Conditional Lower Bounds for Space/Time Tradeoffs

Isaac Goldstein*[1], Tsvi Kopelowitz**[2], Moshe Lewenstein***[1], and Ely Porat ***[1]

[1]Bar-Ilan University , {goldshi,moshe,porately}@cs.biu.ac.il
[2]University of Waterloo , kopelot@gmail.com

Abstract. In recent years much effort has been concentrated towards achieving polynomial time lower bounds on algorithms for solving various well-known problems. A useful technique for showing such lower bounds is to prove them conditionally based on well-studied hardness assumptions such as 3SUM, APSP, SETH, etc. This line of research helps to obtain a better understanding of the complexity inside P.

A related question asks to prove conditional *space* lower bounds on data structures that are constructed to solve certain algorithmic tasks after an initial preprocessing stage. This question received little attention in previous research even though it has potential strong impact.

In this paper we address this question and show that surprisingly many of the well-studied hard problems that are known to have conditional polynomial *time* lower bounds are also hard when concerning *space*. This hardness is shown as a tradeoff between the space consumed by the data structure and the time needed to answer queries. The tradeoff may be either smooth or admit one or more singularity points.

We reveal interesting connections between different space hardness conjectures and present matching upper bounds. We also apply these hardness conjectures to both static and dynamic problems and prove their conditional space hardness.

We believe that this novel framework of polynomial space conjectures can play an important role in expressing polynomial space lower bounds of many important algorithmic problems. Moreover, it seems that it can also help in achieving a better understanding of the hardness of their corresponding problems in terms of time.

*This research is supported by the Adams Foundation of the Israel Academy of Sciences and Humanities

**Part of this work took place while the second author was at University of Michigan. This work is supported in part by the Canada Research Chair for Algorithm Design, NSF grants CCF-1217338, CNS-1318294, and CCF-1514383

***This work was partially supported by an ISF grant #1278/16

F. Ellen et al. (Eds.): WADS 2017, LNCS 10389, pp. 421–436, 2017.
DOI: 10.1007/978-3-319-62127-2_36

1 Introduction

1.1 Background

Lately there has been a concentrated effort to understand the time complexity within P, the class of decision problems solvable by polynomial time algorithms. The main goal is to explain why certain problems have time complexity that seems to be non-optimal. For example, all known efficient algorithmic solutions for the 3SUM problem, where we seek to determine whether there are three elements x, y, z in input set S of size n such that $x + y + z = 0$, take $\tilde{O}(n^2)$ time[1]. However, the only real lower bound that we know is the trivial $\Omega(n)$. Likewise, we know how to solve the *all pairs shortest path*, APSP, problem in $\tilde{O}(n^3)$ time but we cannot even determine whether it is impossible to obtain an $\tilde{O}(n^2)$ time algorithm. One may note that it follows from the time-hierarchy theorem that there exist problems in P with complexity $\Omega(n^k)$ for every fixed k. Nevertheless, such a separation for natural practical problems seems to be hard to achieve.

The collaborated effort to understand the internals of P has been concentrated on identifying some basic problems that are conjectured to be hard to solve more efficiently (by polynomial factors) than their current known complexity. These problems serve as a basis to prove conditional hardness of other problems by using reductions. The reductions are reminiscent of NP-complete reductions but differ in that they are restricted to be of time complexity strictly smaller (by a polynomial factor) than the problem that we are reducing to. Examples of such hard problems include the well-known 3SUM problem, the fundamental APSP problem, (combinatorial) Boolean matrix multiplication, etc. Recently, conditional time lower bounds have been proven based on the conjectured hardness of these problems for graph algorithms [4,30], edit distance [12], longest common subsequence (LCS) [3,14], dynamic algorithms [5,25], jumbled indexing [11,19], and many other problems [1,2,6,7,13,20,23,24,29].

1.2 Motivation

In stark contrast to polynomial *time* lower bounds, little effort has been devoted to finding polynomial *space* conditional lower bounds. An example of a space lower bound appears in the work of Cohen and Porat [17] and Pătraşcu and Roditty [27] where lower bounds are shown on the size of a distance oracle for sparse graphs based on a conjecture about the best possible data structure for a set intersection problem (which we call set disjointness in order to differ it from its reporting variant).

A more general question is, for algorithmic problems, what conditional lower bounds of a space/time tradeoff can be shown based on the set disjointness (intersection) conjecture? Even more general is to discover what space/time tradeoffs can be achieved based on the other algorithmic problems that we assumed are

[1] The $\tilde{O}$ and $\tilde{\Omega}$ notations suppress polylogarithmic factors

hard (in the time sense)? Also, what are the relations between these identified "hard" problems in the space/time tradeoff sense? These are the questions which form the basis and framework of this paper.

Throughout this paper we show connections between different hardness assumptions, show some matching upper bounds and propose several conjectures based on this accumulated knowledge. Moreover, we conjecture that there is a strong correlation between polynomial hardness in time and space. We note that in order to discuss space it is often more natural to consider data structure variants of problems and this is the approach we follow in this paper.

1.3 Our Results

Set Disjointness. In the SetDisjointness problem mentioned before, it is required to preprocess a collection of m sets $S_1, \cdots, S_m \subset U$, where U is the universe of elements and the total number of elements in all sets is N. For a query, a pair of integers (i, j) $(1 \leq i, j \leq m)$ is given and we are asked whether $S_i \cap S_j$ is empty or not. A folklore conjecture, which appears in [16, 27], suggests that to achieve a constant query time the space of the data structure constructed in the preprocessing stage needs to be $\tilde{\Omega}(N^2)$. We call this conjecture the SetDisjointness conjecture. This conjecture does not say anything about the case where we allow *higher* query time. Therefore, we suggest a stronger conjecture which admits a *full tradeoff* between the space consumed by the data structure (denoted by S) and the query time (denoted by T). This is what we call the Strong SetDisjointness conjecture. This conjecture states that for solving SetDisjointness with a query time T our data structure needs $\tilde{\Omega}(N^2/T^2)$ space. A matching upper bound exists for this problem by generalizing ideas from [16] (see also [22]). Our new SetDisjointness conjecture can be used to admit more expressive space lower bounds for a full tradeoff between space and query time.

3SUM Indexing. One of the basic and frequently used hardness conjectures is the celebrated 3SUM conjecture. This conjecture was used for about 20 years to show many conditional *time* lower bounds on various problems. However, we focus on what can be said about its *space* behavior. To do this, it is natural to consider a data structure version of 3SUM which allows one to preprocess the input set S. Then, the query is an external number z for which we need to answer whether there are $x, y \in S$ such that $x + y = z$. It was pointed out by Chan and Lewenstein [15] that all known algorithms for 3SUM actually work within this model as well. We call this problem *3SUM Indexing*. On one hand, this problem can easily be solved using $O(n^2)$ space by sorting $x + y$ for all $x, y \in S$ and then searching for z in $\tilde{O}(1)$ time. On the other hand, by just sorting S we can answer queries by a well-known linear time algorithm. The big question is whether we can obtain better than $\tilde{\Omega}(n^2)$ space while using just $\tilde{O}(1)$ time query? Can it be done even if we allow $\tilde{O}(n^{1-\Omega(1)})$ query time? This leads us to our two new hardness conjectures. The 3SUM-Indexing conjecture states that when using $\tilde{O}(1)$ query time we need $\tilde{\Omega}(n^2)$ space to solve 3SUM-Indexing. In

the Strong 3SUM-Indexing conjecture we say that even when using $\tilde{O}(n^{1-\Omega(1)})$ query time we need $\tilde{\Omega}(n^2)$ space to solve 3SUM-Indexing.

3SUM Indexing and Set Disjointness. We prove connections between the SetDisjointness conjectures and the 3SUM-Indexing conjectures. Specifically, we show that the Strong 3SUM-Indexing conjecture implies the Strong SetDisjointness conjecture, while the SetDisjointness conjecture implies the 3SUM-Indexing conjecture. This gives some evidence towards establishing the difficulty within the 3SUM-Indexing conjectures. The usefulness of these conjectures should not be underestimated. As many problems are known to be 3SUM-hard these new conjectures can play an important role in achieving *space* lower bounds on their corresponding data structure variants. Moreover, it is interesting to point on the difference between SetDisjointness which admits smooth tradeoff between space and query time and 3SUM-Indexing which admits a big gap between the two trivial extremes. This may explain why we are unable to show full equivalence between the hardness conjectures of the two problems. Moreover, it can suggest a separation between problems with smooth space-time behavior and others which have no such tradeoff but rather two "far" extremes.

Generalizations. Following the discussion on the SetDisjointness and the 3SUM-Indexing conjectures we investigate their generalizations.

I. k-Set Disjointness and (k+1)-SUM Indexing. The first generalization is a natural parametrization of both problems. In the SetDisjointness problem we query about the emptiness of the intersection between *two* sets, while in the 3SUM-Indexing problem we ask, given a query number z, whether *two* numbers of the input S sum up to z. In the parameterized versions of these problems we are interested in the emptiness of the intersection between k sets and ask if k numbers sum up to a number given as a query. These generalized variants are called k-SetDisjointness and (k+1)-SUM-Indexing respectively. For each problem we give corresponding space lower bounds conjectures which generalize those of SetDisjointness and 3SUM-Indexing. These conjectures also have corresponding strong variants which are accompanied by matching upper bounds. We prove that the k-SetDisjointness conjecture implies (k+1)-SUM-Indexing conjecture via a novel method using linear equations.

II. k-Reachability. A second generalization is the problem we call k-Reachability. In this problem we are given as an input a directed sparse graph $G = (V, E)$ for preprocessing. Afterwards, for a query, given as a pair of vertices u, v, we wish to return if there is a path from u to v consisting of at most k edges. We provide an upper bound on this problem for every fixed $k \geq 1$. The upper bound admits a tradeoff between the space of the data structure (denoted by S) and the query time (denoted by T), which is $ST^{2/(k-1)} = O(n^2)$. We argue that this upper bound is tight. That is, we conjecture that if query takes T time, the space must be $\tilde{\Omega}(\frac{n^2}{T^{2/(k-1)}})$. We call this conjecture the k-Reachability conjecture.

We give three indications towards the correctness of this conjecture. First, we prove that the base case, where $k = 2$, is equivalent to the SetDisjointness problem. This is why this problem can be thought of as a generalization of SetDisjointness.

Second, if we consider non-constant k then the smooth tradeoff surprisingly disappears and we get "extreme behavior" as $\tilde{\Omega}(\frac{n^2}{T^{2/(k-1)}})$ eventually becomes $\tilde{\Omega}(n^2)$. This means that to answer reachability queries for non-constant path length, we can either store all answers in advance using n^2 space or simply answer queries from scratch using a standard graph traversal algorithm. The general problem where the length of the path from u to v is unlimited in length is sometimes referred to as the problem of constructing efficient reachability oracles. Pătraşcu in [26] leaves it as an open question if a data structure with less than $\tilde{\Omega}(n^2)$ space can answer reachability queries efficiently. Moreover, Pătraşcu proved that for constant time query, truly superlinear space is needed. Our k-Reachability conjecture points to this direction, while admitting full space-time tradeoff for constant k.

The third indication for the correctness of the k-Reachability conjecture comes from a connection to distance oracles. A *distance oracle* is a data structure that can be used to quickly answer queries about the shortest path between two given nodes in a preprocessed undirected graph. As mentioned above, the SetDisjointness conjecture was used to exclude some possible tradeoffs for sparse graphs. Specifically, Cohen and Porat [17] showed that obtaining an approximation ratio smaller than 2 with constant query time requires $\tilde{\Omega}(n^2)$ space. Using a somewhat stronger conjecture Pătraşcu and Roditty [27] showed that a (2,1)-distance oracle for unweighted graphs with $m = O(n)$ edges requires $\tilde{\Omega}(n^{1.5})$ space. Later, this result was strengthened by Pătraşcu et al. [28]. However, these results do not exclude the possibility of compact distance oracles if we allow higher query time. For stretch-2 and stretch-3 in sparse graphs, Agarwal et. al. [9, 10] achieved a space-time tradeoff of $S \times T = O(n^2)$ and $S \times T^2 = O(n^2)$, respectively. Agarwal [8] also showed many other results for stretch-2 and below. We use our k-Reachability conjecture to prove that for stretch-less-than-(1+2/k) distance oracles $S \times T^{2/(k-1)}$ is bounded by $\tilde{\Omega}(n^2)$. This result is interesting in light of Agarwal [8] where a stretch-(5/3) oracle was presented which achieves a space-time tradeoff of $S \times T = O(n^2)$. This matches our lower bound, where $k = 3$, if our lower bound would hold not only for stretch-less-than-(5/3) but also for stretch-(5/3) oracles. Consequently, we see that there is strong evidence for the correctness of the k-Reachability conjecture.

Moreover, these observations show that on one hand k-Reachability is a generalization of SetDisjointness which is closely related to 3SUM-Indexing. On the other hand, k-Reachability is related to distance oracles which solve the famous APSP problem using smaller space by sacrificing the accuracy of the distance between the vertices. Therefore, the k-Reachability conjecture seems as a conjecture corresponding to the APSP hardness conjecture, while also admitting some connection with the celebrated 3SUM hardness conjecture.

SETH and Orthogonal Vectors. After considering space variants of the 3SUM and APSP conjectures it is natural to consider space variants for the Strong Exponential Time Hypothesis (SETH) and the closely related conjecture of orthogonal vectors. SETH asserts that for any $\epsilon > 0$ there is an integer $k > 3$ such that k-SAT cannot be solved in $2^{(1-\epsilon)n}$ time. The orthogonal vectors time conjecture states that there is no algorithm that for every $c \geq 1$, finds if there are at least two orthogonal vectors in a set of n Boolean vectors of length $c \log n$ in $\tilde{O}(n^{2-\Omega(1)})$ time. A discussion about the space variants of these conjectures will appear in the full version of this paper. However, we note that we are unable to connect these conjectures and the previous ones. This is perhaps not surprising as the connection between SETH and the other conjectures even in the time perspective is very loose (see, for example, discussions in [5, 20]).

Boolean Matrix Multiplication. Another problem which receives a lot of attention in the context of conditional time lower bounds is calculating Boolean Matrix Multiplication (BMM). We give a data structure variant of this well-known problem. We then demonstrate the connection between this problem and the problems of SetDisjointness and k-Reachability. The discussion about BMM and its data structure variant will appear in the full version of this paper.

Applications. Finally, armed with the *space* variants of many well-known conditional *time* lower bounds, we apply this conditional space lower bounds to some static and dynamic problems. This gives interesting space lower bound results on these important problems which sometimes also admits clear space-time tradeoff. The list of problems that we prove their conditional space-time hardness includes: edge triangles, histogram indexing, distance oracles for colors, two patterns document retrieval, forbidden pattern document retrieval, (s,t)-reachability, bipartite perfect matching and strong connectivity. All the results regarding the applications of our framework will appear in the full version of this paper. We believe that this is just a glimpse of space lower bounds that can be achieved based on our new framework and that many other interesting results are expected to follow this promising route.

2 Set Intersection Hardness Conjectures

We first give formal definitions of the SetDisjointness problem and its enumeration variant:

Problem 1 (SetDisjointness Problem). Preprocess a family F of m sets, all from universe U, with total size $N = \sum_{S \in F} |S|$ so that given two query sets $S, S' \in F$ one can determine if $S \cap S' = \emptyset$.

Problem 2 (SetIntersection Problem). Preprocess a family F of m sets, all from universe U, with total size $N = \sum_{S \in F} |S|$ so that given two query sets $S, S' \in F$ one can enumerate the set $S \cap S'$.

Conjectures. The SetDisjointness problem was regarded as a problem that admits space hardness. The hardness conjecture of the SetDisjointness problem has received several closely related formulations. One such formulation, given by Pătraşcu and Roditty [27], is as follows:

Conjecture 1. **SetDisjointness Conjecture [Formulation 1].** Any data structure for the SetDisjointness problem where $|U| = \log^c m$ for a large enough constant c and with a constant query time must use $\tilde{\Omega}(m^2)$ space.

Another formulation is implicitly suggested in Cohen and Porat [16]:

Conjecture 2. **SetDisjointness Conjecture [Formulation 2].** Any data structure for the SetDisjointness problem with constant query time must use $\tilde{\Omega}(N^2)$ space.

There is an important distinction between the two formulations, which is related to the sparsity of SetDisjointness instances. This distinction follows from the following upper bound: store an $m \times m$ matrix of the answers to all possible queries, and then queries will cost constant time. The first formulation of the SetDisjointness conjecture states that if we want constant (or poly-logaritmic) query time, then this is the best we can do. At a first glance this makes the second formulation, whose bounds are in terms of N and not m, look rather weak. In particular, why would we ever be interested in a data structure that uses $O(N^2)$ space when we can use one with $O(m^2)$ space? The answer is that the two conjectures are the same if the sets are very sparse, and so at least in terms of N, if one were to require a constant query time then by the second formulation the space must be at least $\Omega(N^2)$ (which happens in the very sparse case).

Nevertheless, we present a more general conjecture, which in particular captures a tradeoff curve between the space usage and query time. This formulation captures the difficulty that is commonly believed to arise from the SetDisjointness problem, and matches the upper bounds of Cohen and Porat [16] (see also [22]).

Conjecture 3. **Strong SetDisjointness Conjecture.** Any data structure for the SetDisjointness problem that answers queries in T time must use $S = \tilde{\Omega}(\frac{N^2}{T^2})$ space.

For example, a natural question to ask is "what is the smallest query time possible with linear space?". This question is addressed, at least from a lower bound perspective, by the Strong SetDisjointness conjecture.

Conjecture 4. **Strong SetIntersection Conjecture.** Any data structure for the SetIntersection problem that answers queries in $O(T + op)$ time, where op is the size of the output of the query, must use $S = \tilde{\Omega}(\frac{N^2}{T})$ space.

3 3SUM-Indexing Hardness Conjectures

In the classic 3SUM problem we are given an integer array A of size n and we wish to decide whether there are 3 distinct integers in A which sum up to zero. Gajentaan and Overmars [18] showed that an equivalent formulation of this problem receives 3 integer arrays A_1, A_2, and A_3, each of size n, and the goal is to decide if there is a triplet $x_1 \in A_1, x_2 \in A_2$, and $x_3 \in A_3$ that sum up to zero.

We consider the data structure variant of this problem which is formally defined as follows:

Problem 3 (3SUM-Indexing Problem). Preprocess two integer arrays A_1 and A_2, each of length n, so that given a query integer z we can decide whether there are $x \in A_1$ and $y \in A_2$ such that $z = x + y$.

It is straightforward to maintain all possible $O(n^2)$ sums of pairs in quadratic space, and then answer a query in $\tilde{O}(1)$ time. On the other extreme, if one does not wish to utilize more than linear space then one can sort the arrays separately during preprocessing time, and then a query can be answered in $\tilde{O}(n)$ time by scanning both of the sorted arrays in parallel and in opposite directions.

We introduce two conjectures with regards to the 3SUM-Indexing problem, which serve as natural candidates for proving polynomial space lower bounds.

Conjecture 5. **3SUM-Indexing Conjecture**: There is no solution for the 3SUM-Indexing problem with truly subquadratic space and $\tilde{O}(1)$ query time.

Conjecture 6. **Strong 3SUM-Indexing Conjecture**: There is no solution for the 3SUM-Indexing problem with truly subquadratic space and truly sublinear query time.

Notice that one can solve the classic 3SUM problem using a data structure for 3SUM-Indexing by preprocessing A_1 and A_2, and answering n 3SUM-Indexing queries on all of the values in A_3.

Next, we prove theorems that show tight connections between the 3SUM-Indexing conjectures and the SetDisjointness conjectures. We note that the proofs of the first two theorems are similar to the proofs of [23], but with space interpretation. These proofs will appear in the full version of this paper.

Theorem 1. *The Strong 3SUM-Indexing Conjecture implies the Strong SetDisjointness Conjecture.*

Theorem 2. *The Strong 3SUM-Indexing Conjecture implies the Strong SetIntersection Conjecture.*

Theorem 3. *The SetDisjointness Conjecture implies the 3SUM-Indexing Conjecture.*

Proof. Given an instance of SetDisjointness, we construct an instance of 3SUM-Indexing as follows. Denote with M the value of the largest element in the SetDisjointness instance. Notice that we may assume that $M \leq N$ (otherwise we can use a straightforward renaming). For every element $x \in U$ that is contained in at least one of the sets we create two integers x_A and x_B, which are represented by $2\lceil \log m \rceil + \lceil \log N \rceil + 3$ bits each (recall that m is the number of sets).

The $\lceil \log N \rceil$ least significant bits in x_A represent the value of x. The following bit is a zero. The following $\lceil \log m \rceil$ bits in x_A represent the index of the set containing x, and the rest of the $2 + \lceil \log m \rceil$ are all set to zero. The $\lceil \log N \rceil$ least significant bits in x_B represent the value of $M - x$. The following $2 + \lceil \log m \rceil$ are all set to zero. The following $\lceil \log m \rceil$ bits in x_B represent the index of the set containing x, and the last bit is set to zero. Finally, the integer x_A is added to A_1 of the 3SUM-Indexing instance, while the integer x_B is added to A_2.

We have created two sets of $n \leq M$ integers. We then preprocess them to answer 3SUM-Indexing queries. Now, to answer a SetDisjointness query on sets S_i and S_j, we query the 3SUM-Indexing data structure with an integer z which is determined as follows. The $\lceil \log N \rceil$ least significant bits in z represent the value of M. The following bit is a zero. The following $\lceil \log m \rceil$ bits represent the index i and are followed by a zero. The next $\lceil \log m \rceil$ bits represent the index j and the last bit is set to zero.

It is straightforward to verify that there exists a solution to the 3SUM-Indexing problem on z if and only if the sets S_i and S_j are not disjoint. Therefore, if there is a solution to the 3SUM-Indexing problem with less than $\tilde{\Omega}(n^2)$ space and constant query time then there is a solution for the SetDisjointness problem which refutes the SetDisjointness Conjecture. □

4 Parameterized Generalization: k-Set Intersection and (k+1)-SUM

Two parameterized generalizations of the SetDisjointness and 3SUM-Indexing problems are formally defined as follows:

Problem 4 (k-SetDisjointness Problem). Preprocess a family F of m sets, all from universe U, with total size $N = \sum_{S \in F} |S|$ so that given k query sets $S_1, S_2, \ldots, S_k \in F$ one can quickly determine if $\cap_{i=1}^{k} S_i = \emptyset$.

Problem 5 ((k+1)-SUM-Indexing Problem). Preprocess k integer arrays $A_1, A_2, \ldots, A_k$, each of length n, so that given a query integer z we can decide if there is $x_1 \in A_1, x_2 \in A_2, \ldots, x_k \in A_k$ such that $z = \sum_{i=1}^{k} x_i$.

It turn out that a natural generalization of the data structure of Cohen and Porat [16] leads to a data structure for k-SetDisjointness as shown in the following lemma.

Lemma 1. *There exists a data structure for the k-SetDisjointness problem where the query time is T and the space usage is $S = O((N/T)^k)$.*

Proof. We call the f largest sets in F *large sets*. The rest of the sets are called *small sets*. In the preprocessing stage we explicitly maintain a k-dimensional table with the answers for all k-SetDisjointness queries where all k sets are large sets. The space needed for such a table is $S = f^k$. Moreover, for each set (large or small) we maintain a look-up table that supports disjointness queries (with this set) in constant time. Since there are f large sets and the total number of elements is N, the size of each of the small sets is at most N/f.

Given a k-SetDisjointness query, if all of the query sets are large then we look up the answer in the k-dimensional table. If at least one of the sets is small then using a brute-force search we look-up each of the at most $O(N/f)$ elements in each of the other $k-1$ sets. Thus, the total query time is bounded by $O(kN/f)$, and the space usage is $S = O(f^k)$. The rest follows.

Notice that for the case of $k = 2$ in Lemma 1 we obtain the same tradeoff of Cohen and Porat [16] for SetDisjointness. The following conjecture suggests that the upper bound of Lemma 1 is the best possible.

Conjecture 7. **Strong k-SetDisjointness Conjecture.** Any data structure for the k-SetDisjointness problem that answers queries in T time must use $S = \tilde{\Omega}(\frac{N^k}{T^k})$ space.

Similarly, a natural generalization of the Strong 3SUM-Indexing conjecture is the following.

Conjecture 8. **Strong (k+1)-SUM-Indexing Conjecture.** There is no solution for the (k+1)-SUM-Indexing problem with $\tilde{O}(n^{k-\Omega(1)})$ space and truly sublinear query time.

We also consider some weaker conjectures, similar to the SetDisjointness and 3SUM-Indexing conjectures.

Conjecture 9. **k-SetDisjointness Conjecture.** Any data structure for the k-SetDisjointness problem that answers queries in constant time must use $\tilde{\Omega}(N^k)$ space.

Conjecture 10. **(k+1)-SUM-Indexing Conjecture.** There is no solution for the (k+1)-SUM-Indexing problem with $\tilde{O}(n^{k-\Omega(1)})$ space and constant query time.

Similar to Theorem 3, we prove the following relationship between the k-SetDisjointness conjecture and the (k+1)-SUM-Indexing conjecture.

Theorem 4. *The k-SetDisjointness conjecture implies the (k+1)-SUM-Indexing conjecture*

Proof. Given an instance of k-SetDisjointness, we construct k instances of (k+1)-SUM-Indexing as follows. Denote by M the value of the largest element in the SetDisjointness instance. Notice that we may assume that $M \leq N$ (otherwise we use a straightforward renaming). For every element $x \in U$ that is contained in

at least one of the sets we create k^2 integers in a matrix $X = \{x_{i,j}\}$ of size $k \times k$, where each integer is represented by $(k-1)\lceil \log m \rceil + \lceil \log N \rceil + k$ bits.

For integer $x_{i,j}$, the $\lceil \log N \rceil + 1$ least significant bits represent the value of $(k-1)x$ if $i = j$, and the value of $M - x$ otherwise. The $(k-1)\lceil \log m \rceil + k - 1$ following bits are all set to zero, except for the bits in indices $(j-1)(\lceil \log m \rceil + 1) + 1, ..., j(\lceil \log m \rceil + 1)$ which represent the index of the set containing x.

We now create k instances of (k+1)-SUM-Indexing where the jth input array A_j for the ith instance is the set of integers $x_{i,j}$ for all $x \in U$ that are contained in at least one set of our family. Thus, the size of each array is at most N. Now, given a k-SetDisjointness query $(i_1, i_2, ..., i_k)$ we must decide if $S_{i_1} \cap S_{i_2} \cap ... \cap S_{i_k} = \emptyset$. To answer this query we will query each of the k instances of (k+1)-SUM-Indexing with an integer z whose binary representation is as follows: the $\lceil \log N \rceil + 1$ least significant bits represent the value of $(k-1)M$, and the bits at locations $(j-1)(\lceil \log m \rceil + 1) + 1, ..., j(\lceil \log m \rceil + 1)$ representing i_j (for $1 \leq j \leq k$). The rest of the bits are padding zero bit (in between representations of various i_j).

If $S_{i_1} \cap S_{i_2} \cap ... \cap S_{i_k} \neq \emptyset$ then by our construction it is straightforward to verify that all of the k (k+1)-SUM-Indexing queries on z will return that there is a solution. If $S_{i_1} \cap S_{i_2} \cap ... \cap S_{i_k} = \emptyset$ then at least one (k+1)-SUM-Indexing query will not be able to find a solution. This is because we can view each instance and query as solving a linear equation. As we construct k instances which represent k independent linear equations, we are guaranteed that only one solution exists. This solution is exactly the one that corresponds to finding a specific x which is contained in all of the k sets. Therefore, we get a correct answer to a k-SetDisjointness query by answering k (k+1)-SUM-Indexing queries.

Consequently, if for some specific constant k there is a solution to the (k+1)-SUM-Indexing problem with less than $\tilde{\Omega}(n^k)$ space and constant query time, then with this reduction we refute the k-SetDisjointness conjecture. □

5 Directed Reachability Oracles as a Generalization of Set Disjointness Conjecture

An open question which was stated by Pătraşcu in [26] asks if it is possible to preprocess a sparse directed graph in less than $\Omega(n^2)$ space so that Reachability queries (given two query vertices u and v decide whether there is a path from u to v or not) can be answered efficiently. A partial answer, given in [26], states that for constant query time truly superlinear space is necessary. In the undirected case the question is trivial and one can answer queries in constant time using linear space. This is also possible for planar directed graphs (see Holm et al. [21]).

We now show that Reachability oracles for sparse graphs can serve as a generalization of the SetDisjointness conjecture. We define the following parameterized version of Reachability. In the *k-Reachability problem* the goal is to preprocess a directed sparse graph $G = (V, E)$ so that given a pair of distinct vertices $u, v \in V$ one can quickly answer whether there is a path from u to v consisting of at most k edges. We prove that 2-Reachability and SetDisjointness are tightly connected.

Lemma 2. *There is a linear time reduction from* SetDisjointness *to 2-Reachability and vice versa which preserves the size of the instance.*

Proof. Given a graph $G = (V, E)$ as an instance for 2-Reachability, we construct a corresponding instance of SetDisjointness as follows. For each vertex v we create the sets $V_{in} = \{u|(u, v) \in E\}$ and $V_{out} = \{u|(v, u) \in E\} \cup \{v\}$. We have $2n$ sets and $2m + n$ elements in all of them ($|V| = n$ and $|E| = m$). Now, a query u, v is reduced to determining if the sets U_{out} and V_{in} are disjoint or not. Notice, that the construction is done in linear time and preserves the size of the instance. In the opposite direction, we are given m sets $S_1, S_2, ..., S_m$ having N elements in total $e_1, e_2, ..., e_N$. We can create an instance of 2-Reachability in the following way. For each set S_i we create a vertex v_i. Moreover, for each element e_j we create a vertex u_j. Then, for each element e_j in a set s_i we create two directed edges (v_i, u_j) and (u_j, v_i). These vertices and edges define a directed graph, which is preprocessed for 2-Reachability queries. It is straightforward to verify that the disjointness of S_i and S_j is equivalent to determining if there is a path of length at most 2 edges from v_i to v_j. Moreover, the construction is done in linear time and preserves the size of the instance. □

Furthermore, we consider k-Reachability for $k \geq 3$. First we show an upper bound on the tradeoff between space and query time for solving k-Reachability.

Lemma 3. *There exists a data structure for k-Reachability with S space and T query time such that $ST^{2/(k-1)} = O(n^2)$.*

Proof. Let $\alpha > 0$ be an integer parameter to be set later. Given a directed graph $G = (V, E)$, we call vertex $v \in V$ a *heavy vertex* if $deg(v) > \alpha$ and a vertex $u \in V$ a *light vertex* if $deg(u) \leq \alpha$. Notice that the number of heavy vertices is at most n/α. For all heavy vertices in V we maintain a matrix containing the answers to any k-Reachability query between two heavy vertices. This uses $O(n^2/\alpha^2)$ space.

Next, we recursively construct a data structure for (k-1)-Reachability. Given a query u, v, if both vertices are heavy then the answer is obtained from the matrix. Otherwise, either u or v is light vertex. Without loss of generality, say u is a light vertex. We consider each vertex $w \in N_{out}(u)$ ($N_{out}(u) = \{v|(u, v) \in E\}$) and query the (k-1)-Reachability data structure with the pair w, v. Since u is a light node, there are no more than α queries. One of the queries returns a positive answer if and only if there exists a path of length at most k from u to v.

Denote by $S(k, n)$ the space used by our k-Reachability oracle on a graph with n vertices and denote by $Q(k, n)$ the corresponding query time. In our construction we have $S(k, n) = n^2/\alpha^2 + S(k - 1, n)$ and $Q(k, n) = \alpha Q(k - 1, n) + O(1)$. For $k = 1$ it is easy to construct a linear space data structure using hashing so that queries can be answered in constant time. Thus, $S = S(k, n) = O((k - 1)n^2/\alpha^2)$ and $T = Q(k, n) = O(\alpha^{k-1})$. □

Notice that for the case of $k = 2$ the upper bounds from Lemma 3 exactly match the tradeoff of the Strong SetDisjointness Conjecture ($ST^2 = \tilde{O}(n^2)$). We

expand this conjecture by considering the tightness of our upper bound for k-Reachability, which then leads to some interesting consequences with regard to distance oracles.

Conjecture 11. **Directed k-Reachability Conjecture.** Any data structure for the k-Reachability problem with query time T must use $S = \tilde{\Omega}(\frac{n^2}{T^{2/(k-1)}})$ space.

Notice that when k is non-constant then by our upper bound $\tilde{\Omega}(n^2)$ space is necessary independent of the query time. This fits nicely with what is currently known about the general question of Reachability oracles: either we spend n^2 space and answer queries in constant time or we do no preprocessing and then answer queries in linear time. This leads to the following conjecture.

Conjecture 12. **Directed Reachability Hypothesis.** Any data structure for the Reachability problem must either use $\tilde{\Omega}(n^2)$ space, or linear query time.

The conjecture states that in the general case of Reachability there is no full tradeoff between space and query time. We believe the conjecture is true even if the path is limited to lengths of some non-constant number of edges.

6 Distance Oracles and Directed Reachability

There are known lower bounds for constant query time distance oracles based on the SetDisjointness hypothesis. Specifically, Cohen and Porat [16] showed that stretch-less-than-2 oracles need $\Omega(n^2)$ space for constant queries. Patrascu et al. [28] showed a conditional space lower bound of $\Omega(m^{5/3})$ for constant-time stretch-2 oracles. Applying the Strong SetDisjointness conjecture to the same argument as in [16] we can prove that for stretch-less-than-2 oracles the tradeoff between S (the space for the oracle) and T (the query time) is by $S \times T^2 = \Omega(n^2)$.

Recent effort was taken toward constructing compact distance oracles where we allow non-constant query time. For stretch-2 and stretch-3 Agarwal et al. [10] [9] achieves a space-time tradeoff of $S \times T = O(n^2)$ and $S \times T^2 = O(n^2)$, respectively, for sparse graphs. Agarwal [8] also showed many other results for stretch-2 and below. Specifically, Agarwal showed that for any integer k a stretch-(1+1/k) oracle exhibits the following space-time tradeoff: $S \times T^{1/k} = O(n^2)$. Agarwal also showed a stretch-(1+1/(k+0.5)) oracle that exhibits the following tradeoff: $S \times T^{1/(k+1)} = O(n^2)$. Finally, Agarwal gave a stretch-(5/3) oracle that achieves a space-time tradeoff of $S \times T = O(n^2)$. Unfortunately, no lower bounds are known for non-constant query time.

Conditioned on the directed k-Reachability conjecture we prove the following lower bound.

Lemma 4. *Assume the directed k-Reachability conjecture holds. Then stretch-less-than-*$(1 + 2/k)$ *distance oracles with query time* T *must use* $S \times T^{2/(k-1)} = \tilde{\Omega}(n^2)$ *space.*

Proof. Given a graph $G = (V, E)$ for which we want to preprocess for k-Reachability, we create a layered graph with k layers where each layer consists of a copy of all vertices of V. Each pair of neighboring layers is connected by a copy of all edges in E. We omit all directions from the edges. For every fixed integer k, the layered graph has $O(|V|)$ vertices and $O(|E|)$ edges. Next, notice that if we construct a distance oracle that can distinguish between pairs of vertices of distance at most k and pairs of vertices of distance at least $k + 2$, then we can answer k-Reachability queries. Consequently, assuming the k-Reachability conjecture we have that $S \times T^{2/(k-1)} = \Omega(n^2)$ for stretch-less-than-$(1 + 2/k)$ distance oracles (For $k = 2$ this is exactly the result we get by the SetDisjointness hypothesis). □

Notice, that the stretch-(5/3) oracle shown by Agarwal [8] achieves a space-time tradeoff of $S \times T = O(n^2)$. Our lower bound is very close to this upper bound since it applies for any distance oracle with stretch-less-than-(5/3), by setting $k = 3$.

References

1. Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proc. of 27th ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1256–1271, 2016.
2. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 98–117, 2015.
3. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 59–78, 2015.
4. Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015.
5. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.
6. Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 39–51, 2014.
7. Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 41–50, 2015.
8. Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In *Algorithms - ESA 2014 - 22th Annual European Symposium on Algorithms, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 49–60, 2014.
9. Rachit Agarwal, Brighten Godfrey, and Sariel Har-Peled. Faster approximate distance queries and compact routing in sparse graphs. *CoRR*, abs/1201.2703, 2012.

10. 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*, pages 1754–1762, 2011.
11. Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 114–125, 2014.
12. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.
13. Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014.
14. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2015.
15. Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40, 2015.
16. Hagai Cohen and Ely Porat. Fast set intersection and two-patterns matching. *Theor. Comput. Sci.*, 411(40-42):3795–3800, 2010.
17. Hagai Cohen and Ely Porat. On the hardness of distance oracle for sparse graph. *CoRR*, abs/1006.1117, 2010.
18. A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.
19. Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In *European Symposium on Algorithms, ESA 2016*, pages 45:1–45:16, 2016.
20. Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30, 2015.
21. Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Planar reachability in linear space and constant time. In *56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 370–389, 2015.
22. Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Dynamic set intersection. In *Proceedings 14th Int'l Symposium on Algorithms and Data Structures (WADS)*, pages 470–481, 2015.
23. Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287, 2016.
24. Kasper Green Larsen, J. Ian Munro, Jesper Sindahl Nielsen, and Sharma V. Thankachan. On hardness of several string indexing problems. *Theor. Comput. Sci.*, 582:74–82, 2015.

25. Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610, 2010.
26. Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011.
27. Mihai Patrascu and Liam Roditty. Distance oracles beyond the Thorup-Zwick bound. *SIAM J. Comput.*, 43(1):300–311, 2014.
28. Mihai Patrascu, 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, 2012.
29. Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.
30. Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654, 2010.