On the Complexity of Algorithms with Predictions for Dynamic Graph Problems

Monika Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Barna Saha
University of California San Diego, La Jolla, CA, USA

Martin P. Seybold
University of Vienna, Austria

Christopher Ye
University of California San Diego, La Jolla, CA, USA

Abstract

Algorithms with predictions is a new research direction that leverages machine learned predictions for algorithm design. So far a plethora of recent works have incorporated predictions to improve on worst-case bounds for online problems. In this paper, we initiate the study of complexity of dynamic data structures with predictions, including dynamic graph algorithms. Unlike online algorithms, the goal in dynamic data structures is to maintain the solution *efficiently* with every update.

We investigate three natural models of prediction: (1) δ-accurate predictions where each predicted request matches the true request with probability δ, (2) list-accurate predictions where a true request comes from a list of possible requests, and (3) bounded delay predictions where the true requests are a permutation of the predicted requests. We give general reductions among the prediction models, showing that bounded delay is the strongest prediction model, followed by list-accurate, and δ-accurate.

Further, we identify two broad problem classes based on lower bounds due to the Online Matrix Vector (OMv) conjecture. Specifically, we show that *locally correctable* dynamic problems have strong conditional lower bounds for list-accurate predictions that are equivalent to the non-prediction setting, unless list-accurate predictions are perfect. Moreover, we show that *locally reducible* dynamic problems have time complexity that degrades gracefully with the quality of bounded delay predictions. We categorize problems with known OMv lower bounds accordingly and give several upper bounds in the delay model that show that our lower bounds are almost tight.

We note that concurrent work by v.d.Brand et al. [SODA '24] and Liu and Srinivas [arXiv:2307.08890] independently study dynamic graph algorithms with predictions, but their work is mostly focused on showing upper bounds.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases Dynamic Graph Algorithms, Algorithms with Predictions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.62

Related Version *Full Version*: `https://arxiv.org/abs/2307.16771` [32]

Funding *Monika Henzinger*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019564) and the Austrian Science Fund (FWF) project Z 422-N, project I 5982-N, and project P 33775-N, with additional funding from the *netidee SCIENCE Stiftung*, 2020–2024.
Barna Saha: This project is partially supported by NSF grants 1652303, 1909046, 2112533, and HDR TRIPODS Phase II grant 2217058.

Acknowledgements We would like to thank Andrea Lincoln for many helpful discussions and insightful comments.

1 Introduction

Modern Machine Learning predictions models are surprisingly accurate in practice and exploiting their, seemingly ever improving, accuracy is a novel direction in theory. Algorithms with predictions provide a framework to study whether machine-learned advice can improve algorithmic performance. Ideally, an algorithm with predictions should improve significantly over a worst-case algorithm if the given predictions are perfect while never performing worse than a worst-case algorithm, regardless of prediction quality. The performance of the algorithm should improve with the quality of the predictions and the performance guarantees of the algorithm should hold without knowledge of prediction quality.

In online problems[1], various prediction models (e.g. [8, 27]) have been studied extensively. There are a wide range of problems where predictions allow to improve quality over worst-case optimal competitive ratios, such as counting sketches [34, 1, 21], bloom filters [38], caching/paging [50, 43, 10, 35, 4], ski rental [47, 9, 5, 51], correlation clustering [52], among many others.[2] The standard assumption is that the algorithm is given access to the predictions for the whole sequence of operations (or requests) *before* the algorithm has to produce its first output, i.e, during preprocessing. Dependent on the quality of the prediction, algorithms with prediction should provide a smooth transition between the online and offline problems.

In this paper, we initiate the study of complexity of dynamic data structures and algorithms with predictions. Unlike in online algorithms, the main goal in the dynamic setting is to maintain the solution *efficiently* with every update. To the best of our knowledge, investigating the potential of algorithms with prediction in the dynamic setting has only just started with our present work, and the independent, concurrent work of van den Brand, Forster, Nazari, and Polak [54] and Liu and Srinivas [42]. Borrowing terminology from online algorithms with prediction, we have the following three goals in designing dynamic algorithms with prediction. 1) The algorithm should be *consistent*, achieving the performance of an optimal offline algorithm when the prediction quality is high. 2) The algorithm should be *robust*, matching the performance of an (online) dynamic algorithm regardless of prediction quality. 3) The algorithm's performance should degrade gracefully between the two extremes as prediction quality deteriorates. Given the predictions, we can allow polynomial preprocessing time. When the actual updates arrive, the dynamic data structure must process them fast provided the available information from preprocessing.

As an example, consider the Online Matrix Vector (OMv) problem which has been instrumental in developing lower bounds for dynamic data structures [29]. In this problem, given a Boolean matrix M of size $n \times n$, and an online sequence of n vectors $\vec{v}_1, \ldots, \vec{v}_n$, the algorithm must compute $M\vec{v}_t$ before seeing $\vec{v}_{t+1}$. While the OMv conjecture states that the total time needed to process these n vectors cannot be sub-cubic, if these vectors are given apriori as part of predictions, then one could have preprocessed them in $O(n^\omega)$ time using fast matrix multiplication where $\omega < 2.373$, and output the results in $O(n)$ time per vector. This is often called the offline-online gap. Of course, it is unrealistic to assume that predictions are completely accurate. But this already showcases ample room for potential improvements in dynamic data structures due to algorithms with predictions.

For dynamic graph problems, a line of work initiated by [29] establishes conditional lower bounds on the time trade-offs between updates and queries for a *large number* of dynamic problems, based on the OMv conjecture (see e.g., [16, 26, 12, 30, 31]). However,

[1] We use *online* to denote problems where the input consists of a sequence of operations, which can modify the input or ask a query about the input, and *offline* to denote that all input is given at once.
[2] See, for example, `https://algorithms-with-predictions.github.io` or the survey [45].

these reductions typically consist of update sequences that present repetitive behaviour, e.g. repeatedly requesting certain updates, asking a query, and then reverting the updates again. Algorithms with predictions might have a tremendous potential for improving the running time bounds on such sequences. The central question that we investigate in this work is:

Can predictions lead to provably faster dynamic graph algorithms?

1.1 Contribution and Paper Outline

Warm Up: OMv with Predictions. Given the central importance of the OMv Problem, we start by investigating if algorithms with predictions can bypass the bounds of the OMv Conjecture (Section 2.2). The OMv problem consists of an $n \times n$ Boolean matrix M which can be preprocessed in polynomial time. Then a sequence of n Boolean vectors $v_1, v_2, \ldots$ arrives and the Boolean product $M\vec{v}_t$ needs to be output before $\vec{v}_{t+1}$ arrives. Suppose the algorithm is given prediction $(\hat{v}_1, \ldots, \hat{v}_n)$. A natural measure to quantify prediction errors could be the maximum ℓ_1 distance or Hamming distance between $\hat{v}_t$ and $\vec{v}_t$ for $t = 1, 2, \ldots, n$. We show a smooth transition in complexity across the offline-online gap for OMv that uses *predictions with bounded Hamming distance* (Theorem 7). Moreover, our lower bound (Theorem 8) shows that the algorithm is essentially optimal under the OMv conjecture.

Our study then turns to analyzing conditions that allow or prevent obtaining similar positive results for more general dynamic data structures and graph problems with appropriate prediction models.

Prediction Models for Dynamic Problems and General Lower Bounds. We propose and analyze three quality measures for predictions for dynamic problems. These are (1) *δ-accurate predictions*, where each predicted request matches the true request with probability at least δ, (2) *list-accurate predictions*, where the prediction consists of a list of possible requests for each time step, and (3) *bounded-delay predictions*, where the true requests are some (unknown) permutations of the predicted requests.

The *δ-accurate predictions* [27] and *bounded-delay predictions* [47, 50, 43, 7, 41] have already been studied in the online algorithms literature. The *list-accurate predictions* are similar to the multi-prediction model [3, 17, 6] where the best prediction from a list needs to be selected at every step. It can also be seen as a generalization of *δ-accurate predictions* by providing a list of possibilities, out of which one is the correct update. For algorithms with *δ-accurate predictions*, where $\delta \in (0, 1)$, we show that any lower bound on the time complexity from the non-prediction setting carries over, reduced by a factor of $1 - \delta$ (Proposition 10). Then we give general reductions between the prediction models, showing that lower bounds for bounded delay predictions imply lower bounds for list-accurate predictions (Lemma 17), which imply lower bounds for δ-accurate predictions (Lemma 12). This provides a natural hierarchy in the power of prediction models. Furthermore, we hope our "Alternating Parallel Simulation" technique (Lemma 17) is of interest for analyzing further prediction models.

Locally Correctable Problems: Hardness for List-Accurate Predictions. In Section 5, we introduce a class of *locally correctable* dynamic problems (defined informally below), and show that OuMv lower bounds for these problems continue to hold for any algorithm with list accurate predictions, unless the predictions are perfect (i.e. the list of each time step has size 1). The OuMv problem is a modification of the OMv problem where we are given two sequences of n vectors $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ and $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ along with the matrix M of $n \times n$

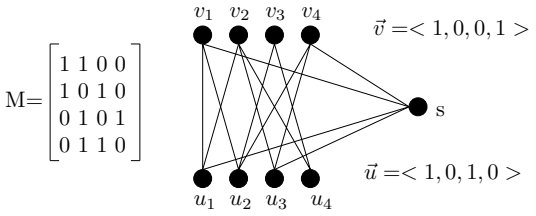

Figure 1 A small example of the OuMv lower bound construction for the #s-△ problem. The diagram shows that a 1 in the matrix or in vectors corresponds to an edge existing in the graph.

dimensions. The product $\vec{u}_t^\top M \vec{v}_t$ needs to be computed before seeing $\vec{u}_{t+1}, \vec{v}_{t+1}$. The OuMv conjecture excludes algorithms with total update and query time that is subcubic in n, and follows from the OMv conjecture (see Theorem 6).

An L-list accurate prediction (Definition 11) for a request sequence of length T consists of T lists of size $\leq L$, such that the t-th list must contain the true request at time t.

▶ **Theorem 1** (Informal, cf. Theorem 27)**.** *Suppose $\mathcal{P}$ is a locally correctable problem. Then, the conditional lower bound from OuMv holds for $\mathcal{P}$ even if the algorithm is given* 2*-list accurate predictions.*

We will illustrate our lower bounds using the simple #s-△ problem: counting the number of triangles containing a special vertex s. The #s-△ problem has a simple lower bound from the OuMv conjecture, as shown in Figure 1. Specifically, for a given $n \times n$ matrix M, we let G_M be a bipartite graph with vertex sets U, V each of size n. In G_M, connect (u_i, v_j) if and only if $M[i][j] = 1$. Our underlying graph is G_M with an additional special vertex s. Given a vector update $(\vec{u}_t, \vec{v}_t)$, connect (s, u_i) whenever $\vec{u}_t[i] = 1$ and (s, v_i) whenever $\vec{v}_t[i] = 1$. Then, #s-△ exactly computes $\vec{u}_t^\top M \vec{v}_t$.

To construct an adversarial request sequence for an algorithm with 2-list accurate predictions, we add an additional vertex so that the underlying graph is $G_M \cup \{s, t\}$. The 2-list accurate prediction is

$$B_t = (\{(s, u_i), (t, u_i)\})_{i=1}^n \circ (\{(s, v_i), (t, v_i)\})_{i=1}^n \circ \{q\}$$

for each vector $(\vec{u}_t, \vec{v}_t)$, where $\circ$ is the concatenation operation. Given the vector update, the adversary can ensure (s, u_i) (resp. (s, v_i)) is an edge if and only $\vec{u}_t[i] = 1$ (resp. $\vec{v}_t[i] = 1$). Whenever the graph already satisfies the condition, the adversary flips (t, u_i) (resp. (t, v_i)) so that on query q, #s-△ computes $\vec{u}_t^\top M \vec{v}_t$.

We now discuss the general lower bound for locally correctable problems. Our lower bound follows from a reduction from OuMv such that the set of request sequences admit efficiently computable 2-list accurate predictions. The basic idea is that for this class of problems a generic "universal list prediction sequence" can be efficiently created (without knowledge of the exact reduction sequence arising in the hardness reduction). In the #s-△ problem, this is the sequence $B_1 \circ \ldots \circ B_n$ defined above. Now any dynamic algorithm (without prediction) can efficiently construct this universal prediction in the preprocessing phase and then execute an algorithm with prediction using the universal prediction. Thus, no efficient dynamic algorithm with predictions can exist unless the OuMv conjecture is false.

Roughly speaking, a *locally correctable* problem $\mathcal{P}$ satisfies the following three properties: 1) any OuMv instance can be simulated by choosing some *subsequence* of a universal request sequence, containing updates needed for answering the query as well as updates that will not be useful for answering the queries, called "junk" updates, 2) any $\mathcal{P}$ instance can be

augmented efficiently into an instance containing both useful and useless updates, and 3) the answer to any query in the augmented instance can be *corrected* efficiently to answer the corresponding query in the original instance. For the #s-$\triangle$ problem, we satisfy 1) any vector update can be simulated by choosing a subsequence of the requests

$$B^* = ((s, u_i))_{i=1}^n \circ ((s, v_i))_{i=1}^n \circ q$$

2) we can augment the instance $G_M \cup \{s\}$ with the extra vertex t to obtain $G_M \cup \{s, t\}$, and 3) since (s, t) is never an edge, #s-$\triangle(G_M \cup \{s\}) =$ #s-$\triangle(G_M \cup \{s, t\})$ so that any query in the augmented instance in fact answers the query in the original instance.

We can then construct the following reduction from an OuMv instance. Typically, an OuMv based reduction encodes the round's query into the problem instance using some subsequence of possible updates to modify the data of $\mathcal{P}$ only where necessary. Instead, our reductions perform the specified update where necessary and otherwise insert a "junk" update. As a result, obtaining a 2-list accurate prediction is simple: The list contains the update itself and an arbitrary "junk" update. For the #s-$\triangle$ problem, the true update is (s, u_i) while the "junk" update is (t, u_i), exactly the 2-list accurate prediction given by B_t.

Locally Reducible Problems: Hardness for Bounded Delay Predictions. In Section 6, we introduce a class of *locally reducible* dynamic problems, where proving lower bounds against algorithms with bounded delay predictions is possible. As with list accurate predictions, our lower bounds rely on constructing an OuMv-based reduction such that a bounded delay prediction for the set of resulting sequences can be constructed efficiently.

Again, we will use #s-$\triangle$ as a concrete example to guide the ideas behind the general reduction. Informally, a prediction has bounded delay d (Definition 15) if the prediction $\hat{\rho} = \pi(\rho)$ where π is d-close to the identity permutation and ρ is the true request sequence.

Roughly speaking, a *locally reducible* problem satisfies two properties: 1) any OuMv instance can be simulated by choosing some subsequence of a universal request sequence and 2) each update, if repeated often enough, say *ord* times, leaves the dynamic data structure unchanged.[3] Inspired by algebra, we call such operations *cyclic*. For the #s-$\triangle$ problem, 1) any vector update is simulated by some subsequence of B^* as in the locally correctable case, and 2) flipping an edge twice leaves the graph unchanged.

As before, a typical OuMv reduction proceeds by choosing a subset of possible updates in the problem instance in order to encode a vector update. Our prediction then simply predicts that in each query vector, every possible update will be necessary. For #s-$\triangle$, this consists of one copy of sequence B^* for each update vector. Denote each set of updates required to encode one vector update a *block*. $\mathcal{P}$ is (u, q)-locally reducible if each block contains $u(n)$ updates and $q(n)$ queries. Note #s-$\triangle$ is $(2n, 1)$-locally reducible.

Since each block consists of some subset of the universal update set, an update will not occur *before* its predicted block, but it is possible an update occurs *after* its predicted block. To ensure that a request does not occur too long after its expected block (say a request has not occurred in *ord* blocks but it was contained in the universal request sequence already *ord* times), we perform the update *ord* times, using property 2) to show that the underlying data structure does not change. It follows that an update cannot occur more than *ord* blocks after its predicted block. Since this sequence has small delay relative to the predicted sequence, the universal request sequence is a prediction with small delay. Our lower bound then follows, as any dynamic algorithm can construct this prediction during the preprocessing phase.

[3] Alternatively, we could model this by giving each update operation a corresponding "undo" operation.

Table 1 List of algorithms with predictions with d bounded delay. Each algorithm has polynomial preprocessing time. A running time involving k states that the algorithm can handle predictions that are d delayed with k outliers. The lower bounds state that no algorithm with polynomial preprocessing time exists for any constant $\varepsilon > 0$ that attains the stated update and query times *simultaneously*, unless the OMv-conjecture fails.

Problem	Upper Bounds			Lower Bounds		
	Update	Query	Reference	Update	Query	Reference
$\#s$-$\triangle$	$d+k$	1	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
$\#s$-$\triangle$	1	$(d+k)^2$	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
Subgraph Connectivity	1	d^2	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
Transitive Closure	1	d^2	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
All Pairs Shortest Path	1	d^2	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
Erickson's Problem	$d+k$	1	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41
Erickson's Problem	1	$(d+k)^2$	[32]	$d^{1-\varepsilon}$	$d^{2-\varepsilon}$	Thm. 41

For the $\#s$-$\triangle$ problem, suppose an edge (s, u_i) has been flipped less than $j-1$ times by the j-th vector update. Then, the adversary flips edge (s, u_i) twice. In particular, the j-th occurrence of any edge flip must occur between the $(j-1)$-th and $(j+2)$-th queries. Since our constraints also ensure that there are roughly $2n$ edge flips between each query, this bounds the overall delay to $O(n)$. In particular, for the $\#s$-$\triangle$ problem, this implies that $O(n)$-delayed predictions cannot offer any improvement over a worst-case algorithm.

▶ **Theorem 2** (Informal, cf. Theorem 33 and Theorem 40)**.** *Suppose $\mathcal{P}$ is (u, q)-locally reducible from OuMv. Then, the conditional lower bound from OuMv holds for $\mathcal{P}$ even if the algorithm is given $O(u(n) + q(n))$-bounded delay predictions.*

Using similar techniques, we show that the lower bound degrades gracefully as the delay error of the prediction decreases (Theorem 41).

Examples of Locally Correctable and Locally Reducible Problems. We use our frameworks to provide lower bounds against algorithms with predictions for the following problems: Subgraph Connectivity [24, 19, 14, 36, 2, 29], Reachability [2, 29], Shortest Path [23, 18, 48, 49], Distance Spanners/Emulators [12], Maximum Matching [28, 11, 53, 16, 37], Maximum Flow [44, 29, 16], Triangle Detection [29], Densest Subgraph [29], d-Failure Connectivity [20, 37], Vertex Color Distance Oracle [33, 15, 39, 22], Weighted Diameter [25, 29], Strong Connectivity [2, 29], Electrical Flows [26], Erickson's Maximum Value Problem [46, 29], Langerman's Zero Prefix Sum Problem [46, 29]. All turn out to be Locally Reducible Problems. Additionally, all are Locally Correctable Problems as well, with exception of Erickson's Maximum Value Problem.

Dynamic Algorithms with Predictions. Finally, we give several algorithms in the fully dynamic edge update model with bounded delay predictions for the problems $\#s$-$\triangle$, Subgraph Connectivity, Transitive Closure, All Pairs Shortest Path, and Erickson's Maximum Value Problem. Some of them can even handle outliers, which are updates that were not at all in the predicted set (Table 1). These algorithms with predictions are conditionally optimal with respect to the prediction quality, matching our conditional lower bounds for d-delayed predictions in either the update or query time. Moreover, none of these algorithms need to know the prediction quality (the error parameter d).

To design our algorithms, we show that the difference between the state of the predicted data and the actual data scales with the delay error of the prediction. Furthermore, this difference can be maintained efficiently. In the preprocessing phase, we compute the predicted data structures, extracting useful intermediate values we require from the predicted data structure. Now, given the online request sequence up to some time step t, we show that by making small changes to the predicted data structure (on the order of the predictions delay) we can recover the result of the query on the actual data structure from precomputed values on the predicted data structure. Details are deferred to the full version [32].

Concurrent Work. Independent work of van den Brand, Forster, Nazari, and Polak [54] and Liu and Srinivas [42] jointly initiate the study of dynamic graph algorithms with predictions, focusing on upper bounds. [54] gives (among many other things) partially dynamic algorithms with bounded delay predictions for transitive closure and all pairs shortest path, and show that these algorithms are optimal with a lower bound giving the same result as Theorem 40. [42] considers the prediction model where a deletion time is predicted for every inserted edge, which is different from our proposed models. To the best of our knowledge, the above summarizes any overlapping contribution with [54] and [42].

2 Preliminaries and OMv with Predictions

We define dynamic data structures in general.

▶ **Definition 3** (Dynamic Data Structure). *Let $\mathcal{P}$ be a dynamic problem. For any instance x of $\mathcal{P}$, let $\mathcal{X}(x)$ be the set of possible* **updates** *on x and let $\mathcal{Q}(x)$ be the set of possible* **queries**. *When the instance is clear, we omit the subscripts and write $\mathcal{X}, \mathcal{Q}$. In the pre-processing step, the algorithm receives as input an initial data structure x_0. At each time step t, the algorithm receives a* **request** *$\rho_t \in \mathcal{X} \cup \mathcal{Q}$. Given a query $\rho_t \in \mathcal{Q}$, the algorithm must answer the query correctly on the current structure x_t, where x_t is obtained by applying request sequence $(\rho_1, \rho_2, \ldots, \rho_{t-1})$ to x_0. The query must be answered before ρ_{t+1} is revealed.*

If the updates $\mathcal{X}$ do not allow element deletions or do not allow element insertions, the data structure is called **partially dynamic**, *otherwise it is* **fully dynamic**.

For a given request sequence ρ, let $\rho_{[a,b]} = (\rho_a, \ldots, \rho_b)$ denote the sub-sequence between the a-th and b-th time step. Let $\rho_{\leq t} = \rho_{[1,t]}$ denote the prefix of the first t requests in ρ and let $\rho_{<t}$ denote $\rho_{\leq t-1}$.

2.1 The Online Matrix Vector Problem

Our hardness results are built on the OMv Conjecture of [29].

▶ **Definition 4** (OMv and OuMv [29, Def. 2.6]). *The OMv (resp. OuMv) problem with parameter n consists of a $n \times n$ Boolean matrix M that can be preprocessed in time $p(n)$. This is followed by n-rounds of processing online input vectors.*

The ***OMv*** *problem consists of n vectors $\vec{v}_1, \ldots, \vec{v}_n$. For all t, the algorithm must output $M\vec{v}_t$ before $\vec{v}_{t+1}$ arrives.*

The ***OuMv*** *problem consists of n vector pairs $(\vec{u}_1, \vec{v}_1), \ldots, (\vec{u}_n, \vec{v}_n)$. For all t, the algorithm must output $\vec{u}_t^\top M \vec{v}_t$ before $(\vec{u}_{t+1}, \vec{v}_{t+1})$ arrives.*

We call the processing of each individual input vector a *round*. Clearly, every *OMv round* can be solved in $O(n^2)$ time, yielding a trivial $O(n^3)$ algorithm. The OMv conjecture claims that this is basically optimal (cf. [29, Conj. 1.1]):

▶ **Conjecture 5.** *For any constant $\varepsilon > 0$, there is no algorithm with polynomial preprocessing time and $O(n^{3-\varepsilon})$ total update time that solves OMv with an error probability at most $1/3$.*

This conjecture leads to the following result for the OuMv problem (cf. [29, Thm. 2.7].). We simply call this the *OuMv conjecture* even though it is just a consequence of the OMv conjecture and *not* a different conjecture.

▶ **Theorem 6.** *For any constant $\varepsilon > 0$, the OMv conjecture implies that there is no algorithm with polynomial preprocessing time and total update time $O(n^{3-\varepsilon})$ that solves OuMv with an error probability at most $1/3$.*

2.2 Upper and Lower Bounds for OMv with Predictions

We begin by discussing how predictions affect the complexity of the OMv Problem, leading to bounds that have a *smooth transition* across the offline-online gap in terms of the prediction quality. We also give conditional lower bounds, showing that this result is optimal.

The extended Hamming distance $EH(s,t)$ of two bit-strings $s,t \in \{0,1\}^n$ is defined as follows. Let $\ell \geq 1$ be the largest index with $s_1 = s_2 = \ldots = s_\ell$ and $t_1 = t_2 = \ldots = t_\ell$, then

$$EH(s_1 \ldots s_n,\ t_1 \ldots t_n) := (s_1 + t_1 \mod 2) + EH(s_{\ell+1} \ldots s_n,\ t_{\ell+1} \ldots t_n)\ .$$

Since each block-difference is only counted once, the EH-distance is at most the Hamming distance, where the latter is equal to the L_1-distance on $\{0,1\}^n$. As a result, upper bounds that are sensitive to EH-distance are stronger than bounds sensitive to L_1.

▶ **Theorem 7** (OMv with Predictions)**.** *Let $M \in \{0,1\}^{n\times n}$ and $\Delta \in [0,n]$. Given predictions $(\hat{v}_1, \ldots, \hat{v}_n)$ that have $EH(\hat{v}_t, \vec{v}_t) \leq \Delta$ for all t, each arithmetic product $M\vec{v}_t$ can be computed in time $Q(n,\Delta) = O(n\Delta)$, after preprocessing of M and $(\hat{v}_1, \ldots \hat{v}_n)$ in $O(n^\omega)$ time.*

Proof. In the preprocessing phase, compute $(\hat{y}_1, \ldots, \hat{y}_n) = M(\hat{v}_1, \ldots, \hat{v}_n)$ using fast matrix multiplication. Compute a Prefix-Sum array D_i for each row i of M. That is, D_i when queried with range $[a,b]$ of column indices, returns the value $\sum_{k\in[a,b]} M_{i,k} = D_i(b) - D_i(a-1)$ in $O(1)$ time. Each D_i can be computed in $O(n)$ time, so preprocessing takes $O(n^\omega)$ time.

To compute the arithmetic result $\vec{y} = M\vec{v}$ for the online input $\vec{v} \in \{0,1\}^n$ of some OMv round, we determine in $O(n)$ time the blocks $b_1, \ldots, b_\Delta$ of index-ranges that contribute to the extended Hamming distance $EH(\hat{v}, \vec{v})$. Next, we initialize the result vector $\vec{y}$ with the precomputed vector $\hat{y}$. Then we iterate, over each row $j \in [n]$ and each block $b_i = [b_i^-, b_i^+]$, and add to the j-th entry in $\vec{y}$ the value $(\vec{v}_{b_i^+} - \hat{v}_{b_i^+}) \sum_{k\in b_i} M_{i,k}$, where $(\vec{v}_{b_i^+} - \hat{v}_{b_i^+}) \in \{-1, +1\}$ since $\vec{v}$ and $\hat{v}$ are bit-vectors. This requires time $O(n\Delta)$. ◀

Note that, after preprocessing, our algorithm with prediction requires time $O(n\Delta)$ per round. This upper bound is also nearly the best possible that can be achieved for algorithms with predictions that have extended Hamming distance at most Δ, unless the OMv-conjecture is false. Specifically, we show a conditional lower bound stating that no algorithm can achieve query time $O(n^{1-c}\Delta + n\Delta^{1-c})$ for any $c > 0$.[4]

▶ **Theorem 8.** *Let $\varepsilon \in (0,1)$ be a constant and $\Delta \in [1, n^\varepsilon]$. There is no algorithm with predictions of EH-distance at most Δ for the OMv problem with amortized time $Q(n,\Delta) = O(n^{2-c}\Delta + n^2\Delta^{1-c})$ for any $c > 0$, if the OMv-conjecture is true.*

[4] This is captured by $\tilde{\delta}(n\Delta)$ with Definition 19.

3 Dynamic Prediction Models with General Lower Bounds

The first question to ask is how to define algorithms with predictions in the dynamic setting. In this section, we propose several prediction models for dynamic problems and show that some definitions of predictions are so weak that almost the same lower bounds on time complexity can be shown as in the setting without prediction.

3.1 δ-Accurate Predictions

We begin with the simplest, very general formulation that lead to improved algorithms (see e.g. [27]). Informally, a predicted request sequence $\hat{\rho}$ is δ-accurate if each predicted request matches the respective online request with probability at least δ.

▶ **Definition 9** (δ-Accurate Predictions). *Let $\delta \in [0,1]$. Consider a dynamic problem with update set $\mathcal{X}$ and query set $\mathcal{Q}$. Let $\mathcal{D}$ be a distribution over sequences of T requests, i.e. over $(\mathcal{X} \cup \mathcal{Q})^T$, and $\hat{\rho} = (\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_T)$ be a sequence of T predicted requests.*

Then $\hat{\rho}$ is an **δ-accurate** *prediction for $\mathcal{D}$, if each $\hat{\rho}_t$ in $\hat{\rho}$ has $\Pr_{\rho \sim \mathcal{D}}[\rho_t = \hat{\rho}_t] \geq \delta$.*

This is a natural model of prediction as accuracy is one of the most common metrics to evaluate the performance of a machine learning model. However, we show that it is too weak of a notion to design efficient dynamic algorithms with prediction. The following proposition shows that for any constant $\delta < 1$, a dynamic problem that is hard in the online setting remains hard even if an δ-accurate prediction is available in advance. For example, even if a dynamic algorithm has a prediction that is correct for 99.9% of future requests, known lower bounds for the online problem still hold.

▶ **Proposition 10** (Request Amplification). *Let $\delta \in (0,1)$ be a constant and f a non-constant, non-decreasing function in n. Suppose there is a dynamic problem $\mathcal{P}$ with queries $\mathcal{Q} \neq \emptyset$ such that any algorithm processing T requests on instances of size n requires time $\Omega(Tf(n))$. Then there exists a distribution $\mathcal{D}$ of request sequences from $\mathcal{P}$ such that any algorithm with δ-accurate predictions for $\mathcal{D}$ requires time $\Omega(T(1-\delta)f(n))$.*

Thus, for any constant $\delta < 1$, the amortized time per request is at least $\Omega(f(n))$. This shows that δ-accurate predictions are not particularly powerful for dynamic problems with known lower bounds. This motivates the search for alternative stronger models of prediction under which it may be possible to harness the power of efficient offline algorithms.

One shortcoming of δ-accurate predictions are their generality. Regardless of the prediction, the support of the distribution $\mathcal{D}$ can be *every* possible request sequence of length T, if we assign small enough probability to request sequences that do not match well with the prediction. We thus investigate also more restrictive models that restrict the possible input sequences $\mathcal{S} \subset (\mathcal{X} \cup \mathcal{Q})^T$ for a given prediction.

3.2 List Accurate Predictions

Next we investigate a deterministic model for predictions that does not require a prediction to exactly specify the t-th request, but only to reveal "some information" about it. If each request is represented by a bit-string of $O(\log|\mathcal{X} \cup \mathcal{Q}|)$ bits, the following prediction model can be thought of as revealing a subset of these bits, i.e. the prediction $\hat{\rho}_t$ for step t is a set, of size at most L, of possible requests such that the t-th request $\rho_t \in \hat{\rho}_t$.

▶ **Definition 11** (L-List Accurate Predictions). *In a dynamic problem with update set $\mathcal{X}$ and query set $\mathcal{Q}$, let $\mathcal{S} \subseteq (\mathcal{X} \cup \mathcal{Q})^T$ be a set of sequences with T requests.*

A sequence of T sets $\hat{\rho} = (\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_T)$, where each $\hat{\rho}_t \subseteq \mathcal{X} \cup \mathcal{Q}$, is called an **$L$-list accurate** *prediction for $\mathcal{S}$, if each set $\hat{\rho}_t$ contains at most L elements and we have for each sequence $\rho \in \mathcal{S}$ and all $t \in [T]$ that $\rho_t \in \hat{\rho}_t$.*

Clearly, having an L-list accurate prediction with $L = 1$ is a perfect prediction. Also note that there is always a $|\mathcal{X} \cup \mathcal{Q}|$-list accurate prediction for all inputs, i.e. $\mathcal{S} = (\mathcal{X} \cup \mathcal{Q})^T$. In the OMv-problem for example, we have that the queries are from $\mathcal{Q} = \{0,1\}^n$ and having an L-list accurate prediction for preprocessing allows to solve each OMv round in $O(n)$ time, after spending $O(Ln^\omega)$ time for preprocessing.

▶ **Lemma 12.** *Given an L-list accurate prediction $\hat{\rho} = (\hat{\rho}_1, \ldots, \hat{\rho}_T)$ for $\mathcal{S}$, one can compute in $O(\sum_t |\hat{\rho}_t|) = O(LT)$ time an $\frac{1}{L}$-accurate prediction $\hat{\rho}'$ for the uniform distribution on $\mathcal{S}$.*

Thus, lower bounds against algorithms with L-list accurate predictions also yield lower bounds against algorithms with δ-accurate predictions for a problem whenever $\delta \leq \frac{1}{L}$.

Though L-list predictions are stronger than δ-accurate predictions, we show that for a wide range of *locally correctable* problems, strong lower bounds (similar to Proposition 10) hold: Any algorithm using L-list accurate predictions is subject to the same conditional lower bounds as a prediction-less online algorithm, unless the list accurate predictions are perfect, i.e. $L = 1$.

We thus seek to investigate even more powerful prediction models in the following.

3.3 Bounded Delay Predictions

Unlike δ-accurate and L-list accurate predictions that aimed at predicting every individual time step $t \in [T]$, we further investigate predictions that know all T requests in advance, though the actual order of a request sequence may have various "small deviations" from the predicted sequence of requests. That is, we will measure prediction accuracy by a notion of closeness for permutations. We consider a permutation $\pi \in \mathrm{Perm}(T)$ as a bijective map $\pi : [T] \to [T]$ on the integers $[T]$, e.g. $\pi(1)$ is the first element of the permutation.

▶ **Definition 13.** *Let $d \geq 0$ and $\pi, \sigma \in \mathrm{Perm}(T)$. We call π and σ d-***close***, denoted $|\pi - \sigma|_\infty \leq d$, if $\left|\pi^{-1}(t) - \sigma^{-1}(t)\right| \leq d$ for all $t \in [T]$.*

To simplify exposition, we overload the notation of a permutation $\pi \in \mathrm{Perm}(T)$ to yield a reordering of a request sequence of length T.

▶ **Definition 14.** *For a request sequence $\rho \in (\mathcal{X} \cup \mathcal{Q})^T$ and $\pi \in \mathrm{Perm}(T)$, let $\pi(\rho) = \left(\rho_{\pi(1)}, \ldots, \rho_{\pi(T)}\right)$ be the request sequence obtained by reordering ρ according to π.*

Next we formalize what it means for predicted request sequence $\hat{\rho} \in (\mathcal{X} \cup \mathcal{Q})^T$ to be a bounded delay prediction for a set of input request sequences.

▶ **Definition 15** (Bounded Delay Predictions). *Let $\mathcal{S} \subseteq (\mathcal{X} \cup \mathcal{Q})^T$ be a set of request sequences of length T and $\hat{\rho} = (\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_T)$ a given sequence of T predicted requests.*

*Then $\hat{\rho}$ is d-***delayed** *for $\mathcal{S}$, if for all $\rho \in \mathcal{S}$, there exists some permutation π with $\pi(\rho) = \hat{\rho}$, and π is d-close $|\pi - \mathsf{id}|_\infty \leq d$ to the identity permutation id.*

Next, we show that bounded delay predictions are a stronger notion than list-accurate predictions, which were a stronger notion than δ-accurate predictions (see Lemma 12).

▶ **Lemma 16.** *Given an integer $d \geq 0$ and a request sequence $\hat{\rho} \in (\mathcal{X} \cup \mathcal{Q})^T$, one can compute in $\Theta((d+1)T)$ time a $(2d+1)$-list accurate prediction $\hat{\rho}' = (\hat{\rho}'_1, \ldots, \hat{\rho}'_T)$ for all request sequence sets $\mathcal{S} \subseteq (\mathcal{X} \cup \mathcal{Q})^T$ for which $\hat{\rho}$ is d-delayed.*

Though the lemma requires that integer $d \geq 0$ is given as input, as opposed to the reduction in Lemma 12, we can still reduce, from bounded delay to L-list accurate algorithms, using an "Alternating Parallel Simulation" that searches for a 2-approximation of a minimum d value for a request sequence ρ.

▶ **Lemma 17** (Alternating Parallel Simulation)**.** *Let $\rho, \hat{\rho} \in (\mathcal{X} \cup \mathcal{Q})^T$ and $d^* \in [0, T]$ be minimal such that prediction $\hat{\rho}$ is d^*-delayed for the request sequence ρ. Suppose there is an algorithm $\mathcal{A}'$ that given an L-list accurate prediction for request sequences of length T solves $\mathcal{P}$ in time $O(Tf'(n, L))$ after at most $P'(n, L)$ preprocessing time. If $LT + P'(n, L) = O(Tf'(n, L))$, then there is an algorithm $\mathcal{A}$ that solves ρ, given prediction $\hat{\rho}$, in $O(f'(n, 4d^* + 1) \log T)$ amortized time, after $O(T)$ time for preprocessing of $\hat{\rho}$.*

Note that the $LT + P'(n, L) = O(Tf'(n, L))$ condition on $\mathcal{A}'$ is very mild, i.e. its preprocessing of a list prediction of size $O(LT)$ takes not more time than solving a request sequence for which the prediction is L-list accurate.

Note that the "alternating parallel simulation" technique to show the reduction in the previous lemma is quite general, though we only use it to reduce from algorithms with bounded delay to algorithms with list-prediction (i.e. taking Lemma 16). The reduction in the previous lemma immediately yields the following general lower bounds.

▶ **Corollary 18.** *Suppose there is a dynamic problem $\mathcal{P}$ such that any algorithm with d-delayed prediction requires time $\Omega(Tf(n, d))$ to processes T requests on instances of size n . Then any algorithm with L-list predictions for $\mathcal{P}$ requires time $\Omega\left(\frac{T}{\log T} f(n, (d-1)/4)\right)$.*

Clearly, there are sets of request sequences that do not admit delay predictions with small d. In Section 6, we will however show that for many problems with OMv-based lower bounds, it is possible to construct sets of request sequences $\mathcal{S}$ admitting bounded predictions while being simultaneously powerful enough to express an arbitrary OMv instance. Concretely, for the class of locally reducible dynamic problems (Definitions 30 and 32) we will show lower bounds for bounded delay predictions (even with no outliers) in Section 6.1.

4 Extensions of the OMv Conjecture

Before discussing our lower bounds against general dynamic problems, we revisit generalizations and extensions of the OMv conjecture. The OMv and OuMv conjectures generalize to non-square dimensions (i.e. Definition 2.1 and 2.6 in [29]). To state this, we need to introduce the $\tilde{\tilde{o}}$-notation for multivariate functions (cf. [29, Definition 1.2]).

▶ **Definition 19** (polynomially lower $\tilde{\tilde{o}}$-notation)**.** *For $f : \mathbb{N}^3 \to \mathbb{N}$ and any constants $c_1, c_2, c_3 \geq 0$, we write $f(n_1, n_2, n_3) = \tilde{\tilde{o}}(n_1^{c_1} n_2^{c_2} n_3^{c_3})$ if and only if there exist constants $\varepsilon, N, C > 0$ such that $f(n_1, n_2, n_3) \leq C(n_1^{c_1-\varepsilon} n_2^{c_2} n_3^{c_3} + n_1^{c_1} n_2^{c_2-\varepsilon} n_3^{c_3} + n_1^{c_1} n_2^{c_2} n_3^{c_3-\varepsilon})$ for all $n_1, n_2, n_3 > N$.*

We use the analogous definition for functions with one or two parameters.

Recall that the standard $\tilde{O}$ and $\tilde{o}$-notation suppresses polylogarithmic factors.

▶ **Definition 20** (Rectangular γ-OMv and γ-OuMv)**.** *Let $\gamma > 0$ be a fixed constant. The γ-OMv (resp. γ-OuMv) problem with parameters n_2, n_3 consists of $n_1 \times n_2$ matrix M given during preprocessing, where $n_1 := \lfloor n_2^\gamma \rfloor$. This is followed by n_3 online vector updates.*

*The γ-**OMv** problem consists of n_3 vectors $\vec{v}_1, \ldots, \vec{v}_{n_3}$, and the algorithm must output $M\vec{v}_t$ before $\vec{v}_{t+1}$ arrives.*

*The γ-**OMv** problem consists of n_3 vector pairs $(\vec{u}_1, \vec{v}_1), \ldots, (\vec{u}_{n_3}, \vec{v}_{n_3})$, and the algorithm must output $M\vec{v}_t$ before $\vec{v}_{t+1}$ arrives. The γ-**uMv** problem is the special case of γ-OuMv with $n_3 = 1$.*

Clearly, the OMv and OuMv problems are the special cases of γ-OMv and γ-OuMv with $\gamma = 1$ and $n_1 = n_2 = n_3 = n$. The OMv conjecture implies an analogous lower bound for the γ-OuMv problem (cf. Theorem 2.2 and 2.7 in [29]).

▶ **Theorem 21** (Hardness of γ-OMv and γ-OuMv). *For any constant $\gamma > 0$, the OMv conjecture implies that there is no algorithm for γ-OMv with parameters n_2, n_3 that has preprocessing time $P(n_2) = poly(n_2)$, total running time for all requests of $\tilde{\delta}(n_1 n_2 n_3) = \tilde{\delta}(n_2^{\gamma+1} n_3)$, where $n_1 = \lfloor n_2^{\gamma} \rfloor$, and error probability at most $1/3$.*

For any constant γ, the OuMv conjecture implies that there is no algorithm for γ-OuMv with parameters n_2, n_3 that has preprocessing time $P(n_2) = poly(n_2)$, total running time for all requests of $\tilde{\delta}(n_1 n_2 n_3) = \tilde{\delta}(n_2^{\gamma+1} n_3)$, and error probability at most $1/3$.

It is possible to solve the OMv problem faster than $\Theta(n^3)$. Green Larsen and Williams [40] gave a non-combinatorial OMv algorithm that runs in $O(n^3/2^{\Omega(\sqrt{\log n})})$ time. Williams [55] gave a combinatorial algorithm that, after $O(n^{2+\varepsilon})$ preprocessing, solves any OMv round in $O(n^2/\log^2 n)$ time. Chakraborty, Kamma and Larsen [13] settled the cell probe complexity, showing that any data structure storing $r \in (n, n^2)$ bits must have a query time t, i.e. the number of reads from memory cells, with $r \cdot t = \Omega(n^3)$ and that this lower bound is tight, by giving an algorithm with $r = t = \tilde{O}(n^{3/2})$ cell probes.

4.1 Sparse OMv Conjecture

We show in this section that the difficulty of the OMv and OuMv problem degrades gracefully with increased sparsity of query vectors. For any integer n, let $[n] = \{1, 2, \ldots, n\}$.

The ***support*** of a vector $\vec{v}$ is the set of non-zero indices, i.e. $\text{supp}(\vec{v}) = \{i \text{ s.t. } \vec{v}[i] \neq 0\}$, and the ***restriction*** $\vec{v}\,|_S$ of $\vec{v}$ to an index-subset S is the vector $(\vec{v}\,|_S)[k] = \begin{cases} \vec{v}[k] & k \in S \\ 0 & k \notin S \end{cases}$.

Next we define the problem variants that have sparse input vectors, with respect to fixed sets of indices.

▶ **Definition 22** (Sparse S-γ-OMv and S-γ-OuMv). *The S-γ-OMv problem differs from the γ-OMv problem only by having an additional input $S_2 \subseteq [n_2]$ of size $|S_2| \leq n_2^t$ for some $t \in (0, 1]$, which is given during the preprocessing phase. In the online phase, each of the n_3 query vectors $\vec{v}_i$ must fulfill support $\text{supp}(\vec{v}_i) \subseteq S_2$.*

The S-γ-OuMv problem differs from the γ-OuMv problem only by having an additional input $S_1 \subseteq [n_1]$ and $S_2 \subseteq [n_2]$ of size $|S_1| \leq n_1^t$ and $|S_2| \leq n_2^t$ for some $t \in (0, 1]$, which are given during the preprocessing phase. In the online phase, each of the n_3 pairs of query vectors $(\vec{u}_i, \vec{v}_i)$ has support $\text{supp}(\vec{u}_i) \subseteq S_1$ and $\text{supp}(\vec{v}_i) \subseteq S_2$.

Clearly, each S-γ-OMv query round can be answered in $O(n_2^{\gamma+t})$ time and each S-γ-OuMv query round can be answered in $O(n_2^{\gamma+t+t})$ time. Under OMv, this is essentially optimal.

▶ **Conjecture 23** (S-γ-OMv and S-γ-OuMv). *Let n_2, n_3, t, γ be parameters for the S-γ-OMv and S-γ-OuMv problem.*

There is no $\tilde{\delta}(n_2^{\gamma+t}n_3)$-time algorithm that solves the S-γ-OMv problem with an error probability of at most $1/3$ after preprocessing in time polynomial in n_2.

There is no $\tilde{\delta}(n_2^{\gamma+t}n_3)$-time algorithm that solves the S-γ-OuMv problem with an error probability of at most $1/3$ after preprocessing in time polynomial in n_2.

For the S-OMv and S-OuMv problems setting $n_1 = n_2 = n_3 = n$, the above conjecture is equivalent to saying there is no $\tilde{\delta}(n^{2+t})$ algorithm for the S-OMv problem and no $\tilde{\delta}(n^{1+2t})$ algorithm for the S-OuMv problem. We observe that Conjecture 23 is a consequence of Conjecture 5.

▶ **Theorem 24.** *Conjecture 23 follows from Conjecture 5.*

5 Locally Correctable Problems: Lower Bounds for List-accurate Predictions

We show in this section that certain problems, which we formally define in Definition 26, allow for remarkably strong lower bounds, in contrast to our general reductions in Section 3.

5.1 Preliminaries: Edge Updates in Dynamic Graphs

We will primarily focus on dynamic graphs with edge updates.

▶ **Definition 25** (Edge-Updates and Queries in Dynamic Graphs)**.** *Let $\mathcal{P}$ be a dynamic graph problem. Let V be a set of n vertices. Let $\mathcal{X}(V) = \{(u, v) \in V \times V \text{ s.t. } u \neq v\}$ denote the set of possible edge flip updates. In an undirected graph, $\mathcal{X}$ contains all unordered pairs of vertices, while in a directed graph $\mathcal{X}$ contains all ordered pairs. $\mathcal{Q}(V)$ denotes the set of queries that are possible for $\mathcal{P}$. When the underlying graph is clear, we omit V and write $\mathcal{X}, \mathcal{Q}$. In the pre-processing step, the algorithm receives as input an initial graph G_0 on vertices V. At each time step t, the algorithm receives some request $\rho_t \in \mathcal{X} \cup \mathcal{Q}$. When given a query $\rho_t \in \mathcal{Q}$, the algorithm must answer the query correctly on the current graph G_t, obtained by applying request sequence $(\rho_1, \rho_2, \ldots, \rho_{t-1})$ to the initial graph G_0. The query must be answered before the following request ρ_{t+1} is revealed.*

In the Maximum Matching problem for example, the query set $\mathcal{Q}$ consists of a single element q: *"What is the size of a maximum matching in the current graph?"*. Our updates are *edge flips*: An edge flip inserts e if it is currently not in the graph and removes it otherwise. In the edge update model of dynamic graphs, a sequence of T requests (updates or queries) arrive online. In our prediction models, we assume that the algorithm is given in preprocessing, some form of prediction for the T requests in the online phase.

For a dynamic graph G on n vertices and an update sequence $\rho = (\rho_1, \rho_2, \ldots, \rho_T)$, we denote with $G_0 = (V, E_0)$ the initial graph and with $G_t(\rho) = (V, E_t(\rho))$ the graph after applying the t-th request of ρ, i.e. $E_t(\rho)$ is the edge set after applying all updates in the first t requests to E_0. When the request sequence is clear, we omit ρ and write $G_t = (V, E_t)$.

5.2 Locally Correctable Dynamic Problems

Next, we formalize our approach of OuMv lower bounds for algorithms with list accurate predictions in our definition of *locally correctable* problems.

In the toy problem for $\#s$-$\triangle$, we observed that every pair of query vectors can be simulated by taking a *subsequence* of a universal update request sequence $(s, u_1), \ldots, (s, u_n), (s, v_1), \ldots, (s, v_n)$, followed by one query, where we only flip the edges

necessary to ensure the edges (s, u_i) (resp. (s, v_i)) correctly encode $\vec{u}$ (resp. $\vec{v}$). However, in the general reduction (Figure 1), the actual subsequence depends heavily on the specific instance. By *augmenting* the graph with one dummy vertex t that remains non-adjacent to s, we restricted the number of edge flips needed at any one time step, without affecting the query result, to a list of *two* update requests. That is flipping (t, u_i) or (s, u_i) for the bits in $\vec{u}$ and flipping (t, v_i) or (s, v_i) for the bits in $\vec{v}$. In our $\#s$-$\triangle$ example, the query computation in the augmented instance immediately yields the correct answer for the non-augmented instance, without further computations needed for *correcting* the query results.

Next, we formally define locally correctable problems and then prove the generalization of the technique in our reduction below.

▶ **Definition 26** (Locally Correctable Dynamic Problem)**.** *Let $\mathcal{P}$ be a dynamic problem.*

Suppose there is no algorithm for $\mathcal{P}$ with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(n_1,n_2)U(n) + q(n_1,n_2)Q(n)\Big) = \tilde{\delta}(n_1n_2n_3)$ if the OuMv conjecture is true, where n_1, n_2, n_3 are integers with $n_1 = \lfloor n_2^\gamma \rfloor$ for some constant $\gamma > 0$, functions $u, q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and $n = n(n_1, n_2)$ is the size of the $\mathcal{P}$ instance in the reduction.

Then, $\mathcal{P}$ is **locally correctable** *if there exists a universal sequence $\bar{\rho} = \bar{\rho}(n_1, n_2)$ of requests from $\mathcal{X} \cup \mathcal{Q}$, an augmentation function f for pre-processing, and a correction function g, satisfying:*

1. *The sequence $\bar{\rho}$ can be partitioned into n_3 subsequences $\bar{\rho} = B_1 \circ \ldots \circ B_{n_3}$, where block B_k contains $u(n_1, n_2)$ updates and $q(n_1, n_2)$ queries.*
2. *For any γ-OuMv instance with $n_1 \times n_2$ matrix M and query vector pairs $\{(\vec{u}_k, \vec{v}_k)\}_{k=1}^{n_3}$ the reduction constructs initial data D_0 of problem $\mathcal{P}$ and a request sequence ρ satisfying:*
   a. *ρ is the concatenation of request sequences $B'_1 \circ B'_2 \circ \ldots \circ B'_{n_3}$, where each B'_k is a subsequence of B_k.*
   b. *For each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$, the result bit $\vec{u}_k^\top M \vec{v}_k$ can be computed in $O(q(n_1, n_2))$ time based on the answers given to the queries in $B'_1 \circ B'_2 \circ \ldots \circ B'_k$.*
3. **Augmentability:** *$f(D_0)$ is an instance of size $O(n)$ and $\mathcal{X}(D_0) \subset \mathcal{X}(f(D_0))$.*
4. **Correctability:** *There is a non-empty subset $\mathcal{X}^* \subseteq \mathcal{X}(f(D_0)) \setminus \mathcal{X}(D_0)$ such that, for all time steps t and queries q, the function g yields $g(q(Y), Y) = q(D_t)$, where D_t is the data structure after request sequence $\rho_{\leq t}$, and Y is the result of request sequence $\rho_{\leq t}$ with arbitrary requests from $\mathcal{X}^*$ inserted, applied to $f(D_0)$.*
5. *f is computable in polynomial time and g is computable in $\tilde{O}(1)$ time.*

The request sequence $\bar{\rho}$ is universal in the sense that it does not depend on any specific γ-OuMv instance. However, the sequence $\bar{\rho}$ depends on the dynamic problem $\mathcal{P}$ and the reduction from γ-OuMv to $\mathcal{P}$. Specifically, $\bar{\rho}$ consists of all updates that *might* be necessary in the reduction from γ-OuMv to encode a vector update $(\vec{u}_k, \vec{v}_k)$ into a $\mathcal{P}$ instance.

The following theorem shows that all problems $\mathcal{P}$, that have a reduction from OuMv satisfying Definition 26, 2-list accurate predictions offer no improvement over a dynamic algorithm with no predictions.

▶ **Theorem 27.** *Suppose $\mathcal{P}$ is a locally correctable problem. Then there is no algorithm solving $\mathcal{P}$ with 2-list accurate predictions with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(n_1,n_2)U(n) + q(n_1,n_2)Q(n)\Big) = \tilde{\delta}(n_1n_2n_3)$ if the OuMv conjecture is true.*

Proof. We begin by constructing a reduction from γ-OuMv to $\mathcal{P}$ that admits efficiently computable 2-list accurate predictions. If an efficient algorithm with predictions exists, then we can design a dynamic algorithm without predictions as follows. First, we compute the efficiently computable predictions, and then run the algorithm with predictions as a sub-routine, violating the lower bound based on the OuMv conjecture.

Consider a γ-OuMv instance with $n_1 \times n_2$ matrix M and vector updates $\{(\vec{u}_k, \vec{v}_k)\}_{k=1}^{n_3}$. By assumption, there is an initial data structure D_0 and request sequence $\rho' = B'_1 \circ B'_2 \circ \ldots \circ B'_{n_3}$ such that each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$ can be computed in $O(q(n_1, n_2))$ time given the answers to the queries in $B'_1 \circ B'_2 \circ \ldots \circ B'_k$. We construct a new request sequence $\rho^* = B^*_1 \circ B^*_2 \circ \ldots \circ B^*_{n_3}$ on the augmented initial data structure $f(D_0)$. Fix an arbitrary update $x^* \in \mathcal{X}^*$. For each update in B_k, B^*_k will contain $\bar{\rho}_t$ if $\bar{\rho}_t \in B'_k$ and x^* otherwise. Consider a query at time step t. Let Y_t be the current state of the data structure, that is $f(D_0)$ with request sequence $\rho^*_{\leq t}$ applied. By assumption, $g(q(Y_t), Y_t) = q(D_{t'})$ where $D_{t'}$ is D_0 with $\rho'_{\leq t'}$ applied and $\rho'_{\leq t'} \subset \rho'$ is the longest prefix such that $\rho'_{\leq t'} \subseteq \rho^*_{\leq t}$. Then, given the query computations after $B^*_1 \circ B^*_2 \circ \ldots \circ B^*_k$, we can compute $\vec{u}_k^\top M \vec{v}_k$ in $\tilde{O}(q(n_1, n_2))$ time as g is efficiently computable and $B'_1 \circ B'_2 \circ \ldots \circ B'_k \subseteq B^*_1 \circ B^*_2 \circ \ldots \circ B^*_k$ contains all the queries required to compute $\vec{u}_k^T M \vec{v}_k$.

Next, we claim that there is an efficient prediction for the above reduction. That is, consider the prediction $\hat{\rho} = \{(\bar{\rho}_t, x^*)\}_{t \in [T]}$. Since ρ^* contains either $\bar{\rho}_t$ or x^* at the t-th position, this is a 2-list accurate prediction. Furthermore $\hat{\rho}$ is efficiently computable.

Therefore, suppose there is an efficient algorithm with 2-list accurate predictions. Then, given a γ-OuMv instance, we compute $f(D_0)$ and $\hat{\rho}$ in the preprocessing phase in polynomial time, providing this as the initial input to the algorithm with predictions. Then, we compute each vector update using the appropriate query computations from ρ^*, therefore obtaining a dynamic algorithm for the γ-OuMv instance. Thus, the update and query times must not satisfy $n_3\Big(u(n_1, n_2)U(n) + q(n_1, n_2)Q(n)\Big) = \tilde{\delta}(n_1 n_2 n_3)$, if the OuMv conjecture is true. ◀

We conclude with a simple argument showing that #s-$\triangle$ is locally correctable. Note that this immediately implies that no algorithm solving #s-$\triangle$ can have total work subcubic in n using Theorem 27.

▶ **Theorem 28.** *The #s-$\triangle$ problem is locally correctable.*

Proof. The universal sequence is n_3 copies of the block, $B_k = ((s, u_i))_{i=1}^n \circ ((s, v_i))_{i=1}^n \circ (q)$ for all blocks B_k, i.e. $\bar{\rho} = B_1 \circ \ldots \circ B_{n_3}$. Then augmentation function f takes $G_M \cup \{s\}$ and outputs $G_M \cup \{s, t\}$ while the correction function g is the identity. It is easy to check that the reduction of Figure 1 uses a subsequence of $\bar{\rho}$ and that #s-$\triangle$ after each block computes $\vec{u}_k^\top M \vec{v}_k$. Clearly, f increases the size instance by 1 and we have argued that #s-$\triangle$ on the augmented graph is exactly #s-$\triangle$ on the original graph, since s, t are not adjacent. ◀

6 Locally Reducible Problems: Lower Bounds for Delay Predictions

In this section, we will provide a framework for proving trade-off conditional lower bounds against algorithms with bounded delay predictions given a conditional lower bound against online algorithms (without predictions). To do so, we introduce the notion of *locally reducible* dynamic problems (Definitions 30 and 32) and show that for this large class of problems, the OuMv-based lower bounds carry through to the setting of algorithms with d-delayed predictions. For each problem, we show that there is a delay threshold (roughly the number of updates and queries used to process one round in the OuMv problem) below which predictions offer no benefit over a generic online algorithm. The basic idea is that, when allowed sufficient delay, every OuMv request sequence can be generated by simply *reordering* one generic request sequence. Thus, even with prediction, a locally reducible dynamic problem is still powerful enough to solve any γ-OuMv instance, and is, thus, still difficult to compute. This implies that a prediction algorithm does not only need to know *what* operations will happen, but also *when* the operations will happen. We also show that the lower bound degrades gracefully as the prediction quality surpasses this threshold.

6.1 Locally Reducible Dynamic Problems

We now define the class of locally reducible dynamic problems. Then we show in Theorems 33 and 40 that for any locally reducible problem, OuMv-based lower bounds extend to dynamic algorithms with bounded delay predictions. For a multi-set S, let $\text{set}(S)$ denote the set of elements that occur in S at least once. For two request sequences ρ_1, ρ_2, let $\rho_1 \circ \rho_2$ denote the concatenation of the two request sequences. We define also the notion of a cyclic update.

▶ **Definition 29.** *Consider a dynamic problem $\mathcal{P}$ with updates $\mathcal{X}$ and queries $\mathcal{Q}$. An update $x \in \mathcal{X}$ has* **cyclic order** $\text{ord}(x)$ *if for any request sequence ρ, inserting or removing exactly $\text{ord}(x)$ identical copies of x into the sequence ρ between indices i_0, i_1 does not change the results of any queries that are not part of ρ between the indices i_0 and i_1. We say x is* **cyclic** *if $1 \leq \text{ord}(x) = O(1)$.*

For example, consider edge insertion and edge deletion not as two separate operations, but as one operation called *(edge) flipping* (which inserts the edge if it exists and deletes it if it does not exist). Edge flipping has cyclic order 2. In order to place strong bounds on the positions of each individual update and query, we will insert redundant updates so that the positions of requests are more predictable. For example, if at the current time, an edge was predicted to be flipped 2 more times than it has already been flipped, we can flip the edge twice without changing the dynamic graph to correct this prediction error. Having updates of small cyclic order therefore allow us to insert redundant updates without significantly blowing up the size of the problem instance and therefore weakening our lower bounds.

In a bit more detail, consider a typical OMv-based lower bound. Let $\mathcal{P}$ be a dynamic problem for which there is an OuMv lower bound. First, for some fixed γ, there is a generic reduction from any arbitrary γ-OuMv instance with arbitrary parameters n_1, n_2, n_3 to an instance of $\mathcal{P}$ of size $n = n(n_1, n_2)$. In this reduction, each of the n_3 rounds of the γ-OuMv instance is simulated separately, i.e. for each round there are $u(n_1, n_2)$ updates and $q(n_1, n_2)$ queries for the problem instance $\mathcal{P}$. To simulate a single OuMv round, we choose some subset of necessary updates from a universal request block to correctly encode the γ-OuMv instance into the $\mathcal{P}$ instance. To construct our prediction, we predict for each block that the whole universal request set occurs. Since the request sequence is a subset of the universal set, an update cannot occur in a block *before* it is predicted to. However, this prediction could very well predict a request to occur in a block long *after* it is predicted to. For example, in the reduction of Figure 1 if there is an index i_0 such that $\vec{u}_k[i_0] = 0$ for all k, then the edge (s, u_{i_0}) will never be flipped in the original request sequence. To solve this, we add (after the query for the vector update $(\vec{u}_k, \vec{v}_k)$ has arrived and before the next vector update arrives) $\text{ord}(x)$ copies of update x whenever an update x has occurred in fewer than $k - \text{ord}(x)$ request blocks when simulating the vector update $(\vec{u}_k, \vec{v}_k)$. These redundant requests do not change the result of any query computation and ensure that a request cannot occur more than $\text{ord}(x)$ blocks later than it is predicted to.

We now give separate definitions for fully dynamic and partially dynamic problems, beginning with the fully dynamic setting.

▶ **Definition 30** (Fully Dynamic Locally Reducible)**.** *Let $\mathcal{P}$ be a fully dynamic problem with update set $\mathcal{X}$ and query set $\mathcal{Q}$. Suppose there is no algorithm for $\mathcal{P}$ with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(n_1, n_2)U(n) + q(n_1, n_2)Q(n)\Big) = \tilde{\delta}(n_1 n_2 n_3)$, if the OuMv conjecture is true, where n_1, n_2, n_3 are integers satisfying $n_1 = \lfloor n_2^\gamma \rfloor$ for some constant $\gamma > 0$, functions $u, q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and $n = n(n_1, n_2)$ is the size of the $\mathcal{P}$ instance in the reduction.*

$\mathcal{P}$ is **(u, q)-locally reducible from γ-OuMv** *if there exists a universal request sequence $\bar{\rho} = \bar{\rho}(n_1, n_2)$ of requests satisfying the following properties:*

1. *The universal request sequence $\bar{\rho}$ consists of n_3 identical copies of the sequence $B = B(n_1, n_2)$, indexed $B_1, B_2, \ldots, B_{n_3}$. The request sequence B contains $u(n_1, n_2)$ updates and $q(n_1, n_2)$ queries.*
2. *For any γ-OuMv instance with $n_1 \times n_2$ matrix M and vector updates $\{(\vec{u}_k, \vec{v}_k)\}_{k=1}^{n_3}$ the reduction constructs an initial data structure D_0 and request sequence ρ satisfying:*
   a. *ρ is the concatenation of request sequences $B'_1 \circ B'_2 \circ \ldots \circ B'_{n_3}$ where $B'_k = \pi_k(B''_k)$ is an ordering of B''_k for some $B''_k \subseteq B$.*
   b. *Each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$, $\vec{u}_k^\top M \vec{v}_k$ can be computed in $O(q(n_1, n_2))$ time based on the answers given to the queries in $B'_1 \circ B'_2 \circ \ldots \circ B'_k$.*
3. *Every update $x \in \bar{\rho}$ in the universal sequence $\bar{\rho}$ is cyclic.*

▶ Remark 31. Our proof assumes that each update has some small finite cyclic order. For example, the edge flip operation has order 2. Alternatively, we can view an update in a fully dynamic algorithm to have an inverse operation. For example, removal and insertion of the same edge are inverse operations. The proof of Theorem 33 for locally reducible fully dynamic problems follows in this case as well. Whenever we insert two edge flips in the proof of Theorem 33, we can insert an operation and its inverse operation counterpart. Both of these sequences of two updates have the desired effect of leaving the underlying data structure unmodified.

Let us compare the Definitions 26 and 30. In both cases, the universal request sequence $\bar{\rho}$ depends only on the reduction from γ-OuMv to the dynamic problem $\mathcal{P}$. It is universal in the sense that $\bar{\rho}$ is *independent* of any specific γ-OuMv instance.

In Definition 30, each block of the request sequence ρ does not have to respect the order of $\bar{\rho}$. Each block of the request sequence in the reduction to a locally correctable problem must be a subsequence of the corresponding block in the universal request sequence. In the reduction to a locally reducible problem, we may instead arbitrarily permute a subsequence of a block of the universal request sequence. To see why this is the case, observe that a list accurate prediction imposes the constraint that a certain update can occur only at $O(1)$ time steps in each block (since only 2 updates can occur at a given time step). Instead, bounded delay predictions allow the update to be placed at any point within a range of the predicted update. We are therefore free to order the subset of block of requests without being forced to adhere to the original order in the universal sequence.

Furthermore, instead of requiring that any instance can be efficiently augmented to a larger instance with an efficient "correction" function to the query computations (Conditions 3, 4, and 5), we now require that each update is cyclic (Condition 3). We now give the definition for partially dynamic locally reducible problems.

▶ **Definition 32** (Partially Dynamic Locally Reducible). *Let $\gamma > 0$ be a constant and n_1, n_2, n_3 be integers satisfying $n_1 = \lfloor n_2^\gamma \rfloor$. Let $u, q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Let $\mathcal{P}$ be an incremental (resp. decremental) dynamic problem with update set $\mathcal{X}$ and query set $\mathcal{Q}$.*

Suppose there is no algorithm for $\mathcal{P}$ with update time $U(n)$ and query time $Q(n)$ satisfying,

$$n_3\Big(u(n_1, n_2)U(n) + q(n_1, n_2)Q(n)\Big) = \tilde{\delta}(n_1 n_2 n_3)$$

*if the OuMv conjecture is true, where $n = n(n_1, n_2)$ is the size of the $\mathcal{P}$ instance in the reduction. $\mathcal{P}$ is (u, q)-**locally reducible from γ-OuMv** if there exists a universal request sequence $\bar{\rho} = \bar{\rho}(n_1, n_2)$ of requests (where only queries can occur more than once) satisfying the following properties:*

1. *The universal request sequence $\bar{\rho}$ consists of n_3 subsequences $\{B_k\}_{k=1}^{n_3}$ where request block B_k contains $u(n_1, n_2)$ updates and $q(n_1, n_2)$ queries.*

2. *For any γ-OuMv instance with $n_1 \times n_2$ matrix M and vector updates $\{(\vec{u}_k, \vec{v}_k)\}_{k=1}^{n_3}$ the reduction constructs an initial data structure D_0 and request sequence ρ satisfying:*
   a. *ρ is the concatenation of request sequences $B'_1 \circ B'_2 \circ \ldots \circ B'_{n_3}$ where $B'_k = \pi_k(B_k)$ is an ordering of B_k.*
   b. *Each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$, $\vec{u}_k^\top M \vec{v}_k$ can be computed in $O(q(n_1, n_2))$ time based on the answers given to the queries in $B'_1 \circ B'_2 \circ \ldots \circ B'_k$.*

We are now ready to present our main lower bound result.

▶ **Theorem 33.** *Let $\gamma > 0$ be a constant and u, q be functions. Suppose $\mathcal{P}$ is a fully dynamic problem that is (u, q)-locally reducible from γ-OuMv and let $C = \max_{x \in \mathcal{X}} \text{ord}(x)$.*

Then there is no algorithm solving $\mathcal{P}$ with $(1+C)(u(n_1, n_2)+q(n_1, n_2))$ delayed predictions with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(n_1, n_2)U(n) + q(n_1, n_2)Q(n)\Big) = \tilde{\delta}(n_1 n_2 n_3)$ if the OuMv conjecture is true.

We begin by defining some useful notation, denoting the position in which the k-th instance of a request occurs.

▶ **Definition 34.** *Let $\mathcal{X}$ denote the set of updates and $\mathcal{Q}$ the set of queries. Let $\rho \in (\mathcal{X} \cup \mathcal{Q})^T$ be a sequence of requests. For a given request $a \in \mathcal{X} \cup \mathcal{Q}$ and $k \in \mathbb{N}$, define $pos(a, k, \rho)$ to be the position in ρ of the k-th occurrence of a. If a does not occur k times in ρ, $pos(a, k, \rho) = \perp$. When the underlying request sequence is clear, we omit the sequence and write $pos(a, k)$.*

We now prove Theorem 33.

Proof. The key ingredient for the lower bound will be a reduction from a γ-OuMv instance to the dynamic problem $\mathcal{P}$. However, we will require the reduction to construct the online request sequence in such a way that we can efficiently compute a very simple prediction with $(1+C)(u(n_1, n_2)+q(n_1, n_2))$ delay. Then, if an efficient algorithm $\mathcal{A}$ with $(1+C)(u(n_1, n_2)+q(n_1, n_2))$ bounded delay predictions exists, we can solve the γ-OuMv problem by constructing the prediction and running the algorithm $\mathcal{A}$ as a sub-routine, violating the γ-OuMv lower bound. Throughout the proof, we defer proofs of certain lemmas to the full version [32].

Preliminaries. Since our reduction will be constructed by modifying an existing reduction, we begin by describing the existing reduction given by Condition 2. Consider a γ-OuMv instance $(M, \mathcal{U})$ consisting of a $n_1 \times n_2$ matrix M and a length-n_3 sequence of vector updates $\mathcal{U} = \{(\vec{u}_k, \vec{v}_k)\}_{k=1}^{n_3}$. We use $\rho'(\mathcal{U}) = B'_1 \circ B'_2 \circ \ldots \circ B'_{n_3}$ to denote the request sequence given by the reduction such that each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$ can be computed in $O(q(n_1, n_2))$ time given the answers to the queries in $B'_1 \circ B'_2 \circ \ldots \circ B'_k$.

We now describe the universal request sequence $\bar{\rho}$. In the given γ-OuMv reduction, each vector update $(\vec{u}_k, \vec{v}_k)$ is encoded into the data structure using some set of updates. For any request $x \in \mathcal{X} \cup \mathcal{Q}$, define $M(x)$ to be the maximum number of times an update $x \in \mathcal{X}$ occurs in a single block $B'_k \subset \rho'(\mathcal{U})$ over all k and all possible vector update sequences $\mathcal{U}$ (not just the worst-case one). The set B then contains $M(x)$ copies of x for all requests $x \in \mathcal{X} \cup \mathcal{Q}$. Note that if $M(x) = 0$, B does not contain any copy of $M(x)$. Additionally, we give an arbitrary, fixed order to the requests in B, so that B is a sequence. The universal request sequence $\bar{\rho}$ consists of n_3 copies of the sequence B. Define the predicted requested sequence $\hat{\rho} = \bar{\rho} = B_1 \circ B_2 \circ \ldots \circ B_{n_3}$ to be the universal request sequence.

Note that for a specific γ-OuMv instance $(M, \mathcal{U})$, we may not need every request in B_k to encode the vector update $(\vec{u}_k, \vec{v}_k)$ (with $k \in [n_3]$) given the state of the data structure after the block computing the previous vector update $(\vec{u}_{k-1}, \vec{v}_{k-1})$. However, by our definition of $M(x)$, it is possible to encode the vector update using some subset $B'_k \subseteq B_k$. This is precisely the block of requests in the request sequence $\rho'(\mathcal{U})$.

Notation. During the proof, we will focus on three request sequences. We use ρ' to denote the request sequence that is generated by the reduction of a *worst-case* γ-OuMv instance to $\mathcal{P}$. We denote by $\hat{\rho}$ the predicted request sequence constructed from the universal sequence $\bar{\rho}$ as discussed above. Note that it can be constructed without knowledge of ρ'. In this proof we will modify ρ' into a request sequence ρ^* encoding the same instance, with the additional property that $\pi(\rho^*) = \hat{\rho}$ for some permutation $|\pi - \mathsf{id}| \leq d$. Finally, we denote by ρ an arbitrary request sequence for $\mathcal{P}$ partitioned into n_3 blocks. For any $1 \leq k \leq n_3$ let $\rho_{(k)}$ denote the first k blocks of ρ and for a request $x \in \mathcal{X} \cup \mathcal{Q}$, let $N(x, k, \rho)$ denote the number of times that x occurs in $\rho_{(k)}$.

Our proof will proceed in three parts. In Part 1, we describe how to modify ρ' into ρ^* for any γ-OuMv instance. In Part 2, we show that $\pi(\rho^*) = \hat{\rho}$ for some permutation $|\pi - \mathsf{id}| \leq d$. In Part 3, we complete the proof by showing how an algorithm with bounded delay can be given $\hat{\rho}$ as prediction and can be used to answer any γ-OuMv instance.

Part 1: Constructing ρ^* from ρ'. We begin by describing the construction of ρ^*. In Lemma 35, we will argue that ρ^* constructed from ρ' correctly encodes the γ-OuMv instance, while satisfying certain properties that we will use in Part 2 to show that $\pi(\rho^*) = \hat{\rho}$ for some permutation π that is d-close to the identity permutation. We construct ρ^* sequentially, appending requests to the end of ρ^*. Recall that we have constructed the universal request sequence $\bar{\rho}$ by imposing an arbitrary order onto B and concatenating n_3 copies of B. Whenever we append a request x, we always append the copy of x that occurs *earliest* in the universal sequence $\bar{\rho}$ out of all requests of $\bar{\rho}$ that we have not already added to ρ^*.

We proceed by induction on k. The sequence after the k-th block is denoted $\rho^*_{(k)}$ and $\rho^*_{(0)}$ is the empty sequence. For $k > 0$ we extend $\rho^*_{(k-1)}$ to $\rho^*_{(k)}$ in the following three steps. For $i \in \{1, 2, 3\}$, let $\rho^*_{(k),i}$ denote the sequence ρ^* after step i when extending $\rho^*_{(k-1)}$ to $\rho^*_{(k)}$. Note $\rho^*_{(k)} = \rho^*_{(k),3}$.

1. We begin by concatenating B'_k to obtain $\rho^*_{(k),1} \leftarrow \rho^*_{(k-1)} \circ B'_k$, emphasizing that we always append the copy of an update that occurs earliest in $\bar{\rho}$ first. In particular, in this step we may in fact append a copy of x from B_j for $j < k$ rather than from B_k.
2. Then, for every update $x \in \mathcal{X}$ such that $N\left(x, k, \rho^*_{(k),1}\right) \leq kM(x) - \mathrm{ord}(x)$, we append $\left\lfloor \frac{kM(x) - N\left(x,k,\rho^*_{(k),1}\right)}{\mathrm{ord}(x)} \right\rfloor \cdot \mathrm{ord}(x)$ copies of x to the end of $\rho^*_{(k),1}$, emphasizing that we always append the copy of an update that occurs earliest in $\bar{\rho}$ first.
3. For every query $q \in \mathcal{Q}$ such that $N\left(q, k, \rho^*_{(k),2}\right) < kM(q)$, append $kM(q) - N\left(q, k, \rho^*_{(k),2}\right)$ copies of q to the end of $\rho^*_{(k),2}$, emphasizing that we always append the copy of a query that occurs earliest in $\bar{\rho}$ first.

Finally, after the final block B'_{n_3}, we add in all remaining unused requests from the universal request sequence $\bar{\rho}$. Thus ρ^* contains exactly all requests of $\hat{\rho}$. We now claim that ρ^* computes the same γ-OuMv instance as ρ', while satisfying certain additional properties we will use in Part 2.

▶ **Lemma 35.** *The constructed sequence ρ^* satisfies the following properties.*

1. *For all $x \in \mathcal{X}$ and $1 \leq k \leq n_3$, $kM(x) - \mathrm{ord}(x) < N(x, k, \rho^*) \leq kM(x)$.*
2. *For all $q \in \mathcal{Q}$ and $1 \leq k \leq n_3$, $N(q, k, \rho^*) = kM(q)$.*
3. *Each γ-OuMv request $(\vec{u}_k, \vec{v}_k)$, $\vec{u}_k^\top M \vec{v}_k$ can be answered in $O(q(n_1, n_2))$ time given the answers to the queries in the request sequence $\rho^*_{(k)} = B^*_1 \circ B^*_2 \circ \ldots \circ B^*_k$.*

Part 2: $\pi(\rho^*) = \hat{\rho}$ for some $\pi \in \mathbf{Perm}(T)$ such that $|\pi - \mathbf{id}| \leq d$. We show that ρ^* can be obtained by re-ordering the predicted sequence $\hat{\rho}$, for a permutation π that is d-close to the identity. In particular, this will show that $\hat{\rho}$ is a prediction with bounded delay d for request sequence set $\mathcal{S}$ consisting of all request sequences ρ^* which can be produced in Part 1.

Recall that $\hat{\rho}$ is the universal request sequence $\bar{\rho}$ obtained by concatenating n_3 copies of B. We will prove that $\hat{\rho}$ has $(C+1)(u(n_1,n_2)+q(n_1,n_2))$ delay with the following steps. First, in Lemma 36, we show that any request occurs at most $O(1)$ blocks away from its predicted position. Then, in Lemma 37, we bound the size of each block. Combining, we show in Lemma 38 that we obtain an upper bound on the delay of prediction $\hat{\rho}$.

▶ **Lemma 36.** *Let $1 \leq k \leq n_3$.*
If $x \in B_k$ is an update, then $x \in B_i^*$ for $k \leq i \leq k + \left\lceil \frac{\mathrm{ord}(x)}{M(x)} \right\rceil - 1$.
If $q \in B_k$ is a query, then $q \in B_k^*$.

▶ **Lemma 37.** *Let ρ^* be a request sequence as constructed in Part 1. Let $C = \max_{x \in B} \mathrm{ord}(x)$. For all $1 \leq k \leq n_3$, $k(u(n_1,n_2)+q(n_1,n_2)) - C \cdot u(n_1,n_2) < \left|\rho^*_{(k)}\right| \leq k(u(n_1,n_2)+q(n_1,n_2))$.*

▶ **Lemma 38.** *Let ρ^* be a request sequence as constructed in Part 1. Let $C = \max_{x \in B} \mathrm{ord}(x)$. Then, there exists permutation π that is $(1+C)(u(n_1,n_2)+q(n_1,n_2))$ close to the identity permutation and $\pi(\rho^*) = \hat{\rho}$.*

Part 3: Proof of Theorem 33 for Fully Dynamic Locally Reducible Problems. We now complete the proof of Theorem 33 for fully dynamic problems. Suppose there exists an algorithm $\mathcal{A}$ with $(1+C)(u(n_1,n_2)+q(n_1,n_2))$ bounded delay predictions solving $\mathcal{P}$ with polynomial preprocessing time, update time $U(n)$ and query time $Q(n)$.

Then, there is algorithm $\mathcal{B}$ (that works without prediction) for the γ-OuMv problem. Let $(M, \mathcal{U})$ be a worst-case γ-OuMv instance. In the preprocessing step, we compute the universal sequence $\bar{\rho} = \bar{\rho}(n_1,n_2)$ in polynomial time, construct the predicted request sequence $\hat{\rho} = \bar{\rho}$, and give $\hat{\rho}$ as input to $\mathcal{A}$. Note we do not need to see the matrix M nor the request sequence $\mathcal{U}$ to construct $\hat{\rho}$. Recall that matrix M is given to $\mathcal{B}$ during preprocessing. It gives M to $\mathcal{A}$, which builds the initial data structure D_0. This completes the preprocessing phase.

Next, given a vector update $(\vec{u}_k, \vec{v}_k)$, $\mathcal{B}$ constructs the sequence B_k^* and asks $\mathcal{A}$ to perform this sequence of requests. $\mathcal{A}$ returns the correct answers to the requests in B_k^* to $\mathcal{B}$ as, by Lemma 38, $\hat{\rho}$ is a $(1+C)(u(n_1,n_2), q(n_1,n_2))$ delayed prediction for $\rho' = \rho'(\mathcal{U})$, and $\mathcal{A}$ is a correct algorithm when given $(1+C)(u(n_1,n_2), q(n_1,n_2))$ delayed predictions. Thus, by Lemma 35, $\mathcal{B}$ can correctly answer $\vec{u}_k^T M \vec{v}_k$ in $O(q(n_1,n_2))$ time given the answers to the queries in B_k'.

We now analyze the complexity of $\mathcal{B}$. In the preprocessing phase, $\mathcal{B}$ constructs $\hat{\rho}$ and D_0, requiring only polynomial time. For each vector update, $\mathcal{B}$ computes B_k^* in $O(u(n_1,n_2)+q(n_1,n_2))$ (Lemma 39), asking $\mathcal{A}$ to perform the updates in B_k^*. Since $\mathcal{B}$ solves γ-OuMv, the OuMv conjecture states that $\mathcal{A}$ cannot satisfy, $n_3\Big(u(n_1,n_2)U(n) + q(n_1,n_2)Q(n)\Big) = \tilde{\delta}(n_1 n_2 n_3)$. We conclude the proof by proving Lemma 39.

▶ **Lemma 39.** *For all $1 \leq k \leq n_3$, B_k^* can be constructed in $O(u(n_1,n_2)+q(n_1,n_2))$ time.*

This concludes our proof of Theorem 33. ◀

For partially dynamic algorithms, in fact a simpler argument shows the analogous result. This lower bound gives a similar result to Theorem 1.3 in the independent work of [54].

▶ **Theorem 40.** *Let $\gamma > 0$ be a constant and u, q be functions. Suppose $\mathcal{P}$ is a partially dynamic problem that is (u, q)-locally reducible from γ-OuMv.*

Then there is no algorithm solving $\mathcal{P}$ with $(u(n_1, n_2) + q(n_1, n_2))$ delayed predictions with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(n_1, n_2)U(n) + q(n_1, n_2)Q(n)\Big) = \tilde{o}(n_1 n_2 n_3)$ if the OuMv conjecture is true.

Above, we have established that algorithms with predictions with delay $(1+C)(u(n_1, n_2)+q(n_1, n_2))$ with $C = \max_{x \in B} \mathrm{ord}(x)$ cannot be more efficient than algorithms with no predictions at all. In the following, we show that for smaller delay the conditional lower bounds based on the OuMv conjecture degrade gracefully with the quality of the predictions.

▶ **Theorem 41.** *Suppose $\mathcal{P}$ is (u, q)-locally reducible from γ-OuMv with $n = n(n_1, n_2)$ a non-decreasing function. Let $t \in (0, 1)$ be a constant. Let $d_1 = \lfloor d_2^{\gamma} \rfloor$ where $d_2 = \lfloor n_2^t \rfloor$.*

Then there is no algorithm solving $\mathcal{P}$ on instances of size $n = n(n_1, n_2)$ with $(1 + C)(u(d_1, d_2) + q(d_1, d_2))$ delayed predictions with update time $U(n)$ and query time $Q(n)$ satisfying $n_3\Big(u(d_1, d_2)U(n) + q(d_1, d_2)Q(n)\Big) = \tilde{o}(d_1 d_2 n_3) = \tilde{o}(n_1^t n_2^t n_3)$ if the OuMv conjecture is true.

In particular, as the guaranteed prediction quality increases (as t decreases towards 0), the lower bound weakens. As an application, we show that $\#s$-$\triangle$ is locally reducible.

▶ **Theorem 42.** *There is no algorithm solving the $\#s$-$\triangle$ problem with $O(n)$ delayed predictions with update time $U(n)$ and query time $Q(n)$ satisfying $n^2U(n) + nQ(n) = \tilde{o}(n^3)$ if the OuMv conjecture is true.*

Proof. We claim the $\#s$-$\triangle$ problem is a fully dynamic locally-reducible problem. Set parameters $n_1 = n_2 = n_3 = n$. We claim that $\#s$-$\triangle$ is $(n_1 + n_2, 1)$-locally reducible where $u(n_1, n_2) = n_1 + n_2$ and $q(n_1, n_2) = 1$. It is easily verified that the reduction of Figure 1 from γ-OuMv to $\#s$-$\triangle$ instances of size $n(n_1, n_2) = 1 + n_1 + n_2 = 2n + 1$ satisfies the required conditions, with $B_k = \{(s, u_i)\}_{i=1}^n \cup \{(s, w_i)\}_{i=1}^n \cup \{q\}$. Finally, we note that each update (edge flip) has cyclic order 2. ◀

For any $d = O(n^{1-\varepsilon})$, there is an algorithm with d delayed predictions overcoming the above lower bound. For $\#s$-$\triangle$, we give an update optimized algorithm with constant update time and $O(d^2)$ query time, as well as a query optimized algorithm with $O(d)$ update time and constant query time. Furthermore, this is tight by Theorem 41 and the above observation that $\#s$-$\triangle$ is locally reducible.

▶ **Theorem 43.** *Let $t \in (0, 1)$ be a constant and $d = \lfloor n^t \rfloor$. There is no algorithm solving the $\#s$-$\triangle$ problem with $O(d)$-delayed predictions with update time $U(n)$ and query time $Q(n)$ satisfying $ndU(n) + nQ(n) = \tilde{o}(nd^2)$ if the OuMv conjecture is true.*

Either update time is not $\tilde{o}(d)$ or query time is not $\tilde{o}(d^2)$, so our algorithms are optimal.

References

1 Anders Aamand, Piotr Indyk, and Ali Vakilian. (learned) frequency estimation algorithms under zipfian distribution. *CoRR*, abs/1908.05198, 2019. arXiv:1908.05198.

2 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Symposium on Foundations of Computer Science (FOCS'14)*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53.

3 Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. Online algorithms with multiple predictions. In *Proc. International Conference on Machine Learning, (ICML'22)*, pages 582–598, 2022. URL: https://proceedings.mlr.press/v162/anand22a.html.

4 Antonios Antoniadis, Joan Boyar, Marek Elias, Lene Monrad Favrholdt, Ruben Hoeksma, Kim S. Larsen, Adam Polak, and Bertrand Simon. Paging with succinct predictions. In *Proc. 40th International Conference on Machine Learning (ICML'23)*, pages 952–968, 2023. URL: https://proceedings.mlr.press/v202/antoniadis23a/antoniadis23a.pdf.

5 Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Learning-augmented dynamic power management with multiple states via new ski rental bounds. In *Proc. Neural Information Processing Systems (NeurIPS'21)*, pages 16714–16726, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/8b8388180314a337c9aa3c5aa8e2f37a-Abstract.html.

6 Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Mixing predictions for online metric algorithms. In *Proc. International Conference on Machine Learning (ICML'23)*, pages 969–983, 2023. URL: https://proceedings.mlr.press/v202/antoniadis23b.html.

7 Antonios Antoniadis, Peyman Jabbarzade Ganje, and Golnoosh Shahkarami. A novel prediction setup for online speed-scaling. In *Proc. 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT'22)*, pages 9:1–9:20, 2022. doi:10.4230/LIPIcs.SWAT.2022.9.

8 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proc. Symposium on Discrete Algorithms (SODA'22)*, pages 35–66, 2022. doi:10.1137/1.9781611977073.3.

9 Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Proc. Neural Information Processing Systems (NeurIPS'20)*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/e834cb114d33f729dbc9c7fb0c6bb607-Abstract.html.

10 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-augmented weighted paging. In *Proc. Symposium on Discrete Algorithms (SODA'22)*, pages 67–89, 2022. doi:10.1137/1.9781611977073.4.

11 Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in o(log n) update time (corrected version). *SIAM J. Comput.*, 47(3):617–650, 2018. doi:10.1137/16M1106158.

12 Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1836–1855. SIAM, 2021. doi:10.1137/1.9781611976465.110.

13 Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. Tight cell probe bounds for succinct boolean matrix-vector multiplication. In *Proc. of the 50th Symposium on Theory of Computing (STOC'18)*, pages 1297–1306, 2018. doi:10.1145/3188745.3188830.

14 Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011. doi:10.1137/090751670.

15 Shiri Chechik. Improved distance oracles and spanners for vertex-labeled graphs. In *Proc. 20th Annual European Symposium (ESA'12)*, pages 325–336, 2012. doi:10.1007/978-3-642-33090-2_29.

16 Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, pages 48:1–48:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.48.

17 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Algorithms with prediction portfolios. In *NeurIPS*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/7f9220f90cc85b0da693643add6618e6-Abstract-Conference.html.

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

19 Ran Duan. New data structures for subgraph connectivity. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proc., Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2010. doi:10.1007/978-3-642-14165-2_18.

20 Ran Duan and Seth Pettie. Connectivity oracles for failure prone graphs. In *Proc. of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 465–474. ACM, 2010. doi:10.1145/1806689.1806754.

21 Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=tilovEHA3YS.

22 Jacob Evald, Viktor Fredslund-Hansen, and Christian Wulff-Nilsen. Near-optimal distance oracles for vertex-labeled planar graphs. In *Proc. 32nd International Symposium on Algorithms and Computation (ISAAC'21)*, pages 23:1–23:14, 2021. doi:10.4230/LIPIcs.ISAAC.2021.23.

23 Shimon Even and Yossi Shiloach. An on-line edge-deletion problem. *J. ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

24 Daniele Frigioni and Giuseppe F. Italiano. Dynamically switching vertices in planar graphs. *Algorithmica*, 28(1):76–103, 2000. doi:10.1007/s004530010032.

25 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proc. of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1150–1162. SIAM, 2012. doi:10.1137/1.9781611973099.91.

26 Gramoz Goranci, Monika Henzinger, and Pan Peng. The power of vertex sparsifiers in dynamic graph algorithms. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.45.

27 Anupam Gupta, Debmalya Panigrahi, Bernardo Subercaseaux, and Kevin Sun. Augmenting online algorithms with ε-accurate predictions. In *Advances in Neural Information Processing Systems*, volume 35, pages 2115–2127, 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/0ea048312aa812b2711fe765a9e9ef05-Paper-Conference.pdf.

28 Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 548–557. IEEE, 2013.

29 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 21–30. ACM, 2015. doi:10.1145/2746539.2746609.

30 Monika Henzinger, Ami Paz, and Stefan Schmid. On the complexity of weight-dynamic network algorithms. In *IFIP Networking Conference, IFIP Networking 2021, Espoo and Helsinki, Finland, June 21-24, 2021*, pages 1–9. IEEE, 2021. doi:10.23919/IFIPNetworking52078.2021.9472803.

31 Monika Henzinger, Ami Paz, and A. R. Sricharan. Fine-grained complexity lower bounds for families of dynamic graphs. In *Proc. 30th European Symposium on Algorithms (ESA'22)*, volume 244 of *LIPIcs*, pages 65:1–65:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.65.

32 Monika Henzinger, Barna Saha, Martin P. Seybold, and Christopher Ye. On the complexity of algorithms with predictions for dynamic graph problems. *CoRR*, abs/2307.16771, 2023. doi:10.48550/arXiv.2307.16771.

33 Danny Hermelin, Avivit Levy, Oren Weimann, and Raphael Yuster. Distance oracles for vertex-labeled graphs. In *Proc. 38th International Colloquium, Automata, Languages and Programming (ICALP'11)*, pages 490–501, 2011. doi:10.1007/978-3-642-22012-8_39.

34 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=r1lohoCqY7.

35 Sungjin Im, Ravi Kumar, Aditya Petety, and Manish Purohit. Parsimonious learning-augmented caching. In *Proc. of the 39th International Conference on Machine Learning*, volume 162 of *Proc. of Machine Learning Research*, pages 9588–9601. PMLR, 17–23 July 2022. URL: https://proceedings.mlr.press/v162/im22a.html.

36 Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proc. of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1131–1142. SIAM, 2013. doi:10.1137/1.9781611973105.81.

37 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proc. of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.ch89.

38 Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proc. of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 489–504. ACM, 2018. doi:10.1145/3183713.3196909.

39 Jakub Lacki, Jakub Ocwieja, Marcin Pilipczuk, Piotr Sankowski, and Anna Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the steiner tree. In *Proc. 47th Symposium on Theory of Computing (STOC'15)*, pages 11–20, 2015. doi:10.1145/2746539.2746615.

40 Kasper Green Larsen and R. Ryan Williams. Faster online matrix-vector multiplication. In *Proc. 28th Symposium on Discrete Algorithms (SODA'17)*, pages 2182–2189, 2017. doi:10.1137/1.9781611974782.142.

41 Alexander Lindermayr and Nicole Megow. Non-clairvoyant scheduling with predictions revisited. *CoRR*, abs/2202.10199, 2022. arXiv:2202.10199.

42 Quanquan C. Liu and Vaidehi Srinivas. The predicted-deletion dynamic model: Taking advantage of ML predictions, for free. *CoRR*, abs/2307.08890, 2023. doi:10.48550/arXiv.2307.08890.

43 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021. doi:10.1145/3447579.

44 Aleksander Madry. *From graphs to matrices, and back: new techniques for graph algorithms*. PhD thesis, Massachusetts Institute of Technology, 2011.

45 Michael Mitzenmacher and Sergei Vassilvitskii. *Algorithms with Predictions*, pages 646–662. Cambridge University Press, 2021. doi:10.1017/9781108637435.037.

46 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proc. of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

47 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9684–9693, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

48 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. doi:10.1007/s00453-010-9401-5.

49 Liam Roditty and Uri Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. *SIAM J. Comput.*, 41(3):670–683, 2012. doi:10.1137/090776573.

50 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proc. of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1834–1845. SIAM, 2020. doi:10.1137/1.9781611975994.112.

51 Yongho Shin, Changyeol Lee, Gukryeol Lee, and Hyung-Chan An. Improved learning-augmented algorithms for the multi-option ski rental problem via best-possible competitive analysis. In *Proc. of the 40th International Conference on Machine Learning*, pages 31539–31561. PMLR, 2023. URL: https://proceedings.mlr.press/v202/shin23c.html.

52 Sandeep Silwal, Sara Ahmadian, Andrew Nystrom, Andrew McCallum, Deepak Ramachandran, and Seyed Mehran Kazemi. Kwikbucks: Correlation clustering with cheap-weak and expensive-strong signals. In *The Eleventh International Conference on Learning Representations ICLR*, 2023. URL: https://openreview.net/pdf?id=p0JSSa1AuV.

53 Shay Solomon. Fully dynamic maximal matching in constant update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, pages 325–334. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.43.

54 Jan van den Brand, Sebastian Forster, Yasamin Nazari, and Adam Polak. On dynamic graph algorithms with predictions. *CoRR*, abs/2307.09961, 2023. doi:10.48550/arXiv.2307.09961.

55 Ryan Williams. Matrix-vector multiplication in sub-quadratic time: (some preprocessing required). In *Proc. 18th Symposium on Discrete Algorithms (SODA'07)*, pages 995–1001, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283490.