Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture*

Monika Henzinger†
University of Vienna
monika.henzinger@univie.ac.at

Sebastian Krinninger
University of Vienna
sebastian.krinninger@univie.ac.at

Danupon Nanongkai‡
KTH Royal Institute of Technology
danupon@gmail.com

Thatchaphol Saranurak§
KTH Royal Institute of Technology
thatchaphol.s@gmail.com

ABSTRACT

Consider the following *Online Boolean Matrix-Vector Multiplication* problem: We are given an $n \times n$ matrix M and will receive n column-vectors of size n, denoted by $v_1, \ldots, v_n$, one by one. After seeing each vector v_i, we have to output the product Mv_i before we can see the next vector. A naive algorithm can solve this problem using $O(n^3)$ time in total, and its running time can be slightly improved to $O(n^3/\log^2 n)$ [Williams SODA'07].

We show that a conjecture that there is no *truly subcubic* ($O(n^{3-\epsilon})$) time algorithm for this problem can be used to exhibit the underlying polynomial time hardness shared by many dynamic problems. For a number of problems, such as subgraph connectivity, Pagh's problem, d-failure connectivity, decremental single-source shortest paths, and decremental transitive closure, this conjecture implies tight hardness results. Thus, proving or disproving this conjecture will be very interesting as it will either imply several tight unconditional lower bounds or break through a common barrier that blocks progress with these problems. This conjecture might also be considered as strong evidence against any further improvement for these problems since refuting it will imply a major breakthrough for combinatorial Boolean matrix multiplication and other long-standing problems if the term "combinatorial algorithms" is interpreted as "Strassen-like algorithms" [Ballard et al. SPAA'11].

The conjecture also leads to hardness results for problems that were previously based on diverse problems and conjectures – such as 3SUM, combinatorial Boolean matrix multiplication, triangle detection, and multiphase – thus providing a uniform way to prove polynomial hardness results for dynamic algorithms; some of the new proofs are also simpler or even become trivial. The conjecture also leads to stronger and new, non-trivial, hardness results, e.g., for the fully-dynamic densest subgraph and diameter problems.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms, Theory

Keywords

dynamic graph algorithms; lower bounds; matrix multiplication

1. INTRODUCTION

Consider the following problem called *Online Boolean Matrix-Vector Multiplication* (OMv): Initially, an algorithm is given an integer n and an $n \times n$ Boolean matrix M. Then, the following protocol repeats for n rounds: At the i^{th} round, it is given an n-dimensional column vector, denoted by v_i, and has to compute Mv_i. It has to output the resulting column vector before it can proceed to the next round. We want the algorithm to finish the computation as quickly as possible.

This problem is a generalization of the classic *Matrix-Vector Multiplication* problem (Mv), which is the special case with only one vector. The main question is whether we can

*The full version of this paper is available at `https://eprints.cs.univie.ac.at/4351/`

†The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement number 340506.

‡Work partially done while at University of Vienna, Austria.

§Work partially done while at University of Vienna, Austria and Saarland University, Germany.

STOC'15, June 14–17, 2015, Portland, Oregon, USA.

http://dx.doi.org/10.1145/2746539.2746609.

preprocess the matrix in order to make the multiplication with n sequentially given vectors faster than n matrix-vector multiplications. This study dates back to as far as 1955 (e.g. [31]), but most major theoretical work has focused on structured matrices; see, e.g. [32, 47] for more information. A naive algorithm can multiply the matrix with each vector in $O(n^2)$ time, and thus requires $O(n^3)$ time in total. It was long known that the matrix can be preprocessed in $O(n^2)$ time in order to compute Mv_i in $O(n^2/\log n)$ time, implying an $O(n^3/\log n)$ time algorithm for OMv; see, e.g., [42, 41] and a recent extension in [30]. More recently, Williams [47] showed that the matrix can be preprocessed in $O(n^{2+\epsilon})$ time in order to compute Mv_i in $O(n^2/\epsilon \log^2 n)$ time for any $0 < \epsilon < 1/2$, implying an $O(n^3/\log^2 n)$ time algorithm for OMv. This is the current best running time for OMv. In this light, it is natural to conjecture that this problem does not admit a so-called *truly subcubic time algorithm*:

CONJECTURE 1.1 (OMv CONJECTURE). *For any constant $\epsilon > 0$, there is no $O(n^{3-\epsilon})$-time algorithm that solves OMv correctly with probability at least $2/3$.*

In fact, it can be argued that this conjecture is implied by the standard *combinatorial Boolean matrix multiplication* (BMM) conjecture which states that there is no truly subcubic ($O(n^{3-\varepsilon})$) time *combinatorial* algorithm for multiplying two $n \times n$ Boolean matrices if the term "combinatorial algorithms" (which is not a well-defined term) is interpreted in a certain way – in particular if it is interpreted as "Strassen-like algorithms", as defined in [2], which captures all known fast matrix multiplication algorithms; see Section 1.2 for further discussion. Thus, breaking Conjecture 1.1 is arguably at least as hard as making a breakthrough for Boolean matrix multiplication and other long-standing open problems (e.g., [12, 49, 1, 34, 20]). This conjecture is also supported by an algebraic lower bound [8].

1.1 OMv-Hardness for Dynamic Algorithms

We show that the OMv conjecture can very well capture the underlying polynomial time hardness shared by a large number of dynamic problems, leading to a unification, simplification, and strengthening of previous results. By dynamic algorithm we mean an algorithm that allows a change to the input. It usually allows three operations: (1) *preprocessing*, which is called when the input is first received, (2) *update*, which is called for every input update, and (3) *query*, which is used to request an answer to the problem. For example, in a typical dynamic graph problem, say s-t shortest path, we will start with an empty graph at the preprocessing step. Each update operation consists of an insertion or deletion of one edge. The algorithm has to answer a query by returning the distance between s and t at that time. Corresponding to the three operations, we have *preprocessing time*, *update time*, and *query time*. There are two types of bounds on the update time: *worst-case bounds*, which bound the time that *each individual* update takes in the worst case, and *amortized bounds* which bound the time taken by *all* updates and then averaging it over all updates. The bounds on query time can be distinguished in the same way. We call a dynamic algorithm *fully-dynamic* if any of its updates can be undone (e.g. an edge insertion can be undone by an edge deletion); otherwise, we call it *partially-dynamic*. We call a partially-dynamic graph algorithm *decremental* if it allows only edge deletions, and *incremental* if it allows only edge insertions. For this type of algorithm, the update time is often analyzed in terms of *total update time*, which is the total time needed to handle *all* insertions or deletions.

Previous hardness results for dynamic problems are often based on diverse conjectures, such as those for 3SUM, combinatorial Boolean matrix multiplication (BMM), triangle detection, all-pairs shortest paths, and multiphase. This sometimes made hardness proofs quite intricate since there are many conjectures to start from, which often yield different hardness results, and in some cases none of these results are tight. Our approach results in stronger bounds which are tight for some problems. Additionally, we show that a number of previous proofs can be unified as they can now start from only one problem, that is OMv, and can be done in a much simpler way (compare, e.g., the hardness proof for Pagh's problem in this paper and in [1]). Thus proving the hardness of a problem via OMv should be a simpler task.

We next explain our main results and the differences to prior work: As shown in Figure 1, we obtain more than 15 new tight[1] hardness results[2]. (Details of these results are provided in the full version.) (1) Generally speaking, for most previous hardness results in [33, 1, 28] that rely on various conjectures, except those relying on the Strong Exponential Time Hypothesis (SETH), our OMv conjecture implies hardness bounds on the amortized time per operation that are the same or better. (2) We also obtain new results such as those for vertex color distance oracles (studied in [25, 11] and used to tackle the minimum Steiner tree problem [29]), restricted top trees with edge query problem (used to tackle the minimum cut problem in [15]), and the dynamic densest subgraph problem [7]. (3) Some minor improvement can in fact immediately be obtained since our conjecture implies a very strong bound for Pătraşcu's *multiphase* problem [33], giving improved bounds for many problems considered in [33]. We can, however, improve these bounds even more by avoiding a reduction via the multiphase problem. (We discuss this further in Section 1.2.) The conjecture leads to an improvement for all problems whose hardness was previously based on 3SUM. (4) A few other improvements follow from converting previous hardness results that hold only for *combinatorial* algorithms into hardness results that hold for *any* algorithm. We note that removing the term "combinatorial" is an important task as there are algebraic algorithms that can break through some bounds for combinatorial algorithms. (We discuss this more in Section 1.2.) (5) Interestingly, all our hardness results

[1]Our results are tight in one of the following ways: (1) the query time of the existing algorithms cannot be improved without significantly increase the update time, (2) the update time of the existing algorithms cannot be improved without significantly increase the query time, (3) the update and query time of the existing algorithms cannot be improved simultaneously, and (4) the approximation guarantee cannot be improved without significantly increasing both query and update time.

[2]For the s-t reachability problem, our result does not subsume the result based on the Boolean matrix multiplication (BMM) conjecture because the latter result holds only for combinatorial algorithms, and it is in fact larger than an upper bound provided by the non-combinatorial algorithm of Sankowski [37] (see Section 1.2 for a discussion). Also note that the result based on the triangle detection problem which is not subsumed by our result holds only for a more restricted notion of amortization (see Section 1.2). This explains the solid lines in Figure 1.

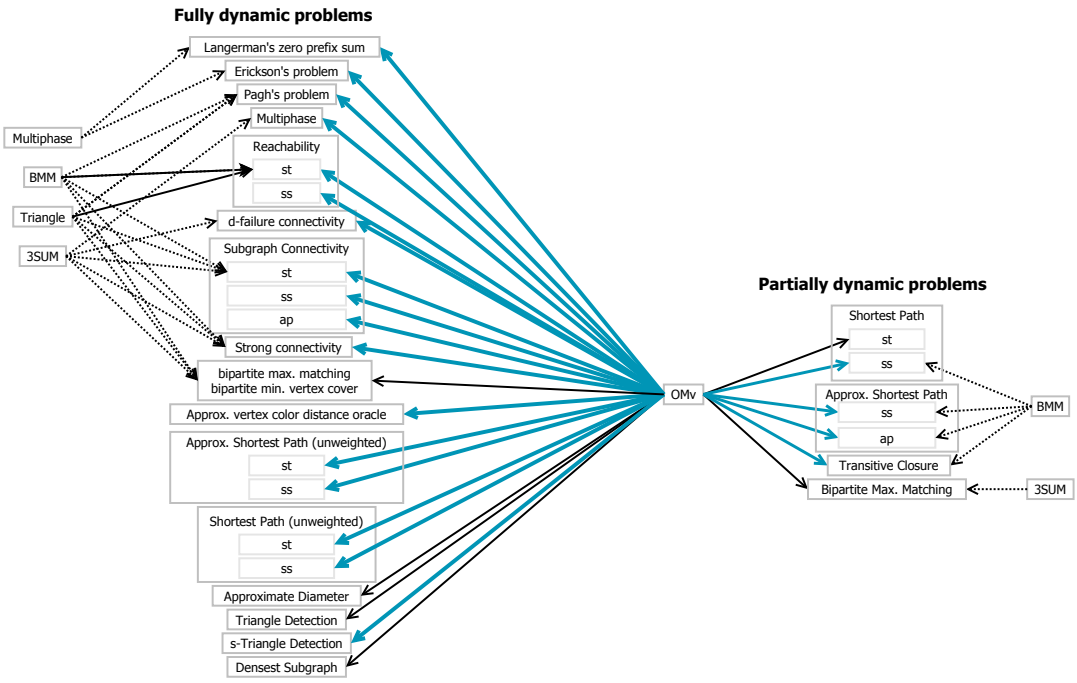

Figure 1: Overview of our and previous hardness results. An arrow from a conjecture to a problem indicates that there is a hardness result based on the conjecture. A thick blue arrow indicates that the hardness result is tight; i.e., there is a matching upper bound. A dotted arrow means that the result is subsumed in our paper. (Footnote 2 discusses results that are not subsumed.) Note that all hardness results based on BMM hold only for combinatorial algorithms.

hold even when we allow an *arbitrary polynomial preprocessing time*. This type of results was obtained earlier only in [1] for hardness results based on SETH. (6) Since the OMv conjecture can replace all other conjectures except SETH, these two conjectures together are sufficient to show that all hardness results for dynamic problems known so far hold even for arbitrary polynomial preprocessing time. (7) We also note that all our results hold for a very general type of amortized running time; e.g. they hold even when there is a large (polynomial) number of updates and, for graph problems, even when we start with an empty graph. No previous hardness results, except those obtained via SETH, hold for this case.

We believe that the *universality* and *simplicity* of the OMv conjecture will be important not only in proving tight hardness results for well-studied dynamic problems, but also in developing faster algorithms; for example, as mentioned earlier it can be used to show the limits of some specific approaches to attack the minimum Steiner tree and minimum cut problems [29, 15]. Below is a sample of our results. A list of all of them and detailed proofs can be found in later sections.

Subgraph Connectivity. In this problem, introduced by Frigioni and Italiano [16], we are given a graph G, and we have to maintain a subset S of nodes where the updates are adding and removing a node of G to and from S, which can be viewed as turning nodes on and off. The queries are to determine whether two nodes s and t are in the same connected component in the subgraph induced by S. The best upper bound in term of m is an algorithm with $\tilde{O}(m^{4/3})$ preprocessing time, $\tilde{O}(m^{2/3})$ amortized update time and $\tilde{O}(m^{1/3})$ amortized query time [10]. An upper bound in terms of n is an algorithm with $\tilde{O}(m)$ preprocessing time, $\tilde{O}(n)$ worst-case update time, and $\tilde{O}(1)$ worst-case query time.

For hardness in terms of m, Abboud and Vassilevska Williams [1] showed that the 3SUM conjecture can rule out algorithms with $m^{4/3-\varepsilon}$ preprocessing time, $m^{\alpha-\epsilon}$ amortized update time and $m^{2/3-\alpha-\epsilon}$ amortized query time, for any constants $1/6 \leq \alpha \leq 1/3$ and $0 < \epsilon < \alpha$. In this paper, we show that the OMv conjecture can rule out algorithms with *polynomial* preprocessing time, $m^{\alpha-\epsilon}$ amortized update time and $m^{1-\alpha-\epsilon}$ amortized query time, for any $0 \leq \alpha \leq 1$. This matches the upper bound of [10] when we set $\alpha = 2/3$.

For hardness in terms of n, Pătraşcu [33] showed that, assuming the hardness of his *multiphase problem*, there is no algorithm with $n^{\delta-\epsilon}$ worst-case update time and query time, for some constant $0 < \delta \leq 1$. By assuming the combinatorial BMM conjecture, Abboud and Vassilevska Williams [1] could rule out combinatorial algorithms with $n^{1-\varepsilon}$ amortized update time, and $n^{2-\epsilon}$ query time. These two bounds cannot rule out some improvement over [27], e.g. a non-combinatorial algorithms with $n^{1-\varepsilon}$ amortized update time, and $O(1)$ amortized query time. In this paper, we show that the OMv conjecture can rule out *any* algorithm with *polynomial* preprocessing time, $n^{1-\epsilon}$ amortized update time and $n^{2-\epsilon}$ amortized query time. Thus, there is no algorithm that can improve the upper bound of [27] without significantly increasing the query time.

Decremental Shortest Paths. In the decremental single-source shortest paths problem, we are given an unweighted

undirected graph G and a source node s. Performing an update means to delete an edge from the graph. A query will ask for the distance from s to some node v. The best exact algorithm for this problem is due to the classic result of Even and Shiloach [14] and requires $O(m)$ preprocessing time, $O(mn)$ total update time, and $O(1)$ query time. Very recently, Henzinger, Krinninger, and Nanongkai [21] showed a $(1+\epsilon)$-approximation algorithm with $O(m^{1+o(1)})$ preprocessing time, $O(m^{1+o(1)})$ total update time, and $O(1)$ query time. Roditty and Zwick [36] showed that the combinatorial BMM conjecture implies that there is no combinatorial *exact* algorithm with $mn^{1-\epsilon}$ preprocessing time and $mn^{1-\epsilon}$ total update time if we need $\tilde{O}(1)$ query time. This leaves the open problem whether we can develop a faster exact algorithm for this problem using algebraic techniques (e.g. by adapting Sankowski's techniques [37, 38, 39]). Our OMv conjecture implies that this is not possible: there is no exact algorithm with polynomial preprocessing and $mn^{1-\epsilon}$ total update time if we need $\tilde{O}(1)$ query time.

For the decremental all-pairs shortest paths problem on undirected graphs, $(1+\epsilon)$-approximation algorithms with $\tilde{O}(mn)$ total update time are also known in both unweighted and weighted cases [35, 6]. For combinatorial algorithms, this is tight even in the static setting under the combinatorial BMM conjecture [12]. Since fast matrix multiplication can be used to break this bound in the static setting when the graph is dense, the question whether we can do the same in the dynamic setting was raised by Bernstein [6]. In this paper, we show that this is impossible under the OMv conjecture. (Our hardness result holds for any algorithm with approximation ratio less than two.)

Pagh's Problem. In this problem, we want to maintain a family X of at most k sets $\{X_i\}_{1\leq i\leq k}$ over $[n]$. An update is by adding the set $X_i \cap X_j$ to X. We have to answer a query of the form "Does element j belong to set X_i?". A trivial solution to this problem requires $O(kn)$ preprocessing time, $O(n)$ worst-case update time and $O(1)$ worst-case query time. Previously, Abboud and Williams [1] showed that, assuming the combinatorial BMM conjecture, for any $n \leq k \leq n^2$ there is no combinatorial algorithm with $k^{3/2-\epsilon}$ preprocessing time, $k^{1/2-\epsilon}$ amortized update time, and $k^{1/2-\epsilon}$ amortized query time. They also obtained hardness for non-combinatorial algorithms but the bounds are weaker. Our OMv conjecture implies that for any $k = poly(n)$ there is no algorithm with $poly(k,n)$ preprocessing time, $n^{1-\varepsilon}$ update time, and $k^{1-\epsilon}$ query time, matching the trivial upper bound. Note that our hardness holds against *all* algorithms, including non-combinatorial algorithms. Also note that while the previous proof in [1] is rather complicated (it needs, e.g., a universal hash function), our proof is almost trivial.

Fully-Dynamic Weighted Diameter Approximation. In this problem, we are give a weighted undirected graph. An update operation adds or deletes a weighted edge. The query asks for the diameter of the graph. For the unweighted case, Abboud and Vassilevska Williams [1] showed that the Strong Exponential Time Hypothesis (SETH) rules out any $(4/3-\epsilon)$-approximation algorithm with polynomial preprocessing time, $n^{2-\epsilon}$ update and query time. Nothing was known for the weighted setting. In this paper, we show that for the weighted case, OMv rules out any $(2-\epsilon)$-approximation algorithm with polynomial preprocessing time, $n^{1/2-\epsilon}$ update time and $n^{1-\epsilon}$ query time. This result is among a few that require a rather non-trivial proof.

1.2 Discussions

OMv vs. Combinatorial BMM. The combinatorial BMM conjecture states that there is no truly subcubic *combinatorial* algorithm for multiplying two $n \times n$ Boolean matrices. There are two important points to discuss here. First, it can be easily observed that any reduction from the OMv problem can be turned into a reduction from the combinatorial BMM problem since, although we get two matrices at once in the BMM problem, we can always pretend that we see one column of the second matrix at a time (this is the OMv problem). This means that bounds obtained via the OMv conjecture will *never* be stronger than bounds obtained via the combinatorial BMM conjecture. However, the latter bounds will hold *only for combinatorial algorithms*, leaving the possibility of an improvement via an algebraic algorithm. This possibility cannot be overlooked since there are examples where an algebraic algorithm can break through the combinatorial hardness obtained by assuming the combinatorial BMM conjecture. For example, it was shown in [1] that the combinatorial BMM conjecture implies that there is no combinatorial algorithm with $n^{3-\epsilon}$ preprocessing time, $n^{2-\epsilon}$ update time, and $n^{2-\epsilon}$ query time for the fully-dynamic s-t reachability and bipartite perfect matching problems. However, we can break these bounds using Sankowski's algebraic algorithms [37, 40] which requires n^{ω} preprocessing time, $n^{1.495}$ worst-case update time, and $O(1)$ worst case query time, where ω is the exponent of the best known matrix multiplication algorithm (currently, $\omega < 2.3728639$ [18]).

Second, it can be argued that the combinatorial BMM conjecture actually implies the OMv conjecture, if the term "combinatorial algorithm" is interpreted in a certain way. Note that while this term has been used very often (e.g., [12, 49, 1, 34, 20]), it is not a well-defined term. Usually it is vaguely used to refer as an algorithm that is different from the "algebraic" approach originated by Strassen [44]; see, e.g., [9, 3, 4]. One formal way to interpret this term is by using the term "Strassen-like algorithm", as defined by Ballard et al. [2]. Roughly speaking, a Strassen-like algorithm divides *both* matrices into constant-size blocks and utilizes an algorithm for multiplying two blocks in order to recursively multiply matrices of arbitrary size (see [2, Section 5.1] for a detailed definition[3]). As pointed out in [2], this is the structure of all the fast matrix multiplication algorithms that were obtained since Strassen's, including the recent breakthroughs by Stothers [43] and Vassilevska-Williams [48]. Since OMv reveals one column of the second matrix at a time, it naturally disallows an algorithm to utilize block multiplications, and thus Strassen-like algorithms cannot be used to solve OMv.

We note that the OMv problem actually excludes even some combinatorial BMM algorithms; for example, the algorithm with $O(n^3(\log\log n)^2/\log^{2.25} n)$ running time of Bansal and Williams [3] cannot be used to solve OMv. Finally, even if one wants to interpret the term "combinatorial algorithm" differently and argue that the combinatorial BMM conjecture does not imply the OMv conjecture, we believe that

[3]Note that Ballard et al. also need to include a technical assumption in the Strassen-like algorithms that they consider in order to prove their results (see [2, Section 5.1.1]). This assumption is irrelevant to us.

breaking the OMv conjecture will still be a breakthrough since it will yield a fast matrix multiplication algorithm that is substantially different from those using Strassen's approach.

OMv vs. Multiphase. Pătraşcu [33] introduced a dynamic version of set disjointness called *multiphase problem*, which can be rephrased as a variation of the Matrix-Vector multiplication problem as follows (see [33] for the original definition). Let k, n, and τ be some parameters. First, we are given a $k \times n$ Boolean matrix M and have $O(nk \cdot \tau)$ time to preprocess M. Second, we are given an n-dimensional vector v and have $O(n \cdot \tau)$ additional computation time. Finally, we are given an integer $1 \leq i \leq n$ and must output $(Mv)[i]$ in $O(\tau)$ time. Pătraşcu conjectured that if there are constants $\gamma > 0$ and $\delta > 0$ such that $k = n^\gamma$, then any solution to the multiphase problem in the Word RAM model requires $\tau = n^\delta$, and used this conjecture to prove polynomial time hardness for several dynamic problems. How strong these hardness bounds are depends on how hard one believes the multiphase problem is. By a trivial reduction, the OMv conjecture implies that the multiphase conjecture holds with $\delta = 1$ when $\gamma = 1$. (We found it quite surprising that viewing the multiphase problem as a matrix problem, instead of a set problem as originally stated, can give an intuitive explanation for a possible value of δ.) This implies the strongest bound possible for the multiphase problem. Moreover, while hardness based on the multiphase problem can only hold for a *worst-case* time bound, it can be shown that under a general condition we can make them hold for an *amortized* time bound too if we instead assume the OMv conjecture (see **??** for details). Thus, with the OMv conjecture it seems that we do not need the multiphase conjecture anymore. Note that, as argued before, we can also conclude that the combinatorial BMM conjecture implies the multiphase conjecture. To the best of our knowledge this is the first connection between these conjectures.

OMv vs. 3SUM and SETH. As mentioned earlier, all previous hardness results that were based on the 3SUM conjecture can be strengthened through the OMv conjecture. However, we do not have a general mechanism that can always convert any hardness proof based on 3SUM into a proof based on OMv. Finding such a mechanism would be interesting.

Techniques for proving hardness for dynamic algorithms based on SETH were very recently introduced in [1]. Results from these techniques are the only ones that cannot be obtained through OMv. SETH together with OMv seems to be enough to prove all the hardness results known to date. It would be very interesting if the number of conjectures one has to start with can be reduced to one.

Remark On the Notion of Amortization. We emphasize that there are two different ways to define the notion of amortized update time. First, we can define it as an amortized update time when we *start from an empty graph*; equivalently, the update time has to be amortized over all edges that ever appear. The second way is to allow the algorithm to preprocess an arbitrary input graph and amortize the update time over all updates (not counting the edges in the initial graph as updates); for example, one can start from a graph with n^2 edges and have only n updates. The first definition is more common in the analysis of dynamic algorithms but it is harder to prove hardness results for it. Our hardness results hold for this type of amortization; in fact, they hold even when there is a large (polynomial) number of updates. Many previous hardness results hold only for the second type of amortization, e.g. results in [1] that are not based on SETH and 3SUM.

1.3 Notations

All matrices and vectors in this paper are Boolean and $\tilde{O}(\cdot)$ hides logarithmic factors in $O(\cdot)$. We will also use the following non-standard notation.

DEFINITION 1.2 ($\tilde{\tilde{o}}$ NOTATION). *For any parameters n_1, n_2, n_3, we say that a function $f(n_1, n_2, n_3) = \tilde{\tilde{o}}(n_1^{c_1} n_2^{c_2} n_3^{c_3})$ iff there exists some constant $\epsilon > 0$ such that $f = O(n_1^{c_1-\epsilon} n_2^{c_2} n_3^{c_3} + n_1^{c_1} n_2^{c_2-\epsilon} n_3^{c_3} + n_1^{c_1} n_2^{c_2} n_3^{c_3-\epsilon})$.*

1.4 Organization

Instead of starting from OMv, our reductions will start from an intermediate problem called OuMv. We describe this and prove necessary results in Section 2. In Section 3, we provide examples of our hardness proofs (more proofs can be found in the full version). In Section 4 we discuss some open problems.

2. INTERMEDIATE PROBLEMS

In this section we show that the OMv conjecture implies that OMv is hard even when there is a polynomial preprocessing time and different dimension parameters. Then we present the problem whose hardness can be proved assuming the OMv conjecture, namely the *online vector-matrix-vector multiplication* (OuMv) problem, which is the key starting points for our reductions.

We first define a more general version of the OMv problem: (1) we allow the algorithm to preprocess the matrix before the vectors arrive and (2) we allow the matrix to have arbitrary dimensions with a promise that the size of minimum dimension is not too "small" compared to the size of maximum dimension.

DEFINITION 2.1 (γ-OMv). *Let $\gamma > 0$ be a fixed constant. An algorithm for γ-OMv problem is given* parameters *n_1, n_2, n_3 as input with a promise that $n_1 = \lfloor n_2^\gamma \rfloor$. Next, it is given a matrix M of size $n_1 \times n_2$ that can be preprocessed. Let $p(n_1, n_2)$ denote the* preprocessing time. *After the preprocessing, an online sequence of vectors $v^1, \ldots, v^{n_3}$ is presented one after the other and the task is to compute each Mv^t before v^{t+1} arrives. Let $c(n_1, n_2, n_3)$ denote the* computation time *over the whole sequence.*

Note that the γ-OMv problem can be trivially solved with $O(n_1 n_2 n_3)$ total computing time and without preprocessing time. Obviously, the OMv conjecture implies that this running time is (almost) tight when $n_1 = n_2 = n_3 = n$. Interestingly, it also implies that this running time is tight for other values of n_1, n_2, and n_3:

THEOREM 2.2. *For any constant $\gamma > 0$, Conjecture 1.1 implies that there is no algorithm for γ-OMv with parameters n_1, n_2, n_3 using preprocessing time $p(n_1, n_2) = poly(n_1, n_2)$ and computation time $c(n_1, n_2, n_3) = \tilde{o}(n_1 n_2 n_3)$, and has error probability* $1/3$.

See the full version for the proof of Theorem 2.2. We now present the *online vector-matrix-vector multiplication problem* (OuMv). Although we base our results on the hardness

of OMv, the starting point of most of our reductions is the OuMv problem. In this problem, we multiply the matrix with two vectors, one from the left and one from the right:

DEFINITION 2.3 (γ-OuMv PROBLEM). *Let $\gamma > 0$ be a fixed constant. An algorithm for the γ-OuMv problem is given* parameters n_1, n_2, n_3 *as input with a promise that $n_1 = \lfloor n_2^\gamma \rfloor$. Next, it is given a matrix M of size $n_1 \times n_2$ that can be preprocessed. Let $p(n_1, n_2)$ denote the* preprocessing time. *After the preprocessing, an online sequence of vector pairs $(u^1, v^1), \ldots, (u^{n_3}, v^{n_3})$ is presented one after the other and the task is to compute each $(u^t)^\top M v^t$ before (u^{t+1}, v^{t+1}) arrives. Let $c(n_1, n_2, n_3)$ denote the* computation time *over the whole sequence. The γ-uMv problem with parameters n_1, n_2 is the special case of γ-OuMv where $n_3 = 1$.*

We also write OuMv and uMv to refer to, respectively, γ-OuMv and γ-uMv without the promise. Our reductions will exploit the fact that the result of this multiplication is either 0 or 1. This simplifies the reduction and improves the lower bound for the query time. Using a technique for finding "witnesses", which will be defined below, when the result of a vector-matrix-vector multiplication is 1, we can reduce the γ-OMv problem to the γ-OuMv problem and establish the following hardness for γ-OuMv.

THEOREM 2.4. *For any constant $\gamma > 0$, Conjecture 1.1 implies that there is no algorithm for γ-OuMv with parameters n_1, n_2, n_3 using preprocessing time $p(n_1, n_2) = poly(n_1, n_2)$ and computation time $c(n_1, n_2, n_3) = \tilde{o}(n_1 n_2 n_3)$, and has error probability* 1/3.

The proof of Theorem 2.4 can be found in the full version.

An γ-uMv algorithm with preprocessing time $p(n_1, n_2)$ and computation time $c(n_1, n_2)$ implies an γ-OuMv algorithm with preprocessing time $\tilde{O}(p(n_1, n_2))$, computation time $\tilde{O}(n_3 c(n_1, n_2))$ and the same error probability by a standard application of Chernoff bound. Therefore, we also get the following:

COROLLARY 2.5. *For any constant $\gamma > 0$, Conjecture 1.1 implies that there is no algorithm for γ-uMv with parameters n_1, n_2 using preprocessing time $p(n_1, n_2) = poly(n_1, n_2)$ and computation time $c(n_1, n_2) = \tilde{o}(n_1 n_2)$, and has error probability* 1/3.

3. EXEMPLARY HARDNESS RESULTS

In the following we provide some exemplary hardness results that can be obtained by reductions from the OuMv and uMv problems. Our proofs usually follow two simple steps. First, we show the reductions in lemmas that given a dynamic algorithm $\mathcal{A}$ for some problem, one can solve uMv by running the preprocessing step of $\mathcal{A}$ on some graph and then making some number of updates and queries. Then, we conclude in corollaries that if $\mathcal{A}$ has low update/query time then this contradicts Conjecture 1.1.

Given a matrix $M \in \{0,1\}^{n_1 \times n_2}$, we denote a bipartite graph $G_M = ((L, R), E)$ where $L = \{l_1, \ldots, l_{n_1}\}$, $R = \{r_1, \ldots, r_{n_2}\}$, and $E = \{(r_j, l_i) \mid M_{ij} = 1\}$.

Partially-Dynamic s-t Shortest Path (st-SP):

LEMMA 3.1. *Given an incremental (respectively decremental) dynamic algorithm $\mathcal{A}$ for st-SP algorithm, one can solve 1-OuMv with parameters n_1, n_2, n_3 by running the preprocessing step of $\mathcal{A}$ on a graph with $\Theta(\sqrt{m})$ vertices and $O(m)$ edges which is initially empty (respectively initially has $\Theta(m)$ edges), and then making $\sqrt{m}$ queries, where m is such that $n_1 = n_2 = n_3 = \sqrt{m}$.*

PROOF. Given an input matrix M of 1-OuMv, we construct a bipartite graph G_M, and also two paths P and Q with n_3 vertices each. Let $P = (p_1, p_2, \ldots, p_{n_3})$ where $p_1 = s$, and $Q = (q_1, q_2, \ldots, q_{n_3})$ where $q_1 = s'$. Add the edge (p_t, l_i) and (q_t, r_j) for all $i \leq n_1, j \leq n_2$ and $t \leq n_3$. There are $\Theta(n_1 + n_2 + n_3) = \Theta(\sqrt{m})$ vertices and $O(n_1 n_2 + n_3) = O(m)$ edges.

Once u^t and v^t arrive, we disconnect p_t from l_i iff $u_i^t = 0$, and disconnect q_t from r_j iff $v_j^t = 0$. We have that if $(u^t)^\top M v^t = 1$, then $d(s, s') = 2t + 1$, otherwise $d(s, s') \geq 2t + 2$. This is because, before u^{t+1} and v^{t+1} arrive, we disconnect p_t from l_i for all $i \leq n_1$ and disconnect q_t from r_j for all $j \leq n_2$. So for each t, we need 1 query, and hence $n_3 = \sqrt{m}$ queries in total. □

COROLLARY 3.2. *For any n and $m \leq O(n^2)$, Conjecture 1.1 implies that there is no partially dynamic st-SP algorithm $\mathcal{A}$ on a graph with n vertices and at most m edges with polynomial preprocessing time, total update time $\tilde{\tilde{o}}(m^{3/2})$, query time $\tilde{\tilde{o}}(m)$ per query and answer all queries correctly with probability* 2/3.

PROOF. Suppose there is such an algorithm $\mathcal{A}$. By Lemma 3.1, we construct an algorithm $\mathcal{B}$ for 1-OuMv with parameters $n_1 = \sqrt{m}, n_2 = \sqrt{m}, n_3 = \sqrt{m}$ by running $\mathcal{A}$ on a graph with $n_0 = \Theta(\sqrt{m})$ vertices and $m_0 = O(m)$ edges. Note that $m_0 = O(n_0^2)$. Since $\mathcal{A}$ uses polynomial preprocessing time, total update time $\tilde{\tilde{o}}(m^{3/2})$ and total query time $O(\sqrt{m} q(m)) = \tilde{\tilde{o}}(m^{3/2})$ $\mathcal{B}$ has polynomial preprocessing time and $\tilde{\tilde{o}}(m^{3/2})$ computation time contradicting Conjecture 1.1 by Theorem 2.4. □

Partially-Dynamic Single Source Shortest Path (ss-SP):

LEMMA 3.3. *Given an incremental (respectively decremental) dynamic algorithm $\mathcal{A}$ for ss-SP, one can solve $(\frac{\delta}{1-\delta})$-OuMv with parameters n_1, n_2, n_3 by running $\mathcal{A}$ on a graph with $\Theta(m^\delta + m^{1-\delta})$ vertices and $\Theta(m)$ edges which is initially empty (respectively initially has $\Theta(m)$ edges), and then making $m^{2(1-\delta)}$ queries, where m is such that $n_1 = m^{1-\delta}, n_2 = m^\delta$ and $n_3 = m^{1-\delta}$.*

PROOF. Given an input matrix M of $(\frac{\delta}{1-\delta})$-OuMv, we construct the bipartite graph G_M, and also a path $Q = (q_1, q_2, \ldots, q_{n_3})$ where q_1 is also denoted by s. Add the edge (q_t, r_j) for all $j \leq n_2, t \leq n_3$. There are $\Theta(n_1 + n_2 + n_3) = \Theta(m^\delta + m^{1-\delta})$ vertices and $\Theta(n_1 n_2 + n_2 n_3) = \Theta(m)$ edges.

Once u^t and v^t arrive, we disconnect q_t from r_j iff $v_j^t = 0$. We have that if $(u^t)^\top M v^t = 1$, then $d(s, l_i) = t + 1$ for some i where $u_i = 1$, otherwise $d(s, l_i) \geq t + 2$ for all i where $u_i = 1$. So n_1 queries are enough to decide this. Before u^{t+1} and v^{t+1} arrive, we disconnect q_t from r_j for all $j \leq n_2$. So for each t, we need n_1 queries, and hence $n_1 n_3 = m^{2(1-\delta)}$ queries in total. □

COROLLARY 3.4. *For any n, $m = \Theta(n^{1/(1-\delta)})$ and constant $\delta \in (0, 1/2]$, Conjecture 1.1 implies that there is no partially dynamic algorithm $\mathcal{A}$ for ss-SP for a graph with n*

vertices and at most m edges with preprocessing time $p(m) = poly(m)$, total update time $u(m) = \tilde{o}(mn)$, query time $q(m) = \tilde{o}(m^\delta)$ per query and answer all queries correctly with probability 2/3.

PROOF. Suppose there is such an algorithm $\mathcal{A}$. By Lemma 3.3, we construct an algorithm $\mathcal{B}$ for $(\frac{\delta}{1-\delta})$-OuMv with parameters $n_1 = m^{1-\delta}, n_2 = m^\delta$ and $n_3 = m^{1-\delta}$ by running $\mathcal{A}$ on a graph with $n_0 = \Theta(m^\delta + m^{1-\delta}) = \Theta(m^{1-\delta})$ vertices and $m_0 = \Theta(m)$ edges. Note that $m_0 = \Theta(n_0^{1/(1-\delta)})$. Since $\mathcal{A}$ uses polynomial preprocessing time, total update time $\tilde{\tilde{o}}(mn) = \tilde{\tilde{o}}(m^{2-\delta})$ and total query time $O(m^{2(1-\delta)}q(m)) = \tilde{\tilde{o}}(m^{2-\delta})$, $\mathcal{B}$ has polynomial preprocessing time and $\tilde{\tilde{o}}(m^{2-\delta})$ computation time contradicting Conjecture 1.1 by Theorem 2.4. □

Corollary 3.4 implies that the algorithm by [14], which has total update time $O(mn)$, is tight.

Single Source Subgraph Connectivity (ss-Subconn):

LEMMA 3.5. *Let $\delta \in (0,1)$ be any constant. Given a partially dynamic algorithm $\mathcal{A}$ for ss-SubConn, after polynomial preprocessing time, one can solve $(\frac{1-\delta}{\delta})$-uMv with parameters n_1, n_2 by running the preprocessing step of $\mathcal{A}$ on a graph with $\Theta(m^{1-\delta}+m^\delta)$ nodes and $O(m)$ edges, then making $O(m^{1-\delta})$ insertions (or $O(m^{1-\delta})$ deletions) and $O(m^\delta)$ queries, where m is such that $m^\delta = n_1$ (so $m^{1-\delta} = n_2$).*

PROOF. We show only the decremental case because the incremental case is symmetric. Given M, we construct the bipartite graph G_M, with an additional vertex s and edges (r_j, s) for all $j \le n_2$. Thus, the total number of edges is $n_1 n_2 + n_2 = O(m)$. In the beginning, every node is turned on. Once u and v arrive, we turn off r_j iff $v_j = 0$. If $u^\top M v = 1$, then s is connected to l_i for some i where $u_i = 1$. Otherwise, s is not connected to l_i for all i where $u_i = 1$. In total, we need to do $n_2 = m^{1-\delta}$ updates and $n_1 = m^\delta$ queries. □

COROLLARY 3.6. *For any n, $m \le O(\min\{n^{1/\delta}, n^{1/(1-\delta)}\})$, and constant $\delta \in (0,1)$, unless Conjecture 1.1 fails, there is no partially dynamic algorithm $\mathcal{A}$ for ss-SubConn for a graph with n vertices and at most m edges with preprocessing time $p(m) = poly(m)$, worst-case update time $u(m) = \tilde{\tilde{o}}(m^\delta)$ and query time $q(m) = \tilde{\tilde{o}}(m^{1-\delta})$ that can answer all queries correctly with probability at least 2/3. Moreover, this is true also for fully dynamic algorithm with amortized update time.*

PROOF. Suppose there is such a partially dynamic algorithm $\mathcal{A}$. That is, on a graph with n_0 vertices and $m_0 = O(\min\{n_0^{1/\delta}, n_0^{1/(1-\delta)}\})$ edges, $\mathcal{A}$ has worst-case update time $u(m_0) = \tilde{\tilde{o}}(m_0^\delta)$ and query time $q(m_0) = \tilde{\tilde{o}}(m_0^{1-\delta})$. We will construct an algorithm $\mathcal{B}$ for $(\frac{1-\delta}{\delta})$-uMv with parameter n_1, n_2 which contradicts Conjecture 1.1. Using Lemma 3.5, by running $\mathcal{A}$ on a graph with $n_0 = \Theta(m^\delta + m^{1-\delta})$ vertices and $m_0 = O(m)$ edges, where m is such that $m^\delta = n_1$ and so $m^{1-\delta} = n_2$ (note that, indeed, $m_0 = O(\min\{n_0^{1/\delta}, n_0^{1/(1-\delta)}\})$), $\mathcal{B}$ has preprocessing time $poly(m)$ and computation time $O(m^{1-\delta}u(m) + m^\delta q(m)) = \tilde{\tilde{o}}(m)$ which contradicts Conjecture 1.1 by Corollary 2.5.

Next, suppose that $\mathcal{A}$ is fully dynamic and guarantees only amortized bound. We will construct an algorithm $\mathcal{C}$ for 1-OuMv with parameter n_1, n_2, n_3 which again contradicts Conjecture 1.1 by running $\mathcal{A}$ on the same graph for solving 1-uMv while the number of updates and queries needed is multiplied by $O(n_3)$. This can be done because $\mathcal{A}$ is fully dynamic: for each vector pair (u, v) for $\mathcal{C}$, if $\mathcal{A}$ makes k updates to the graph, then $\mathcal{A}$ can undo these updates with another k updates so that the updated graph revert to the state right after preprocessing. Recall that, by the notion of amortization, if there are t updates, then $\mathcal{A}$ takes $O((t+m_0)\cdot u(m_0))$ time where m_0 is a number of edges ever appeared in the graph. By choosing $n_3 = m^\delta$, we have that $\mathcal{C}$ has preprocessing time $poly(m)$ and computation time $O((m^{1-\delta}n_3 + m)u(m) + m^\delta n_3 q(m)) = \tilde{\tilde{o}}(m^{1+\delta})$ which contradicts Conjecture 1.1 by Theorem 2.4. □

Corollary 3.6 implies tightness of two state-of-the-art algorithms by [13] and [10] which have worst-case update time $O(m^{4/5})$ and query time $O(m^{1/5})$, and amortized update time $O(m^{2/3})$ and query time $O(m^{1/3})$ respectively.

In the full version of this paper, we also show another hardness result with high query time:

COROLLARY 3.7. *For any n and $m \le O(n^2)$, unless Conjecture 1.1 fails, there is no partially dynamic algorithm $\mathcal{A}$ for ss-SubConn for a graph with n vertices and at most m edges with preprocessing time $p(m) = poly(m)$, worst update time $u(m) = \tilde{\tilde{o}}(\sqrt{m})$ and query time $q(m) = \tilde{\tilde{o}}(m)$ that can answer all queries correctly with probability at least 2/3. Moreover, this is true also for fully dynamic algorithm with amortized update time.*

Pagh's problem:

LEMMA 3.8. *Given an algorithm $\mathcal{A}$ for Pagh's problem, for any constant $\gamma > 0$, one can solve γ-uMv with parameters k, n by running the preprocessing step of $\mathcal{A}$ on k initial sets in universe $[n]$ using $poly(k,n)$ preprocessing time, k updates (adding k more sets into the family) and n queries.*

PROOF. Given a matrix M, let $\bar{M}$ be such that $\bar{M}_{i,j} = 1 - M_{i,j}$. Let M_i be the i-th row of M and M_i is treated as a subset of $[n]$ i.e. $j \in M_i$ iff $M_{i,j} = 1$. We similarly treat the i-th row $\bar{M}_i$ of $\bar{M}$ as a set. Note that $u^\top M e_j = 1$ iff $j \in \bigcup_{u_i=1} M_i$ iff $j \notin \bigcap_{u_i=1} \bar{M}_i$. Thus, at the beginning, we compute $\bar{M}_i$ for each $i \le k$. Once u and v arrive, we compute $\bigcap_{u_i=1} \bar{M}_i$ using k updates. There is a j where $v_j = 1$ such that $j \notin \bigcap_{u_i=1} \bar{M}_i$ iff $u^\top M v = 1$. We need n queries to check if such a j exists. □

COROLLARY 3.9. *For any constant $\gamma > 0$, Conjecture 1.1 implies that there is no dynamic algorithm $\mathcal{A}$ for Pagh's problem maintaining k sets over $[n]$ where $k = n^\gamma$, with $poly(k,n)$ preprocessing time, $u(k,n) = \tilde{\tilde{o}}(n)$ amortized update time and $q(k,n) = \tilde{\tilde{o}}(k)$ query time that can answer all queries correctly with probability at least 2/3.*

PROOF. Recall that, by the notion of amortization, if $\mathcal{A}$ initially maintains k sets, then the total update time of $\mathcal{A}$ is $O((t+k)\cdot u(t+k,n))$. By Lemma 3.8, we can solve uMv with parameters k, n by running $\mathcal{A}$ and making k updates and n queries in time $O(2k\cdot u(2k,n) + n\cdot q(2k,n)) = \tilde{\tilde{o}}(nk)$. This contradicts Conjecture 1.1 by Corollary 2.5. □

$(2-\epsilon)$ **Approximate Diameter on Weighted Graphs:** This reduction is inspired by a lower bound in distributed computation [17].

LEMMA 3.10. *For any $\gamma > 0$, Given a fully dynamic algorithm $\mathcal{A}$ for $(2-\epsilon)$-approximate diameter on $\{0,1\}$-weighted graph, one can solve γ-uMv with parameters n_1, n_2 by running the preprocessing step of $\mathcal{A}$ on a graph with $O(n_1\sqrt{n_2})$ vertices, and then making $n_2 + O(n_1\sqrt{n_2})$ updates and n_1 queries to $\mathcal{A}$.*

PROOF. First, let us define a graph H^v, called *vector graph*, from a vector v of size n_2. A vector graph $H^v = (B^v \cup C^v)$ has two halves, called *upper* and *lower halves* denoted by B^v and C^v respectively. B^v and C^v are both cliques of size $\sqrt{n_2}$. Let $\{b_x^v\}_{x\in[\sqrt{n_2}]}$ and $\{c_x^v\}_{x\in[\sqrt{n_2}]}$ be vertices of B^v and C^v respectively. There is an edge (b_x^v, c_y^v) iff $v_{x\sqrt{n_2}+y} = 0$. The weight of all edges in H^v is 1.

Given a matrix M of size $n_1 \times n_2$, let M_i be the i-th row of M. We construct a vector graph H^{M_i} for each $i \in [n_1]$, and another vector graph H^v where v is a zero vector. There are two special vertices a and z. Connect a to all vertices in H^v with weight one. Connect z to a and all vertices in H^{M_i} with weight zero, for every $i \in [n_1]$.

Once (u, v) arrives, we update H^v to be the vector graph of v in n_2 updates. Then we work in stages i, for each i where $u_i = 1$. Before going to the next stage, we undo all the updates. In stage i, 1) disconnect z from each vertex in H^{M_i}, 2) connect a to each vertex in H^{M_i} with weight one, and 3) add 0-weight *matching edges* $(b_x^{M_i}, b_x^v)$ and $(c_y^{M_i}, c_y^v)$ for all $x, y \le \sqrt{n_2}$. All three steps need $O(\sqrt{n_2})$ updates. Let $G(M_i, v)$ denote the resulting graph. We query the diameter of $G(M_i, v)$ and will use the result to solve γ-uMv. After finishing all stages, there are $n_2 + O(n_1\sqrt{n_2})$ updates and n_1 queries in total.

If there is some stage i where the diameter of the graph $G(M_i, v)$ is 2, then report $u^\top Mv = 1$. Otherwise report 0. The following claim justifies this answer.

CLAIM 3.11. *The diameter of $G(M_i, v)$ is 1 if $M_i^\top v = 0$. Otherwise, the diameter is 2.*

PROOF. First, every edge incident to z has weight 0. So we can treat all the adjacent vertices of z, which are exactly a and those in $H^{M_{i'}}$ where $i' \neq i$, as a single vertex. Therefore, we just have to analyze the distance among vertices in H^{M_i}, H^v and the vertex a.

Note that a is connected to all vertices in H^{M_i} and H^v by 1-weigth edges. The distance between vertices among the upper halves B^{M_i} and B^v is at most 1, because B^{M_i} and B^v are cliques and there are matching edges of weight zero. Similarly, for the lower halves C^{M_i} and C^v. Now, we are left with analyzing the distance between a vertex in the upper halves and another vertex in the lower ones. There are two cases.

If $M_i^\top v = 0$, then for each $x, y \le \sqrt{n_2}$, there is either an edge $(b_x^{M_i}, c_y^{M_i})$ or an edge (b_x^v, c_y^v). So given any $b_x^{M_i}$ and $c_y^{M_i}$, there is either a path $(b_x^{M_i}, b_x^v, c_y^v, c_y^{M_i})$ or a path $(b_x^{M_i}, c_y^{M_i})$ both of weight 1. Since $d(b_x^{M_i}, c_y^{M_i}) = 1$, the distance among $b_x^{M_i}, b_x^v, c_y^{M_i}, c_y^v$ is at most 1. Since this is true for any x, y, the diameter of the graph is 1.

If $M_i^\top v = 1$, then there is x, y where there is neither the edge $(b_x^{M_i}, c_y^{M_i})$ nor the edge (b_x^v, c_y^v). To show that $d(b_x^{M_i}, c_y^{M_i}) \ge 2$, it is enough to show that $d(b_x^{M_i}, c_y^{M_i}) > 1$ because every non-zero edge weight is 1. The set of vertices with distance 1 from $b_x^{M_i}$ includes exactly the neighbors of $b_x^{M_i}$ in H^{M_i} and their "matching" neighbors in H^v, the neighbors of b_x^v in H^v and their "matching" neighbors in H^{M_i}, and the vertex a, which combines z and all vertices in $H^{M_{i'}}$ where $i' \neq i$. But this set does not include $c_y^{M_i}$. Therefore $d(b_x^{M_i}, c_y^{M_i}) \ge 2$ and we are done. □

COROLLARY 3.12. *Assuming Conjecture 1.1, there is no fully dynamic algorithm for $(2-\epsilon)$-approximate diameter on $\{0,1\}$-weighted graphs with n vertices with preprocessing time $p(n) = poly(n)$, amortized update time $u(n) = \tilde{\delta}(\sqrt{n})$ and query time $q(n) = \tilde{\delta}(n)$ that can answer all queries correctly with probability at least 2/3.*

PROOF. Suppose such a fully dynamic algorithm $\mathcal{A}$ exists. By Lemma 3.10 and by "undoing" the operations as the proof of Corollary 3.6, we can solve 2-OuMv with parameters $(n_1, n_2, n_3) = (\sqrt{n}, n, n)$ by running $\mathcal{A}$ on a graph with $\Theta(n_1\sqrt{n_2}) = \Theta(n)$ vertices, and then making $O(n_2 + n_1\sqrt{n_2}) \times n_3 = O(n^2)$ updates and $n_1 n_3 = O(n\sqrt{n})$ queries in total. The computation time is $O(n^2 u(n) + n\sqrt{n} q(n)) = \tilde{\delta}(n^2\sqrt{n})$, contradicting Conjecture 1.1 by Theorem 2.4. Note that we choose $n_3 = n\sqrt{n}$ to make sure that the number of updates is at least the number of edges in the graph, so that we can use the amortized time bound. □

4. OPEN PROBLEMS

Of course it is very interesting to settle the OMv conjecture. Besides this, there are still many problems for which this work does not provide tight lower bounds, and it is interesting to prove such lower bounds based on the OMv or other reasonable conjectures.

Minimum Cut. Thorup and Karger [46] presented a $(2 + o(1))$-approximation algorithm with polylogarithmic amortized update time. Thorup [45] showed that in $\tilde{O}(\sqrt{n})$ worst-case update time the minimum cut can be maintained exactly when the minimum cut size is small and $(1+\epsilon)$-approximately otherwise. Improving this result using amortization is mentioned as a major open problem in [45]. Very recently Fakcharoenphol et al. [15] showed some related hardness results (e.g. for some subroutine used in Thorup's algorithm). However, currently there is no evidence that the minimum cut cannot be maintained in polylogarithmic update time (there is no hardness result even for the weighted case). In fact, it is not even known if polylogarithmic update time is possible or impossible for a key subroutine in Thorup's algorithm called *min-tree cut*, where we are given edge updates on a graph and its spanning tree and have to maintain the minimum cut among the cuts obtained by removing one edge from the spanning tree. We believe that understanding this subroutine is an important step in understanding the dynamic minimum cut problem.

Approximation Algorithms for Non-Distance Problems. In this paper we provide hardness results for several approximation algorithms for distance-related problems. However, we could not extend the techniques to non-distance graph problems such as approximating maximum matching, minimum cut, maximum flow, and maximum densest subgraph.

Total Update Time for Partially-Dynamic Algorithms. While there are many hardness results in the partially-dynamic setting in this and previous work, quite a few problems are still open. Of a particular interest are problems that are known to be easy when one type of updates is allowed but

become challenging when the other type of updates is allowed. For example, the single-source reachability problem can be solved in $O(m)$ total update time in the incremental setting [26] but the best time in the decremental setting is still larger than $mn^{0.9}$ (e.g. [22, 23]). This is also the case for the minimum cut problem where the incremental setting can be $(1+\epsilon)$-approximated in $\tilde{O}(m)$ total update time [24] while the current best decremental $(1+\epsilon)$-approximation algorithm requires $\tilde{O}(m+n\sqrt{n})$ total update time [45], and the topological ordering problem which is trivial in the decremental setting but challenging otherwise (e.g. [19, 5]). Since known hardness techniques – including those presented in this paper – usually work for both the incremental and the decremental setting, proving non-trivial hardness results for the above problems seems to be challenging.

Worst-Case Update Time. While there are fully-dynamic algorithms with polylogarithmic *amortized* update time for many problems, not much is known for *worst-case* update time. The only exception that we are aware of is the connectivity problem (due to the recent breakthrough of Kapron et al. [27]). Other basic graph problems, such as minimum spanning tree, 2-edge connectivity, biconnectivity, maximum matching, and densest subgraph are not known to have polylogarithmic worst-case update time. A polynomial hardness result for worst-case update time for these problems based on a natural assumption will be very interesting. The challenge in obtaining this is that such a result must hold only for the worst-case update time and not for the amortized one. Such results were published previously (e.g. those in [1, 33, 28]), but most of these results are now known to hold for amortized update time as well assuming OMv and SETH (some exceptions are those for partially-dynamic problems).

Deterministic Algorithms. Derandomizing the current best randomized algorithms is an important question for many problems, e.g., approximate decremental single-source and all-pairs shortest paths [6, 21] and worst-case connectivity and spanning tree [27]. This is important since deterministic algorithms do not have to limit the power of the adversary generating the sequence of updates and queries. Proving that derandomization is impossible for some problems will be very interesting. The challenge is that such hardness results must hold only for deterministic algorithm and not for randomized algorithms.

Trade-off between Query and Update Time In this paper we present hardness results with a trade-off between query and update time for several problems. Are these hardness results tight? This seems to be a non-trivial question since not much is known about the upper bounds for these problems. A problem for which it is particularly interesting to study this question is the subgraph connectivity problem, since it is the starting point of many reductions that lead to hardness results with a trade-off. In this paper, we show that for any $0 < \alpha < 1$ getting an $O(m^\alpha)$ update time requires a query time of $\Omega(m^{1-\alpha})$. This matches two known upper bounds in [13, 10] when $\alpha = 4/5$ and $\alpha = 2/3$. It is reasonable to conjecture that there is a matching upper bound for all $0 < \alpha < 1$; however, it is not clear if this is true or not.

Acknowledgement. D. Nanongkai would like to thank Parinya Chalermsoomsook, Bundit Laekhanukit, and Virginia Vassilevska Williams for comments. T. Saranurak thanks Markus Bläser and Gorav Jindal for discussions.

5. REFERENCES

[1] A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443, 2014.

[2] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Graph expansion and communication costs of fast matrix multiplication. *J. ACM*, 59(6):32:1–32:23, 2012.

[3] N. Bansal and R. Williams. Regularity lemmas and combinatorial algorithms. *Theory of Computing*, 8(1):69–94, 2012. Announced at FOCS'09.

[4] J. Basch, S. Khanna, and R. Motwani. On Diameter Verification and Boolean Matrix Multiplication. Technical report, Stanford University, 1995.

[5] M. A. Bender, J. T. Fineman, and S. Gilbert. A new approach to incremental topological ordering. In C. Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 1108–1115. SIAM, 2009.

[6] A. Bernstein. Maintaining shortest paths under deletions in weighted directed graphs: [extended abstract]. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13*, pages 725–734. ACM, 2013.

[7] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. *STOC*, 2015.

[8] M. Bläser. Lower bounds for online matrix-vector multiplication. *Manuscript*, 2014.

[9] G. E. Blelloch, V. Vassilevska, and R. Williams. A new combinatorial approach for sparse graph problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 108–120, 2008.

[10] T. M. Chan, M. Patrascu, and L. Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011. Announced at FOCS'08.

[11] S. Chechik. Improved distance oracles and spanners for vertex-labeled graphs. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 325–336, 2012.

[12] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000. Announced at FOCS'96.

[13] R. Duan. New data structures for subgraph connectivity. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 201–212, 2010.

[14] S. Even and Y. Shiloach. An On-Line Edge-Deletion Problem. *Journal of the ACM*, 28(1):1–4, 1981.

[15] J. Fakcharoenphol, T. Kumpijit, D. Nanongkai, T. Saranurak, and P. Sukprasert. Fully-dynamic

minimum cut on plane graphs in poylogarithmic time. *Manuscript*, 2014.

[16] D. Frigioni and G. F. Italiano. Dynamically switching vertices in planar graphs. *Algorithmica*, 28(1):76–103, 2000.

[17] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1150–1162, 2012.

[18] F. L. Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

[19] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Transactions on Algorithms*, 8(1):3, 2012.

[20] M. Henzinger, S. Krinninger, and D. Nanongkai. Dynamic approximate all-pairs shortest paths: Breaking the $o(mn)$ barrier and derandomization. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 538–547, 2013.

[21] M. Henzinger, S. Krinninger, and D. Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *FOCS*, pages 146–155, 2014.

[22] M. Henzinger, S. Krinninger, and D. Nanongkai. Sublinear-time decremental algorithms for single-source reachability and shortest paths on directed graphs. In *STOC*, pages 674–683, 2014.

[23] M. Henzinger, S. Krinninger, and D. Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In *Manuscript*, 2015.

[24] M. R. Henzinger. A static 2-approximation algorithm for vertex connectivity and incremental approximation algorithms for edge and vertex connectivity. *J. Algorithms*, 24(1):194–220, 1997.

[25] D. Hermelin, A. Levy, O. Weimann, and R. Yuster. Distance oracles for vertex-labeled graphs. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 490–501, 2011.

[26] G. F. Italiano. Amortized efficiency of a path retrieval data structure. *Theor. Comput. Sci.*, 48(3):273–281, 1986.

[27] B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1131–1142, 2013.

[28] T. Kopelowitz, S. Pettie, and E. Porat. 3sum hardness in (dynamic) data structures. *CoRR*, abs/1407.6756, 2014.

[29] J. Lacki, J. Ocwieja, M. Pilipczuk, P. Sankowski, and A. Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the Steiner tree. *STOC*, 2015.

[30] E. Liberty and S. W. Zucker. The mailman algorithm: A note on matrix-vector multiplication. *Inf. Process. Lett.*, 109(3):179–182, 2009.

[31] T. Motzkin. Evaluation of polynomials and evaluation of rational functions. *Bull. Amer. Math. Soc*, 61(163):10, 1955.

[32] V. Y. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[33] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 603–610, 2010.

[34] L. Roditty and R. Tov. Approximating the girth. *ACM Trans. Algorithms*, 9(2):15:1–15:13, Mar. 2013. Announced at SODA'11.

[35] L. Roditty and U. Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. In *FOCS*, pages 499–508. IEEE Computer Society, 2004.

[36] L. Roditty and U. Zwick. On Dynamic Shortest Paths Problems. *Algorithmica*, 61(2):389–401, 2011. Announced at ESA, 2004.

[37] P. Sankowski. Dynamic transitive closure via dynamic matrix inverse. In *45th Symposium on Foundations of Computer Science (FOCS 2004), Proceedings*, pages 509–517, 2004.

[38] P. Sankowski. Shortest paths in matrix multiplication time. In *Algorithms - ESA 2005, 13th Annual European Symposium, Proceedings*, pages 770–778, 2005.

[39] P. Sankowski. Subquadratic algorithm for dynamic shortest distances. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Proceedings*, pages 461–470, 2005.

[40] P. Sankowski. Faster dynamic matchings and vertex connectivity. In N. Bansal, K. Pruhs, and C. Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pages 118–126. SIAM, 2007.

[41] N. Santoro and J. Urrutia. An improved algorithm for Boolean matrix multiplication. *J. Computing*, 36(4):375–382, Dec. 1986.

[42] J. E. Savage. An algorithm for the computation of linear forms. *SIAM J. Comput.*, 3(2):150–158, 1974.

[43] A. J. Stothers. *On the complexity of matrix multiplication*. PhD thesis, 2010.

[44] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[45] M. Thorup. Fully-dynamic min-cut. *Combinatorica*, 27(1):91–127, 2007. Announced at STOC'01.

[46] M. Thorup and D. R. Karger. Dynamic graph algorithms with applications. In *Algorithm Theory-SWAT 2000*, pages 1–9. Springer, 2000.

[47] R. Williams. Matrix-vector multiplication in sub-quadratic time: (some preprocessing required). In *SODA*, pages 995–1001, 2007.

[48] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In H. J. Karloff and T. Pitassi, editors, *STOC*. ACM, 2012.

[49] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.