Approximating Approximate Distance Oracles

Michael Dinitz[*1] **and Zeyu Zhang**[†2]

1 **Department of Computer Science, Johns Hopkins University, Baltimore, USA**
mdinitz@cs.jhu.edu
2 **Department of Computer Science, Johns Hopkins University, Baltimore, USA**
zyzhang92@gmail.com

Abstract

Given a finite metric space (V, d), an approximate distance oracle is a data structure which, when queried on two points $u, v \in V$, returns an approximation to the the actual distance between u and v which is within some bounded stretch factor of the true distance. There has been significant work on the tradeoff between the important parameters of approximate distance oracles (and in particular between the size, stretch, and query time), but in this paper we take a different point of view, that of per-instance optimization. If we are given an particular input metric space and stretch bound, can we find the smallest possible approximate distance oracle for that particular input? Since this question is not even well-defined, we restrict our attention to well-known classes of approximate distance oracles, and study whether we can optimize over those classes.

In particular, we give an $O(\log n)$-approximation to the problem of finding the smallest stretch 3 Thorup-Zwick distance oracle, as well as the problem of finding the smallest Pătraşcu-Roditty distance oracle. We also prove a matching $\Omega(\log n)$ lower bound for both problems, and an $\Omega(n^{\frac{1}{k}-\frac{1}{2^{k-1}}})$ integrality gap for the more general stretch $(2k-1)$ Thorup-Zwick distance oracle. We also consider the problem of approximating the best TZ or PR approximate distance oracle *with outliers*, and show that more advanced techniques (SDP relaxations in particular) allow us to optimize even in the presence of outliers.

1998 ACM Subject Classification E.1 Data Structures

Keywords and phrases Distance Oracles, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.52

1 Introduction

Given a finite metric space (V, d), an approximate distance oracle is a data structure which can approximately answer distance queries. It is usually a combination of a preprocessing algorithm to compute a data structure, and a query algorithm which returns a distance $d'(u, v)$ whenever queried on a pair of vertices $u, v \in V$. An approximate distance oracle is said to have *stretch* t if $d(u, v) \leq d'(u, v) \leq t \cdot d(u, v)$. Note that there is a trivial stretch 1 distance oracle that uses $\Theta(n^2)$ space: we could just store the entire metric space. So the goal is to reduce the space, i.e., to build a small data structure that also has small stretch and small query time.

The seminal work on approximate distance oracles is due to Thorup and Zwick [19]. They showed that for every integer $k \geq 1$, every finite metric space has an approximate distance oracle with stretch $(2k-1)$ and query time $O(k)$ which uses only $O(kn^{1+\frac{1}{k}})$ space.

* Partially supported by NSF grants 1464239 and 1535887.
† Partially supported by NSF grants 1464239.

A significant fraction of more recent results have built off of the ideas developed in [19], and much of this follow-up work has stored the exact same (or very similar) data structure, just with improved query algorithms or slightly different information in the storage (see, e.g., [17, 21, 5, 6]). Most notably, Pătraşcu and Roditty [16] gave a different distance oracle (still using some of the basic ideas from [19]) that has multiplicative stretch of 2 and additive stretch of 1, with size $O(n^{\frac{5}{3}})$. This broke through the stretch 3 barrier from [19]. Later this result was improved to more general multiplicative/additive stretches [1].

In this paper we ask a natural but very different type of question about approximate distance oracles: can we find (or approximate) the *best* approximate distance oracle? If we are given an input metric space and a stretch bound, is it possible to find the *smallest* approximate distance oracle for that particular input? This is an unusual question in two ways. First, most data structures are by design forced to store all of the input data; the question is *how* to store it and what *extra* information should be stored. This is the case in other settings where instance-optimality of data structures has been considered, e.g., static or dynamic optimality of splay trees. Second, it is not clear whether this question is even well-defined: lower bounds on data structures are commonly arrived at through information or communication complexity (see, e.g., [15]) but when we ask for the optimal data structure on one particular instance this approach becomes meaningless.

However, approximate distance oracles are different in ways which allow us to make meaningful progress towards these optimization questions. First, since we are allowed to return only approximate distances (up to some stretch factor), we are allowed to store only part of the input (and indeed this is the entire point of such an oracle). The second problem is a bit more tricky: given an input, how can we optimize over "the space of all approximate distance oracles"? What does this mean, and what does this space look like?

To get around this issue, we make an observation: many modern distance oracles (and in particular Thorup-Zwick, Pătraşcu-Roditty, and almost all of their variants) have a similar structure. The preprocessing algorithm chooses a subset of the original distances to store which has some particular structure, and the query algorithm can return a valid distance estimate efficiently as long as the stored distances satisfy the required structure. Thus we can optimize for *these particular* distance oracles by choosing the *best possible* set of distances to remember subject to the required structure. By characterizing this structure for different types of distance oracles, we can optimize over those types.

For example, the stretch-3 Thorup-Zwick distance oracle uses a subtle but simple method to choose the set of distances to store. It randomly samples a subset of approximately $\sqrt{n}$ vertices, without using any information about the original metric space, and then creates a data structure which is related (in a well-defined, important way) to these vertices. The correctness of the query algorithm does not depend on the choice of the vertices. Thus instead of simply choosing the subset of vertices uniformly at random, we can instead try to optimize the set of chosen vertices with respect to the actual input metric space.

In this paper, we give matching $\Theta(\log n)$ upper and lower bounds for optimizing stretch-3 Thorup-Zwick distance oracles, and matching $\Theta(\log n)$ upper and lower bounds for optimizing the Pătraşcu-Roditty distance oracle. These upper bounds both use a similar LP relaxation, but by giving an $\Omega(n^{\frac{1}{k}-\frac{1}{2^{k-1}}})$ integrality gap for optimizing stretch-$(2k-1)$ Thorup-Zwick distance oracles, we show that this relaxation is not enough to give nontrivial approximations when extended to larger stretch values.

As an extension, we also study the problem of optimizing distance oracles *with outliers*: if we are allowed to not answer queries for some of the vertices (of our choosing), can we have much smaller storage space? We give an $(O(\log n), 1+\varepsilon)$-bicriteria approximation to

both stretch-3 Thorup-Zwick and Pătraşcu-Roditty distance oracles with outliers. We also give a true approximation to stretch-3 Thorup-Zwick distance oracle with outliers when the number of outliers is small.

1.1 Relationship to Spanners

It is worth noting that this paper is motivated by a similar line of research on *graph spanners* (subgraphs which approximately preserve distances). Spanners and distance oracles tend to be related (although there is no known formal connection between them), and the traditional questions asked of spanners (what is the tradeoff between the stretch and the size?) are similar to the traditional questions asked of distance oracles. Recently, there has been significant progress in looking at spanners from an optimization point of view: given an input graph and an allowed stretch bound, can we find the sparsest possible spanner meeting that stretch bound? In the last few years, upper and lower bounds have been developed for these problems in the basic case, the directed case, with a degree objective, with fault-tolerance, etc. See, e.g., [9, 2, 11, 8, 7].

It is natural to ask these kinds of optimization questions for distance oracles as well, but the definitions become much more difficult. For spanners, the space we are optimizing over (all subgraphs) is very clear and well-defined. But for distance oracles, as discussed, it is much harder to define the space of all data structures. Thus in this paper we optimize over restricted classes, where this space is more well-defined. We view our definitions of these restricted optimization questions as one of the major contributions of this work.

2 Definitions and Preliminaries

We begin with some basic definitions, including formal definitions of the problems that we will be working on.

▶ **Definition 1.** An approximate distance oracle with (m, a)-stretch, size s, preprocessing time g, and query time h is a pair of algorithms, *preprocess* and *query*, with the following properties.

- *preprocess* is a randomized preprocessing algorithm $preprocess(V, d, m, a, r)$ which takes as input a metric space (V, d), stretch bound (m, a), and random string r and outputs a data structure S where the expected output size is at most $\mathbb{E}_r[|S|] \leq s(|V|, m, a)$ and the expected preprocessing time is at most $g(|V|, m, a)$.
- *query* takes as input a data structure $S = preprocess(V, d, m, a, r)$ (the output of the preprocess algorithm) with two vertices $u, v \in V$, and outputs a value $d'(u, v) \in \mathbb{R}$ such that $d(u, v) \leq d'(u, v) \leq m \cdot d(u, v) + a$. The running time of *query* is at most $h(|V|, m, a)$.

We will frequently refer to these just as "distance oracles" rather than "approximate distance oracles" when the stretch bound is clear from context.

The query algorithm guarantees here are deterministic: the randomness only affects the size of the data structure. Note that one could easily define distance oracles so that either the correctness (with respect to the stretch bound) or the query running time (or both) hold only in expectation or with high probability, but as discussed in Section 1, essentially all existing distance oracles (and in particular the Thorup-Zwick distance oracle) have deterministic guarantees on the queries.

This naturally leads us to the following question: If we fix a particular distance oracle and metric space, can we find the *best* possible data structure? Here we will focus on the output size, not the preprocessing time (as long as the preprocessing time is polynomial). In

other words, since the query algorithm work on *any* of the possible data structures which the preprocessing algorithm might output, can we actually find the smallest such data structure? This gives the following natural optimization problem.

▶ **Definition 2.** Given an approximate distance oracle $\mathcal{A} = (preprocess, query)$, the $\mathcal{A}$-optimization problem takes as input a metric space (V, d) and a stretch bound (m, a), and the goal is to find a string r which minimizes $|preprocess(V, d, m, a, r)|$.

In this paper we will focus on two distance oracles (Thorup-Zwick [19] and Pătraşcu-Roditty [16]), so we now introduce these oracles.

2.1 Thorup-Zwick Distance Oracle

For every integer $k \geq 1$, Thorup and Zwick [19] provided an approximate distance oracle with $(2k-1, 0)$-stretch, size $O(n^{1+\frac{1}{k}})$, preprocessing time $O(kn^{2+\frac{1}{k}})$, and query time $O(k)$. We call this distance oracle TZ_k.

Their preprocessing algorithm first constructs a chain of subsets $\varnothing = A_k \subseteq A_{k-1} \subseteq \ldots \subseteq A_0 = V$ by repeated sampling. Each set A_i, where $i \in [k-1]$, is obtained by including each element of A_{i-1} independently with probability $n^{-\frac{1}{k}}$.

Let $R_{iu} = \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\}$ for all $u \in V$ and $i \in [k]$ (where by convention $\min_{w \in \varnothing} d(u,w) = \infty$ for all $u \in V$ to handle the $i = k$ case). The output data structure is obtained by storing (in a 2-level hash table) the distance from each node u to each node in $\bigcup_{i=1}^{k} R_{iu}$.

The data structure also stores a little more information. Each vertex u remembers $k-1$ pivots: $\arg\min_{w \in A_i} d(u,w)$ for all $i \in [k-1]$, and the distance from u to these pivots. However, this is a fixed space cost, and also negligible, so when analyzing the size of the oracle we will ignore the cost of storing the pivots

Clearly the output data structure is determined once $A_1, \ldots, A_{k-1}$ are fixed. The size of the data structure is:

$$cost(A_1, \ldots, A_{k-1}, V, d) = \sum_{u \in V} \sum_{i=1}^{k} |R_{iu}| = \sum_{u \in V} \sum_{i=1}^{k} \left| \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\} \right|.$$

We will refer to $\sum_{u \in V} |R_{iu}|$ as the cost in level i.

Let us look back on the definition of approximate distance oracle. The random string r is only used to generate A_i's, and the query algorithm will return a correct distance estimate no matter what the sets A_i are, but the size is determined by the sets. Therefore, the TZ_k-optimization problem is to find the subsets $\varnothing = A_k \subseteq A_{k-1} \subseteq \ldots \subseteq A_0 = V$ in order to minimize the total cost.

2.2 Pătraşcu-Roditty Distance Oracle

Pătraşcu and Roditty [16] provided an approximate distance oracle with $(2, 1)$-stretch, size $O(n^{\frac{5}{3}})$, preprocessing time $O(n^2)$, and query time $O(1)$. We call this distance oracle PR. Note that PR works only for metric spaces with integer distances.

Their preprocessing algorithm first construct a set $A \subseteq V$ via a complicated correlated sampling (informally, they sample a large set and a small set, and then define A to be everything in the large set and everything contained in a ball around the small set delimited by the large set). The data structure consists of a 2-level hash table for the distance from each node in A to each node in V, as well as a 2-level hash table storing the distance between each pair $\{u, v\} \subseteq V$ such that $d(u,v) < \min_{w \in A} d(u,w) + \min_{w \in A} d(v,w) - 1$.

As with Thorup-Zwick, the output data structure is completely determined once A is fixed. Let $R = \{\{u, v\} \subseteq V \mid d(u, v) < \min_{w \in A} d(u, w) + \min_{w \in A} d(v, w) - 1\}$. Then the size of the data structure is

$$\begin{aligned} cost(A, V, d) &= n \cdot |A| + |R| \\ &= n \cdot |A| + \left| \left\{ \{u, v\} \subseteq V : d(u, v) < \min_{w \in A} d(u, w) + \min_{w \in A} d(v, w) - 1 \right\} \right|. \end{aligned}$$

As before, the random string r is only used to generate the set A, and any $A \subseteq V$ gives a data structure on which the query algorithm works. Therefore, the PR-optimization problem is to find the subset $A \subseteq V$ in order to minimize the total cost.

2.3 Distance Oracles With Outliers

In some cases, a small set of outlier vertices may make the size of the data structure blow up. Yet in some applications it is acceptable to ignore these outliers. This was the motivation behind a line of work on distance oracles with slack ([3], [4]), in which the data structure could ignore the stretch bound on a small fraction of the distances.

In this paper, we consider the case that we can refuse to answer distance queries for some outlier vertices. In other words, we can essentially remove an outlier set F out of V when computing the distance oracle. This gives us the problem of optimizing distance oracle with outliers, in which we not only need to find the random string to determine the output data structure, we also need to find the set of outliers to minimize the final cost. More formally, we have the following type of problem.

▶ **Definition 3.** Given an approximate distance oracle $\mathcal{A} = (preprocess, query)$, the $\mathcal{A}$-optimization problem with outliers takes as input a metric space (V, d), a stretch bound (m, a), and a bound on the number of outliers $f \in \mathbb{N}$. The goal is to find a string r as well as a set $F \subseteq V$ where $|F| \leq f$, in order to minimize $|preprocess(V \backslash F, d, m, a, r)|$.

We will provide both true approximation results and (α, β)-bicriteria results, in which we slightly violate the bound on the number of outliers. Formally, an (α, β)-approximation algorithm for the $\mathcal{A}$-optimization problem with outliers is an algorithm which on any input $((V, d), (m, a), f)$ returns a solution with cost at most $\alpha \cdot OPT$ that has at most $\beta \cdot f$ outliers (where OPT is the minimum cost of any solution with at most f outliers).

2.4 Our Results and Techniques

With these definitions in hand, we can now formally state our results.

In Section 3 we discuss the problem of optimizing the 3-stretch Thorup-Zwick distance oracle, i.e., the TZ_2-optimization problem. It is straightforward to obtain an $O(\log n)$-approximation by reducing to the non-metric facility location problem.

▶ **Theorem 4.** *There is an $O(\log n)$-approximation algorithm for the TZ_2-optimization problem.*

To prove a matching lower bound, we use a reduction from Label Cover to the TZ_2-optimization problem. We use a proof which is similar to the proof of the hardness of Set Cover in [20] (based on [13]). However, we cannot use a reduction directly from Set Cover since we will need some extra properties of the starting instances, and thus are forced to start from Label Cover. We introduce a new notion of (m, l, δ)-set families and show that these can still be plugged into existing hardness results to get the extra structural properties that we need. This lets us prove the following theorem:

▶ **Theorem 5.** *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$*, the* TZ_2*-optimization problem does not admit a polynomial-time* $o(\log n)$*-approximation.*

For larger stretch values, a natural approach is to realize that a simple LP relaxation suffices to give Theorem 4 in the stretch 3 case, and try to extend this basic LP to larger stretches. In Section 4, we show that this does not work for the more general TZ_k-optimization problem: the integrality gap jumps up to become a polynomial. The instance is very simple: it is just the metric space formed by shortest paths on the n-cycle. It turns out to be straightforward to calculate the optimal fractional LP cost, but proving that the optimal integral solution is large is surprisingly involved.

▶ **Theorem 6.** *The basic LP relaxation for the* TZ_k*-optimization problem has an* $\Omega(n^{\frac{1}{k}-\frac{1}{2^{k-1}}})$ *integrality gap when* $k > 2$*.*

In Section 5 we discuss the problem of optimizing the Pătraşcu-Roditty distance oracle. The basic LP and a simple rounding algorithm gives us an $O(\log n)$-approximation algorithm.

▶ **Theorem 7.** *There is an* $O(\log n)$*-approximation algorithm for* PR*-optimization problem.*

A reduction from set cover problem also gives us a matching lower bound.

▶ **Theorem 8.** *Unless* $\mathbf{P} = \mathbf{NP}$*, the* PR*-optimization problem does not admit a polynomial-time* $o(\log n)$*-approximation.*

In Section 6 we move to the outliers setting. For both TZ_2- and PR-optimization problems, a semidefinite programming relaxation and a simple rounding algorithm gives us an $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$-approximation algorithm. Here, using an SDP relaxation seems to be necessary – the corresponding LP relaxation requires violating the number of outliers by a factor of 2 rather than a factor of $1+\varepsilon$. We can also get a true approximation on TZ_2-optimization problem with outliers if the number of outliers is low. These results form the following theorems.

▶ **Theorem 9.** *There is an* $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$*-approximation algorithm for the* TZ_2*-optimization problem with outliers.*

▶ **Theorem 10.** *There is an* $O(\log n)$*-approximation algorithm for* TZ_2*-optimization problem with outliers if the number of outliers is at most* $\sqrt{n}$*.*

▶ **Theorem 11.** *There is an* $(O(\frac{\log n}{\varepsilon}), 1+\varepsilon)$*-approximation algorithm for the* PR*-optimization problem with outliers.*

3 TZ_2-Optimization Problem

We first give an $O(\log n)$-approximation for TZ_2-optimization (Theorem 4), and follow this with a matching lower bound.

3.1 Upper Bound

We will prove our upper bound by a reduction to the non-metric facility location problem.

▶ **Definition 12.** In the *non-metric facility location* problem we are given a set F of facilities, a set D of clients, an opening cost function $f : F \to \mathbb{R}^+$, and a connection cost function $c : D \times F \to \mathbb{R}^+$. The goal is to find the set $S \subseteq F$ which minimizes $\sum_{i\in S} f(i) + \sum_{i\in D} \min_{j\in S} c(i,j)$ (i.e. the sum of the opening and connection costs).

Non-metric facility location is a classic problem, and much is known about it, including the following upper bound due to Hochbaum.

▶ **Theorem 13** ([14]). *There is a polynomial time algorithm which gives an $O(\log n)$-approximation to the non-metric facility location problem.*

Hochbaum's algorithm is a greedy algorithm, but it is also straightforward to design an algorithm with similar bounds using an LP relaxation. Since it is not necessary we do not present the relaxation here, but generalizations of the relaxation will prove its importance in the more general TZ_k setting (see Section 4).

We now show that the TZ_2-optimization problem is essentially a special case of non-metric facility location problem. First, simple arithmetic manipulation of the cost function of the TZ_2-optimization problem gives the following:

$$\begin{aligned} cost(A_1, V, d) &= \sum_{u \in V} |R_{1u}| + \sum_{u \in V} |R_{2u}| \\ &= \sum_{u \in V} \left| \{v \in V \mid d(u,v) < \min_{w \in A_1} d(u,w)\} \right| + \sum_{u \in V} |\{v \in A_1 \mid d(u,v) < \infty| \\ &= \sum_{u \in V} \left| \{v \in V \mid d(u,v) < \min_{w \in A_1} d(u,w)\} \right| + n|A_1| \\ &= \sum_{w \in A_1} n + \sum_{u \in V} \min_{w \in A_1} |\{v \in V \mid d(u,v) < d(u,w)\}| \,. \end{aligned}$$

Given an instance (V, d) of the TZ_2-optimization problem, we create an instance of non-metric facility location by setting $F = D = V$, opening costs $f(v) = n$ for all $v \in V$, and connection costs $c(u, w) = |\{v \in V \mid d(u, v) < d(u, w)\}|$ for all $u, w \in V$. Then the cost function of the TZ_2-optimization problem is exactly the same as the cost function of non-metric facility location problem. Therefore TZ_2 is a special case of non-metric facility location, which together with Theorem 13 implies Theorem 4.

3.2 Lower Bound

Proving an $\Omega(\log n)$ hardness of approximation (Theorem 5) turns out to be surprisingly difficult. Details appear in the full version [10]; here we provide an informal overview. Technically we reduce directly to TZ_2-optimization from a version of the Label Cover problem that corresponds to applying parallel repetition [18] to 3SAT-5, which is a standard starting point for hardness reductions. Informally, though, we are "really" reducing from Set Cover: given an instance of Set Cover, we show how to create an instance of TZ_2-optimization where the cost of the optimal solution is the same (up to a constant and a polynomial scaling factor). But in order for our reduction to work, we actually need more than just an arbitrary Set Cover instance: we need a version of Set Cover in which it is hard even to cover *most* of the elements, not just all of them.

So we have to also give a new reduction from Label Cover to Set Cover, showing that even this version of Set Cover is hard. It turns out that Feige's reduction [13], reinterpreted by Vazirani [20], essentially already gives us what we need. We just need to analyze it a bit more carefully. In particular, a key component of this reduction is what Vazirani called (m, l)*-set systems*, which can be thought of as nearly-unbiased sample spaces. We generalize this notion to (m, l, δ)-set systems, given in the following definition.

▶ **Definition 14.** A set B (the universe) and a collection of subsets $C_1, \ldots, C_m$ of B form an *(m, l, δ)-set system* if any collection of l sets in $\{C_1, \ldots, C_m, \overline{C_1}, \ldots, \overline{C_m}\}$ whose union contains at least $(1-\delta)|B|$ elements must include both C_i and $\overline{C_i}$ for some i.

An (m, l)-set system is just a $(m, l, 0)$-set system. While not all (m, l)-set systems are (m, l, δ)-set systems for larger δ, the construction of (m, l)-set systems in [20] actually does generalize directly to larger values of δ. With this tool in hand, we follow through the rest of the reduction and it gives us the type of Set Cover instances which we need. Technically our reduction skips this step by going directly from Label Cover to TZ_2-optimization, but generating these kinds of Set Cover instances is intuitively what the first part of the reduction is doing.

4 TZ_k-Optimization Problem

We now move to the more general TZ_k-optimization problem. While we are not able to give nontrivial upper bounds for this problem, we can at least show that the basic LP relaxation (as discussed in Section 3.1) does not give polylogarithmic bounds in this more general setting.

4.1 The LP

Let $B_u(v) = \{w \in V \mid d(u,w) \leq d(u,v)\}$. For every $v \in V$ and $i \in [k]$, let $x_v^{(i)}$ be a variable which is supposed to be an indicator for whether $v \in A_i$. Similarly, for all $u, v \in V$ and $i \in [k]$, let $y_{uv}^{(i)}$ be a variable which is supposed to be an indicator for whether $v \in R_{iu}$. (Recall that $R_{iu} = \{v \in A_{i-1} \mid d(u,v) < \min_{w \in A_i} d(u,w)\}$) We can easily write an LP relaxation for this problem:

$$\begin{array}{lll} (LP_{TZ_k}): \min & \sum_{i=1}^{k} \sum_{u,v \in V} y_{uv}^{(i)} & \\ s.t. & 0 = x_v^{(k)} \leq x_v^{(k-1)} \leq \ldots \leq x_v^{(1)} \leq x_v^{(0)} = 1 & \forall v \in V \\ & y_{uv}^{(i)} \geq x_v^{(i-1)} - \sum_{w \in B_u(v)} x_w^{(i)} & \forall u, v \in V, i \in [k] \\ & y_{uv}^{(i)} \geq 0 & \forall u, v \in V, i \in [k] \end{array}$$

It can easily be shown that this is a valid relaxation (the proof can be found in the full version [10]). When restricted to the special case of $k = 2$, it is not hard to see that this LP is essentially a special case of the basic LP relaxation for non-metric facility location, which can be used to prove the $O(\log n)$ bound of Theorem 4. But for larger values of k the behavior is different, and does not result in a polylogarithmic integrality gap.

4.2 Integrality Gap

The integrality gap instance is quite simple: the metric (V, d) induced by shortest-path distances in a cycle. Slightly more formally, we let $V = [n]$, and use the cycle distance $d(u,v) = \min\{|u-v|, n + \min\{u,v\} - \max\{u,v\}\}$.

Details can be found in the full version [10]. It turns out to be relatively easy to find a fractional solution to LP_{TZ_k} with cost $O(n^{1+\frac{1}{2^{k-1}}})$ on this instance. The tricky part is lower bounding the optimal solution, i.e., showing that the optimal integral solution has cost at least $\Omega(n^{1+\frac{1}{k}})$. Combining these two results gives us an $\Omega(n^{\frac{1}{k}-\frac{1}{2^{k-1}}})$ integrality gap, proving Theorem 6.

5 PR-Optimization Problem

We now move from Thorup-Zwick distance oracles to Pătraşcu-Roditty distance oracles. We show that from an optimization perspective, they are similar to TZ_2 in that we can find matching bounds: an $O(\log n)$-approximation, and $\Omega(\log n)$-hardness.

5.1 Upper Bound

In this section we prove Theorem 7 by using an LP and randomized rounding to give an $O(\log n)$-approximation to the PR-optimization problem.

Let $B_u(v) = \{w \in V \mid d(u,w) \leq d(u,v)\}$, and $B(u,r) = \{w \in V \mid d(u,w) \leq r\}$. We can see $B_u(v) = B(u, d(u,v))$. Now, let x_v be a variable which is supposed to be an indicator for whether $v \in A$, and let y_{uv} be a variable which is supposed to be an indicator for whether $\{u,v\} \in R$ (recall that $R = \{\{u,v\} \subseteq V \mid d(u,v) < \min_{w\in A} d(u,w) + \min_{w \in A} d(v,w) - 1\}$). We can write the following LP relaxation:

$$\begin{array}{rll} (LP_{PR}): \min & \sum_{v\in V} n \cdot x_v + \sum_{\{u,v\}\subseteq V} y_{uv} & \\ s.t. & y_{uv} \geq 1 - \sum_{w \in B(u,r) \cup B(v,d(u,v)-r)} x_w & \forall u,v \in V, \forall r \in [0, d(u,v)] \\ & x_v \in [0,1] & \forall v \in V \\ & y_{uv} \geq 0 & \forall u,v \in V \end{array}$$

At first blush it may not be obvious that the first type of constraint in this LP really captures the characterization of pairs in R. But it is actually not that hard to see that this is a valid relaxation (a formal proof can be found in the full version [10]). Note that while the number of constraints appears to be exponential (recall that we assume integer weights, but not necessarily unit weights, and hence $d(u,v)$ is not necessarily polynomial in the input size), it is in fact possible to solve this LP in polynomial time. We can do this by noting that for each $u,v \in V$, only at most n different value of r actually yield *different* constraints, so we can simply write the constraints for those values.

Our algorithm is relatively straightforward. We first solve LP_{PR} and get an optimal fractional solution (x^*_v, y^*_{uv}). We then use independent randomized rounding, adding each $v \in V$ to A independently with probability $\min\{4 \ln n \cdot x^*_v, 1\}$.

▶ **Lemma 15.** *If $y^*_{uv} \leq \frac{1}{2}$, then the probability that $\{u,v\} \in R$ is at most $\frac{1}{n}$.*

Proof. If $y^*_{uv} \leq \frac{1}{2}$, then the first constraint implies that $\sum_{w \in B(u,r) \cup B(v,d(u,v)-r)} x^*_w \geq \frac{1}{2}$ for all $r \in [0, d(u,v)]$. Therefore, the probability that $A \cap (B(u,r) \cup B(v, d(u,v) - r)) = \varnothing$ for a specific $r \in [0, d(u,v)]$ is at most

$$\prod_{w \in B(u,r) \cup B(v,d(u,v)-r)} (1 - \min\{4 \ln n \cdot x^*_w, 1\}) \leq e^{-\sum_{w \in B(u,r) \cup B(v,d(u,v)-r)} 4 \ln n \cdot x^*_w} \leq \frac{1}{n^2}$$

A union bound over all the different values of r we used in our LP implies that the probability that there exists an $r \in [0, d(u,v)]$ where $A \cap (B(u,r) \cup B(v, d(u,v) - r)) = \varnothing$ is at most $\frac{1}{n^2} \cdot n = \frac{1}{n}$. We claim that the existence of such an r is implied by $\{u,v\} \in R$, and hence the probability that $\{u,v\} \in R$ is at most $\frac{1}{n}$. To see this, suppose that $\{u,v\} \in R$, i.e. suppose that $d(u,v) < \min_{w\in A} d(u,w) + \min_{w \in A} d(v,w) - 1$. Then if we set $r = \min_{w \in A} d(u,w) - 1$, this implies that $\min_{w \in A} d(v,w) > d(u,v) - r$. But then this would imply that no element of A is in $B(u,r) \cup B(v, d(u,v) - r)$. ◀

Let $OPT_{LP_{PR}}$ denote the optimal cost of LP_{PR}. Then the above lemma implies that the expected cost of the rounding algorithm is at most

$$\begin{aligned}\mathbf{E}[n|A| + |R|] &\leq \sum_{v \in V} n \cdot x_v^* \cdot 4 \ln n + 2 \cdot \sum_{u,v \in V} y_{uv}^* + n^2 \cdot \frac{1}{n} \leq O(\log n) \cdot OPT_{LP_{PR}} + n \\ &\leq O(\log n) \cdot OPT\end{aligned}$$

(where we use the fact that $OPT \geq \Omega(n)$). This completes the proof of Theorem 7.

5.2 $\Omega(\log n)$-hardness

We now show a matching hardness bound for the PR-optimization problem by reducing from the Set Cover problem.

Consider a set cover instance $(\mathcal{U}, \mathcal{S})$ where $|\mathcal{U}| + |\mathcal{S}| = n$. For each $e \in \mathcal{U}$, we create a group of vertices G_e where $|G_e| = 3n$. For each $S \in \mathcal{S}$, we also create a group of vertices G_S where $|G_S| = 3n$.

Now we construct the following metric space: $V = (\bigcup_{e \in \mathcal{U}} G_e) \cup (\bigcup_{S \in \mathcal{S}} G_S)$ and

$$d(u,v) = \begin{cases} 1, & \text{if } u \in G_e, v \in G_e \\ 1, & \text{if } u \in G_S, v \in G_S \\ 1, & \text{if } u \in G_e, v \in G_S, e \in S \\ 2, & \text{otherwise.} \end{cases}$$

In the full version [10] we show that if there is a solution $\mathcal{S}^*$ to the set cover instance $(\mathcal{U}, \mathcal{S})$ where $|\mathcal{S}^*| = t$, then there is a set A where $cost(A, V, d) \leq t|V|$. We also show that if there is a set $A \subseteq V$ where $cost(A, V, d) \leq t|V|$, then there exists a solution $\mathcal{S}^*$ to the set cover instance $(\mathcal{U}, \mathcal{S})$ where $|\mathcal{S}^*| = t$. These two claims, together with an appropriate hardness theorem for Set Cover [12], imply Theorem 8.

6 Distance Oracles With Outliers

We now move to the more difficult outliers setting, where we can also optimize over a set of vertices to ignore. Recall that for an approximate distance oracle $\mathcal{A}$, our goal is now to find a set of vertices F (the outliers) where $|F| \leq f$ as well as a string r in order to minimize $|preprocess(V \setminus F, d, m, a, r)|$. In other words, we are going to try to solve the same problems as before, but where we can choose a set F to remove. We begin with TZ_2, and then move to PR.

6.1 TZ_2-Optimization Problem With Outliers

For this problem, it is easy to see that the cost function becomes:

$$\begin{aligned}cost(A, F, V, d) &= (n - f)|A| + \sum_{u \in V \setminus F} |R_{1u}| \\ &= (n - f)|A| + \sum_{u \in V \setminus F} \left| \{v \in V \setminus F \mid d(u,v) < \min_{w \in A} d(u,w)\} \right|.\end{aligned}$$

A natural approach is to use an LP which is similar to LP_{TZ_k} to solve this problem (but for TZ_2), suitably adapted to handle outliers. Let x_v be a variable which is supposed to be

an indicator for whether $v \in A$, let y_{uv} be a variable which is supposed to be an indicator for whether $v \in R_{1u}$, and let z_v be a variable which is supposed to be an indicator for whether $v \in F$. Then we can write the following natural LP relaxation:

$$\begin{array}{rll} (LP_{TZ_2O}): \min & \sum_{v\in V}(n-f)\cdot x_v + \sum_{u,v\in V} y_{uv} & \\ s.t. & y_{uv} \geq 1 - z_u - z_v - \sum_{w\in B_u(v)} x_w & \forall u,v \in V \\ & \sum_{v\in V} z_v \leq f & \\ & x_v \in [0,1] & \forall v \in V \\ & y_{uv} \geq 0 & \forall u,v \in V \\ & z_v \in [0,1] & \forall v \in V \end{array}$$

Unfortunately, this LP can not give an (α, β)-approximation with $\beta = 2 - \epsilon$. To see this, consider the case that $f = \frac{n}{2}$. Then the optimal solution to LP_{TZ_2O} is 0, by setting all z_v to $\frac{1}{2}$, all x_v to 0, and all $y_{uv} = 0$. Thus any integral solution, to be competitive with this fractional solution, must treat *all* nodes as outliers, requiring β to be at least 2.

Fortunately we can give a stronger semidefinite programming relaxation, allowing for a better approximation. As in LP_{TZ_2O}, let $\vec{x}_v$ be a variable which is supposed to be an indicator for whether $v \in A$, let $\vec{y}_{uv}$ be a variable which is supposed to be an indicator for whether $v \in R_{1u}$, and let $\vec{z}_v$ be a variable which is supposed to be an indicator for whether $v \in F$. We can then write this SDP:

$$\begin{array}{rll} (SDP_{TZ_2O}): \min & \sum_{v\in V}(n-f)\cdot \|\vec{x}_v\|^2 + \sum_{u,v\in V} \|\vec{y}_{uv}\|^2 & \\ s.t. & \|\vec{y}_{uv}\|^2 \geq 1 - \vec{z}_u \cdot \vec{z}_v - \sum_{w\in B_u(v)} \|\vec{x}_w\|^2 & \forall u,v \in V \\ & \sum_{v\in V} \|\vec{z}_v\|^2 \leq f & \\ & \|\vec{x}_v\|^2 \leq 1 & \forall v \in V \\ & \|\vec{y}_{uv}\|^2 \leq 1 & \forall u,v \in V \\ & \|\vec{z}_v\|^2 \leq 1 & \forall v \in V \end{array}$$

Our approximation algorithm first solves SDP_{TZ_2O} to get an optimal solution $(\vec{x}^*_v, \vec{y}^*_{uv}, \vec{z}^*_v)$. We then use independent randomized rounding to construct A, adding each $v \in V$ to A independently with probability $\min\{\frac{3\ln n}{\varepsilon} \cdot \|\vec{x}^*_v\|^2, 1\}$ where ε is a small constant. Finally, we use threshold rounding to construct F by adding each $v \in V$ to F if $\|\vec{z}^*_v\|^2 \geq \frac{1}{1+\varepsilon}$.

We want to show that this is an $(O(\log n), 1+\varepsilon)$-approximation. It is easy to see that $|F| \leq (1+\varepsilon)f$ because $\sum_{v\in V} \|\vec{z}^*_v\|^2 \leq f$. In order to prove Theorem 9 it only remains to prove that the expected cost is at most $O(\log n) \cdot OPT$. This proof can be found in the full version [10].

When $f \leq \sqrt{n}$ we can actually give a true $O(\log n)$-approximation (Theorem 10). The algorithm is almost the same; we just need to change the threshold rounding for outliers to instead pick the f vertices with largest $\|\vec{z}_v\|^2$ value. Details appear in the full version [10].

6.2 PR-Optimization Problem With Outliers

For this problem, the cost function becomes:

$$\begin{aligned} cost(A,F,V,d) &= (n-f)\cdot|A| + |R| \\ &= (n-f)\cdot|A| + \left|\{\{u,v\} \subseteq V\backslash F \mid d(u,v) < \min_{w\in A} d(u,w) + \min_{w\in A} d(v,w) - 1\}\right|. \end{aligned}$$

We will again use an SDP relaxation. Let $\vec{x}_v$ be a variable which is supposed to be an indicator for whether $v \in A$, let $\vec{y}_{uv}$ be a variable which is supposed to be an indicator for whether $\{u,v\} \in R$, and let $\vec{z}_v$ be a variable which is supposed to be an indicator for whether

$v \in F$. We have the following relaxation (which we call SDP_{PR}) which is similar to both LP_{PR} and SDP_{TZ_2O}:

$$\begin{array}{lll} \min & \sum_{v\in V}(n-f)\cdot\|\vec{x}_v\|^2 + \sum_{\{u,v\}\subseteq V}\|\vec{y}_{uv}\|^2 & \\ s.t. & \|\vec{y}_{uv}\|^2 \geq 1 - \vec{z}_u\cdot\vec{z}_v - \sum_{w\in B(u,r)\cup B(v,d(u,v)-r)}\|\vec{x}_w\|^2 & \forall u,v\in V, r\in[0,d(u,v)] \\ & \sum_{v\in V}\|\vec{z}_v\|^2 \leq f & \\ & \|\vec{x}_v\|^2 \leq 1 & \forall v\in V \\ & \|\vec{y}_{uv}\|^2 \leq 1 & \forall u,v\in V \\ & \|\vec{z}_v\|^2 \leq 1 & \forall v\in V \end{array}$$

Note that SDP_{PR} is solvable in polynimial time for the same reason that LP_{PR} is solvable: for each pair of (u,v), we can find n different values of r that give all of the distinct constraints.

The rounding algorithm is basically the same as the TZ_2-optimization problem with outliers. We first solve the SDP_{PR} and get an optimal solution $(\vec{x}^*_v, \vec{y}^*_{uv}, \vec{z}^*_v)$. We then use independent randomized rounding to get A, adding each $v \in V$ to A independently with probability $\min\{\frac{6\ln n}{\varepsilon}\cdot\|\vec{x}^*_v\|^2, 1\}$ where ε is a small constant. Then we use threshold rounding to get F, adding each $v \in V$ to F if $\|\vec{z}^*_v\|^2 \geq \frac{1}{1+\varepsilon}$.

This is an $(O(\log n), 1+\varepsilon)$-approximation. It is easy to see that $|F| \leq (1+\varepsilon)f$ because $\sum_{v\in V}\|\vec{z}^*_v\|^2 \leq f$. The proof that the expected cost is at most $O(\log n)\cdot OPT$ is in the full version [10], which completes the proof of Theorem 11.

7 Conclusion and Future Work

In this paper we initiate the study of *approximating* approximate distance oracles. This is a different take on the question of optimizing data structures, where we attempt to find the best data structure for a particular input, rather than for a class of inputs. In order to make this tractable (or even well-defined), we restrict our attention to known classes of distance oracles, and show that it is sometimes possible to find the best of these restricted oracles. We also extended our approaches to optimize in the presence of outliers.

For future work, the major question is clearly whether we can approximately optimize higher level (i.e., higher stretch) Thorup-Zwick distance oracles. Although we show an integrality gap for the basic LP, it is quite conceivable that a stronger LP or SDP could be used to give a logarithmic approximation ratio. Beyond this, there are other distance oracles which could be optimized – we chose Thorup-Zwick and Pătraşcu-Roditty since they are well-known and in some ways canonical, but it would be interesting to extend these ideas to other oracles. At a higher level, we believe that the definitions and ideas we have introduced here could lead to many interesting questions about optimizing data structures for given inputs: can we find near-optimal distance labels? Or compact routing schemes? Or connectivity oracles? Or fault-tolerant oracles? Essentially any data structure question in which there is a choice of *which* data to store, rather than how to store it, can be put into our optimization framework. Exploring this space is an exciting future direction.

References

1 Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 404–415. Springer, 2011.

2 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed steiner forest. *Information and Computation*, 222:93–107, 2013.

3 T-H Hubert Chan, Kedar Dhamdhere, Anupam Gupta, Jon Kleinberg, and Aleksandrs Slivkins. Metric embeddings with relaxed guarantees. *SIAM Journal on Computing*, 38(6):2303–2329, 2009.

4 T-H Hubert Chan, Michael Dinitz, and Anupam Gupta. Spanners with slack. In *European Symposium on Algorithms*, pages 196–207. Springer, 2006.

5 Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 654–663. ACM, 2014.

6 Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 1–10. ACM, 2015.

7 Eden Chlamtác and Michael Dinitz. Lowest degree k-spanner: Approximation and hardness. In *Proceedings of the 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 28, pages 80–95. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

8 Eden Chlamtac, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 758–767. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.61.

9 Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pages 323–332. ACM, 2011.

10 Michael Dinitz and Zeyu Zhang. Approximating approximate distance oracles. Full version. URL: https://arxiv.org/abs/1612.05623.

11 Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 821–840. SIAM, 2016.

12 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

13 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

14 Dorit S Hochbaum. Heuristics for the fixed cost median problem. *Mathematical programming*, 22(1):148–162, 1982.

15 Guy Joseph Jacobson. *Succinct Static Data Structures*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1988.

16 Mihai Patrascu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 815–823. IEEE, 2010.

17 Mihai Patrascu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 738–747. IEEE, 2012.

18 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

19 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

20 Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

21 Christian Wulff-Nilsen. Approximate distance oracles with improved query time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 539–549. Society for Industrial and Applied Mathematics, 2013.