Distance oracles for unweighted graphs : breaking the quadratic barrier with constant additive error

Surender Baswana*, Akshay Gaur**, Sandeep Sen**, and Jayant Upadhyay**

Abstract. Thorup and Zwick, in the seminal paper [Journal of ACM, 52(1), 2005, pp 1-24], showed that a weighted undirected graph on n vertices can be preprocessed in subcubic time to design a data structure which occupies only subquadratic space, and yet, for any pair of vertices, can answer distance query approximately in constant time. The data structure is termed as approximate distance oracle. Subsequently, there has been improvement in their preprocessing time, and presently the best known algorithms [4, 3] achieve expected $O(n^2)$ preprocessing time for these oracles. For a class of graphs, these algorithms indeed run in $\Theta(n^2)$ time. In this paper, we are able to break this quadratic barrier at the expense of introducing a (small) constant additive error for unweighted graphs. In achieving this goal, we have been able to preserve the optimal size-stretch trade offs of the oracles. One of our algorithms can be extended to weighted graphs, where the additive error becomes $2 \cdot w_{\max}(u, v)$ - here $w_{\max}(u, v)$ is the heaviest edge in the shortest path between vertices u, v.

1 Introduction

Let $G = (V, E)$ be a graph on $|V| = n$ vertices and $|E| = m$ edges, and $\delta(u, v)$ denote the distance between any pair of vertices $u, v \in V$ in graph G. The all-pairs shortest paths (APSP) problem requires preprocessing the given graph G so as to build a data structure using which we can retrieve distance or the shortest path between any pair of vertices efficiently. APSP is undoubtedly one of the most fundamental algorithmic graph problems of computer science. Despite being a classical problem with widespread applications, there exists a huge gap between the lower bound $\Omega(n^2)$ and the worst case upper bound $O(n^3/\log^2 n)$ (due to Chan [6]) of the time complexity of APSP problem. Furthermore, $\Theta(n^2)$ space requirement is a major bottleneck for graphs in many large scale applications. These two factors have motivated researchers to design efficient algorithms (or data structures) for reporting *approximate* distances. In the last fifteen years,

* Department of Comp. Sc. & Engg. Indian Institute of Technology Kanpur, Kanpur - 208016, India. Email : sbaswana@iitk.ac.in. The work was supported by a fellowship from Research I Foundation, CSE, IIT Kanpur.

** Department of Comp. Sc. & Engg., Indian Institute of Technology Delhi, New Delhi-110016, India. Email : {manu,ssen,jayant}@cse.iitd.ernet.in

many novel algorithms [1, 8, 7, 2, 11] have been designed which work for undirected graphs. However, among all these algorithms the *approximate distance oracles* designed by Thorup and Zwick [12] deserve special mention. They showed that any given weighted undirected graph on n vertices can be preprocessed in sub-cubic time for any integer $t \geq 3$ to build a data structure of sub-quadratic size which for any pair of vertices u, v reports t-approximate distance - at least $\delta(u, v)$ and at most $t\delta(u, v)$. There are two very impressive features of their data structure. First, the trade-off between *stretch* t and the size of data structure is essentially optimal assuming a 1963 girth lower bound conjecture of Erdős [9] and second, in spite of its sub-quadratic size their data structure can answer any distance query in *constant* time, hence the name "oracle". More precisely, Thorup and Zwick achieved the following result.

Theorem 1. *[12] For any integer $k \geq 1$, an undirected weighted graph on n vertices and m edges can be preprocessed in expected $O(kmn^{1/k})$ time to build a data structure of size $O(kn^{1+1/k})$ that can answer any $(2k-1)$-approximate distance query in $O(k)$ time.*

Having achieved optimal size-stretch trade offs, and essentially constant query time, it is only the preprocessing time of these oracles which may be improved. The preprocessing time has been improved to $O(\min(n^2, kmn^{1/k}))$ for unweighted graphs [4], and recently for weighted graphs as well [3]. Therefore, a natural question is whether it is possible to achieve $O(m+n^{2-\epsilon})$ - a subquadratic upper bound for constructing approximate distance oracles. Note that any approximate all pairs shortest path algorithm takes $\Omega(n^2)$ steps because of the output size. Therefore, a sub-quadratic time oracle construction provides a clear advantage over such algorithms when we are not interested in all the pair-wise distances. The main objective here is to achieve sub-quadratic preprocessing time for approximate distance oracles without violating the size-stretch trade off. It may be noted that the quadratic upper bound of the existing preprocessing algorithms [12][4] for these oracles is indeed tight - there exists a family of graphs on which these algorithms would execute in $\Theta(n^2)$ time.

In this paper, we design approximate distance oracles which, at the expense of constant additive error, are constructable in sub-quadratic time and preserve size stretch trade-off optimally. More precisely, we show the following. *For any $k > 1$, there is a data-structure which occupies $O(kn^{1+1/k})$ space such that for any pair of vertices $u, v \in V$, it takes $O(k)$ time to return $\hat{\delta}(u, v)$ satisfying*

$$\delta(u, v) \leq \hat{\delta}(u, v) \leq (2k-1)\delta(u, v) + c_k \quad \textit{where} \;\; c_k = 2 \;\; \textit{for} \; k \geq 3 \;\; \textit{and} \;\; c_2 = 8$$

As a natural extension of $(2k-1)$-approximate distance oracle of [12], we denote the above oracle by $(2k-1, c_k)$-approximate distance oracle, where the first term $(2k-1)$ is the stretch (multiplicative error) and c_k is the surplus (additive error). The expected preprocessing time for $(2k-1, c_k)$ oracle is $O(m+kn^{2-\alpha_k})$, where α_k takes value in the interval $[\frac{1}{12}, \frac{1}{2})$ - takes value $\frac{1}{12}$ for $k = 2$ and approaches $\frac{1}{2}$ steadily as k increases (see Table 1).

In short, the small additive error has allowed us to break the quadratic barrier of preprocessing of approximate distance oracles. It would be very important to

Stretch	Space	Preprocessing Time	Reference
$(2k-1,0)$	$O(kn^{1+1/k})$	$O(\min(n^2, kmn^{1/k}))$	[12, 3, 4]
$(3,8)$	$O(n^{3/2})$	$O(\min(m+n^{\frac{23}{12}}, m\sqrt{n}))$	**this paper**
$(2k-1,2)$, $k \geq 3$	$O(kn^{1+1/k})$	$O(\min(m+kn^{\frac{3}{2}+\frac{1}{2k}+\frac{1}{2k-2}}, kmn^{\frac{1}{k}}))$	**this paper**

Table 1. Comparing the new algorithms with the existing algorithms for approximate distance oracles.

explore the limits to which the preprocessing time can be further improved. The result of this paper can be viewed as the first significant step in this direction.

1.1 Overview of the new algorithms

The observation which forms the basis of our algorithms is the simple fact that the $O(kmn^{1/k})$ time complexity of the algorithm of Thorup and Zwick [12] is already sub-quadratic provided the graph is sparse enough. In order to utilize this observation, we use the idea of partitioning the graph into sparse and dense subgraphs. Previously this idea was used by the algorithms which compute all-pairs approximate distance with purely additive error only [1, 8]. Using a random sample $S \subseteq V$ of vertices, we define a sparse subgraph with $o(n^{2-1/k})$ edges, and execute Thorup and Zwick algorithm on this sub graph. This algorithm will execute in $o(n^2)$ time and will easily take care of the case when the shortest paths between a pair of vertices is fully preserved in the sparse graph. Novelty of our algorithms is to handle the other case. Our algorithms make use of a combination of old and new ideas which enables achieving sub-quadratic space without compromising the optimal size-stretch trade-off. In order to make these ideas work, our algorithms effectively use suitable emulators and spanners which are sufficiently sparse.

Definition 1 *An (α, β)-spanner of a graph $G = (V, E)$ is a subgraph (V, E'), $E' \subseteq E$ with the property that distance between any two vertices $u, v \in V$ in the spanner is at least $\delta(u, v)$ and at most $\alpha(\delta(u, v)) + \beta$.*

Definition 2 *An (α, β)-emulator of a graph $G = (V, E)$ is a weighted graph (V, E^*) such that the distance $\delta^*(u, v)$ between any two vertices $u, v \in V$ in the emulator is at least $\delta(u, v)$ and at most $\alpha\delta(u, v) + \beta$.*

In the following section we describe the notations and lemmas which will be used throughout this paper. In section 3, we describe our $(3, 8)$-approximate distance oracle. In section 4, we describe $(2k-1, 2)$-approximate distance oracle for $k > 2$.

2 Preliminaries

For a given graph $G = (V, E)$, and any subset $S \subseteq V$, we shall use the following notations :

- $\mathcal{N}(v)$: the set consisting of v and every neighbor of v in the graph G.
- $\mathcal{N}(S)$: $\cup_{v \in S} \mathcal{N}(v)$.
- $p_S(v)$: the vertex from set S which is nearest to v (break the tie arbitrarily in case there are multiple nearest vertices).
- $\delta(v, S)$: distance between v and $p_S(v)$.
- $E(v)$: the set of edges in G which are incident on v.
- $\mathcal{E}_S(v)$: the set $E(v)$ if v is not adjacent to any vertex of set S, and $\emptyset$ otherwise.
- $\mathcal{E}_S$: $\cup_{v \in V} \mathcal{E}_S(v)$.
- G_S : the subgraph $(V, \mathcal{E}_S)$.
- $\mathcal{O}_t^{\mathcal{G}}$: the t-approximate distance oracle of Thorup and Zwick [12] created on a subgraph $\mathcal{G}$ of G.

Our data structure will store information about $p_S(v)$ and $\delta(v, S)$ for each vertex in the given graph G. To compute this information, the graph G can be processed in just $O(m)$ time as follows : *insert a dummy vertex o in to the graph, connect it to all the vertices of set S, and perform a BFS traversal on the graph starting from o.* We shall use $\mathcal{T}_S$ to denote the set of edges of this BFS tree excluding the edges incident on the dummy vertex.

Lemma 1. *The edge set $\mathcal{T}_S$ preserves the shortest path between v and $p_S(v)$ for all $v \in V$. The size of $\mathcal{T}_S$ is $O(n)$.*

Now we redefine an important concept (due to Thorup and Zwick [12]) of *ball* around a vertex.

Definition 3 *[12] For a vertex $u \in V$ and a set $S \subseteq V$ in a graph $G = (V, E)$, we define $ball(u, V, S)$ as the sub graph induced by all those vertices $v \in V$ which satisfy $\delta(u, v) < \delta(u, S)$ (i.e., for u, it is v which is nearer than $p_S(u)$).*

We now state the following Lemma about the number of vertices and edges in $ball(u, V, S)$ when S is formed by random sampling.

Lemma 2. *[12, 4] For a given graph $G = (V, E)$, let $S \subseteq V$ be a set formed by selecting each vertex from V independently with probability $q > 0$. Then the expected number of vertices and expected number of edges in $ball(u, V, S)$ are $O(1/q)$ and $O(1/q^2)$ respectively.*

We shall now state a few important Lemmas about the sparse subgraph $(V, \mathcal{E}_S)$.

Lemma 3. *If set $S \subseteq V$ is formed by selecting each vertex independently with probability $q > 0$, the expected size of the set $\mathcal{E}_S$ would be $O(n/q)$.*

Lemma 4. *If on the shortest path between any two vertices $u, v \in V$ in the graph $G = (V, E)$ there are no two consecutive vertices in set $\mathcal{N}(S)$, then the shortest path between u and v is preserved exactly in the subgraph $(V, \mathcal{E}_S)$.*

The above property of the edge set $\mathcal{E}_S$ will prove to be very useful in our construction. For the other case we observe the following.

Lemma 5. *If the shortest path between u and v in the graph G contains at least 2 consecutive vertices from $\mathcal{N}(S)$, then $\delta(u, S) + \delta(v, S) \leq \delta(u, v) + 1$.*

Proof. Suppose on the shortest path between u and v, u' be the vertex from the set $\mathcal{N}(S)$ nearest to u, and v' be the vertex from the set $\mathcal{N}(S)$ nearest to v. Since $u' \in \mathcal{N}(S)$ either u' belongs to S or some neighbor of u' belongs to S. This implies that $\delta(u,u') \geq \delta(u,S) - 1$. Similarly $\delta(v,v') \geq \delta(v,S) - 1$. Also note that $\delta(u,u') + \delta(u',v') + \delta(v',v) = \delta(u,v)$. Furthermore, $\delta(u',v') \geq 1$ since there are at least two vertices from $\mathcal{N}(S)$ on the shortest path between u and v. Therefore, $\delta(u,u') + \delta(v',v) + 1 \leq \delta(u,v)$. This along with the lower bounds on $\delta(u,u')$ and $\delta(v,v')$ derived above imply that $\delta(u,S) + \delta(v,S) \leq \delta(u,v) + 1$.

For construction of our $(2k-1,2)$-oracle for $k > 2$, we shall employ the following result on spanners.

Theorem 2. *[10] For a given unweighted graph $G = (V,E)$ and any integer $k > 1$, there exists an $O(m)$ time algorithm for computing a $(2k-1)$-spanner of size $O(n^{1+1/k})$.*

3 A $(3,c)$-approximate distance oracle in expected $O(n^{2-\frac{1}{12}})$ time

Let G be the given undirected unweighted graph. Let S be a set formed by selecting each vertex independently with probability $n^{-\frac{5}{12}}$. Our preprocessing algorithm for $(3,c)$-approximate distance oracle, where $c = 8$, will employ the sparse subgraph $(V, \mathcal{E}_S)$ and an *emulator* of the given graph G. We shall need a $(3,2)$-emulator which also satisfies some additional properties which are very crucial (see Lemma 7). We describe the construction and properties of this emulator in the following subsection first.

3.1 The emulator (V, E^*) : its construction and properties

In the construction of the emulator, we shall employ the $(3,2)$-spanner designed by Baswana et al. [5].

Theorem 3. *[5] For a given graph $G = (V,E)$, let S' be a set formed by selecting each vertex from V independently with probability $p = n^{-\frac{1}{3}}$. It takes expected $O(m)$ time to construct a $(3,2)$-spanner of size $O(n^{4/3})$ that satisfies the following additional properties for each $u \in V$.*

1. *If $u \in V$ has no neighbor from set S' in G, then every edge incident onto u will be in the spanner.*
2. *If u has one or more neighbors from set S' in G, then for some unique neighbor among them, denoted by $c(u)$, the following assertions hold true.*
   (a) *the edge $(u, c(u))$ is present in the spanner also.*
   (b) *for each edge $(u,v) \in E$ not present in the spanner, there is a path between $c(u)$ and $c(v)$ in the spanner with length at most 3.*

Algorithm for building emulator (V, E^*)

1. Add the edges of $\mathcal{T}_S$ (see Lemma 1)to E^*.
2. Let S' be the set formed by selecting each vertex with probability $n^{-\frac{1}{3}}$, and let *span* be the $(3,2)$-spanner of the graph as stated in Theorem 3. Add all the edges of *span* to E^*.
3. From each vertex $v \in V \backslash S$ perform BFS traversal up to level $\delta(v,S)-1$ to compute distance to each $x \in S'$ which is present in $ball(v,V,S)$, and add an edge (v,x) of weight $\delta(v,x)$ to E^*.

Lemma 6. *The emulator (V,E^*) output by the above algorithm has size $O(n^{4/3})$ and can be constructed in expected $O(m+n^{11/6})$ time.*

Proof. The first two steps take expected $O(m)$ time [5]. The third step is performed by executing BFS traversal from each $v \in V \backslash S$ up to level $\delta(v,S)-1$. This will involve traversing only the edges of the subgraph $ball(v,V,S)$. So it follows from Lemma 2 that the expected time spent per vertex v in step 3 is $O(n^{\frac{5}{6}})$. So the expected time spent for constructing E^* will be $O(n^{\frac{11}{6}}+m)$.

Let us analyze the number of edges of E^*. The first two steps will contribute $O(n^{\frac{4}{3}})$ edges to E^*. It follows from elementary probabilistic analysis that the expected number of vertices of set S' in $ball(v,V,S)$ will be $O(n^{\frac{1}{12}})$. So the expected number of weighted edges contributed to E^* by each vertex $v \in V \backslash S$ is $O(n^{\frac{1}{12}})$. Hence the expected number of edges in E^* will be $O(n^{\frac{13}{12}}+n^{\frac{4}{3}}) = O(n^{\frac{4}{3}})$. We repeat the above algorithm if $|E^*|$ exceeds twice its expected value; it follows from Markov inequality that the expected number of repetitions needed will be at most 2. So we can conclude that the above algorithm can be made to run in expected $O(m+n^{\frac{11}{6}})$ time to compute emulator (V,E^*) with $|E^*| = O(n^{\frac{4}{3}})$.

It follows from the second step in the above algorithm that (V,E^*) is a $(3,2)$-emulator of the given graph. Furthermore, it *nearly* preserves exact distance from every vertex u to all the vertices with in $ball(u,V,S)$. The following Lemma formalizes this fact.

Lemma 7. *For each vertex u and any vertex $v \in ball(u,V,S)$, the distance $\delta^*(u,v)$ in the emulator (V,E^*) is at least $\delta(u,v)$ and at most $\delta(u,v)+4$*

Proof. Consider the shortest path between u and v in the original graph. If $v \in S'$, it follows that $\delta^*(u,v) = \delta(u,v)$. Otherwise, let w be the vertex nearest from v(excluding v) on this path which belongs to $\mathcal{N}(S')$. If no such w exists, then the entire path is preserved in the emulator as follows from Theorem 3(*1*). Otherwise it follows from Theorem 3(*2*) that there is a vertex $x \in S'$ such that the edge (x,w) is present in the $(3,2)$-spanner (and hence in the emulator too). Note that x must belong to $ball(u,V,S)$ also, therefore, there is a weighted edge (u,x) in the emulator with weight $\delta(u,x)$. Now there are two cases.

1. $(w, v) \in E$: In this case, $\delta(u, x) \leq \delta(u, v)$. Now consider the sub-case when $v \in \mathcal{N}(S')$, it follows from Theorem 3(*2*) that for some vertex $c(v) \in S'$, the edge $(v, c(v))$ as well as a path between x and $c(v)$ of length at most 3 is present in the spanner, and hence in the emulator. This implies that $\delta^*(u, v) \leq \delta(u, x) + \delta^*(x, c(v)) + \delta(c(v), v) \leq \delta(u, v) + 4$.
   If $v \notin \mathcal{N}(S')$ then the edge (w, v) is preserved in emulator as follows from Theorem 3(*1*). Thus the distance $\delta^*(u, v) \leq \delta(u, x) + 2 \leq \delta(u, w) + 3 \leq \delta(u, v) + 2$.
2. $(w, v) \notin E$: In this case, $\delta(u, x) \leq \delta(u, w) + 1$. Furthermore, it follows from definition of w and Theorem 3(*1*) that the entire path between w and v is preserved in the spanner, and hence in the emulator. The weighted edge (u, x), the edge (x, w) and the $w \leftrightarrow v$ path constitute a path in the emulator of length at most $\delta(u, v) + 2$.

Now we shall describe the preprocessing algorithm for the $(3, c)$-approximate distance oracle.

3.2 Preprocessing and query algorithm for $(3, c)$-Oracle

The following is the preprocessing algorithm for $(3, c)$-oracle.

1. Choose a random sample $S \subseteq V$ by picking each vertex independently with probability $n^{-5/12}$.
2. Build a 3-approximate distance oracle $\mathcal{O}_3^{G_S}$ using Thorup and Zwick [12] for the subgraph $G_S = (V, \mathcal{E}_S)$.
3. For each $s \in S$, perform Dijkstra's single source shortest path (SSSP) algorithm on emulator (V, E^*). Let $Dist$ be the matrix which stores shortest distance between each pair of vertices from set S in the emulator.
4. Compute $p_S(u)$ and $\delta(u, S)$ for all $u \in V \backslash S$; Let $\mathcal{R}$ be an array such that $\mathcal{R}[u] = \delta(u, S)$.

The $(3, c)$-oracle will consist of $\mathcal{O}_3^{G_S}$, matrix $Dist[]$, and array $\mathcal{R}[]$. We now describe the query algorithm.

Query(u, v) {
    $d_1 \leftarrow \mathcal{O}_3^{G_S}(u, v)$.
    $d_2 \leftarrow \mathcal{R}[u] + \mathcal{R}[v] + Dist[p_S(u), p_S(v)]$.
return $\min(d_1, d_2)$ }

3.3 Analysis of the new oracle

Let us first analyze the stretch of the approximate distance reported by **Query**(u, v) for any two vertices $u, v \in V$. If the shortest path between u and v in G doesn't contain two consecutive vertices in $\mathcal{N}(S)$, then from Lemma 4, it follows that

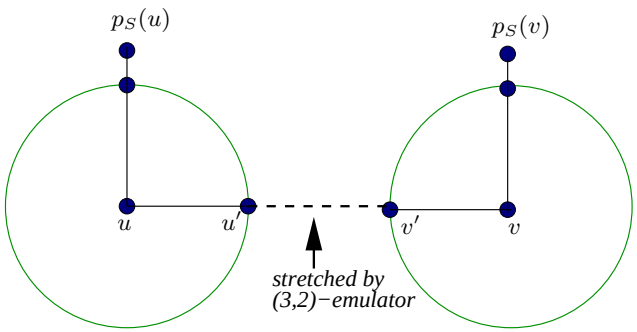

Fig. 1. Analyzing the $(3, c)$-approximate distance oracle when there are at least two consecutive vertices from $\mathcal{N}(S)$ on the shortest path between u and v

this shortest path is captured in the sparse subgraph $G_S = (V, \mathcal{E}_S)$ as well. It follows from Theorem 1 that the data structure $\mathcal{O}_3^{G_S}$ will report a stretch-3 estimate of shortest distance, and we are done. Let us consider the case where the shortest path between u and v contains two or more consecutive vertices from $\mathcal{N}(S)$. It follows from Lemma 5 that $\mathcal{R}[u] + \mathcal{R}[v] \leq \delta(u, v) + 1$. In this case $ball(u, V, S)$ and $ball(v, V, S)$ will be non-overlapping (see Figure1). We divide the shortest path between u and v into 3 separate sub-paths : the path $u \leftrightarrow u'$ lying inside $ball(u, V, S)$, the path $v' \leftrightarrow v$ lying inside $ball(v, V, S)$, and the path $u' \leftrightarrow v'$ not covered by either of the two balls. It follows from Lemma 7 that $\delta^*(u, u') \leq \delta(u, u') + 4 = (\delta(u, S) - 1) + 4 = \mathcal{R}[u] + 3$, and similarly $\delta^*(v', v) \leq \mathcal{R}[v] + 3$. Moreover, since the emulator has stretch $(3, 2)$, so

$$\begin{aligned}\delta^*(u', v') &\leq 3\delta(u', v') + 2 \leq 3(\delta(u, v) - \mathcal{R}[u] - \mathcal{R}[v] + 2) + 2 \\ &= 3\delta(u, v) - 3\mathcal{R}[u] - 3\mathcal{R}[v] + 8\end{aligned}$$

Combining the length of the three sub-paths in the emulator, it follows that that $\delta^*(u, v) \leq 3\delta(u, v) - 2\mathcal{R}[u] - 2\mathcal{R}[v] + 14$. Thus the distance reported by **Query**(u, v) is at most :

$$\mathcal{R}[u] + \mathcal{R}[v] + Dist[p_S(u), p_S(v)] \leq 2\mathcal{R}[u] + 2\mathcal{R}[v] + \delta^*(u, v) \leq 3\delta(u, v) + 14$$

Additive error can be reduced to 8 by a more careful analysis. However, for sake of simplicity, we have omitted the details in this version.

We now analyze the space and preprocessing time of this $(3, c)$-oracle.

Lemma 8. *The size of $(3, c)$-approximate distance oracle is $O(n^{3/2})$.*

Proof. It follows from Theorem 1 that the size of $\mathcal{O}_3^{G_S}$ is $O(n^{3/2})$. The size of S will be concentrated around its expected value of $O(n^{7/12})$ (this is because $|S|$ is sum of n independent Bernoulli random variables, and we can apply Chernoff's bound). Therefore, with high probability, the size of matrix $Dist$ is $|S| \times |S| =$

$O(n^{14/12})$ which is $o(n^{3/2})$. For each $u \in V$ we also store vertex $p_S(u)$ and distance $\mathcal{R}[u]$; this will require additional $O(n)$ space for all the vertices. So the overall space required is concentrated around $O(n^{3/2})$ with high probability. We may rebuild the oracle again from scratch if the space exceeds its expected value by some large constant; using Markov inequality, it follows that the expected number of repetitions will be just a constant.

Lemma 9. *The expected time taken by preprocessing algorithm is* $O(n^{2-\alpha_2}+m)$ *where* $\alpha_2 = \frac{1}{12}$.

Proof. Steps 1 and 4 of the preprocessing algorithm take $O(m)$ time. In step 2, it follows from Theorem 1 that the $\mathcal{O}_3^{G_S}$ construction takes $O(|\mathcal{E}_S| \cdot n^{\frac{1}{2}})$ time, and we know from Lemma 3 that the expected value of $|\mathcal{E}_S|$ is $O(n^{1+\frac{5}{12}})$. Thus constructing $\mathcal{O}_3^{G_S}$ takes expected $O(n^{2-\frac{1}{12}})$ time. Emulator construction takes expected $O(m+n^{\frac{11}{6}})$ time which follows from Lemma 6. In Step 3 of the preprocessing algorithm, we execute Dijkstra's single source shortest path algorithm $\forall s \in S$ on E^* which will take expected $O(|S| \cdot (|E^*| + n \log n))$ time. Also note that $|E^*| = O(n^{\frac{4}{3}})$ (see Lemma 6). Thus the expected time required in step 3 of the preprocessing algorithm is $O(n^{\frac{7}{12}+\frac{4}{3}}) = O(n^{2-\frac{1}{12}})$. Thus the expected running time of the preprocessing algorithm is $O(n^{\frac{23}{12}})$.

We can thus conclude the following Theorem.

Theorem 4. *A given unweighted graphs on* n *vertices and* m *edges can be preprocessed in expected* $O(m + n^{\frac{23}{12}})$ *time to build a* $(3, 8)$*-approximate distance oracle of size* $O(n^{\frac{3}{2}})$.

4 A $(2k-1, 2)$ approximate distance oracle in $o(n^2)$ time

In this section we describe a $(2k-1, 2)$-approximate distance oracle, for any integer $k \geq 3$, which can be constructed in expected sub-quadratic time. The space occupied by the oracle is $O(kn^{1+1/k})$.

Let $S \subseteq V$ be formed by selecting each vertex independently with probability $n^{-\frac{1}{2}-\frac{1}{2k(k-1)}}$. Thus the expected size of S is $O(n^{\frac{1}{2}-\frac{1}{2k(k-1)}})$. Just like our $(3, c)$-approximate distance oracle, our preprocessing algorithm for $(2k-1, 2)$-approximate distance oracle will employ an emulator (V, E^*). However, this emulator will be a proper sub graph of the original graph G as can be observed from its construction described below.

Constructing emulator (V, E^*)
1. Add the edges of $\mathcal{T}_S$ (see Lemma 1)to E^*. 2. Compute a $(2k-3)$-spanner *span* of size $O(n^{1+\frac{1}{k-1}})$ for the given graph using $O(m)$ time algorithm of Theorem 2. Add all edges of *span* to E^*.

Lemma 10. *The subgraph (V, E^*) output by the above algorithm is an $(2k-3)$-emulator of size $O(n^{1+\frac{1}{k-1}})$. It is computed in $O(m)$ time. Furthermore, this emulator preserves distance between u and $p_S(u)$ for each $u \in V$.*

4.1 Preprocessing and query algorithm for $(2k-1, 2)$-approximate distance oracle

We preprocess the graph to obtain $(2k-1, 2)$-oracle as follows.

1. Choose a random sample $S \subseteq V$ by picking each vertex independently with probability $n^{-\frac{1}{2}-\frac{1}{2k(k-1)}}$.
2. Construct a $(2k-1)$-approximate distance oracle of Thorup and Zwick [12] on $G_S = (V, \mathcal{E}_S)$. Let us denote this data structure by $\mathcal{O}^{G_S}_{2k-1}$.
3. For each $s \in S$, execute SSSP algorithm on emulator (V, E^*) to obtain matrix $Dist$, which stores distance $\delta^*(u, v)$ between u and v in the emulator for all $u, v \in S$.
4. Compute $p_S(u)$ for each $u \in V$, and construct an array $\mathcal{R}$ such that $\mathcal{R}[u] = \delta(u, S)$ for each $u \in V$.

The oracle will consist of $\mathcal{O}^{G_S}_{2k-1}$, matrix $Dist[]$, and array $\mathcal{R}[]$. We now describe the query algorithm.

Query(u, v) {
  $d_1 \leftarrow \mathcal{O}^{G_S}_{2k-1}(u, v)$.
  $d_2 \leftarrow \mathcal{R}[u] + \mathcal{R}[v] + Dist[p_S(u), p_S(v)]$.
return $\min(d_1, d_2)$ }

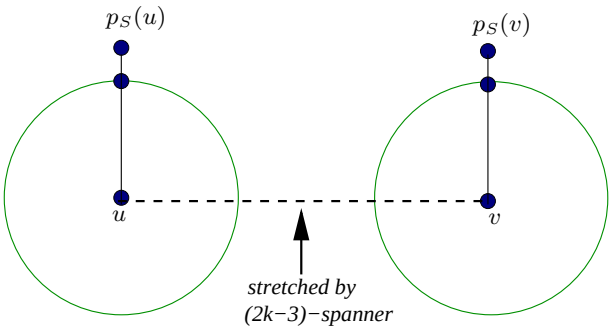

Fig. 2. Proving stretch bound in $(2k-1, 2)$-approximate distance oracle

4.2 Analysis of stretch, space and preprocessing time of oracle

Let us analyze the distance reported by **Query**(u, v) for any two vertices $u, v \in V$. We consider the two cases to prove the stretch bound as follows:

1. If the shortest path between u and v doesn't contain two or more consecutive vertices in $\mathcal{N}(S)$, then it follows from Lemma 4 that this shortest path is captured in the sparse subgraph $G_S = (V, \mathcal{E}_S)$ as well. By Theorem 1, the data structure $\mathcal{O}_{2k-1}^{G_S}$ will report a stretch-$(2k-1)$ estimate of the shortest distance between u and v.
2. If the shortest path between u and v contains two or more consecutive vertices in $\mathcal{N}(S)$, then it follows from Lemma 5 that $\mathcal{R}(u) + \mathcal{R}(v) = \delta(u, S) + \delta(v, S) \leq \delta(u, v) + 1$, where $\delta(u, v)$ is the length of the shortest path between u and v in G. In this case, it is easy to observe that $ball(u, V, S)$ and $ball(v, V, S)$ do not overlap (see Figure 2). It follows from Lemma 10 that $\delta^*(u, p_S(u)) = \delta(u, p_S(u))$, and similarly $\delta^*(v, p_S(v)) = \delta(v, p_S(v))$. Hence, $Dist(p_S(u), p_S(v)) \leq \delta(u, S) + \delta(v, S) + \delta^*(u, v)$. Now $\delta^*(u, v) \leq (2k-3)\delta(u, v)$ since (V, E^*) is a $(2k-3)$-emulator. Combining these inequalities and using Lemma 5, the approximate distance returned by **Query**(u, v) can be bounded by :

$$\begin{aligned}\mathcal{R}[u] + \mathcal{R}[v] + Dist[p_S(u), p_S(v)] &\leq 2\delta(u, S) + 2\delta(v, S) + \delta^*(u, v) \\ &\leq (2k-1)\delta(u, v) + 2\end{aligned}$$

Lemma 11. *The size of the $(2k-1, 2)$-oracle constructed is $O(kn^{1+1/k})$.*

Proof. It follows from Theorem 1 that $\mathcal{O}_{2k-1}^{G_S}$ uses $O(kn^{1+\frac{1}{k}})$ space. *Dist* matrix will use $|S|^2 = O(n^{1-\frac{1}{k(k-1)}})$ space, and array $\mathcal{R}$ will use $O(n)$ space.

Lemma 12. *The expected time taken in computing a $(2k-1, 2)$-approximate distance oracle is $O(kn^{2-\alpha_k} + m)$ where $\alpha_k = (\frac{1}{2} - \frac{1}{2k} - \frac{1}{2k-2})$.*

Proof. It follows from Theorem 1 that the construction of $\mathcal{O}_{2k-1}^{G_S}$ takes $O(k|\mathcal{E}_S| \cdot n^{\frac{1}{k}})$ time, and we know from Lemma 3 that the expected value of $|\mathcal{E}_S|$ is $O(n^{\frac{3}{2}+\frac{1}{2k(k-1)}})$. So the construction of $\mathcal{O}_{2k-1}^{G_S}$ will take expected $O(kn^{\frac{3}{2}+\frac{1}{2k}+\frac{1}{2k-2}} + m)$ time. In Step 3 of the preprocessing algorithm, the construction of emulator takes $O(m)$ time. We run Dijkstra's SSSP algorithm for each $s \in S$ on $(2k-3)$-emulator (V, E^*) of size $O(n^{1+\frac{1}{(k-1)}})$.[1] This will take expected $O(|S| \cdot n^{1+\frac{1}{(k-1)}}) = O(n^{\frac{3}{2}+\frac{1}{2k}+\frac{1}{(2k-2)}})$ time. Step 4 will take $O(m)$ time. Thus the expected time complexity of computing $(2k-1, 2)$-approximate distance oracle is $O(kn^{\frac{3}{2}+\frac{1}{2k}+\frac{1}{2k-2}} + m)$.

We can thus conclude the following Theorem.

Theorem 5. *For any integer $k > 2$, an unweighted undirected graphs on n vertices and m edges can be preprocessed in expected $O(kn^{\frac{3}{2}+\frac{1}{2k}+\frac{1}{2k-2}} + m)$ time to compute $(2k-1, 2)$-approximate distance oracle of $O(kn^{1+\frac{1}{k}})$ size.*

[1] Although this space exceeds $O(n^{1+1/k})$, it is only required during the construction of the oracle and does not contribute to the size of the oracle.

5 Concluding Remarks

For the case of $k \geq 3$, we can obtain further improvements in time bounds by some modifications in the techniques we used. In particular for $k = 3$, we can obtain a $(5, 4)$ oracle in $O(n^{11/6} + m)$ time and for $k \geq 4$ we can get a $(2k - 1, 2k - 2)$ oracle in $O(kn^{2+\frac{1}{k}-\lfloor\frac{k}{2}\rfloor\cdot\frac{1}{k}} + m)$ running time, which is better for all even values of k. However, the additive stretch here is not a constant but depends on k.

A more careful analysis of the algorithm for $k \geq 3$ yields a $(2k-1, 2w_{\max}(u, v))$ oracle for weighted graphs where $w_{\max}(u, v)$ is the weight of the heaviest edge between u, v. In fact when edge weights are identical, we obtain the algorithm in the previous section as a special case. Details are straight forward and omitted from this version.

References

1. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths(without matrix multiplication). *SIAM Journal on Computing*, 28:1167–1181, 1999.
2. S. Baswana, V. Goyal, and S. Sen. All-pairs nearly 2-approximate shortest paths in $O(n^2 \operatorname{polylog} n)$ time. In *Proceedings of 22nd Annual Symposium on Theoretical Aspect of Computer Science*, volume 3404 of *LNCS*, pages 666–679. Springer, 2005.
3. S. Baswana and T. Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *Proceedings of the 47th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 591–602, 2006.
4. S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms*, 2:557–577, 2006.
5. S. Baswana, K. Telikepalli, K. Mehlhorn, and S. Pettie. New construction of (α, β)-spanners and purely additive spanners. In *Proceedings of 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005.
6. T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Proceedings of 39th Annual ACM Symposium on Theory of Computing*, pages 590–598, 2007.
7. E. Cohen and U. Zwick. All-pairs small stretch paths. *Journal of Algorithms*, 38:335–353, 2001.
8. D. Dor, S. Halperin, and U. Zwick. All pairs almost shortest paths. *Siam Journal on Computing*, 29:1740–1759, 2000.
9. P. Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice,1963)*, pages 29–36, Publ. House Czechoslovak Acad. Sci., Prague, 1964.
10. S. Halperin and U. Zwick. Linear time deterministic algorithm for computing spanners for unweighted graphs. *unpublished manuscript*, 1996.
11. L. Roditty, M. Thorup, and U. Zwick. Deterministic construction of approximate distance oracles and spanners. In *Proceedings of 32nd International Colloquim on Automata, Languagaes and Programming (ICALP)*, volume 3580 of *LNCS*, pages 261–272. Springer, 2005.
12. M. Thorup and U. Zwick. Approximate distance oracles. *Journal of Association of Computing Machinery*, 52:1–24, 2005.