On the Space Usage of Approximate Distance Oracles with Sub-2 Stretch

Tsvi Kopelowitz
Bar-Ilan University, Ramat-Gan, Israel

Ariel Korin
Bar-Ilan University, Ramat-Gan, Israel

Liam Roditty
Bar-Ilan University, Ramat-Gan, Israel

Abstract

For an undirected unweighted graph $G = (V, E)$ with n vertices and m edges, let $d(u, v)$ denote the distance from $u \in V$ to $v \in V$ in G. An (α, β)-stretch approximate distance oracle (ADO) for G is a data structure that given $u, v \in V$ returns in constant (or near constant) time a value $\hat{d}(u, v)$ such that $d(u, v) \leq \hat{d}(u, v) \leq \alpha \cdot d(u, v) + \beta$, for some reals $\alpha > 1, \beta$.

Thorup and Zwick [34] showed that one cannot beat stretch 3 with subquadratic space (in terms of n) for general graphs. Pătraşcu and Roditty [27] showed that one can obtain stretch 2 using $O(m^{1/3}n^{4/3})$ space, and so if m is subquadratic in n then the space usage is also subquadratic. Moreover, Pătraşcu and Roditty [27] showed that one cannot beat stretch 2 with subquadratic space even for graphs where $m = \tilde{O}(n)$, based on the set-intersection hypothesis.

In this paper we explore the conditions for which an ADO can beat stretch 2 while using subquadratic space. In particular, we show that if the maximum degree in G is $\Delta_G \leq O(n^{1/k-\varepsilon})$ for some $0 < \varepsilon \leq 1/k$, then there exists an ADO for G that uses $\tilde{O}(n^{2-\frac{k\varepsilon}{3}})$ space and has a $(2, 1-k)$-stretch. For $k = 2$ this result implies a subquadratic sub-2 stretch ADO for graphs with $\Delta_G \leq O(n^{1/2-\varepsilon})$.

Moreover, we prove a conditional lower bound, based on the set intersection hypothesis, which states that for any positive integer $k \leq \log n$, obtaining a sub-$\frac{k+2}{k}$ stretch for graphs with $\Delta_G = \Theta(n^{1/k})$ requires $\tilde{\Omega}(n^2)$ space. Thus, for graphs with maximum degree $\Theta(n^{1/2})$, obtaining a sub-2 stretch requires $\tilde{\Omega}(n^2)$ space.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Graph algorithms, Approximate distance oracle, data structures, shortest path

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.101

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2310.12239`

Funding This work was supported in part by the Israel Science Foundation (ISF) grant 1926/19 and by the United States - Israel Binational Science Foundation (BSF) grants 2018364, 2020356.

1 Introduction

One of the most fundamental and classic problems in algorithmic research is the task of computing distances in graphs. Formally, given an undirected unweighted graph $G = (V, E)$, $|V| = n$ and $|E| = m$, the distance between two vertices $u, v \in V$, denoted $d(u, v)$, is the length of a shortest path between u and v. A central problem in distance computations is the *all-pairs shortest paths* (APSP) problem [18, 10, 20, 16, 37, 12, 22] where the objective is to compute the distances between every pair of vertices in the graph. A main disadvantage in handling the output of the APSP problem is that storing the distances between every

© Tsvi Kopelowitz, Ariel Korin, and Liam Roditty;
licensed under Creative Commons License CC-BY 4.0

pair of vertices in the graph requires $\Omega(n^2)$ space. As in many other problems in computer science, the lack of space efficiency in solving the APSP problem has motivated researchers to search for a tradeoff between space and accuracy. As a result, one central form of the APSP problem emerging from this line of research is constructing an *approximate distance oracle* where we sacrifice accuracy for space efficiency.

Approximate Distance Oracles. An approximate distance oracle (ADO) is a space efficient data structure that produces distance estimations between any two vertices in the graph in constant or near-constant time. Formally, given two vertices, $u, v \in V$, an ADO returns an estimation $\hat{d}(u,v)$ for the distance between u and v that satisfies: $d(u,v) \le \hat{d}(u,v) \le \alpha \cdot d(u,v)$, for some real $\alpha > 1$ which is called the *stretch* of the ADO. If the estimation of the ADO satisfies $d(u,v) \le \hat{d}(u,v) \le \max\{d(u,v),\ \alpha \cdot d(u,v) + \beta\}$ for some reals $\alpha > 1$ and β (which can be negative), we say that the stretch of the ADO is an (α, β)-stretch.

ADO for general graphs. ADOs were originally presented by Thorup and Zwick [34] who designed a randomized algorithm that for any positive integer k constructs an ADO for *weighted* undirected graphs in $O(kmn^{1/k})$ time that uses $O(kn^{1+1/k})$ space and returns a $2k-1$-stretch in $O(k)$ query time.

Thorup and Zwick [34] showed that the space usage of their ADO construction for their given stretch is optimal for *general graphs* based on the girth conjecture of Erdős. Moreover, for stretch 3 (when $k = 2$), the appropriate case of the girth conjecture is known to be true (due to complete bipartite graphs), and so the quadratic (in n) space lower bound for this case is unconditional. Notice that constructing an exact distance oracle in quadratic space is trivial.

In the case where one allows for an additive error, Pătraşcu and Roditty [27] designed an algorithm which constructs a $(2,1)$-stretch ADO for unweighted graphs using $O(n^{5/3})$ space and $O(1)$ query time. Their result demonstrates that in such a case the multiplicative error can be reduced while still using subquadratic space.

ADO for sparse graphs. The (conditional) lower bound of Thorup and Zwick [34] does not apply to sparser graphs with $m = o(n^{1+1/k})$, and indeed additional results show that it is possible to use subquadratic space and return a sub-3 stretch in such cases. Specifically, Pătraşcu and Roditty [27], designed an algorithm that constructs a 2-stretch ADO using $O(m^{1/3}n^{4/3})$ space, and so for subquadratic m the space usage is subquadratic. Pătraşcu, Roditty and Thorup [28] presented additional tradeoffs for sub-3 stretch using subquadratic space for graphs where $m = \tilde{O}(n)$[1]. Roditty and Tov [30] improved the stretch of the ADO presented by Thorup and Zwick [34] while using the same space for graphs with $m = \tilde{O}(n)$.

Conditional lower bounds and set-intersection. Pătraşcu and Roditty [27] proved a lower bound for the space usage of sub-2 stretch ADOs (i.e., ADOs which satisfy $d(u,v) \le \hat{d}(u,v) < 2d(u,v)$) that holds (even) for sparse graphs, conditioned on the space usage for data structures that solve the following set intersection problem.

▶ **Problem 1.** *Let $X = \log^c N$ for a large enough constant c. Construct 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?" in constant time.*

[1] Throughout this paper we use $\sim$ when suppressing poly-logarithmic factors in asymptotic complexities.

The lower bound proof by Pătraşcu and Roditty [27] is based on the following hypothesis.

▶ **Hypothesis 2** ([27, 31, 17]). *A data structure that solves Problem 1 requires $\tilde{\Omega}(N^2)$ space.*

Since understanding the reduction by Pătraşcu and Roditty [27] is useful for our results, we provide an overview of their reduction tailored to our needs. Given an instance of Problem 1, we construct a 3 *layered* graph, where edges are only between adjacent layers, as follows. The first layer is $V_L = \{v_1, \dots, v_N\}$, the second layer is $V_M = X$, and the third layer is $V_R = \{u_1, \dots, u_N\}$. Vertices v_i and u_i represent S_i, and so for each set S_i and each $x \in S_i$, we add edges (v_i, x) and (x, u_i). Notice that the graph contains $\Theta(N)$ vertices. It is straightforward to observe that $S_i \cap S_j \neq \emptyset$ if and only if there is a path of length 2 between v_i and u_j. Moreover, since the graph is a 3 layered graph and the representatives of the sets are at the outer layers, there are no paths of length 3 between representatives of sets. Thus, one can solve Problem 1 using a solution to the following problem (for $a = 2$ and $b = 4$).

▶ **Problem 3.** *For positive integers a and b, an (a,b)-*distinguisher oracle *for a graph $G = (V, E)$, is a data structure that, given $u, v \in V$ establishes in constant time whether $d(u,v) \leq a$ or $d(u,v) \geq b$. If $a < d(u,v) < b$ then the data structure can return any arbitrary answer.*

We conclude that a $(2,4)$-distinguisher oracle that uses $f(n)$ space can be used to solve Problem 1 using $f(N)$ space by applying the oracle onto the 3 layered graph. Finally, since a sub-2 stretch ADO is a $(2,4)$-distinguisher oracle, any sub-2 stretch ADO must use at least $\Omega(n^2)$ space.

1.1 Main results: When can we beat stretch 2 with subquadratic space?

The line of work by [27, 28, 30] is a natural research path given the observation that the (conditional) lower bounds of [34] apply only to graphs with $m = \Omega(n^{1+1/k})$. Similarly, a natural goal, which we address in this paper, is to understand for which families of graphs can an ADO beat stretch 2 using subquadratic space. In particular, the conditional lower bound proof of Pătraşcu and Roditty [27] does not apply to graphs with maximum degree of $n^{\frac{1}{2}-\Omega(1)}$, since in such graphs the number of paths of length 2 is $n^{2-\Omega(1)}$, and so constructing a subquadratic space $(2,4)$-distinguisher oracle is straightforward (by explicitly storing all length 2 paths).

Thus, a natural goal, which we investigate in this paper, is to understand the relationship between the maximum degree of G, denoted by Δ_G, and the best possible stretch obtainable for an ADO using subquadratic space. To address this question, we present two main results. The first result is an upper bound for $\Delta_G = n^{\frac{1}{k}-\Omega(1)}$ which is summarized in the following Theorem.

▶ **Theorem 4.** *For any graph G, positive real constant c, and positive integer k for which $\Delta_G \leq cn^{\frac{1}{k}-\varepsilon}$ for some real $0 < \varepsilon \leq 1/k$, there exists an ADO for G that uses $\tilde{O}(c^k n^{2-\frac{k\varepsilon}{3}})$ space and has a $(2, 1-k)$-stretch.*

For $k = 2$, Theorem 4 implies a subquadratic sub-2 stretch ADO for graphs for which $\Delta_G \leq n^{\frac{1}{2}-\Omega(1)}$.

The second result is a conditional lower bound, conditioned on Hypothesis 2, that applies when $\Delta_G = \Theta(n^{1/k})$, for all integers $k \geq 1$ [2]. The conditional lower bound is summarized in the following Theorem.

[2] The case $k = 1$ was proven by Thorup and Zwick [34] to hold unconditionally. Thus, Theorem 5 focuses on $k \geq 2$.

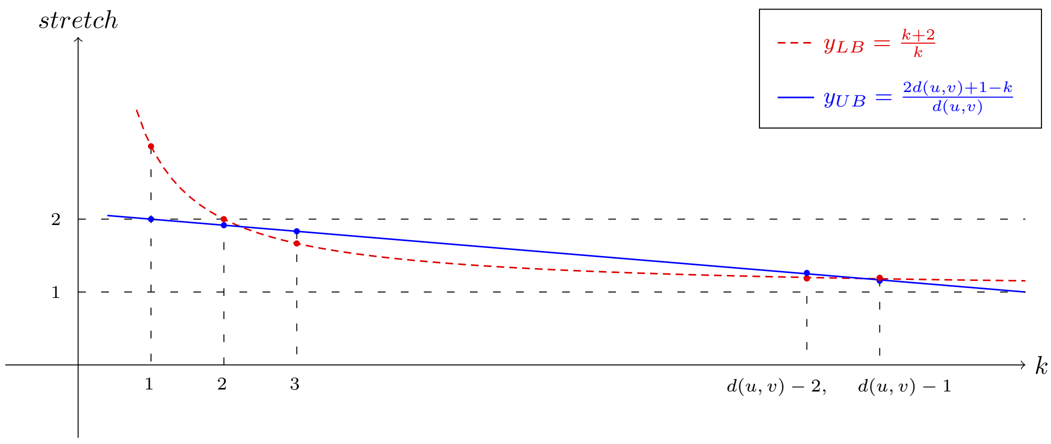

Figure 1 A comparison between the upper bound for graphs with $\Delta_G \leq O(n^{\frac{1}{k}-\varepsilon})$ (Theorem 4) and the lower bound for graphs with $\Delta_G = \Theta(n^{\frac{1}{k}})$ (Theorem 5). The figure demonstrates the stretch as a function of k for a fixed (sufficiently large) value of $d(u,v)$. The curves intersect at $2 < k_1 < 3$ and at $d(u,v) - 2 < k_2 < d(u,v) - 1$ which implies that for $k = 1, 2$ the ADO from Theorem 4 for graphs with $\Delta_G \leq O(n^{\frac{1}{k}-\varepsilon})$ produces a better stretch than the best possible stretch for graphs with $\Delta_G = \Theta(n^{\frac{1}{k}})$. On the other hand, the intersection point k_2 does not provide strong enough results; see the discussion following the statement of Theorem 5.

▶ **Theorem 5.** *Let $2 \leq k \leq \log n$. Assuming Hypothesis 2, a sub-$\frac{k+2}{k}$ stretch ADO for graphs with n vertices and maximum degree $\Theta(n^{1/k})$ must use $\tilde{\Omega}(n^2)$ space.*

When $k = 2$, the lower bound of Theorem 5 implies that Theorem 4 is essentially optimal, in the sense that there is no ADO for graphs with a maximum degree of $\Theta(n^{1/2})$ that still uses subquadratic space and has a sub-2 stretch.

For larger values of k, although the upper and lower bounds are defined only for integer values of k, the values lie on two curves: $y_{UB} = \frac{2d(u,v)+1-k}{d(u,v)}$ for the upper bound and $y_{LB} = \frac{k+2}{k}$ for the lower bound. See Figure 1 for a depiction and comparison of the two curves for a fixed value of $d(u,v)$. When $d(u,v) \leq 6$ (which is not the case shown in Figure 1), the upper bound will always produce a stretch which is equal or smaller to the lower bound. Otherwise, the curves intersect at $2 < k_1 < 3$ and $d(u,v) - 2 < k_2 < d(u,v) - 1$. We emphasize that the bounds on k_1 are independent of $d(u,v)$, and so for $k = 1, 2$, the fact that the upper bound curve is beneath the lower bound curve is relevant for all for all possible distances. On the other hand, the bounds on k_2 depend on $d(u,v)$, and so while it is true that for a fixed value of large enough $d(u,v)$ there are many values of k for which the ADO of Theorem 4 provides an approximation which beats the lower bound, it is also true that for any integer value of $k \geq 3$ there will always exist some distances for which the upper bound is above the lower bound. Thus, the intersection at k_2 is unfortunately not meaningful enough.

1.2 Organization

The rest of this paper is organized as follows. In Section 1.3 we provide an overview of the main ideas used in this paper. In Section 1.4 we survey some additional related work. In Section 2 we provide some definitions that are used in the more technical parts of the paper. In Section 3 we prove some useful lemmas that are used in the proof of Theorem 4, which is described in Section 4 together with the construction of our new ADO. In Section 5 we prove Theorem 5. Finally, in Section 6 we provide some conclusions and describe a natural open problem.

1.3 Overview of Results and Techniques

In this section we describe an overview of the intuition and techniques used to obtain our main results.

1.3.1 Upper Bound: A New ADO

Since our new ADO is based on the ADO of Agarwal and Godfrey [4], that has a $(2, 1)$-stretch and uses $\tilde{O}(n^{5/3})$ space (which simplifies the ADO of [27]), we provide an overview of the construction of their ADO.

The ADO constructed by Agarwal and Godfrey [4] uses the concept of *bunches* and *clusters* introduced by Thorup and Zwick [33]. Following the conventions of Thorup and Zwick [33, 34], for a vertex $v \in V$ and set $A \subseteq V$, let $p_A(v)$ be the vertex in A which is closest to v (breaking ties arbitrarily). The *bunch* $B_A(v)$ of v with respect to A is defined as $B_A(v) = \{w \in V \mid d(v, w) < d(v, p_A(v))\}$. The *cluster* $C_A(w)$ of w with respect to A is defined as $C_A(w) = \{v \in V \mid d(w, v) < d(v, p_A(v))\}$. We omit A from the notation when it is clear from context. Thorup and Zwick [33] presented an algorithm that computes a set A of size $\tilde{O}(s)$ such that $|B(v)|, |C(v)| \leq 4n/s$, for every $v \in V$. The ADO of Agarwal and Godfrey [4] uses a hitting set A of size $\tilde{O}(n^{2/3})$ such that for every $v \in V$, $|B(v)|, |C(v)| \leq O(n^{1/3})$.

Given two vertices $u, v \in V$, the ADO first tests whether $B(u) \cap B(v) \neq \emptyset$, and, if so, then the ADO returns the exact distance $d(u, v)$. The method for testing whether the two bunches intersect is based on the observation (which follows from the definitions of bunch and cluster) that $B(u) \cap B(v) \neq \emptyset$ if and only if $u \in C(B(v))$[3]. Thus, each vertex $v \in V$ stores the exact distances to all vertices in $C(B(v))$, and now, the case of $B(u) \cap B(v) \neq \emptyset$ costs constant time and returns an exact distance. To deal with the case of $B(u) \cap B(v) = \emptyset$, the oracle stores the distances of pairs in $V \times A$, and the ADO returns the minimum of either the length of the shortest path between u and v passing through $p(u)$ or the length of the shortest path between u and v passing through $p(v)$. The space usage is $O(n^{5/3})$ for storing $C(B(v))$ for every $v \in V$, and $\tilde{O}(n|A|) = \tilde{O}(n^{5/3})$ for storing the distances for all pairs in $V \times A$.

Intuition for the new ADO. Our new ADO construction is based on the following intuition regarding the ADO construction of Agarwal and Godfrey [4]. If we enlarge $B(u)$ by moving $p(u)$ to a further vertex (from u), then we would increase the likelihood of $B(u) \cap B(v) \neq \emptyset$, and so the ADO would return exact distances for more pairs of vertices. However, in such a case, the quality guarantee on the stretch obtained by approximating $d(u, v)$ with the shortest path from u to v that passes through $p(u)$ becomes worse. Part of the challenge is to balance the size of $B(u)$ which affects the usefulness of the intersections and the role of $p(u)$ when approximating the distances.

Our approach, intuitively, is to separate the definition of $p(u)$ used for the approximations and the set chosen for the intersections. Specifically, the definition of $p(u)$ remains unchanged relative to A (we do however change the size of A), but instead of testing whether $B(u) \cap B(v) \neq \emptyset$, we use a larger set $N(u)$ (which contains $B(u)$), and test whether $N(u) \cap B(v) \neq \emptyset$ (or $B(u) \cap N(v) \neq \emptyset$). Testing whether $N(u) \cap B(v) \neq \emptyset$ is implemented by storing all of the distances between u and vertices in $C(N(u))$. We remark that one may consider the possibility of testing whether $N(u) \cap N(v) \neq \emptyset$ instead of testing whether $N(u) \cap B(v) \neq \emptyset$, however, such an approach seems to require too much space.

[3] For $S \subseteq V$, let $C(S) = \bigcup_{u \in S} C(u)$.

Recall that Agarwal and Godfrey [4] obtained a $(2,1)$-stretch. In our algorithm, we choose $N(u)$ in such a way that when $N(u) \cap B(v) = \emptyset$ then $\min_{x \in B(u), y \in B(v)}\{d(x,y)\} > k$ if $\Delta_G \leq n^{\frac{1}{k}-\Omega(1)}$, which ends up reducing the additive component of the stretch by at least k (see Claim 13). Thus, the approximation of the ADO is always at most $(2, 1-k)$, which is less than stretch 2 for $k \geq 2$.

1.3.2 Conditional Lower bound

In Section 5 we prove the following lemma, which directly implies Theorem 5 since a sub-$\frac{k+2}{k}$ stretch ADO is also a $(k, k+2)$-distinguisher oracle.

▶ **Lemma 6.** *Let $2 \leq k \leq \log n$. Assuming Hypothesis 2, any $(k, k+2)$-distinguisher oracle for graphs with n vertices and maximum degree $\Theta(n^{1/k})$ must use $\tilde{\Omega}(n^2)$ space.*

Notice that it is straightforward to construct a $(k, k+2)$-distinguisher oracle for graphs with n vertices and maximum degree $\Theta(n^{\frac{1}{k}-\varepsilon})$ in $O(n^{2-k\varepsilon})$ space by storing all pairs of vertices at distance exactly k. Thus, Lemma 6 shows that such a construction is essentially optimal.

The challenges. There are two issues that need to be addressed in order to extend the reduction by Pătraşcu and Roditty [27] in a way that proves Lemma 6. The first is to adjust the distances so that $S_i \cap S_j \neq \emptyset$ if and only if $d(v_i, u_j) = k$, and otherwise, $d(v_i, u_j) \geq k+2$. The second issue is that the degrees of vertices in V_M need to be adjusted to be at most $\tilde{O}(N^{1/k})$. In order to simplify our intuitive explanation, we focus our attention to the special case where $X = \{x\}$ has only one element.

One straightforward way of dealing with the first issue is to replace vertex $x \in V_M$ with a path $P_x = (w_1, \ldots, w_{k-1})$ of length $k-2$, and for each S_i that contains x we add edges (v_i, w_1) and (w_{k-1}, u_i). Thus, the constructed graph would be a $k+1$ layered graph. The number of vertices in such a graph is $O(k+N) = O(N)$, and the distance between vertices in the first and last layers are $k + 2q$ for some integer $q \geq 0$. However, we still need to address the second issue of bounding the maximum degree, since w_1 and w_{k-1} may have a very high degree corresponding to the number of sets containing x.

On the other hand, one initial idea (that does not work) for dealing with the second issue is to replace $x \in V_M$ (in the original 3 layered graph) with N vertices $y_1, y_2, \ldots, y_N$, and for each S_i that contains x we add edges (v_i, y_i) and (y_i, u_i). Now the maximum degree of each node is constant, however, for $i \neq j$ such that $x \in S_i \cap S_j$, there is no path from v_i to u_j. This idea is missing the functionality of the path P_x which allows us to connect more than one pair of vertices from $V_L \times V_R$.

Combining approaches. Our reduction makes use of an underlying $k+1$ layered *infrastructure graph* $\mathcal{L}$, commonly known as the *butterfly graph* (see [26, 28]), which has the following three properties: (i) each layer contains N vertices, (ii) there is a path of length k from every vertex in the first layer to every vertex in the last layer, and (iii) the degree of every vertex is at most $2N^{1/k}$. The layers of $\mathcal{L}$ are numbered 0 to k. The vertices in each layer are (separately) indexed with integers from 1 to N, and the construction of $\mathcal{L}$ is based on the base $N^{1/k}$ representation of the these indices: for $1 \leq t \leq k$, vertices from layer $t-1$ are connected with vertices from layer t if and only if the base $N^{1/k}$ representation of their corresponding indices are the same, except for possibly the t'th digit. Similar to before, we denote the first layer of $\mathcal{L}$ by $V_L = \{v_1, \ldots, v_N\}$ and the last layer by $V_R = \{u_1, \ldots, u_N\}$.

Finally, we construct a $k + 1$ layered graph G_x which is intuitively obtained by removing from $\mathcal{L}$ edges touching either v_i or u_i for every S_i that does not contain x. Thus, in G_x, if $x \in S_i \cap S_j$ then there is a path of length k from v_i to u_j in G_x, and otherwise, there is no path from v_i to u_j in G_x.

We remark that in the general case, where $|X|$ may be larger than 1, we combine G_x for different $x \in X$ in a special way, and so we may introduce paths from v_i to u_j even if $S_i \cap S_j = \emptyset$. However, since the resulting graph is still a $k+1$ layered graph, and we are interested in paths between vertices in the first layer and vertices in the last layer, the lengths of such paths must be at least $k+2$. Thus, a $(k, k+2)$-distinguisher oracle on the combined graph suffices for solving Problem 1. See Section 5 for more details.

1.4 Additional related work

Different aspects of Thorup and Zwick [34] ADOs were studied since they were introduced for the first time. Chechik [14, 13] reduced the query time from $O(k)$ to $O(1)$, while keeping the stretch and the space unchanged. (See also [36, 23].) Roditty, Thorup, and Zwick [29] presented a deterministic algorithm that constructs an ADO in $\tilde{O}(mn^{1/k})$ time while keeping the stretch and the space unchanged. Baswana and Kavitha [9] presented an algorithm with $O(n^2 \log n)$ running time[4]. Baswana, Goyaland and Sen [8] presented an $\tilde{O}(n^2)$ time algorithm that computes a $(2, 3)$-distance oracle with $\tilde{O}(n^{5/3})$ space. Sommer [32] presented an $\tilde{O}(n^2)$ time algorithm that computes a $(2, 1)$-distance oracle with $\tilde{O}(n^{5/3})$ space. Akav and Roditty [7] presented the first sub-quadratic time algorithm that constructs an ADO with stretch better than 3. They presented an $O(n^{2-\epsilon})$-time algorithm that constructs a ADO with $O(n^{11/6})$ space and $(2 + \epsilon, 5)$-stretch. Chechik and Zhang [15] improved the result of Akav and Roditty [7]. Among their results is an $O(m + n^{1.987})$ time algorithm that constructs an ADO with $(2, 3)$-stretch and $\tilde{O}(n^{5/3})$ space. Following the work by Pǎtraşcu and Roditty [27] who constructed an ADO for unweighted graphs that uses $O(n^{5/3})$ space and returns a $(2, 1)$-stretch in $O(1)$ time, Abraham and Gavoille [2] extended the ADO by Pǎtraşcu and Roditty [27] for all even stretch values, by constructing for any integer $k \geq 2$, an ADO of size $\tilde{O}(n^{1+2/(2k-1)})$ with a $(2k - 2, 1)$-stretch returned in $O(k)$ time. Pǎtraşcu, Roditty and Thorup [28] focused on analyzing sparse graphs where $m = \tilde{O}(n)$ and noted that both the ADOs by Thorup and Zwick [34], and the ADOs by Abraham and Gavoille [2] use a space complexity that can be described by the curve $S(\alpha, m) = \tilde{O}(m^{1+2/(\alpha+1)})$ where α is the stretch of the ADO and m is the number of edges in the graph. Pǎtraşcu, Roditty and Thorup [28] extended the curve $S(\alpha, m)$ to work for non integer stretch values $\alpha > 2$. Although our research focuses on constant query time ADOs, another branch of research includes ADOs that have non constant query time [6, 24, 5, 3, 11].

In the lower bound regime, the problem of constructing a $(2, 4)$-distinguisher oracle was analyzed from the perspective of time complexity as well. For graphs with degree of at most $n^{1/2}$, the problem of determining for each edge in the graph whether it is in a triangle in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$ was shown to be hard under either the 3SUM [25, 21] or APSP [35] hypotheses. Since there exists a standard reduction from the problem of determining for each edge in the graph whether it is in a triangle to the problem of constructing a $(2, 4)$-distinguisher oracle (see [1]), a $(2, 4)$-distinguisher oracle is also hard to construct in subquadratic time for graphs with degree of at most $n^{1/2}$ under either the 3SUM or APSP hypotheses.

[4] For $k = 2$ the query time is $O(\log n)$. For $k > 2$ the query time is $O(k)$.

The problem of constructing a $(k, k+2)$-distinguisher oracle for a general integer $k \geq 2$ was also studied in the past in terms of time complexity. Dor, Halperin and Zwick [19] showed that if all distances in an undirected n vertex graph can be approximated with an additive error of at most 1 in $O(A(n))$ time, then *Boolean matrix multiplication* on matrices of size $n \times n$ can also be performed in $O(A(n))$ time. Dor, Halperin and Zwick [19] conclude that constructing a $(k, k+2)$-distinguisher oracle for an integer $k \geq 2$ is at least as hard as multiplying two Boolean matrices.

2 Preliminaries

Let $d_G(u, v)$ be the distance between vertices u and v in the graph G. The eccentricity of a vertex $v \in V$ in a graph G, denoted by $ecc_G(v)$, is defined as $ecc_G(v) = \max_{u \in V}\{d_G(v, u)\}$. The diameter of G is defined as $diam_G = \max_{v \in V}\{ecc_G(v)\}$ and the radius of G is defined as $rad_G = \min_{v \in V}\{ecc_G(v)\}$.

The eccentricity of a vertex v can be thought of as the distance between v and the last vertex met during a *Breadth First Search* (BFS) of the graph starting at v. Since our goal is to construct an ADO that uses subquadratic space, we cannot afford to store a separate BFS tree for each vertex. Instead, the construction algorithm of the ADO from Theorem 4 will store only a partial BFS tree for each vertex by truncating the BFS scan after some number of vertices. Motivated by this notion of a truncated scan, we introduce the following generalization of eccentricity which turns out to be useful for our purposes.

Let $N_G(v, s)$ be the first s vertices met during a BFS[5] starting from v in the graph G, i.e., the s closest vertices to v (excluding v). If s is not an integer, then let $N(v, s) = N(v, \lfloor s \rfloor)$. For an integer $r \geq 0$, define $L_G(v, r) = \{u \in V \setminus \{v\} \mid d_G(u, v) = r\}$ and $T_G(v, r) = \{u \in V \mid 0 < d_G(u, v) \leq r\}$. Notice that $T_G(v, r) = \bigcup_{i=1}^{r} L_G(v, i)$. For any real $1 \leq s \leq n-1$, define $ecc_G(v, s)$ to be the maximum integer $k \in [0, ecc_G(v)]$ for which $T_G(v, k) \subseteq N_G(v, s)$. Notice that $ecc_G(v, n-1) = ecc_G(v)$. Define $rad_G(s) = \min_{v \in V}\{ecc_G(v, s)\}$. Notice that $rad_G(n-1) = rad_G$. We omit the subscript G when using the definitions above whenever G is clear from context.

3 Useful Lemmas

In this section we prove several useful properties of the graph attributes defined in Section 2 which will be used throughout the paper.

The following observation and corollary address the relationship between $T(v, r)$ and $N(v, s)$, and follow from the definition of BFS.

▶ **Observation 7.** *Let $G = (V, E)$ be an unweighted undirected graph, with $|V| = n$. For any $v \in V$ and integers s and r such that $1 \leq s < n$ and $1 \leq r \leq diam_G$, either (i) $T(v, r) \subset N(v, s)$, (ii) $N(v, s) \subset T(v, r)$, or (iii) $N(v, s) = T(v, r)$*

▶ **Corollary 8.** *Let $G = (V, E)$ be an unweighted undirected graph, with $|V| = n$. For any $v \in V$ and integers s and r such that $1 \leq s < n$ and $1 \leq r \leq diam_G$, (i) if $|T(v, r)| < |N(v, s)|$ then $T(v, r) \subset N(v, s)$, (ii) if $|N(v, s)| < |T(v, r)|$ then $N(v, s) \subset T(v, r)$, and (iii) if $|N(v, s)| = |T(v, r)|$ then $N(v, s) = T(v, r)$.*

The following useful property addresses the relationship between $T(v, r)$ and $N(v, s)$ for the special cases where either $r = ecc(v, s)$ or $r = ecc(v, s) + 1$.

[5] The traversal order of vertices in the same layer during the BFS execution does not matter as long as the order is consistent.

▶ **Property 9.** *Let $G = (V, E)$ be an unweighted undirected graph, with $|V| = n$. For any $v \in V$ and integer s such that $1 \leq s < n$, we have: (i) $T(v, ecc(v, s)) \subseteq N(v, s)$, and (ii) if $s < n - 1$, then $T(v, ecc(v, s) + 1) \not\subseteq N(v, s)$.*

Proof. By definition, $ecc(v, s)$ is the largest integer $k \in [0, ecc(v)]$ for which $T(v, k) \subseteq N(v, s)$. Thus, (i) $T(v, ecc(v, s)) \subseteq N(v, s)$, and (ii) if $ecc(v, s) < ecc(v)$ then $T(v, ecc(v, s) + 1) \not\subseteq N(v, s)$. If $s < n - 1$, it must be that $ecc(v, s) < ecc(v)$, since if we assume towards a contradiction that $ecc(v, s) = ecc(v)$ for some $s < n - 1$ then $T(v, ecc(v, s)) = T(v, ecc(v)) = V\backslash\{v\} = N(v, n-1)$ but on the other hand, by definition of $ecc(v, s)$, we have $T(v, ecc(v, s)) \subseteq N(v, s) \subset N(v, n - 1)$, which is a contradiction. ◀

The following lemma states that $ecc_G(v, s)$ exhibits a behavior that is similar to the behavior of the distance function which cannot decrease when removing edges and vertices from G.

▶ **Lemma 10.** *Let $G = (V, E)$ be an unweighted undirected graph, $V' \subseteq V$, and let G' be the subgraph of G induced by the vertices in V'. For any vertex $v \in V'$ and for any integer s such that $1 \leq s < |V'|$, it holds that $ecc_G(v, s) \leq ecc_{G'}(v, s)$.*

Proof. Given an integer $1 \leq s < |V'|$, let $r = ecc_G(v, s)$ and $r' = ecc_{G'}(v, s)$. We want to show that $r \leq r'$. By definition of $ecc_{G'}(v, s)$, $r' = ecc_{G'}(v, s)$ is the largest value for which $T_{G'}(v, r') \subseteq N_{G'}(v, s)$. Thus, in order to show that $r \leq r'$, it suffices to show that $T_{G'}(v, r) \subseteq N_{G'}(v, s)$.

For any vertex pair $u, w \in V'$, we have $d_G(u, w) \leq d_{G'}(u, w)$ since G' is a subgraph of G. Thus,

$$\begin{aligned} T_{G'}(v, r) &= \{u \in V' \mid 0 < d_{G'}(v, u) \leq r\} \\ &\subseteq \{u \in V \mid 0 < d_G(v, u) \leq r\} \\ &= T_G(v, r) \\ &\underset{\substack{\uparrow \\ \text{Property 9}}}{\subseteq} N_G(v, s). \end{aligned}$$

This implies that $|T_{G'}(v, r)| \leq |N_G(v, s)| = s = |N_{G'}(v, s)|$. By Corollary 8, since $|T_{G'}(v, r)| \leq |N_{G'}(v, s)|$ then $T_{G'}(v, r) \subseteq N_{G'}(v, s)$, as required. ◀

3.1 The Logarithmic-Like Behavior of Eccentricity

In the following lemma, which is an important ingredient in the analysis of our new ADO, we show that $ecc(v, s)$ satisfies a logarithmic-like behavior. Specifically, $\log(xy) = \log x + \log y$. The reason for this behavior is that the number of vertices in each layer of a BFS tree expands in a similar way to an exponential function. For a tree-graph G with minimum degree δ rooted at a vertex v, for integers $0 \leq i < t < ecc(v)$ it holds that $|L(v, t)| \geq \delta^i \cdot |L(v, t - i)|$. Since the number of vertices in every layer of the rooted tree grows exponentially, the eccentricity $ecc(v, s)$ grows logarithmically (in relation to s). Unlike in trees where the expansion of the number of vertices in every layer of a BFS can be analyzed using δ, for general graphs, in order to achieve a lower bound for the expansion rate of the eccentricity of the vertices, we use $rad_G(s)$ instead.

▶ **Lemma 11.** *Let $G = (V, E)$ be an unweighted undirected graph, with $|V| = n$. For any vertex $v \in V$ and integers $s_1, s_2 \geq 1$ such that $s_1(s_2 + 1) < n - 1$, it holds that $ecc_G(v, s_1(s_2 + 1)) \geq ecc_G(v, s_1) + rad_G(s_2)$.*

Proof. Assume towards a contradiction that $ecc_G(v, s_1(s_2 + 1)) < ecc_G(v, s_1) + rad_G(s_2)$. Thus, $ecc_G(v, s_1(s_2 + 1)) + 1 \leq ecc_G(v, s_1) + rad_G(s_2)$.

Let T_{BFS} be a BFS tree rooted at v in graph G. Let $\ell = |L(v, ecc_G(v, s_1))|$ and let $u_1, u_2, \ldots, u_\ell$ be the vertices in $L(v, ecc_G(v, s_1))$. For any i, where $1 \leq i \leq \ell$, let V_i be the set of descendant of u_i in T_{BFS} and let G_i be the graph induced by V_i in G[6].

Let $\mu = \min_{i\in[1,\ell]}\{ecc_{G_i}(u_i, s_2)\}$. We will show that:

$$\{u \in V \mid ecc_G(v, s_1) < d_G(v, u) \leq ecc_G(v, s_1) + \mu\} \subseteq \bigcup_{i=1}^{\ell} N_{G_i}(u_i, s_2). \tag{1}$$

Let $w \in \{u \in V \mid ecc_G(v, s_1) < d_G(v, u) \leq ecc_G(v, s_1) + \mu\}$. By the definition of BFS, since $d_G(v, w) > ecc_G(v, s_1)$, w must be a descendant of some vertex u_j, and so $w \in V_j$. Since T_{BFS} is a shortest path tree rooted at v and since $w \in V_j$, it must be that $d_G(v, w) = d_G(v, u_j)+d_G(u_j, w) = ecc_G(v, s_1)+d_G(u_j, w)$. By definition of w, $d_G(v, w) \leq ecc_G(v, s_1)+\mu$. Thus, $ecc_G(v, s_1) + d_G(u_j, w) \leq ecc_G(v, s_1) + \mu$ and so $d_G(u_j, w) \leq \mu$.

By definition, $T_{G_j}(u_j, ecc_{G_j}(u_j, s_2)) = \{x \mid x \in V_j \wedge 0 < d_{G_j}(u_j, x) \leq ecc_{G_j}(u_j, s_2)\}$. Since T_{BFS} is a shortest path tree, for any $y \in V_j$ it must be that $d_{G_j}(u_j, y) = d_G(u_j, y)$. Thus, $T_{G_j}(u_j, ecc_{G_j}(u_j, s_2)) = \{x \mid x \in V_j \wedge 0 < d_G(u_j, x) \leq ecc_{G_j}(u_j, s_2)\}$.

Since $w \in V_j$, and since $d_G(u_j, w) \leq \mu \leq ecc_{G_j}(u_j, s_2)$, then $w \in T_{G_j}(u_j, ecc_{G_j}(u_j, s_2))$. By Property 9, $T_{G_j}(u_j, ecc_{G_j}(u_j, s_2)) \subseteq N_{G_j}(u_j, s_2)$. It follows that every vertex in $\{u \in V \mid ecc_G(v, s_1) < d_G(v, u) \leq ecc_G(v, s_1) + \mu\}$ must be included in $N_{G_i}(u_i, s_2)$ for some i, thus confirming Equation (1).

By Property 9, $\{u \in V \mid 0 < d_G(v, u) \leq ecc_G(v, s_1)\} = T(v, ecc_G(v, s_1)) \subseteq N(v, s_1)$. Combining with Equation (1) we have that $\{u \in V \mid 0 < d_G(v, u) \leq ecc_G(v, s_1)\} \cup \{u \in V \mid ecc_G(v, s_1) < d_G(v, u) \leq ecc_G(v, s_1) + \mu\} \subseteq N(v, s_1) \cup \left(\bigcup_{i=1}^{\ell} N_{G_i}(u_i, s_2)\right)$, and so:

$$\begin{aligned}\{u \in V \mid 0 < d_G(v, u) \leq ecc_G(v, s_1) + \mu\} &= T(v, ecc_G(v, s_1) + \mu)\\ &\subseteq N(v, s_1) \cup \left(\bigcup_{i=1}^{\ell} N_{G_i}(u_i, s_2)\right).\end{aligned}$$

Now,

$$\begin{aligned}ecc_G(v, s_1) + \mu &\underset{\substack{\uparrow\\ \text{definition of } \mu}}{=} ecc_G(v, s_1) + \min_{i\in[1,\ell]}\{ecc_{G_i}(u_i, s_2)\}\\ &\underset{\substack{\uparrow\\ \text{by Lemma 10}}}{\geq} ecc_G(v, s_1) + \min_{i\in[1,\ell]}\{ecc_G(u_i, s_2)\}\\ &\geq ecc_G(v, s_1) + \min_{u\in V}\{ecc_G(u, s_2)\}\\ &\underset{\substack{\uparrow\\ \text{definition of } rad_G(s_2)}}{=} ecc_G(v, s_1) + rad_G(s_2)\\ &\underset{\substack{\uparrow\\ \text{assumption}}}{\geq} ecc_G(v, s_1(s_2 + 1)) + 1.\end{aligned}$$

[6] It is important to note that for all scans referenced in this proof, which include a BFS procedure of G starting at v and BFS procedures of G_i starting at u_i for $1 \leq i \leq \ell$, we require a consistent order of scanning, i.e., that for a given $1 \leq i \leq \ell$, and vertices $x, x' \in V_i \subseteq V$, if x is scanned before x' in G then x should also be scanned before x' in G_i (and vice versa). This is a valid requirement since for any vertices $y, y' \in V_i \subseteq V$, $d_G(v, y) \leq d_G(v, y')$ if and only if $d_{G_i}(u_i, y) \leq d_{G_i}(u_i, y')$.

Thus, $T(v, ecc_G(v, s_1(s_2+1))+1) \subseteq T(v, ecc_G(v, s_1)+\mu) \subseteq N(v, s_1) \cup \left(\bigcup_{i=1}^{\ell} N_{G_i}(u_i, s_2)\right)$.

Notice that $\ell \le s_1$, since, by Property 9, $\ell = |L(v, ecc_G(v, s_1))| \le |T(v, ecc_G(v, s_1))| \le N(v, s_1) = s_1$. Therefore,

$$\begin{aligned}
|T(v, ecc_G(v, s_1(s_2+1))+1)| &\le |N(v, s_1) \cup \left(\bigcup_{i=1}^{\ell} N_{G_i}(u_i, s_2)\right)| \\
&\le |N(v, s_1)| + \sum_{i=1}^{\ell} |N_{G_i}(u_i, s_2)| \\
&\le s_1 + \ell \cdot s_2 \\
&\le s_1 + s_1 \cdot s_2 \\
&= s_1(1+s_2) \\
&= |N(v, s_1(s_2+1))|.
\end{aligned}$$

By Corollary 8, it follows that $T(v, ecc_G(v, s_1(s_2+1))+1) \subseteq N(v, s_1(s_2+1))$, which contradicts Property 9. ◀

4 The new ADO

In this section we prove Theorem 4 by introducing a new ADO which uses subquadratic space and produces a $(2, 1-k)$-stretch for graphs for which $\Delta_G \le O(n^{1/k-\varepsilon})$ for a positive integer k and real constant $0 < \varepsilon \le 1/k$. The ADO is parameterized by a parameter $0 \le \alpha < 1/3$ which quantifies the tradeoff between the space and the stretch of the ADO. When $\alpha = 0$ the ADO is very similar to the ADO of Agarwal and Godfrey [4] which uses $\tilde{O}(n^{5/3})$ space and has a $(2,1)$-stretch. For $0 < \alpha < 1/3$, the ADO uses additional space and is able to improve the stretch of the ADO for the family of graphs for which $\Delta_G \le O(n^{3\alpha/k})$.

4.1 The Construction Algorithm

The description of our construction algorithm follows the notations and definitions described in Section 1.3.1. The construction begins with an algorithm of Thorup and Zwick [33] that computes a set A of size $\tilde{O}(s)$ such that $|B(v)|, |C(v)| \le 4n/s$, for every $v \in V$. In our case we set $s = n^{2/3+\alpha}$, thus $|A| = \tilde{O}(n^{2/3+\alpha})$ and $|B(v)|, |C(v)| \le 4n^{1/3-\alpha}$, for every $v \in V$.

For every vertex $v \in V$, the ADO explicitly stores the distances between v and every vertex in $C(N(v, 16\hat{c}\,n^{1/3+2\alpha}))$ for some constant $\hat{c}$ to be decided later. In addition, for every vertex $v \in V$ the ADO stores $p(v)$, $d(v, p(v))$ and the distances between v and every vertex in A.

A distance query between vertices u and v is answered as follows. If one of the following conditions holds (i) $u \in A$ or $v \in A$, (ii) $u \in C(N(v, 16\hat{c}\,n^{1/3+2\alpha}))$ or $v \in C(N(u, 16\hat{c}\,n^{1/3+2\alpha}))$, then the exact distance is returned. Otherwise, the ADO returns $\hat{d}(u, v) = \min\{d(u, p(u)) + d(p(u), v),\ d(u, p(v)) + d(p(v), v)\}$. Notice that the query time is constant.

In Claim 12, we show that the space complexity of the ADO is $\tilde{O}(\hat{c}n^{5/3+\alpha})$ and in Claim 13, we show that the ADO satisfies a $(2, 1 - rad(\hat{c}n^{3\alpha}))$-stretch.

▷ Claim 12. The space complexity of the ADO is $\tilde{O}(\hat{c}n^{5/3+\alpha})$.

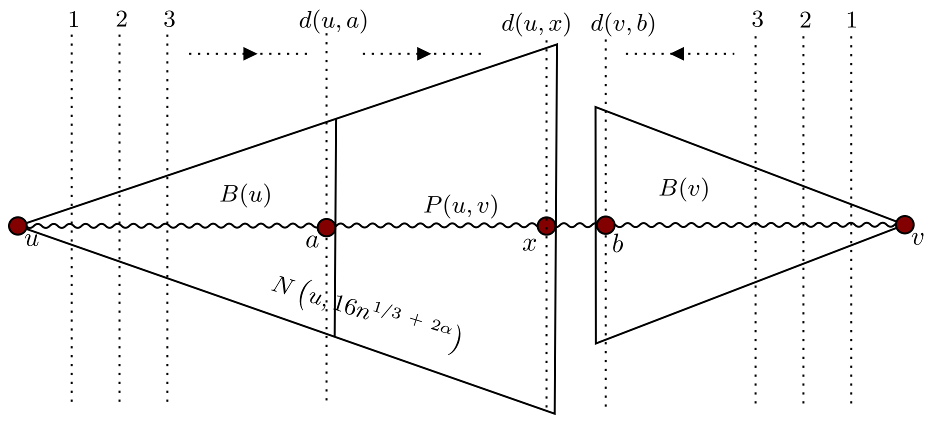

Figure 2 A query for $u, v \in V$ in the case that $N(u, 16\hat{c}n^{1/3+2\alpha}) \cap B(v) = \emptyset$. Since $x \in N(u, 16\hat{c}n^{1/3+2\alpha})$, $b \in B(v)$ and x and b are both on a shortest path between u and v, it must be that $d(u,x) \le d(u,b) - 1$.

Proof. Storing $p(v)$, $d(v, p(v))$ and the distances between v and every vertex in A, for all vertices $v \in V$, uses $\tilde{O}(n \cdot n^{2/3+\alpha}) = \tilde{O}(n^{5/3+\alpha})$ space. As mentioned in the construction phase, $|B(v)|, |C(v)| \le O(n^{1/3-\alpha})$ for every $v \in V$. Thus, storing the distances between every vertex v and $C(N(v, 16\hat{c}n^{1/3+2\alpha}))$ requires $O(n \cdot n^{1/3-\alpha} \cdot 16\hat{c}n^{1/3+2\alpha}) = \tilde{O}(\hat{c}n^{5/3+\alpha})$ space as well, leading to an overall space complexity of $\tilde{O}(\hat{c}n^{5/3+\alpha})$. ◁

▷ Claim 13. The distance estimation $\hat{d}(u,v)$ returned by the ADO satisfies $d(u,v) \le \hat{d}(u,v) \le 2d(u,v) + 1 - rad(\hat{c}n^{3\alpha})$.

Proof. Notice that $d(u,v) \le \hat{d}(u,v)$ since the ADO always returns a length of some path in the graph between u and v. It is left to show that $\hat{d}(u,v) \le 2d(u,v) + 1 - rad(\hat{c}n^{3\alpha})$.

If the exact distance is stored in the ADO then $\hat{d}(u,v) = d(u,v)$ and the claim follows. Consider the case that the exact distance is not stored. This implies that $u, v \notin A$ and $v \notin C(N(u, 16\hat{c}n^{1/3+2\alpha}))$. Assume towards a contradiction that $N(u, 16\hat{c}n^{1/3+2\alpha}) \cap B(v) \ne \emptyset$ and let w be a vertex such that $w \in N(u, 16\hat{c}n^{1/3+2\alpha}) \cap B(v)$. From the definitions of bunch and cluster, we have that $w \in B(v)$ if and only if $v \in C(w)$. Thus, $v \in C(w)$, and since $w \in N(u, 16\hat{c}n^{1/3+2\alpha})$, it must be that $v \in C(N(u, 16\hat{c}n^{1/3+2\alpha}))$ which is a contradiction. Thus, we have that $N(u, 16\hat{c}n^{1/3+2\alpha}) \cap B(v) = \emptyset$.

Let $P(u,v)$ be a shortest path between u and v. Let a be the furthest vertex from u in $B(u) \cap P(u,v)$, let x be the furthest vertex from u in $N(u, 16\hat{c}n^{1/3+2\alpha}) \cap P(u,v)$ and let b be the furthest vertex from v in $B(v) \cap P(u,v)$ (see Figure 2). Notice that, by definition of x and $ecc(v,s)$, if $T(u, d(u,x)) \subseteq N(u, 16\hat{c}n^{1/3+2\alpha})$ then $d(u,x) = ecc(u, 16\hat{c}n^{1/3+2\alpha})$ and if $T(u, d(u,x)) \nsubseteq N(u, 16\hat{c}n^{1/3+2\alpha})$ then $d(u,x) = ecc(u, 16\hat{c}n^{1/3+2\alpha}) + 1$. Thus, we get that $d(u,x) \ge ecc(u, 16\hat{c}n^{1/3+2\alpha})$.

Since $N(u, 16\hat{c}n^{1/3+2\alpha}) \cap B(v) = \emptyset$, $x \in N(u, 16\hat{c}n^{1/3+2\alpha})$, $b \in B(v)$ and x and b are both on a shortest path between u and v, it must be that $d(u,x) \le d(u,b) - 1$. Since b is on a shortest path between u and v, it holds that $d(u,b) = d(u,v) - d(v,b)$, and so $d(u,x) \le d(u,v) - d(v,b) - 1$. Since $d(u,x) \ge ecc(u, 16\hat{c}n^{1/3+2\alpha})$, it follows that:

$$ecc(u, 16\hat{c}n^{1/3+2\alpha}) \le d(u,v) - d(v,b) - 1. \quad (2)$$

By Lemma 11, $ecc(u, \lceil 4n^{1/3-\alpha} \rceil) + rad(\lceil \hat{c}n^{3\alpha} \rceil) \le ecc(u, (4n^{1/3-\alpha} + 1)(\hat{c}n^{3\alpha} + 2))$. Since $a \in B(u)$ and $|B(u)| \le 4n^{1/3-\alpha}$, it follows from the definitions of $ecc(v,s)$ and bunch that $d(u,a) \le ecc(u, 4n^{1/3-\alpha})$. We have that:

$$\begin{aligned}
d(u,a) + rad(\hat{c}n^{3\alpha}) &\le ecc(u, 4n^{1/3-\alpha}) + rad(\hat{c}n^{3\alpha}) \\
&\le ecc(u, \left\lceil 4n^{1/3-\alpha} \right\rceil) + rad(\lceil \hat{c}n^{3\alpha} \rceil) \\
&\le ecc(u, (4n^{1/3-\alpha}+1)(\hat{c}n^{3\alpha}+2)) \\
&\le ecc(u, 16\hat{c}\, n^{1/3+2\alpha}) \\
&\underset{\substack{\uparrow \\ \text{Equation (2)}}}{\le} d(u,v) - d(v,b) - 1.
\end{aligned}$$

Thus, $d(u,a) + d(b,v) \le d(u,v) - rad(\hat{c}n^{3\alpha}) - 1$. It follows that:

$$2 \min\{d(u,a), d(b,v)\} \le d(u,v) - rad(\hat{c}n^{3\alpha}) - 1. \tag{3}$$

Notice that by the definitions of bunch, a and $p(u)$, it holds that $d(u,p(u)) = d(u,a)+1$. Similarly, $d(v,p(v)) = d(v,b)+1$. Thus:

$$\begin{aligned}
\hat{d}(u,v) &\le \min\{d(u,p(u)) + d(p(u),v), d(u,p(v)) + d(p(v),v)\} \\
&\underset{\substack{\uparrow \\ \text{triangle inequallity}}}{\le} \min\{2d(u,p(u)) + d(u,v), 2d(v,p(v)) + d(u,v)\} \\
&\underset{\substack{\uparrow \\ d(u,p(u))=d(u,a)+1 \text{ and } d(v,p(v))=d(v,b)+1}}{=} \min\{2(d(u,a)+1) + d(u,v), 2(d(v,b)+1) + d(u,v)\} \\
&\le 2\min\{d(u,a), d(v,b)\} + 2 + d(u,v) \qquad (4) \\
&\underset{\substack{\uparrow \\ \text{Equation (3)}}}{\le} d(u,v) - rad(\hat{c}n^{3\alpha}) + 1 + d(u,v) \\
&= 2d(u,v) + 1 - rad(\hat{c}n^{3\alpha}). \qquad (5)
\end{aligned}$$

◁

By combining our ADO construction with Claims 12 and 13 we have proven the following lemma.

▶ **Lemma 14.** *For any graph G with n vertices, real $0 \le \alpha < \frac{1}{3}$ and constant $\hat{c} \ge 1$, it is possible to construct an ADO that uses $\tilde{O}(\hat{c}n^{\frac{5}{3}+\alpha})$ space and has a $(2, 1 - rad(\hat{c}n^{3\alpha}))$-stretch.*

4.2 Proof of Main Upper Bound Theorem

The following lemma connects Δ_G and $rad(s)$, which is the last ingredient needed for proving Theorem 4.

▶ **Lemma 15.** *Let $G = (V,E)$ be an unweighted undirected graph, with $|V| = n$. For any real s such that $1 \le s < n$, it holds that $rad_G(s) \ge \lfloor \log_{\Delta_G}(s/2) \rfloor$.*

Proof. For any vertex v and integer $t \ge 1$, $T(v,t)$ cannot include more than $\Delta_G \cdot \sum_{i=0}^{t-1}(\Delta_G - 1)^i$ vertices. Since $\Delta_G, t \ge 1$ we have that $\Delta_G \cdot \sum_{i=0}^{t-1}(\Delta_G-1)^i \le 2 \cdot \Delta_G^t$ and so for any integer $t \ge 1$ such that $2 \cdot \Delta_G^t < n$ it must be that $T(v,t) \subseteq N(v, 2\cdot\Delta_G^t)$. By definition, $ecc(v,s)$ is equal to the largest integer $x \in [0, ecc(v)]$ for which $T(v,x) \subseteq N(v,s)$. Thus, $t \le ecc(v, 2\cdot\Delta_G^t)$. Since $t \le ecc(v, 2\cdot\Delta_G^t)$ for any vertex v, it follows from the definition of $rad(s) = \min_{v\in V}\{ecc(v,s)\}$ that $t \le rad(2\cdot\Delta_G^t)$. Setting $s \ge 2\cdot\Delta_G^t$, or $t \le \log_{\Delta_G}(s/2)$, it follows that for any integer t such that $t \le \log_{\Delta_G}(s/2)$ it must be that $t \le rad(2\cdot\Delta_G^t) \le rad(s)$. Thus, $\lfloor \log_{\Delta_G}(s/2) \rfloor \le rad(s)$. ◀

Finally, we are ready to prove Theorem 4.

Proof of Theorem 4. It holds that $k = \lfloor \log_{\Delta_G}(\Delta_G^k) \rfloor \leq \lfloor \log_{\Delta_G}(c^k n^{1-k\varepsilon}) \rfloor$, and by Lemma 15, $\lfloor \log_{\Delta_G}(c^k n^{1-k\varepsilon}) \rfloor \leq rad_G(2c^k n^{1-k\varepsilon})$. Thus, the ADO from Lemma 14 constructed for G using $\alpha = \frac{1-k\varepsilon}{3}$ and $\hat{c} = 2c^k$ uses $\tilde{O}(c^k n^{2-\frac{k\varepsilon}{3}})$ space and produces a distance estimation that satisfies $d(u,v) \leq \hat{d}(u,v) \leq \max\{d(u,v), 2d(u,v)+1-rad_G(2c^k n^{1-k\varepsilon})\} \leq \max\{d(u,v), 2d(u,v)+1-k\}$. ◀

5 Reduction from the Set Intersection Problem

Proof of Lemma 6. Given an instance of Problem 1, we construct a graph G with $n = \tilde{O}(N)$ vertices and $\Delta_G = O(n^{1/k})$, such that a $(k, k+2)$-distinguisher oracle for G solves the instance of Problem 1.

We begin by focusing on a $k+1$ layered graph $\mathcal{L} = (V_{\mathcal{L}}, E_{\mathcal{L}})$, which we call the *infrastructure graph*. The infrastructure graph has three important properties: (i) each layer contains N vertices, (ii) there is a path of length k from every vertex in the first layer to every vertex in the last layer, and (iii) the degree of every vertex is at most $2N^{1/k}$.

We then construct for each $x \in X$ a graph $G_x = (V_x, E_x)$, which is a subgraph of (a copy of) $\mathcal{L}$, by removing some of the edges between the first (last) and second (second to last) layers of $\mathcal{L}$ in a way that expresses which sets contain x and which do not. Finally, we construct the graph G which is *specialized* union of all of the graphs G_x for all $x \in X$, and enables solving the instance of Problem 1 by using a $(k, k+2)$-distinguisher oracle on G.

The infrastructure graph. The infrastructure graph $\mathcal{L}$ is a $k+1$ layered graph where each layer contains N vertices, and each layer of N vertices is locally indexed from 1 to N. The layers are numbered 0 to k.

The edges of $\mathcal{L}$ are defined using the following labels. Assign a *label* $\ell(v)$ to every vertex v in $\mathcal{L}$ which is the k digit representation in base[7] $N^{1/k}$ of the local index (an integer between 1 and N) of v. Then, for every $1 \leq t \leq k$, connect u from layer $t-1$ with v from layer t if and only if the digits of $\ell(u)$ and the digits of $\ell(v)$ all match, except for possibly the t'th digit. It is straightforward to observe (since each digit has $N^{1/k}$ options) that the degree of every vertex in $\mathcal{L}$ is $2N^{1/k}$, except for the vertices in the first and last layers which have degree $N^{1/k}$. The following claim shows that there is a path of length k from every vertex in the first layer and every vertex in the last layer.

▷ Claim 16. Let v be a vertex in the first layer of $\mathcal{L}$ and let u be a vertex in the last layer of $\mathcal{L}$. Then there exists a path of length k from u to v in $\mathcal{L}$.

Proof. We describe the path of length k between u and v. For any $0 \leq t \leq k$, consider the vertex w_t in layer t of $\mathcal{L}$ with the label of the following form: the first t digits are the first t digits of $\ell(u)$, and the last $k-t$ digits are the last $k-t$ digits of $\ell(v)$. Thus, for $0 \leq t \leq k-1$, the edge (w_t, w_{t+1}) is in $\mathcal{L}$ since $\ell(w_t)$ and $\ell(w_{t+1})$ are the same, except for possibly the $(t+1)$-th digit. The set of edges which we described form a path of length k between v and u. ◁

[7] We assume for convenience that $N^{1/k}$ is an integer, since otherwise, one can increase N slightly without affecting the asymptotic complexities.

Constructing G_x. We construct V_x by making copies of each vertex in $V_{\mathcal{L}}$. Denote the first layer of $\mathcal{L}$ by $V_L = \{v_1, \ldots, v_N\}$ and the last layer by $V_R = \{u_1, \ldots, u_N\}$. Let $\hat{E}_x = \{(v_i, w) | x \notin S_i \wedge (v_i, w) \in E_{\mathcal{L}}\} \cup \{(u_i, w) | x \notin S_i \wedge (u_i, w) \in E_{\mathcal{L}}\}$. Thus, $\hat{E}_x$ is the set of edges in $\mathcal{L}$ that touch vertices in the first or last layers whose index corresponds to the index of sets that do not contain x. We construct E_x by making copies of all edges in $E_{\mathcal{L}} \setminus \hat{E}_x$. The reason for removing the edges in $\hat{E}_x$ is so that vertices in the first and last layers of G_x whose edges are in $\hat{E}_x$ are not connected to any other vertex in G_x. Thus, for each v_i (u_j) in the first (last) layer of G_x, $x \in S_i$ if and only if there are edges between v_i (u_j) and the second (second to last) layer in G_x. By Claim 16, if $S_i \cap S_j \neq \emptyset$ then there exists a path of length k between v_i and u_j, and otherwise, there is no path in G_x between v_i and u_j. Finally, since G_x is a partial copy of $\mathcal{L}$, the maximum degree in G_x is $2N^{1/k}$.

Constructing G. We construct the $k+1$ layered graph G by performing the following special union of G_x for all x: for $1 \leq t \leq k-1$ the t'th layer of G is the union of the t'th layer of all of the G_x graphs taken over all $x \in X$. Thus, each of the $k-1$ inner layers (excluding the first and last layer of G) has $|X|N$ vertices. For the first (last) layer G, instead of taking the union of all of the first (last) layers from all of the G_x graphs, we merge them all into one layer of N vertices. So the i'th vertex in the first (last) layer of G is a vertex obtained by merging the i'th vertex in the first (last) layer of every G_x, for all $x \in X$. Thus, the first and last layers of G_x contain N vertices each. Since the vertices in the first and last layer of G correspond directly to the vertices V_L and V_R in $\mathcal{L}$, respectively, we treat the first layer of G as $V_L = \{v_1, \ldots v_n\}$ and the last layer of G by $V_R = \{u_1, \ldots, u_N\}$. Thus, each node in $V_L \cup V_R$ has maximum degree at most $|X|N^{1/k} = \tilde{O}(N^{1/k})$.

Answering a set intersection query. Notice that for a set intersection query between S_i and S_j, if $S_i \cap S_j \neq \emptyset$, then there exists some $x \in S_i \cap S_j$, and since G contains G_x as a subgraph, the distance between v_i and u_j is at most (and actually exactly) k. On the other hand, if there exists a path P of length k between v_i and u_j, then by the construction of G, P must be completely contained within some G_x for some $x \in X$. By the construction of G_x, and specifically E_x, the existence of P in G_x implies that $x \in S_i$ and $x \in S_j$. So, in such a case $S_i \cap S_j \neq \emptyset$.

Notice that, since G is a $k+1$ layered graph, any path between a vertex in the first layer and a vertex in the last layer must be of length $k+2q$ for some integer $q \geq 0$. Thus, to answer a set intersection query, it suffices to establish whether the distance in G between v_i and u_j is either k or at least $k+2$, which the $(k, k+2)$-distinguisher oracle returns in constant time.

Analysis. We conclude that a $(k, k+2)$-distinguisher oracle for graphs with $n = \tilde{O}(N)$ vertices and maximum degree $\tilde{O}(n^{1/k})$ also solves the instance of Problem 1 (of size N). Thus, according to Hypothesis 2, an ADO for graphs with $n = \tilde{O}(N)$ vertices, for which the maximum degree is $\tilde{O}(n^{1/k})$, must use $\tilde{O}(N^2) = \tilde{O}(n^2)$ space. We note that the maximum degree can be reduced to $O(n^{1/k})$ by artificially adding $\tilde{O}(n) = \tilde{O}(N)$ isolated vertices to G. ◀

6 Conclusions and Open Problems

In this paper we provide an algorithm (Theorem 4) and a conditional lower bound (Theorem 5) for subquadratic space ADOs as a function of the maximum degree. As mentioned in Section 1, the case of $k = 2$ in Theorem 5 essentially matches the upper bound of Theorem 4. Although the upper bound from Theorem 4 improves the additive approximation of the ADO for larger values of k, a natural remaining open problem is whether it is also possible to reduce the multiplicative approximation of the ADO and design a sub-$\frac{k+2}{k}$ stretch ADO for graphs with maximum degree $\Theta(n^{\frac{1}{k}-\Omega(1)})$ while using subquadratic space for integers $k \geq 3$.

References

1 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. doi:10.1145/3519935.3520066.

2 Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In David Peleg, editor, *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings*, volume 6950 of *Lecture Notes in Computer Science*, pages 404–415. Springer, 2011. doi:10.1007/978-3-642-24100-0_39.

3 Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2014. doi:10.1007/978-3-662-44777-2_5.

4 Rachit Agarwal and Philip Brighten Godfrey. Brief announcement: a simple stretch 2 distance oracle. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 110–112. ACM, 2013. doi:10.1145/2484239.2484277.

5 Rachit Agarwal and Philip Brighten Godfrey. Distance oracles for stretch less than 2. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 526–538. SIAM, 2013. doi:10.1137/1.9781611973105.38.

6 Rachit Agarwal, Philip Brighten Godfrey, and Sariel Har-Peled. Approximate distance queries and compact routing in sparse graphs. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 1754–1762. IEEE, 2011. doi:10.1109/INFCOM.2011.5934973.

7 Maor Akav and Liam Roditty. An almost 2-approximation for all-pairs of shortest paths in subquadratic time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1–11. SIAM, 2020. doi:10.1137/1.9781611975994.1.

8 Surender Baswana, Vishrut Goyal, and Sandeep Sen. All-pairs nearly 2-approximate shortest paths in I time. *Theor. Comput. Sci.*, 410(1):84–93, 2009. doi:10.1016/J.TCS.2008.10.018.

9 Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 591–602. IEEE, 2006.

10 Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

11 Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Improved approximate distance oracles: Bypassing the thorup-zwick bound in dense graphs. *arXiv preprint*, 2023. arXiv:2307.11677.

12 Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010. doi:10.1137/08071990X.

13 Shiri Chechik. Approximate distance oracles with constant query time. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 654–663. ACM, 2014. doi:10.1145/2591796.2591801.

14 Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 1–10, 2015.

15 Shiri Chechik and Tianyi Zhang. Nearly 2-approximate distance oracles in subquadratic time. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 551–580. SIAM, 2022. doi:10.1137/1.9781611977073.26.

16 Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Math. Program.*, 73:129–174, 1996. doi:10.1007/BF02592101.

17 Hagai Cohen and Ely Porat. On the hardness of distance oracle for sparse graph. *CoRR*, abs/1006.1117, 2010. arXiv:1006.1117.

18 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

19 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000. doi:10.1137/S0097539797327908.

20 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

21 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.CH89.

22 Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh M. Basalamah. A survey of shortest-path algorithms. *CoRR*, abs/1705.02044, 2017. arXiv:1705.02044.

23 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 109–118. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.65.

24 Ely Porat and Liam Roditty. Preprocess, set, query! *Algorithmica*, 67(4):516–528, 2013.

25 Mihai Puatracscu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

26 Mihai Puatracscu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011. doi:10.1137/09075336X.

27 Mihai Puatracscu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 815–823. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.83.

28 Mihai Puatracscu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 738–747. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.44.

29 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 2005. doi:10.1007/11523468_22.

30 Liam Roditty and Roei Tov. Approximate distance oracles with improved stretch for sparse graphs. *Theor. Comput. Sci.*, 943:89–101, 2023. doi:10.1016/J.TCS.2022.11.016.

31 Christian Sommer. Shortest-path queries in static networks. *ACM Comput. Surv.*, 46(4):45:1–45:31, 2014. doi:10.1145/2530531.

32 Christian Sommer. All-pairs approximate shortest paths and distance oracle preprocessing. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 55:1–55:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ICALP.2016.55.

33 Mikkel Thorup and Uri Zwick. Compact routing schemes. In Arnold L. Rosenberg, editor, *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10. ACM, 2001. doi:10.1145/378580.378581.

34 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

35 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 786–797. IEEE, 2020. doi:10.1109/FOCS46700.2020.00078.

36 Christian Wulff-Nilsen. Approximate distance oracles with improved query time. In *Encyclopedia of Algorithms*, pages 94–97, 2016. doi:10.1007/978-1-4939-2864-4_568.

37 Uri Zwick. Exact and approximate distances in graphs - A survey. In Friedhelm Meyer auf der Heide, editor, *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2001. doi:10.1007/3-540-44676-1_3.