\star_C(\cdot)$, some vertex of this path must be missing from both $\mathcal{B}_{N(C)}(s)$ and $\mathcal{B}_{N(C)}(t)$. Let this vertex be $v$. If $d(s,v) < d(v,t)$, then $d(s,v) \leq \frac{1}{2}d(s,t)$. Also, the nearest neighbor of $s$ in $N(C)$ (call it $w$) is at distance at most $d(s,v)$, since $v$ was not in the ball. We can report the path $s \rightsquigarrow w \rightsquigarrow t$, which, by the triangle inequality, has length at most $2 \cdot d(s,w) + d(s,t) \leq 2 \cdot d(s,t)$. The case when the nearest neighbor of $t$ is closer than $d(s,w)$ is symmetric.

It remains to bound the size of the second component:

**Claim 11.** *For any $u \in V$, the number of vertices $v$ with $\mathcal{B}^\star_C(u) \cap \mathcal{B}^\star_C(v) \neq \emptyset$ is at most $n/\alpha^{1/3}$ in expectation.*

*Proof.* Let $S$ be the set of vertices $v$ with $\mathcal{B}^\star_C(u) \cap \mathcal{B}^\star_C(v) \neq \emptyset$. By changing from $\mathcal{B}^\star_C(v)$ to $\mathcal{B}^\star_A(v)$, this becomes a superset of what we want to count.

Imagine a Dijkstra exploration around $u$. Whenever a vertex is added to the ball, all its incident edges are added to $\mathcal{B}^\star_C(u)$. This means that all vertices $v$ whose balls $\mathcal{B}^\star_A(v)$ contain one of these edges are added to $S$.

The process certainly stops before $v \in B$ would be added. Indeed, $C$ contains all balls $\mathcal{B}^\star_A(v)$ for $v \in B$. Thus, the edge that would cause $v$ to be added is in $C$, marking the end of $\mathcal{B}^\star_C(u)$.

This means that $|S|$ is bounded from above by the first occurrence of a vertex from $B$. The order in which vertices are considered is independent of $B$ (since we changed to using $\mathcal{B}^\star_A$). Thus, $|S|$ is bounded by a geometric random variable with rate $\alpha^{1/3}/n$. □

# 4 The Lower Bound

In this section, we prove Theorem 5: a 2-approximate distance oracle for unweighted graphs with $m = O(n^2/\alpha)$ edges requires space $\widetilde{\Omega}(n^2/\sqrt{\alpha})$, assuming Conjecture 7 about the hardness of set intersection.

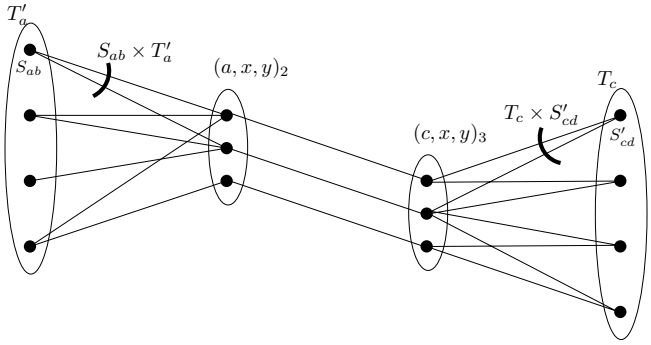


Figure 2: The structure of a source–destination path in our 4-level graphs.

### 4.1 Graph Structure

Our instances will be graphs with four layers of vertices, with a rather particular structure; see Figure 2. Let $A$ and $X$ be parameters to be decided; we will set $|X| = \lg^{O(1)} n$ and $A \in \left[\sqrt{n}, \frac{n}{X}\right]$.

The first and last levels will each have a vertex set of $[A] \times [\frac{n}{A}]$. We denote a node in these levels by $(a,b)_i$, where $i \in \{1,4\}$. The middle two levels each have a vertex set of $[A] \times [X]^2$, and we denote a vertex by $(a,x,y)_i$ for $i \in \{2,3\}$.

The edges are build from four families of sets: $S_{ab}$ and $S'_{ab}(\forall)(a,b) \in [A] \times [\frac{n}{A}]$; $T_a$ and $T'_a(\forall)a \in [A]$. The three layers of edges are:

**Left:** A vertex $(a,b)_1$ is connected to $(a,x,y)_2$ iff $x \in S_{ab}$ and $y \in T'_a$. That is, the neighbors of $(a,b)_1$ are $\{a\} \times S_{ab} \times T'_a$.

**Middle:** A vertex $(a,x,y)_2$ is connected to all vertices $(c,x,y)_3$, $(\forall)c \in [A]$.

**Right:** A vertex $(c,d)_4$ is connected to $(c,x,y)_3$ iff $x \in T_c$ and $y \in S'_{cd}$. That is, the neighbors of $(c,d)_4$ are $\{c\} \times T_c \times S'_{cd}$.

The query will always have the source on level one, and the destination on level four. The minimum distance between them is thus 3. The following characterizes paths of length 3:

**Fact 12.** *Source $(a,b)_1$ and destination $(c,d)_4$ are connected by a path of length 3 if and only if $S_{ab} \cap T_c \neq \emptyset$ and $S'_{cd} \cap T'_a \neq \emptyset$.*

*Proof.* A path of length 3, if it exists, has the form $(a,b)_1 \rightsquigarrow (a,x,y)_2 \rightsquigarrow (c,x,y)_3 \rightsquigarrow (c,d)_4$. By construction, $x \in S_{ab}$ and $x \in T_c$; also $y \in T'_a$ and $y \in S'_{cd}$. □

When a path of length 3 exists, the distance oracle must find a path of length at most $2 \cdot 3 = 6$. However, our level structure forces all paths between sources and destinations to be of *odd* length. Thus, if a path of length 3 exists, the oracle is forced to return a path of length at most 5. The following characterizes such paths:

**Fact 13.** *Source $(a,b)_1$ and destination $(c,d)_4$ are connected by a path of length 5 if and only if $S_{ab} \cap T_c \neq \emptyset$ or $S'_{cd} \cap T'_a \neq \emptyset$.*

*Proof.* On two levels, the path must spend only one edge per level; it may spend 3 edges on the remaining "bad level." The bad level could be:

1. One can traverse $(a,b)_1 \rightsquigarrow (a,x,y)_2 \rightsquigarrow (a,b')_1 \rightsquigarrow (a,x',y')$. We have no limitation on $x'$ in this setting, but it must be that $y' \in T'_a$. The path must continue $(a,x',y') \rightsquigarrow (c,x',y') \rightsquigarrow (c,d)_1$, so $y' \in S_{cd}$. Therefore, $T'_a$ and $S'_{cd}$ intersect.

2. One can traverse $(a,x,y)_2 \rightsquigarrow (c',x,y)_2 \rightsquigarrow (a',x,y)_2 \rightsquigarrow (c,x,y)_2$. However, the values of $x$ and $y$ cannot change by moving inside the second level. Thus, it must be that $x \in S_{ab} \cap T_c$ and $y \in S'_{cd} \cap T'_a$.

3. Analogous to case 1, we have $x \in S_{ab} \cap T_c$. □

Thus, returning an approximate path of length 5 is as hard as solving one of two intersection queries "does $S_i$ intersect $T_j$?", where $i \in [n]$ and $j \in [A]$. Intuitively, this will require space $\widetilde{\Omega}(n \cdot A)$. To understand this lower bound in terms of $n$ and $m$, note that our graph has $O(n)$ vertices and $m = 2nX^2 + X^2A^2$ edges. We will fix $X = \lg^c n$ for a large enough constant $c$ to make set intersection hard. Then, the number of edges is $m = \widetilde{O}(A^2)$. When aiming for graphs of density $m = n^2/\alpha$, we set $A = \widetilde{O}(n/\sqrt{\alpha})$ and thus the space lower bound is $\widetilde{\Omega}(nA) = \widetilde{\Omega}(n^2/\sqrt{\alpha})$.

### 4.2 A Formal Reduction

We now give a formal reduction from Conjecture 7 to the distance oracle problem in the layered graphs defined above. First we convert to the asymmetric version of the problem: we are given sets $S_1, \ldots, S_n$ and $T_1, \ldots, T_A$, where $A \leq N$, and we are asked queries of the form "does $S_i$ intersect $T_j$?" Conjecture 7 trivially implies a lower bound of $\widetilde{\Omega}(nA)$ on the space, since we can solve the original problem using $\lceil \frac{n}{A} \rceil$ instances of this problem.

Let $p_1$ be the probability, according to $\mathcal{D}$, that $S_i$ and $T_j$ intersect (the correct query answer is affirmative). We must have $0.1 \leq p_1 \leq 0.9$, since otherwise an algorithm could have constant output, and have small error.

We will now describe an algorithm for the set intersection problem on $\mathcal{D} \times [n]^2$. First toss a random coin $R$ to determine whether you will use your input as $(S,T)$ or as $(S',T')$. Assume the former by symmetry ($R = 0$). At construction time, the algorithm sees $S,T$, but does not know $i,j$ yet. Pick $(S',T')$ from $\mathcal{D}$. (Since we made Conjecture 7 in the cell-probe model, it does not matter is $\mathcal{D}$ is easy to sample algorithmically.) Permute $S,T,S',T'$ randomly, and build a graph based on them. At query time, $(i,j)$ is given. After applying the same permutation as for $S$ and $T$, $i$ translates into $(a,b)$ and $j$ translates into $c$. Now choose $d$ randomly; through the permutation of $S'$ and $T'$, this gives $(i',j')$. Query the distance oracle for the distance $(a,b)_1 \rightsquigarrow (c,d)_4$.

We claim that the distance oracle has no information about $R$: the marginal distribution it sees is identical if $R = 0$ and $R = 1$. Indeed, $(S,T,S',T')$ are always chosen from $\mathcal{D} \times \mathcal{D}$. Since $(i,j)$ is uniform in $[n] \times [A]$ and $d$ is uniform in $[n/A]$, the distribution of $(i,j,i',j')$ is completely uniform.

Let us now reason from the point of view of the distance oracle. The probability that there is a path of length 3 ($S_i \cap T_j \neq \emptyset$ and $S'_{i'} \cap T'_{j'} \neq \emptyset$) is precisely $p_1^2$. Indeed, fix $a,b,c,d$. Since the permutations for $S,T,S',T'$ were independently random, the queries $(i,j)$ and $(i',j')$ are independently random. Thus, they each have probability $p_1$ to hit intersecting sets.

Similarly, the probability that the shortest path has length 5 is $2p_1(1-p_1)$. Let $P_5$ be the probability that the distance oracle answers 5 *conditioned* on the shortest path having length 5. The algorithm *must* answer 3 or 5 if a length-3 path exists, and cannot if the shortest path is $\geq 7$. Then, the overall probability that it answers 5 is $p_1^2 + P_5 \cdot 2p_1(1-p_1)$.

Now we switch to the view of the set intersection algorithm. If the true answer to the query is "disjoint," we will have a path of length 5 with probability $p_1$, and a path of length $\geq 7$ otherwise. If the true answer is "intersect," we will have a path of length 3 with probability $p_1$, and a path of length 5 otherwise. Since the distance oracle cannot predict $R$, it must behave the same whether a path of length 5 arises for the first reason (the query sets are disjoint, but in the manufactured input, the query is positive) or the second reason (the query sets intersect, but in the manufactured input the query is negative). Thus, if the original query should be answered "disjoint," the probability that a path of length 5 is reported is $p_1 \cdot P_5$. If the original query should be answered "intersect," the probability that a path of length 3 or 5 is reported is $p_1 + (1 - p_1)P_5$.

Now observe that $p_1 + (1 - p_1)P_5 \gg p_1 P_5$. Indeed, the gap between them is $p_1 + (1 - 2p_1)P_5$. This expression has minimum value $p_1$ if $p_1 \leq \frac{1}{2}$, and $1 - p_1$ if $p_1 \geq \frac{1}{2}$. Since $p_1 \in [0.1, 0.9]$, the gap is at least $\frac{1}{10}$. Thus, the algorithm can repeat the reduction a constant number of times, and differentiate between "intersect" and "disjoint" queries with any constant success probability. Observe that $p_1$ and $p_5$ are absolute constants, so they can be fixed into the algorithm.

# 5 Conclusions

By breaking the status quo in the field of distance oracles, our work potentially opens a road to a large number of exciting problems:

- Can the space of the Thorup–Zwick distance oracles [TZ05] be improved for approximation greater than 2? For instance, can one achieve approximation 3 with space $o(n^{1.5})$ whenever the graph has $o(n^{1.5})$ edges?
- Can one derive lower bounds for approximation above 2 based on the hardness of set intersection queries? This would significantly improve the lower bound of Sommer et al. [SVY09].
- So far, work has focused on constant-time data structures. It would also be interesting to study the regime of linear-space data structures, allowing polynomial query times. For example, by applying the techniques of Section 3, we can obtain a data structure with linear space that answers 3-approximate distance queries in $O(\sqrt{m})$ time. What else is possible?

# References


[ADD+93] Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

[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.

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of $(\alpha, \beta)$-spanners and purely additive spanners. In *Proc. 16th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005.

[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.

[CP10] Hagai Cohen and Ely Porat. On the hardness of distance oracle for sparse graph. arXiv:1006.1117v1, 2010.

[CSTW09] Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. Compact routing in power-law graphs. In *Proc. 23rd International Symposium on Distributed Computing (DISC)*, pages 379–391, 2009.

[DHZ00] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. See also FOCS'96.

[EP04] Michael Elkin and David Peleg. $(1 + \varepsilon, \beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004. See also STOC'01.

[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.

[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.

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

[TZ06] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proc. 17th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 802–809, 2006.

[Woo06] David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 389–398, 2006.

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.