Complexity Classes for Online Problems with and without Predictions *

Magnus Berg
IBA International Business Academy, Kolding, Denmark
mbmo@iba.dk

Joan Boyar Lene M. Favrholdt Kim S. Larsen
University of Southern Denmark, Odense, Denmark
{joan,lenem,kslarsen}@imada.sdu.dk

June 11, 2026

Abstract

With the developments in machine learning, there has been a surge in interest and results focused on algorithms utilizing predictions, not least in online algorithms where most new results incorporate the prediction aspect for concrete online problems. While the structural computational hardness of problems with regards to time and space is quite well developed, not much is known about online problems where time and space resources are typically not in focus. Some information-theoretical insights were gained when researchers considered online algorithms with oracle advice, but predictions of uncertain quality is a very different matter.

We initiate the development of a complexity theory for online problems with predictions, considering minimization problems and one prediction bit per request. Based on the most generic hard online problem

*Supported in part by the Independent Research Fund Denmark, Natural Sciences, grants DFF-0135-00018B and DFF-4283-00079B and in part by the Innovation Fund Denmark, grant 9142-00001B, Digital Research Centre Denmark, project P40: Online Algorithms with Predictions. An extended abstract of this paper was published in the International Joint Conference on Theoretical Computer Science – Frontier of Algorithmic Wisdom (IJTCS-FAW), Track A Best Paper Award, volume 15828 of Lecture Notes in Computer Science, pages 49-63. Springer, 2025.

type, string guessing, we define a family of hierarchies of complexity classes (indexed by pairs of error measures) and develop notions of reductions, class membership, hardness, and completeness. Our framework contains all the tools one expects to find when working with complexity, and we illustrate our tools by analyzing problems with different characteristics. In addition, we show that known lower bounds for paging with discard predictions apply directly to all hard problems for each class in the hierarchy based on the canonical pair of error measures. This paging problem is not complete for these classes.

Our work also implies corresponding complexity classes for classic online problems without predictions, with the corresponding complete problems.

1 Introduction

In computational complexity theory, one aims at classifying computational problems based on their hardness, by relating them via hardness-preserving mappings, referred to as reductions. Most commonly seen are time and space complexity, where problems are classified based on how much time or space is needed to solve the problem. Our primary aim is to classify online minimization problems with predictions based on the *competitiveness* of best possible deterministic online algorithms for each problem. In initiating this line of research, we consider minimization problems with binary predictions, encoding an optimal solution and given one bit at a time together with the requests. Our framework has recently been extended to maximization problems [8].

An *online problem* is an optimization problem where the input is revealed to an online algorithm in a piece-wise fashion in the form of *requests*. When a request arrives, an online algorithm must make an irrevocable decision about the request before the next request arrives. When comparing the quality of online algorithms, we use the standard *competitive analysis* framework [42] (see [12, 36]), where the competitiveness of an online algorithm is computed by comparing the algorithm's performance to an offline optimal algorithm's performance. Competitive analysis is a framework for worst-case guarantees, where we say that an algorithm is c-competitive if, asymptotically over all possible input sequences, its cost is at most a factor c times the cost of the optimal offline algorithm.

With the increased availability and improved quality of predictions from machine learning software, efforts to utilize predictions in online algorithms have increased dramatically [3]. Typically, one studies the competitiveness

of online algorithms that have access to additional information about the instance through (unreliable) predictions. Ideally, such algorithms should perform perfectly when the predictions are error-free (the competitiveness in this case is called the *consistency*), and perform as well as the best purely online algorithm when the predictions are erroneous (*robustness*). There is also a desire that an algorithm's competitiveness degrades gracefully from the consistency to the robustness as the predictions get worse (often referred to as *smoothness*). In particular, the performance should not plummet due to minor errors. To establish smoothness, it is necessary to have some measure of how wrong a prediction is. Thus, results of this type are based on some *error measure*.

The complexity of algorithms with predictions has also been considered in a different context, dynamic graph problems [31]. However, Henzinger et al. study the *time* complexity of dynamic data structures, whereas we create complexity classes where the hardness is based on *competitiveness*.

The basis for our complexity classes is a parameterized version of *asymmetric string guessing* [17], a generic hard online problem, where each request is simply a prompt for the algorithm to guess a bit. String guessing (not necessarily asymmetric) [9] has played a fundamental role in what is often referred to as advice complexity [10, 26, 29, 32, 16], where online algorithms have access to oracle-produced information about the instance which, in our context, can be considered infallible predictions. The same standard string guessing problem has also been used for the advice complexity of priority algorithms [11, 21, 13]. Specifically, we use *Online* $(1,t)$*-Asymmetric String Guessing with Unknown History and Predictions* (ASG_t), which will be our base family of complete problems, establishing a strict hierarchy based on the parameter, t. The *cost* of processing an input is the number of guesses of 1 plus t times the number of incorrect guesses of 0. Other variants of Asymmetric String Guessing have been used before in [39], considering connections between advice complexity and randomization, as well as in [18] where weighted versions of problems proven hard with respect to advice complexity in [17] are considered.

We define complexity classes, $\mathcal{C}^t_{\eta_0,\eta_1}$, parameterized by $t \in \mathbb{Z}^+ \cup \{\infty\}$ and a pair of error measures, (η_0, η_1), with certain properties. To prove that a problem, P, is $\mathcal{C}^t_{\eta_0,\eta_1}$*-hard*, one must show that P is as hard as ASG_t, and to prove *membership* in $\mathcal{C}^t_{\eta_0,\eta_1}$, one must show that ASG_t is as hard as P. If both are true, P is $\mathcal{C}^t_{\eta_0,\eta_1}$*-complete*. The as-hard-as relation is transitive, so our framework provides all the usual tools: if a subproblem of some problem

is hard, the problem itself is hard, one can reduce from the most convenient complete problem, etc. Thus, working with our complexity classes is similar to working with, e.g., NP, MAX-SNP [40], the W-hierarchy [27], and APX [6], in that hardness results are obtained by proving the existence of special types of reductions that preserve properties related to hardness. However, we obtain performance bounds that are independent of any conjectures.

Deriving lower bounds on the competitiveness of algorithms based on the hardness of string guessing has been considered before [9, 11, 29], with different objectives. The closest related work is in [17], where one of the base problems we use in this paper, $(1, \infty)$-Asymmetric String Guessing with Unknown History, was used as the base problem for the complexity class AOC; AOC-complete problems are hard online problems with advice. Note that despite the similarities, working with advice and predictions are quite different matters. In advice complexity, the competitive ratio is a function of the number of advice bits available. Working with predictions, competitiveness is a function of the quality, not the quantity, of information about the input. Thus, results cannot be translated between AOC and the complexity classes of this paper. Moreover, AOC is only one complexity class, not a hierarchy.

Strong lower bounds in the form of hardness results from our framework can be seen as indicating the insufficiency of a binary prediction scheme for a problem. Proving that a problem is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard suggests that one cannot solve it better than blindly trusting the predictions, when using binary predictions, giving a rather poor result. Hence, proving that a problem is hard serves as an argument for needing a richer prediction scheme for the problem, or possibly a more accurate way of measuring prediction error.

Our main contribution is an initial framework enabling a complexity theory for online algorithms with binary predictions. In this framework, proving that a problem is hard for a class currently requires that the predictions be binary encodings of the output the online algorithm should produce. Using this framework, we prove hardness and class membership results for several problems, including showing the completeness of Online t-Bounded Degree Vertex Cover (VC_t) for $\mathcal{C}^t_{\eta_0,\eta_1}$. Thus, VC_t, or any other complete problem, could be used as the basis for the complexity classes instead of ASG_t. However, we follow the tradition from advice complexity and use a string guessing problem, ASG_t, as its lack of structure offers simpler proofs. We illustrate the relative hardness of the problems we investigate in Figure 1. Worth noting is that by choosing the appropriate pair of error measures, our set-up immediately gives the same hardness results for purely online problems, that

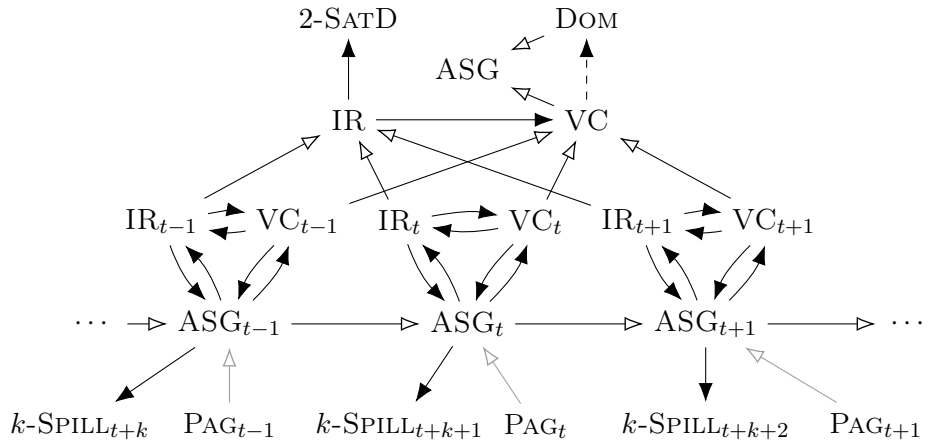


Figure 1: A hardness graph based on our complexity hierarchy. The problems shown are defined in Definitions 6, 34, 40, 44, 46, 49, and 57. Given two problems P and Q, we write $P \to Q$ to indicate that Q is as hard as P (see Definition 16). If the arrowhead is only outlined, P is not as hard as Q. If the arrow is dashed, P is asymptotically as hard as Q (see Definition 50). We leave out most arrows that can be derived by transitivity. The gray arrows hold with respect to the pair of error measures (μ_0, μ_1) (see Definition 3), and the remaining arrows hold with respect to all pairs of *insertion monotone* error measures (see Definition 5).

is, for algorithms without predictions.

As a soundness test for our framework, we consider the paging problem. Intuitively, researchers in online algorithms would expect that problem to be easier than, for instance, ASG_t or VC_t, because it seems that much more information is available. And, indeed, we can prove that, as opposed to those two problems, paging with discard predictions is not a complete problem for the complexity class $\mathcal{C}^t_{\mu_0,\mu_1}$ from our framework. Furthermore, maybe surprisingly, this non-complete paging problem gives rise to new lower bounds for $\mathcal{C}^t_{\mu_0,\mu_1}$-complete problems.

2 Preliminaries

In this paper, we consider online problems with binary predictions. From now on, we will not keep emphasizing that they are binary but simply refer to them as *predictions*. The algorithms have to make an irrevocable decision,

y_i, for each request. With the exception of the paging problem, we focus on problems, where these decisions (the y_i's) are also binary.

For any problem, P, as just described, we let $\mathcal{I}_P$ be the collection of instances of P, we let $\mathcal{A}_P$ be the set of algorithms for P, and we let $\textsc{Opt}_P$ be a fixed optimal algorithm for P. In our notation, an instance of P is a triple $I = (x, \hat{x}, r)$, consisting of two bitstrings $x, \hat{x} \in \{0,1\}^n$, and a sequence of requests $r = \langle r_1, r_2, \ldots, r_n \rangle$ for P. The bitstring x is an encoding of $\textsc{Opt}_P$'s solution, and $\hat{x}$ is a prediction of x. When an algorithm, Alg, receives the request r_i, it also receives the prediction $\hat{x}_i$ to aid its decision (represented by a bit, y_i) for r_i. What information is contained in each request, r_i, and the meaning of the bits x_i, $\hat{x}_i$, and y_i, will be specified for each problem. When there can be no confusion, we write Opt instead of $\textsc{Opt}_P$.

Given an algorithm, $\textsc{Alg} \in \mathcal{A}_P$, and an instance, $I \in \mathcal{I}_P$, we let $\textsc{Alg}[I]$ be Alg's solution to I, and $\textsc{Alg}(I)$ be the cost of $\textsc{Alg}[I]$.

2.1 Competitiveness and Error Measures

For purely online algorithms, we use the following definition of competitiveness: An algorithm, Alg, for a minimization problem, is *c-competitive* if there exists a constant, κ, called the *additive term*, such that for all instances $I = (x, r) \in \mathcal{I}_P$,

$$\textsc{Alg}(I) \leqslant c \cdot \textsc{Opt}(I) + \kappa.$$

We extend the definition of competitiveness to online algorithms with predictions, based on [4]. Here, the competitiveness of an algorithm is written as a function of two error measures[1], η_0 and η_1, where η_b is a function of the bits incorrectly predicted to be b.

Definition 1 Let (η_0, η_1) be a pair of error measures, let P be an online minimization problem with predictions, and let Alg be a deterministic online algorithm for P. If there exist three maps $\alpha, \beta, \gamma \colon \mathcal{I}_P \to \mathbb{R}_{\geqslant 0}$ and an additive constant, κ, such that for all $I \in \mathcal{I}_P$,

$$\textsc{Alg}(I) \leqslant \alpha \cdot \textsc{Opt}(I) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) + \kappa,$$

[1] We work with separate error measures for predicted 0s and 1s to allow for more detailed results. Our reductions and structural results would also work if we used only one error measure. For instance, letting $\eta_{t-1,1}(x, \hat{x}) = (t-1) \cdot \mu_0(x, \hat{x}) + \mu_1(x, \hat{x})$, Theorem 9 implies that FtP is $(1,1)$-competitive with respect to $\eta_{t-1,1}$. However, combining the two error measures into one, we lose some detail such as the trade-offs between α, β, and γ given in Theorem 60.

then ALG is (α, β, γ)-*competitive with respect to* (η_0, η_1). When (η_0, η_1) is clear from the context, we simply write that ALG is (α, β, γ)-competitive. If $\kappa \leqslant 0$, for all $I \in \mathcal{I}_P$, then ALG is *strictly* (α, β, γ)-*competitive*. If ALG is not (α, β, γ)-competitive for any α, β, γ, we simply say that it is *not competitive*. □

In our notation, α, β, and γ's dependency on I is kept implicit, as is tradition. For the results in this paper, it is not necessary that α, β, and γ are functions of the input. However, we are developing a framework that should be generally applicable, and many results from standard competitive analysis have competitive ratios that are functions of the input (size). Examples include graph coloring, dynamic binary search trees, and some scheduling problems.

Further, observe that any purely online α-competitive algorithm is $(\alpha, 0, 0)$-competitive with respect to any pair of error measures (η_0, η_1).

Since the performance of an algorithm is not measured by a competitive ratio, but by a triple, (α, β, γ), we cannot establish a total ordering of algorithms. Instead, we consider the concept of Pareto-optimality.

Definition 2 An (α, β, γ)-competitive algorithm is called *Pareto-optimal* for a problem, P, if, for any $\varepsilon > 0$, there cannot exist an $(\alpha - \varepsilon, \beta, \gamma)$-, an $(\alpha, \beta - \varepsilon, \gamma)$-, or an $(\alpha, \beta, \gamma - \varepsilon)$-competitive algorithm for P. □

2.1.1 Error Measures of Interest

We define a pair of error measures, (μ_0, μ_1), equivalent to the pair of error measures from [4], where μ_b is the number of incorrect predictions of $b \in \{0, 1\}$:

Definition 3 For any instance, $I = (x, \hat{x}, r)$,

$$\mu_0(I) = \sum_{i=1}^{n} x_i \cdot (1 - \hat{x}_i) \quad \text{and} \quad \mu_1(I) = \sum_{i=1}^{n} (1 - x_i) \cdot \hat{x}_i.$$

□

The following pair of error measures allows us to extend the results of this paper to purely online algorithms.

Definition 4 For any instance, $I = (x, \hat{x}, r)$, $Z_0(I) = Z_1(I) = 0$. □

In this paper, we will consider error measures with the property that insertion of a finite number of correctly predicted bits into an instance does not increase the error:

Definition 5 An error measure η is called *insertion monotone* if for any instance I, $\eta(I') \leqslant \eta(I)$, where I' is obtained by inserting a finite number of correctly predicted requests into I. □

Clearly μ_b and Z_b, $b \in \{0, 1\}$, are insertion monotone.[2] Although formally, the predictions $\hat{x}$ exist in general results that assume insertion monotone measures, these predictions are not used in the analysis when Z_b is used; the competitiveness must be obtained independently of the correctness of the prediction. Thus, these results hold for purely online algorithms, where the predictions do not exist.

2.2 Graph Problems

All graph problems mentioned in this paper are studied in the *vertex-arrival* model, the most standard for online graph problems. Hence, for any graph $G = (V, E)$, each request r_i is a vertex $v_i \in V$ that is revealed together with all edges of the form $(v_j, v_i) \in E$, where $j \leqslant i$.

We only consider simple unweighted graphs.

3 Asymmetric String Guessing: A Collection of Hard Problems

Given a bitstring, x, and a $t \in \mathbb{Z}^+ \cup \{\infty\}$, the task of an algorithm for $(1, t)$-Asymmetric String Guessing (ASG_t) is to correctly guess the contents of x. The cost of a solution is the number of guesses of 1 plus t times the number of incorrect guesses of 0. When $t = \infty$, the problem corresponds to the string guessing problem considered in [17]. We now define the problem more formally.

Definition 6 For any $t \in \mathbb{Z}^+ \cup \{\infty\}$, an instance of the problem *Online $(1, t)$-Asymmetric String Guessing with Unknown History and Predictions* (ASG_t) is a triple $I = (x, \hat{x}, r)$, where $x = \langle x_1, \ldots x_n \rangle$ and $\hat{x} = \langle \hat{x}_1, \ldots, \hat{x}_n \rangle$ are bitstrings and $r = \langle r_1, \ldots, r_n \rangle$ is a sequence of requests. Each request,

[2] Other examples of insertion monotone functions are L^p norms. Moreover, any linear combination of insertion monotone functions is insertion monotone.

r_i, is a prompt for the algorithm to output a bit, y_i. Together with r_i, $\hat{x}_i$ is revealed, but x is only revealed after the last request.

Given an instance $I \in \mathcal{I}_{\mathrm{ASG}_t}$ of ASG_t with $t \in \mathbb{Z}^+$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} \left(y_i + t \cdot x_i \cdot (1 - y_i) \right),$$

where y_i is Alg's i'th guess.

When $t = \infty$, we abbreviate ASG_t by ASG and rewrite the objective function as:

$$\textsc{Alg}(I) = \begin{cases} \sum_{i=1}^{n} y_i, & \text{if } \sum_{i=1}^{n} x_i \cdot (1 - y_i) = 0 \\ \infty, & \text{otherwise.} \end{cases}$$

□

3.1 ASG_t without Predictions

For all $t \in \mathbb{Z}^+ \cup \{\infty\}$, we may consider ASG_t as a purely online problem by omitting $\hat{x}$. We briefly state the main results on ASG_t without predictions. The following observation is well-known.

Observation 7 ([17]) For any purely online algorithm, Alg, for ASG, there is no function, f, such that Alg is $f(n)$-competitive. □

The observation follows from the fact that if an algorithm, Alg, ever guesses 0, there is an instance where Alg guesses 0 on a true 1, and thus incurs cost ∞. On the other hand, if Alg only guesses 1, there is an instance consisting of only 0's, such that Opt will incur cost 0 while the cost of Alg is equal to the length of the sequence.

Theorem 8 Let $t \in \mathbb{Z}^+$ and $\varepsilon > 0$. Then, for ASG_t, the following hold.

(i) The algorithm that always guesses 0 is t-competitive.

(ii) There is no purely online $(t - \varepsilon)$-competitive deterministic algorithm.

Proof We consider each item separately.

Towards (i): Let Alg be the algorithm that always guesses 0. Then, for

any instance $I = (x, r) \in \mathcal{I}_{\text{ASG}_t}$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} t \cdot x_i = t \cdot \sum_{i=1}^{n} x_i = t \cdot \textsc{Opt}(I).$$

Towards (ii): Assume towards contradiction that there exists a deterministic purely online $(t - \varepsilon)$-competitive algorithm, $\textsc{Alg}$, for some $\varepsilon > 0$. Then, there exists a constant $0 < \varepsilon' < 1$ such that $\textsc{Alg}$ is $(t - \varepsilon')$-competitive. This means that there exists an additive term, $\kappa \in O(1)$, such that, for all $I = (x, r) \in \mathcal{I}_{\text{ASG}_t}$,

$$\textsc{Alg}(I) \leqslant (t - \varepsilon') \cdot \textsc{Opt}(I) + \kappa. \tag{1}$$

We define a family $\{I^n\}_{n \in \mathbb{Z}^+}$ of instances, where, for any $n \in \mathbb{Z}^+$ and any $i \in \{1, 2, \ldots, n\}$, the i'th true bit in $I^n = (x^n, r^n)$ is

$$x_i^n = \begin{cases} 0, & \text{if } y_i^n = 1, \\ 1, & \text{if } y_i^n = 0, \end{cases}$$

where y_i^n is $\textsc{Alg}$'s i'th guess when run on I^n. Since $\textsc{Alg}$ is deterministic, the collection $\{I^n\}_{n \in \mathbb{Z}^+}$ is well-defined.

For each $i = 1, 2, \ldots, n$, if $y_i^n = 0$, then $x_i^n = 1$, and so $\textsc{Alg}$ incurs cost t, and $\textsc{Opt}$ incurs cost 1. On the other hand, if $y_i^n = 1$, then $x_i^n = 0$, and so $\textsc{Alg}$ incurs cost 1, and $\textsc{Opt}$ incurs cost 0. Hence, for each $n \in \mathbb{Z}^+$,

$$\begin{aligned} \textsc{Alg}(I^n) &= t \cdot \textsc{Opt}(I^n) + 1 \cdot (n - \textsc{Opt}(I^n)) \\ &= (t - 1) \cdot \textsc{Opt}(I^n) + n. \end{aligned} \tag{2}$$

Combining Equation (2) with Inequality (1), we get

$$(t - 1) \cdot \textsc{Opt}(I^n) + n \leqslant (t - \varepsilon') \cdot \textsc{Opt}(I^n) + \kappa.$$

Solving for κ, we get

$$\begin{aligned} \kappa &\geqslant n - (1 - \varepsilon') \cdot \textsc{Opt}(I^n) \\ &\geqslant \varepsilon' \cdot n, \text{ since } 1 - \varepsilon' > 0 \text{ and } \textsc{Opt} \leqslant n. \end{aligned} \tag{3}$$

This contradicts that $\kappa \in O(1)$. □

3.2 ASG_t with Predictions and Error Measures (μ_0, μ_1)

Now, we turn to the hardness of ASG_t with predictions, focusing on the pair (μ_0, μ_1) of error measures.

An obvious algorithm for ASG_t is *Follow-the-Predictions* (FtP), which always sets its guess, y_i, to the given prediction, $\hat{x}_i$.

Theorem 9 For any $t \in \mathbb{Z}^+$ and any α and β such that $\alpha \geqslant 1$ and $\alpha+\beta \geqslant t$, FtP is strictly $(\alpha, \beta, 1)$-competitive for ASG_t with respect to (μ_0, μ_1).

Proof Consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$ with $n = |r|$, and let y_i be FtP's output on r_i, $1 \leqslant i \leqslant n$. Then,

$$\begin{aligned}
\textsc{FtP}(I) &= \sum_{i=1}^{n} (y_i + t \cdot x_i \cdot (1 - y_i)) \\
&= \sum_{i=1}^{n} (\hat{x}_i + t \cdot x_i \cdot (1 - \hat{x}_i)) \\
&\leqslant \sum_{i=1}^{n} \hat{x}_i + (\alpha + \beta) \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)), \text{ since } \alpha + \beta \geqslant t \\
&= \alpha \cdot \sum_{i=1}^{n} x_i + \beta \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)) + \sum_{i=1}^{n} (1 - \alpha \cdot x_i) \cdot \hat{x}_i \\
&\leqslant \alpha \cdot \sum_{i=1}^{n} x_i + \beta \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)) + \sum_{i=1}^{n} (1 - x_i) \cdot \hat{x}_i, \text{ since } \alpha \geqslant 1 \\
&= \alpha \cdot \textsc{Opt}(I) + \beta \cdot \mu_0(I) + \mu_1(I).
\end{aligned}$$

□

Corollary 10 For any $t \in \mathbb{Z}^+$, FtP is strictly $(1, t-1, 1)$-competitive for ASG_t with respect to (μ_0, μ_1).

In the following, we extend a lower bound on Paging with Discard Predictions by Antoniadis et al. [4] to ASG_t. For completeness, we state the theorem here, though restricted to ASG_t and without proof. In Theorem 60 in Section 7.1.1, it is restated and proven in a more general form.

Theorem 11 Let $t \in \mathbb{Z}^+$. Then, for any (α, β, γ)-competitive algorithm for ASG_t with respect to (μ_0, μ_1),

(i) $\alpha + \beta \geqslant t$ and

(ii) $\alpha + (t-1) \cdot \gamma \geqslant t$.

Next, we prove a negative result on the competitiveness of algorithms for ASG_t that does not follow from Theorem 11. Together with Theorems 8, 9, and 11, this negative result gives a complete classification of all Pareto-optimal algorithms for ASG_t with respect to (μ_0, μ_1).

Lemma 12 Let ALG be an (α, β, γ)-competitive algorithm for ASG_t with respect to (μ_0, μ_1). If $\alpha < t$, then $\gamma \geqslant 1$.

Proof Assume towards contradiction that ALG is (α, β, γ)-competitive, where $\alpha < t$ and $\gamma < 1$. Then there exists an $\varepsilon > 0$ such that $\alpha \leqslant t - \varepsilon$ and $\gamma \leqslant 1 - \varepsilon$.

Consider the family $\{I^n\}_{n \in \mathbb{Z}^+}$ of instances with $I^n = (x^n, \hat{x}^n, r^n)$, where

- $\hat{x}^n = \langle 1^n \rangle$ and
- $x_i^n = 1 - y_i^n$, where y_i^n is ALG's output on r_i^n, $1 \leqslant i \leqslant n$.

Observe that x^n is well-defined since ALG is deterministic.

Then,

$$\begin{aligned}
\mathrm{ALG}(I^n) &= \sum_{i=1}^{n} \left(y_i^n + t \cdot (1 - y_i^n)\right) \\
&= \sum_{i=1}^{n} \left(1 - x_i^n + t \cdot x_i^n\right), \text{ by definition of } x^n \\
&= n + (t-1) \cdot \sum_{i=1}^{n} x_i^n \qquad (4)
\end{aligned}$$

Since ALG is $(t - \varepsilon, \beta, 1 - \varepsilon)$-competitive, there exists a $\kappa \in O(1)$, such that, for each n,

$$\begin{aligned}
\mathrm{ALG}(I^n) &\leqslant (t - \varepsilon) \cdot \mathrm{OPT}(I^n) + (1 - \varepsilon) \cdot \mu_1(I^n) + \kappa, \text{ since } \mu_0(I^n) = 0 \\
&= (t - \varepsilon) \cdot \sum_{i=1}^{n} x_i^n + (1 - \varepsilon) \cdot \sum_{i=1}^{n} (1 - x_i^n) + \kappa \\
&= (t-1) \cdot \sum_{i=1}^{n} x_i^n + (1 - \varepsilon) \cdot n + \kappa \qquad (5)
\end{aligned}$$

Combining (4) and (5) and solving for κ, we get

$$n + (t-1) \cdot \sum_{i=1}^{n} x_i^n \leqslant (t-1) \cdot \sum_{i=1}^{n} x_i^n + (1-\varepsilon) \cdot n + \kappa \Leftrightarrow$$
$$\varepsilon \cdot n < \kappa,$$

contradicting that $\kappa \in O(1)$, since $\varepsilon > 0$. □

Theorem 13 Let ALG be an (α, β, γ)-competitive algorithm for ASG_t. Then, ALG is Pareto-optimal with respect to (μ_0, μ_1), if and only if

(i) $\alpha = t$ and $\beta = \gamma = 0$, or

(ii) $\alpha < t$, $\beta = t - \alpha$, and $\gamma = 1$.

Proof We consider an (α, β, γ)-competitive algorithm and split the proof into two cases based on the value of α.

Case $\alpha \geqslant t$: By Theorem 8(i), there exists a $(t, 0, 0)$-competitive algorithm. This algorithm is Pareto-optimal, since neither the second nor the third entry can be improved and by Theorem 8(ii), no $(t-\varepsilon, 0, 0)$-competitive algorithm exists. Thus, since $\alpha \geqslant t$, ALG is Pareto-optimal, if and only if $\alpha = t$ and $\beta = \gamma = 0$.

Case $\alpha < t$: We note that $\alpha \geqslant 1$, since no algorithm can be better than 1-consistent. Thus, by Theorem 9, the algorithm FTP is $(\alpha, t - \alpha, 1)$-competitive. Hence, for ALG to be Pareto-optimal, we must have that

(a) $\beta < t - \alpha$,

(b) $\gamma < 1$, or

(c) $\beta = t - \alpha$ and $\gamma = 1$.

By Theorem 11(i), (a) is impossible, and, by Lemma 12, (b) is impossible. Thus, ALG is Pareto-optimal, if and only if $\beta = t - \alpha$ and $\gamma = 1$. □

Note that by Theorem 8(i), the algorithm that always guesses zero is t-robust with respect to (μ_0, μ_1).

Corollary 14 FTP and the algorithm that always guesses 0 are both Pareto-optimal with respect to (μ_0, μ_1).

Theorem 15 There is no competitive algorithm for ASG with respect to (μ_0, μ_1).

Proof Suppose towards contradiction that for all sequences I, $\textsc{Alg}(I) \leqslant \alpha\,\textsc{Opt}(I)+\beta\mu_0(I)+\gamma\mu_1(I)+\kappa$, where $\alpha, \beta, \gamma \colon \mathcal{I}_P \to \mathbb{R}_{\geqslant 0}$ and κ is a constant. Consider the family $\{I^n\}_{n\in\mathbb{Z}^+}$ of instances with $I^n = (x^n, \hat{x}^n, r^n)$, where

- $\hat{x}^n = \langle 0^n \rangle$ and
- $x^n_i = 1 - y^n_i$, where y^n_i is $\textsc{Alg}$'s output on r^n_i, $1 \leqslant i \leqslant n$.

Observe that x^n is well-defined since $\textsc{Alg}$ is deterministic. Also, note that $\mu_1(I^n) = 0$, so $\textsc{Alg}(I^n) \leqslant \alpha\,\textsc{Opt}(I^n) + \beta\mu_0(I^n) + \kappa$.

If $y^n_i = 0$ for some i, then $x^n_i = 1$, so $\textsc{Alg}$ incurs an infinite cost on r_i. Thus, since neither α nor β can be infinite, we have a contradiction.

Otherwise, $y^n_i = 1$ for all i, so $\textsc{Alg}(I^n) = n$, $\textsc{Opt}(I^n) = 0$ and $\mu_0(I^n) = 0$, implying that $n \leqslant \alpha \cdot 0 + \beta \cdot 0 + \kappa = \kappa \in O(1)$, giving a contradiction.

Thus, $\textsc{Alg}$ is not competitive. □

4 Hierarchies of Complexity Classes

In this section, we formally introduce the complexity classes, prove that, for each pair of error measures, they form a strict hierarchy, and show multiple fundamental structural properties of the complexity classes.

4.1 Relative Hardness and Reductions

We define relative hardness as follows:

Definition 16 Let P and Q be two online problems with predictions and error measures (η_0, η_1) and (φ_0, φ_1). We say that Q is *as hard as P with respect to (φ_0, φ_1) and (η_0, η_1)*, if the existence of an (α, β, γ)-competitive algorithm for Q with respect to (φ_0, φ_1) implies the existence of an (α, β, γ)-competitive algorithm for P with respect to (η_0, η_1). If the error measures are clear from the context, we simply say that Q is *as hard as P*. If Q is as hard as P, we use the notation $Q \geq_{\text{o}} P$. We also say that P is *no harder than Q*, denoting this $P \leq_{\text{o}} Q$. □

It is not hard to see that the as-hard-as relation is both reflexive and transitive, but for completeness we give the proof here.

Lemma 17 The relation as-hard-as is reflexive and transitive.

Proof We prove each property separately.

Towards reflexivity: Proving reflexivity translates to proving that the existence of an (α, β, γ)-competitive algorithm with respect to (η_0, η_1) for P implies the existence of an (α, β, γ)-competitive algorithm with respect to (η_0, η_1) for P, which is a tautology.

Towards transitivity: Let P, W, and Q be online maximization problems with predictions with error measures (η_0, η_1), (ξ_0, ξ_1), and (φ_0, φ_1), and assume that $Q \geq_{\text{o}} W$ and that $W \geq_{\text{o}} P$. Let $\textsc{Alg}_Q \in \mathcal{A}_Q$ be an (α, β, γ)-competitive algorithm for Q with respect to (φ_0, φ_1). Since $Q \geq_{\text{o}} W$ with respect to (φ_0, φ_1) and (ξ_0, ξ_1), the existence of $\textsc{Alg}_Q$ implies the existence of an (α, β, γ)-competitive algorithm with respect to (ξ_0, ξ_1) for W, say $\textsc{Alg}_W$. Since $W \geq_{\text{o}} P$ with respect to (ξ_0, ξ_1) and (η_0, η_1), the existence of $\textsc{Alg}_W$ also implies the existence of an (α, β, γ)-competitive algorithm for P with respect to (η_0, η_1), and so $Q \geq_{\text{o}} P$ with respect to (φ_0, φ_1) and (η_0, η_1). □

As a tool for proving hardness, we introduce the notion of reductions. A reduction from a problem, P, to another problem, Q, consists of a mapping of instances of P to instances of Q and a mapping from algorithms for Q to algorithms for P, with the requirement that (α, β, γ)-competitive algorithms for Q map to (α, β, γ)-competitive algorithms for P. In this paper, we use a restricted type of reduction:

Definition 18 Let P and Q be online minimization problems with predictions, and let (η_0, η_1) and (φ_0, φ_1) be pairs of error measures for the predictions in P and Q, respectively. Let $\rho = (\rho_{\text{A}}, \rho_{\text{I}})$ be a tuple consisting of two maps, $\rho_{\text{A}} \colon \mathcal{A}_Q \to \mathcal{A}_P$ and $\rho_{\text{I}} \colon \mathcal{A}_Q \times \mathcal{I}_P \to \mathcal{I}_Q$.

If there exists a constant, a, called the *reduction term* of ρ, such that for each instance $I_P \in \mathcal{I}_P$ and each algorithm $\textsc{Alg}_Q \in \mathcal{A}_Q$, letting $\textsc{Alg}_P = \rho_{\text{A}}(\textsc{Alg}_Q)$ and $I_Q = \rho_{\text{I}}(\textsc{Alg}_Q, I_P)$,

(O1) $\textsc{Alg}_P(I_P) \leqslant \textsc{Alg}_Q(I_Q) + a$,

(O2) $\textsc{Opt}_Q(I_Q) \leqslant \textsc{Opt}_P(I_P)$,

(O3) $\varphi_0(I_Q) \leqslant \eta_0(I_P)$, and $\varphi_1(I_Q) \leqslant \eta_1(I_P)$,

then ρ is called a *strict online reduction from P to Q with respect to* (η_0, η_1) *and* (φ_0, φ_1). If the pairs (η_0, η_1) and (φ_0, φ_1) are clear from the context, then we simply say that ρ is a *strict online reduction* from P to Q, and write $\rho \colon P \xrightarrow{r} Q$. □

In many of the examples to follow, we do not need a (nonzero) reduction term to establish the reductions, and we only mention the reduction term

when needed.

Lemma 19 Let $\rho = (\rho_{\text{A}}, \rho_{\text{I}})$ be a strict online reduction from a problem, P, with predictions and error measures (η_0, η_1) to a problem, Q, with predictions and error measures (φ_0, φ_1). Let $\text{ALG}_Q \in \mathcal{A}_Q$ be an (α, β, γ)-competitive algorithm for Q with respect to (φ_0, φ_1), and let $\text{ALG}_P = \rho_{\text{A}}(\text{ALG}_Q)$. Then, ALG_P is an (α, β, γ)-competitive algorithm for P with respect to (η_0, η_1).

Proof Consider any instance, $I_P \in \mathcal{I}_P$, and let $I_Q = \rho_{\text{I}}(\text{ALG}_Q, I_P)$. Then, by Condition (O1) and Definition 1, there exists constants, a and κ, such that

$$\begin{aligned}
\text{ALG}_P(I_P) &\leqslant \text{ALG}_Q(I_Q) + a \\
&\leqslant \alpha \cdot \text{OPT}_Q(I_Q) + \beta \cdot \varphi_0(I_Q) + \gamma \cdot \varphi_1(I_Q) + \kappa + a \\
&\leqslant \alpha \cdot \text{OPT}_P(I_P) + \beta \cdot \eta_0(I_P) + \gamma \cdot \eta_1(I_P) + \kappa + a, \\
&\quad \text{by (O2) and (O3)}
\end{aligned}$$

□

Observation 20 Lemma 19 implies that strict online reductions serve the desired purpose of reductions: If there exists a strict online reduction $\rho\colon P \xrightarrow{r} Q$, then $Q \geq_{\text{o}} P$. □

When using strict online reductions, we will often simply use the term *reduction*.

For the rest of this paper, we only consider reductions where the quality of predictions is measured using the same pair of error measures for both problems.

4.2 Introducing the Complexity Classes

For any pair, (η_0, η_1), of error measures and any $t \in \mathbb{Z}^+ \cup \{\infty\}$, we define the complexity classes $\mathcal{C}^t_{\eta_0,\eta_1}$ as the set of minimization problems with predictions that are no harder than ASG_t with respect to (η_0, η_1):

Definition 21 For each $t \in \mathbb{Z}^+ \cup \{\infty\}$ and each pair of error measures, (η_0, η_1), the complexity class $\mathcal{C}^t_{\eta_0,\eta_1}$ is the closure of ASG_t under the as-hard-as relation with respect to (η_0, η_1). Hence, for an online minimization problem, P,

- $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, if $\text{ASG}_t \geq_{\text{o}} P$,
- P is $\mathcal{C}^t_{\eta_0,\eta_1}$-*hard*, if $P \geq_{\text{o}} \text{ASG}_t$, and

- P is $\mathcal{C}^t_{\eta_0,\eta_1}$-*complete*, if $P \in \mathcal{C}^t_{\eta_0,\eta_1}$ and P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

The notation $\mathcal{C}^\infty_{\eta_0,\eta_1}$ is usually abbreviated $\mathcal{C}_{\eta_0,\eta_1}$. □

Observation 22 By Observation 20 and Lemma 19, if P and Q are $\mathcal{C}^t_{\eta_0,\eta_1}$-complete problems, then there exists an (α,β,γ)-competitive algorithm for P if and only if there exists an (α,β,γ)-competitive algorithm for Q. Thus, all of the results for ASG_t hold for the complete problems in $\mathcal{C}^t_{\eta_0,\eta_1}$, and the upper bounds for ASG_t hold for all of the problems in $\mathcal{C}^t_{\eta_0,\eta_1}$. □

Since the as-hard-as relation is reflexive, ASG_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete for any pair of error measures, (η_0,η_1), and any t. Further, due to transitivity, we have the following simple but important properties that one would expect should hold for complexity classes.

Theorem 23 Let $t \in \mathbb{Z}^+ \cup \{\infty\}$ and let (η_0,η_1) be any pair of error measures.

(i) If $P \in \mathcal{C}^t_{\eta_0,\eta_1}$ and $P \geq_{\mathrm{o}} Q$, then $Q \in \mathcal{C}^t_{\eta_0,\eta_1}$.

(ii) If P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard and $Q \geq_{\mathrm{o}} P$, then Q is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We prove each item separately.

Towards (i): Since $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, $\mathrm{ASG}_t \geq_{\mathrm{o}} P$. Since $P \geq_{\mathrm{o}} Q$, transitivity implies that $\mathrm{ASG}_t \geq_{\mathrm{o}} Q$, and thus $Q \in \mathcal{C}^t_{\eta_0,\eta_1}$.

Towards (ii): Since P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, $P \geq_{\mathrm{o}} \mathrm{ASG}_t$. Since $Q \geq_{\mathrm{o}} P$, transitivity implies that $Q \geq_{\mathrm{o}} \mathrm{ASG}_t$, and thus we conclude that Q is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard. □

This theorem implies results concerning special cases of a problem:

Corollary 24 Let $t \in \mathbb{Z}^+ \cup \{\infty\}$ and let (η_0,η_1) be any pair of error measures. Let P and P_{sub} be online minimization problems such that $\mathcal{I}_{P_{\mathrm{sub}}} \subseteq \mathcal{I}_P$.

(i) If $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, then $P_{\mathrm{sub}} \in \mathcal{C}^t_{\eta_0,\eta_1}$.

(ii) If P_{sub} is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, then P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof There exists a trivial online reduction, $\rho\colon P_{\mathrm{sub}} \xrightarrow{r} P$, obtained by setting $\rho_{\mathrm{A}}(\textsc{Alg}) = \textsc{Alg}$ and $\rho_{\mathrm{I}}(\textsc{Alg}, I) = I$, for all algorithms $\textsc{Alg} \in \mathcal{A}_P$ and all instances $I \in \mathcal{I}_{P_{\mathrm{sub}}}$. Hence, this is a consequence of Theorem 23. □

Note that Theorem 23 implies the following.

Observation 25 Any $\mathcal{C}^t_{\eta_0,\eta_1}$-complete problem can be used instead of ASG_t as the base problem when defining $\mathcal{C}^t_{\eta_0,\eta_1}$. □

As shown in Lemmas 35 and 38, the better known Online t-Bounded Degree Vertex Cover with Predictions is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete with respect to a wide range of pairs of error measures and may therefore be used as the basis of these complexity classes instead of ASG_t. However, we chose to define the complexity classes as the closure of ASG_t, due to it being a generic problem that is easily analyzed. Also, after establishing the first $\mathcal{C}^t_{\eta_0,\eta_1}$-hard problems, we may reduce from any $\mathcal{C}^t_{\eta_0,\eta_1}$-hard problem to prove hardness. Finally, a problem Q is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, if and only if $Q \geq_{\mathrm{o}} P$, for all $P \in \mathcal{C}^t_{\eta_0,\eta_1}$. This is in line with the structure of other complexity classes such as NP and APX, where a problem Q is NP-hard, respectively APX-hard, if and only if there exists a polynomial-time reduction, respectively PTAS-reduction, from any problem in NP, respectively APX, to Q.

Proposition 26 For every problem P in $\mathcal{C}^t_{\mu_0,\mu_1}$, there exists a $(t,0,0)$-competitive algorithm and for every $1 \leqslant \alpha \leqslant t$, there exists a $(\alpha, t-\alpha, 1)$-competitive algorithm for P. If P is $\mathcal{C}^t_{\mu_0,\mu_1}$-complete, this is tight for Pareto-optimal algorithms for P.

Proof By Theorems 9 and 13, the theorem holds for ASG_t. Thus, by Observation 22, it also holds for any problems in $\mathcal{C}^t_{\eta_0,\eta_1}$. □

Note that by Proposition 26, all of the problems in $\mathcal{C}^t_{\mu_0,\mu_1}$ have a $(1,t-1,1)$-competitive algorithm, so there are algorithms with consistency 1. Moreover, problems in the class indexed by t have a t-robust algorithm, and for complete problems this is tight.

In introducing some initial complexity classes to the field of online algorithms (with and without predictions), we define reductions from complete problems for each class to all members of the class. The reductions map algorithms for one problem to algorithms for another problem. To show inclusion of a problem P in a class, the map is from P to a complete problem. The original complete problems for each class have binary outputs and predictions. In contrast, the Paging problem, with discard predictions for whether or not the current page will be in a selected Opt's cache the next time it is requested, obtains an optimal solution by choosing pages to evict among those that would not be in Opt's cache. Thus, the predictions are an encoding of all optimal solutions that would have a page in cache, if the selected Opt would. The sequence of pages chosen for eviction is the output.

4.3 Establishing the Hierarchy

In this subsection, we show that our complexity classes form a strict hierarchy by showing that ASG_{t+1} is strictly harder than ASG_t, for any $t \in \mathbb{Z}^+$.

Lemma 27 For any $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1),

(i) $\mathrm{ASG}_{t+1} \geq_{\mathrm{o}} \mathrm{ASG}_t$, and

(ii) ASG_t is not as hard as ASG_{t+1}.

Proof We prove each item separately.

Towards (i): We give a reduction, $\rho = (\rho_{\mathrm{A}}, \rho_{\mathrm{I}})$, from ASG_t to ASG_{t+1}. For any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$ and $\textsc{Alg}_{t+1} \in \mathcal{A}_{\mathrm{ASG}_{t+1}}$, we let $\rho_{\mathrm{I}}(\textsc{Alg}_{t+1}, I) = I$. We let $\textsc{Alg}_t = \rho_{\mathrm{A}}(\textsc{Alg}_{t+1})$ be the algorithm which, for each request, passes the prediction bit on to $\textsc{Alg}_{t+1}$ and outputs the same bit as $\textsc{Alg}_{t+1}$.

We verify that this reduction satisfies Conditions (O1)–(O3) from Definition 18. Since $\rho_{\mathrm{I}}(\textsc{Alg}_{t+1}, I) = I$, Conditions (O2)–(O3) are immediate for any pair of error measures (η_0, η_1). Towards Condition (O1), denote by $y_1, y_2, \ldots, y_n$ the bits guessed by $\textsc{Alg}_{t+1}$, and therefore also by $\textsc{Alg}_t$. Then,

$$\begin{aligned}
\textsc{Alg}_{t+1}(I) - \textsc{Alg}_t(I) &= \sum_{i=1}^{n} (y_i + (t+1) \cdot (1 - y_i) \cdot x_i) \\
&\quad - \sum_{i=1}^{n} (y_i + t \cdot (1 - y_i) \cdot x_i) \\
&= \sum_{i=1}^{n} (1 - y_i) \cdot x_i \geqslant 0,
\end{aligned}$$

and Condition (O1) follows.

Towards (ii): Assume towards contradiction that $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$. Then, for any (α, β, γ)-competitive algorithm for ASG_t, there exists an (α, β, γ)-competitive algorithm for ASG_{t+1}. By Theorem 8(i), the algorithm that always guesses 0 is $(t, 0, 0)$-competitive for ASG_t. Hence, there exists a $(t, 0, 0)$-competitive algorithm for ASG_{t+1}, which contradicts Theorem 8(ii). □

Lemma 28 For any $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1), $\mathcal{C}^t_{\eta_0,\eta_1} \subsetneq \mathcal{C}^{t+1}_{\eta_0,\eta_1}$.

Proof We first prove that $\mathcal{C}^t_{\eta_0,\eta_1} \subseteq \mathcal{C}^{t+1}_{\eta_0,\eta_1}$. For any $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, $\mathrm{ASG}_t \geq_{\mathrm{o}} P$, and, by Lemma 27(i), $\mathrm{ASG}_{t+1} \geq_{\mathrm{o}} \mathrm{ASG}_t$. Thus, by transitivity (Lemma 17),

$\mathrm{ASG}_{t+1} \geq_{\mathrm{o}} P$, and so $P \in \mathcal{C}^{t+1}_{\eta_0,\eta_1}$. To see that $\mathcal{C}^{t}_{\eta_0,\eta_1} \subsetneq \mathcal{C}^{t+1}_{\eta_0,\eta_1}$, observe that $\mathrm{ASG}_{t+1} \in \mathcal{C}^{t+1}_{\eta_0,\eta_1}$ and, by Lemma 27(ii), $\mathrm{ASG}_{t+1} \notin \mathcal{C}^{t}_{\eta_0,\eta_1}$. □

Lemma 29 For any $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1), no $\mathcal{C}^{t+1}_{\eta_0,\eta_1}$-hard problem is in $\mathcal{C}^{t}_{\eta_0,\eta_1}$.

Proof Assume towards contradiction that $\mathcal{C}^{t}_{\eta_0,\eta_1}$ contains a $\mathcal{C}^{t+1}_{\eta_0,\eta_1}$-hard problem, P. Then, $\mathrm{ASG}_t \geq_{\mathrm{o}} P$, and $P \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$. Thus, due to transitivity, $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$, contradicting Lemma 27. □

By similar arguments, we get the following:

Lemma 30 For any $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1),

- $\mathcal{C}^{t}_{\eta_0,\eta_1} \subsetneq \mathcal{C}_{\eta_0,\eta_1}$, and
- $\mathcal{C}^{t}_{\eta_0,\eta_1}$ contains no $\mathcal{C}_{\eta_0,\eta_1}$-hard problems.

Theorem 31 For any pair of error measures (η_0, η_1), we have a strict hierarchy of complexity classes:

$$\mathcal{C}^{1}_{\eta_0,\eta_1} \subsetneq \mathcal{C}^{2}_{\eta_0,\eta_1} \subsetneq \mathcal{C}^{3}_{\eta_0,\eta_1} \subsetneq \cdots \subsetneq \mathcal{C}_{\eta_0,\eta_1}.$$

4.4 Purely Online Algorithms

Observe that our complexity theory extends to a complexity theory for purely online algorithms as well. In particular, one may consider the complexity classes, $\mathcal{C}^{t}_{Z_0,Z_1}$. Recall that $Z_0(I) = Z_1(I) = 0$, for any instance $I = (x, \hat{x}, r)$. In this framework, any (α, β, γ)-competitive algorithm, ALG, for an online minimization problem P, satisfies that

$$\begin{aligned} \textsc{Alg}(I) &\leqslant \alpha \cdot \textsc{Opt}(I) + \beta \cdot Z_0(I) + \gamma \cdot Z_1(I) + \kappa \\ &= \alpha \cdot \textsc{Opt}(I) + \kappa, \end{aligned}$$

for all instances $I \in \mathcal{I}_P$, and so ALG is an α-competitive purely online algorithm for P. Since ASG_t has a t-competitive algorithm (it guesses all 0s) and is as hard as all problems in $\mathcal{C}^{t}_{Z_0,Z_1}$, all problems in $\mathcal{C}^{t}_{Z_0,Z_1}$ are (at worst) t-competitive.

By the above and Theorem 23, if a purely online problem, P, has an α-competitive algorithm, it is in $\mathcal{C}^{t}_{Z_0,Z_1}$, for any $t \geqslant \alpha$. If, in addition, P has no $(t - \varepsilon)$-competitive algorithm, P is $\mathcal{C}^{t}_{Z_0,Z_1}$-complete.

Since most results in this paper hold for any insertion monotone error measure, and Z_0 and Z_1 are insertion monotone, we obtain a similar complexity theory for purely online algorithms.

Observation 32 All results in this paper on problems and complexity classes with respect to insertion monotone error measures also hold for the same problems and classes without predictions. □

Note that our complexity theory may help translating results for one purely online problem directly to other purely online problems. In Section 7.1.1, we discuss a strategy for proving lower bounds for all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems. Using the same strategy, the lower bound from Theorem 8(ii) on the competitive ratio of any purely online algorithm for ASG_t extends to a lower bound on the competitiveness of any purely online algorithm for any $\mathcal{C}^t_{Z_0,Z_1}$-hard problems.

Observation 33 Assume that some minimization problem, Q, is as hard as some minimization problem, P, with respect to any pair of insertion monotone error measures. Then any lower bound on the competitiveness of the purely online version of P is a lower bound on the competitiveness of the purely online version of Q. Similarly, any upper bound for the purely online version of Q is an upper bound for the purely online version of P. □

5 A Template for Online Reductions via Simulation

In this section, we introduce a template for creating online reductions from ASG_t to a problem, Q, implying $\mathcal{C}^t_{\eta_0,\eta_1}$-hardness of Q with respect to any pair of insertion monotone error measures. Recall from Definition 5 that an error measure is insertion monotone if insertion of a finite number of correctly predicted bits into an instance does not increase the error. The reduction strategy is outlined in Algorithm 1. It might be an advantage to read this template in conjunction with the first example of its use for Vertex Cover (Lemma 35 in Section 6.1) to better understand how elements from the concrete problems enter into the template.

Algorithm 1 is a template where the challenge requests and the blocks need to be defined when a concrete online reduction is created. For each request, r_i, of the ASG_t instance, we create and give a challenge request, c_i, to the algorithm, $\textsc{Alg}'$, for Q in a way such that no information about the true bit, x'_i, can be inferred from the information about the instance obtained so far. If, for instance, Q is VC_t, each challenge request will be a vertex which is isolated at arrival. After we have treated the entire ASG_t instance, we give $\textsc{Alg}'$ one (possibly empty) block of requests per challenge request. For VC_t, the ith block consists of a (possibly empty) set of neighbors of the ith challenge request, which an optimal algorithm can reject. The purpose of the

Algorithm 1 Reducing ASG_t to Q

Input: $\textsc{Alg}' \in \mathcal{A}_Q$ and $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$
Output: $\textsc{Alg} \in \mathcal{A}_{\mathrm{ASG}_t}$ and $I' = (x', \hat{x}', r') \in \mathcal{I}_Q$ ▷ I' may be longer than I

1: **for** each request, r_i, in r **do**
2: Give $\textsc{Alg}'$ the *challenge request* c_i and the prediction $\hat{x}'_i = \hat{x}_i$
3: Output $y_i = y'_i$, where y'_i is the output of $\textsc{Alg}'$ for c_i
4: Receive $x = x_1, x_2, \ldots, x_n$ ▷ Finishes the ASG_t instance
5: **for** $j = 1, 2, \ldots, n$ **do**
6: **for** each request, r, in *block* $B_Q(x_j, y'_j, j)$ **do**
7: Give $\textsc{Alg}'$ the request r together with a correct prediction of 0

blocks is to ensure that $x'_i = x_i$, for $1 \leqslant i \leqslant n$, and $x'_i = 0$, for $i > n$, encode an optimal solution for Q. Thus, for each incorrect bit, $\hat{x}'_i$, in $\hat{x}'$, the bit $\hat{x}_i$ is also incorrect, though $\hat{x}'$ has additional, correct, predictions associated with the blocks. Since the error measures, η_0 and η_1, are insertion monotone, $\eta_0(I') \leqslant \eta_0(I)$ and $\eta_1(I') \leqslant \eta_1(I)$.

Summing up, in the input created for $\textsc{Alg}'$, the requests are the challenge requests given in Line 2 followed by the block requests (Lines 5–7), the prediction consists of the prediction bits for the ASG_t instance (Line 2) followed by a 0 for each block request, and the optimal solution is the correct solution to the ASG_t instance (Line 4) followed by a 0 for each block request. On each request, r_i, $\textsc{Alg}$ outputs the same as $\textsc{Alg}'$ does on c_i.

Note that $\textsc{Alg}'$ may not recognize when the challenge requests end and the blocks begin, in which case it may not answer all of the challenges in the blocks correctly, despite the correct predictions. However, this is not a problem, since the idea is to prove that $\textsc{Alg}$ does no worse than $\textsc{Alg}'$.

In the online reductions, we need to know which request we are dealing with to create the right blocks. However, most often this is clear from the context, and then we leave out the third parameter to $B_Q(x_j, y'_j, j)$.

6 Examples of $\mathcal{C}^t_{\eta_0,\eta_1}$-Hard Problems

We use the reduction template from Section 5 for proving $\mathcal{C}^t_{\eta_0,\eta_1}$-hardness of Vertex Cover (Lemma 35), k-Spill (Theorem 45), and Interval Rejection (Lemma 41). In addition, we demonstrate the use of transitivity to prove hardness, creating reductions from Interval Rejection to 2-SAT-Deletion

(Theorem 47) and from Vertex Cover to Dominating Set (Theorem 55).

6.1 Vertex Cover

Given a graph, $G = (V, E)$, an algorithm for Vertex Cover finds a subset $V' \subseteq V$ of vertices such that for each edge $e = (u, v) \in E$, $u \in V'$ or $v \in V'$. The cost of the solution is given by the size of V', and the goal is to minimize this cost. In t-Bounded Degree Vertex Cover, all vertices have degree at most t.

Other work on Vertex Cover includes the following. In the purely online setting, there exists a t-competitive algorithm for t-Bounded Degree Vertex Cover, and the problem does not allow for a $(t-\varepsilon)$-competitive algorithm, for any $\varepsilon > 0$ [24]. Considering advice, the problem is AOC-complete [17]. In the offline setting, Vertex Cover is NP-complete [33] and APX-complete [41, 25], and t-Bounded Degree Vertex Cover is MAX-SNP-hard [40].

We formally define the Vertex Cover problem studied in this section:

Definition 34 The input to *Online t-Bounded Degree Vertex Cover with Predictions* (VC_t) is a graph, $G = (V, E)$, with maximum degree at most t. When an algorithm, ALG, receives a vertex, v_i, it outputs $y_i = 1$ to accept this vertex or $y_i = 0$ to reject it.

For the algorithm's solution to be feasible, it must be a vertex cover, i.e.,

$$y_i = 1 \vee y_j = 1, \text{ for each edge } (v_i, v_j) \in E.$$

For an instance, $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{VC}_t}$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} y_i.$$

The set $\{v_i \mid x_i = 1\}$ is an optimal vertex cover. □

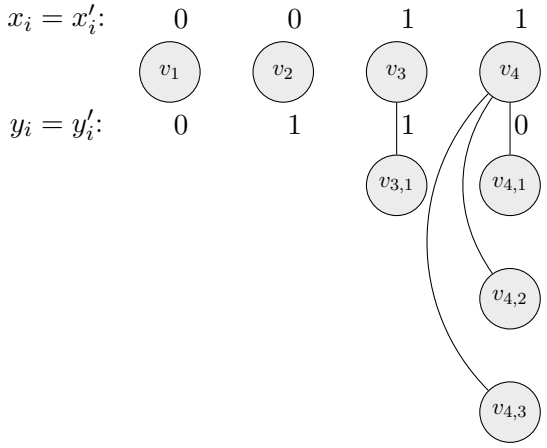
The standard unbounded Online Vertex Cover with Predictions is also considered, and is abbreviated VC.

First, we show hardness results for VC_t:

Lemma 35 For any $t \in \mathbb{Z}^+$ and any pair of insertion monotone error measures, (η_0, η_1), VC_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We give a strict online reduction $\rho = (\rho_{\mathrm{A}}, \rho_{\mathrm{I}})$ from ASG_t to VC_t, using the reduction template of Algorithm 1 in Section 5 (see Figure 2 for

an example). Consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$ and any $\textsc{Alg}' \in \mathcal{A}_{\mathrm{VC}_t}$. Let $\textsc{Alg} = \rho_{\mathrm{A}}(\textsc{Alg}')$ and $I' = \rho_{\mathrm{I}}(\textsc{Alg}', I)$ be the ASG_t algorithm and VC_t instance created in the reduction described below.



$\textsc{Opt} = \textsc{Opt}'$:	0	0	1	1
$\textsc{Alg}$:	0	1	1	t
$\textsc{Alg}'$:	0	1	$\geqslant 1$	t

Figure 2: A VC_3 instance constructed from an ASG_3 instance as described in the proof of Lemma 35. For each challenge request, the bits x_i and y_i are shown above and below v_i, respectively. Below the graph, the cost of each challenge request and its corresponding block requests is shown for $\textsc{Opt}$, $\textsc{Opt}'$, $\textsc{Alg}$, and $\textsc{Alg}'$.

The ith challenge request is an isolated vertex, v_i. Clearly, $v_1, v_2, \ldots, v_i$ give no information about x_i, as required. Recall from the template that $\hat{x}'_i = \hat{x}_i$ and that $\textsc{Alg}$ outputs $y_i = y'_i$, where y'_i is the output of $\textsc{Alg}'$ for v_i.

Following all the n challenge requests, we give the n blocks, where the ith block, $B(x_i, y'_i)$, is constructed as follows, with all prediction bits equal to 0 as described in the template.

- If $x_i = 0$, then $B(x_i, y'_i)$ is empty. Thus, no optimal solution will contain v_i.
- If $x_i = y'_i = 1$, then $B(x_i, y'_i)$ contains one request to a vertex, $v_{i,1}$, connected to v_i, ensuring that there is an optimal solution containing v_i.

- If $x_i = 1$ and $y'_i = 0$, then $B(x_i, y'_i)$ contains requests to t new vertices, $v_{i,j}$, $j = 1, 2, \ldots, t$, each connected to v_i, giving ALG$'$ a cost of t and ensuring that there is an optimal solution containing v_i.

Observe that $\{v_i \mid x_i = 1\}$ is an optimal solution to the VC_t instance constructed, as required by the template. This shows that $\textsc{Opt}(I) = \textsc{Opt}(I')$, satisfying Condition (O2) of Definition 18. It also shows that all block requests are correctly predicted, since all block predictions are zero. Since $\hat{x}'_i = \hat{x}_i$, for each challenge request, v_i, and η_0 and η_1 are insertion monotone, we conclude that $\eta_0(I') \leqslant \eta_0(I)$ and $\eta_1(I') \leqslant \eta_1(I)$, satisfying Condition (O3) of Definition 18.

To prove that Condition (O1) is also satisfied, we argue that $\textsc{Alg}(I) \leqslant \textsc{Alg}'(I')$. For each request, r_i in r, such that $x_i = 0$, ALG and ALG$'$ both have a cost of $y_i = y'_i$ on r_i. If $x_i = y'_i = 1$, both algorithms have a cost of 1 on r_i. Finally, if $x_i = 1$ and $y'_i = 0$, then ALG has a cost of t on r_i and ALG$'$ has a cost of t on $B(x_i, y'_i)$. □

A *star graph* is a tree where all vertices, except at most one, are leaves. A *star forest* is a disjoint collection of star graphs. Since the graphs constructed in the proof of Lemma 35 are star forests, we obtain the following corollary.

Corollary 36 For any $t \in \mathbb{Z}^+$ and any pair of insertion monotone error measures, (η_0, η_1), VC_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, even on star forests.

Since star forests are interval graphs, we also obtain the following.

Corollary 37 For any $t \in \mathbb{Z}^+$ and any pair of insertion monotone error measures, (η_0, η_1), VC_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, even on interval graphs.

Note that Figure 2 is an (interval) graph representation of the intervals in Figure 3.

Now, we turn to showing membership by reducing to ASG_t.

Lemma 38 For any $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1), $\mathrm{VC}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$.

Proof We define a strict online reduction, $\rho\colon \mathrm{VC}_t \xrightarrow{r} \mathrm{ASG}_t$. Consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{VC}_t}$ and any ALG$' \in \mathcal{A}_{\mathrm{ASG}_t}$. We define an instance, $I' = (x', \hat{x}', r') \in \mathcal{I}_{\mathrm{ASG}_t}$, and an algorithm, ALG, for handling I using the output of ALG$'$ on I':

For each request, r_i in r, give $\hat{x}'_i = \hat{x}_i$ to ALG$'$ and let y'_i be the output of ALG$'$. If v_i has a neighbor among $v_1, \ldots, v_{i-1}$ which is not in the vertex

cover constructed so far, ALG outputs $y_i = 1$, ensuring that ALG always outputs a vertex cover. Otherwise, it outputs $y_i = y'_i$. After the last request of I, compute an optimal solution, x, for I and present $x' = x$ to ALG$'$, in order to finish the instance I'.

Since we let $x' = x$ and $\hat{x}' = \hat{x}$, Condition (O3) from Definition 18 is trivially satisfied for any pair of error measures. Moreover, since $x' = x$, OPT$(I) =$ OPT(I'), and so Condition (O2) is also satisfied. Hence, it only remains to check Condition (O1). To this end, we prove that ALG$(I) \leqslant$ ALG$'(I')$.

We consider the cost of ALG and ALG$'$ on request r_i:

- $y'_i = y_i = 0$:
  - $x_i = 0$: Both algorithms have a cost of 0.
  - $x_i = 1$: ALG$'$ has a cost of t and ALG has a cost of 0.
- $y'_i = 0$, $y_i = 1$: ALG has a cost of 1.
  - $x_i = 0$: ALG$'$ has a cost of 0.
  - $x_i = 1$: ALG$'$ has a cost of t.
- $y'_i = y_i = 1$: Both algorithms have a cost of 1.

Note that ALG has a higher cost than ALG$'$, only when $x_i = y'_i = 0$ and $y_i = 1$. In this case, the cost of ALG is exactly one higher than that of ALG$'$. However, from $y_i \neq y'_i$, it follows by the definition of ALG that v_i has a neighbor, v_j, $j < i$, such that $y_j = 0$, in which case also $y'_j = 0$. Moreover, $x_i = 0$ implies that $x_j = 1$, since x encodes a feasible vertex cover. Since v_j has at most t neighbors, and since the cost of ALG$'$ on r'_j is t higher than that of ALG on r_j, we conclude that the total cost of ALG is no larger than that of ALG. Hence, Condition (O1) is satisfied. □

Our results about Vertex Cover are summarized in Theorem 39. Note that Items (i) and (v) follow directly from Lemmas 35 and 38.

Theorem 39 Consider any $t \in \mathbb{Z}^+$.

For any pair of insertion monotone error measures, (η_0, η_1),

(i) VC$_t$ is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete,

(ii) VC$_t$ on star forests is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete,

(iii) VC$_t$ on interval graphs is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete,

(iv) VC is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, and VC $\notin \mathcal{C}^t_{\eta_0,\eta_1}$.

For any pair of error measures, (η_0, η_1),

(v) $\mathrm{VC}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$,

(vi) VC_t on star forests is contained in $\mathcal{C}^t_{\eta_0,\eta_1}$,

(vii) VC_t on interval graphs is contained in $\mathcal{C}^t_{\eta_0,\eta_1}$,

(viii) VC $\in \mathcal{C}_{\eta_0,\eta_1}$, and VC is not $\mathcal{C}_{\eta_0,\eta_1}$-hard.

Proof First, we consider insertion monotone error measures:

Towards (i): This is a direct consequence of Lemmas 35 and 38.

Towards (ii): This is a direct consequence of Corollary 36 and of Lemma 38 in combination with Corollary 24(i).

Towards (iii): This is a direct consequence of Corollary 37 and of Lemma 38 in combination with Corollary 24(i).

Towards (iv): The hardness of VC is a direct consequence of Lemma 35 in combination with Corollary 24(i). Now, assume towards contradiction that VC $\in \mathcal{C}^t_{\eta_0,\eta_1}$, for some $t \in \mathbb{Z}^+$. This means that $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{VC}$. By Lemma 35, VC $\geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$, and so, by transitivity (Lemma 17), $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$, which contradicts Lemma 27(ii).

Next, we consider general error measures:

Towards (v): This is a direct consequence of Lemma 38.

Towards (vi)–(vii): These are direct consequences of Lemma 38 in combination with Corollary 24(i).

Towards (viii): To prove that VC $\in \mathcal{C}_{\eta_0,\eta_1}$, we give a strict online reduction $\rho\colon \mathrm{VC} \xrightarrow{r} \mathrm{ASG}$. To this end, consider any $\textsc{Alg}' \in \mathcal{A}_{\mathrm{ASG}}$ and any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{VC}}$. Let $I' = \rho_{\mathrm{I}}(\textsc{Alg}', I) = (x', \hat{x}', r')$, where $x' = x$, $\hat{x}' = \hat{x}$, and r' is a sequence of prompts for guessing the next bit as usual for ASG. Moreover, let $\textsc{Alg} = \rho_{\mathrm{A}}(\textsc{Alg}')$ be the algorithm that always outputs the same as $\textsc{Alg}'$ and, at the end, computes $\textsc{Opt}_{\mathrm{VC}_t}[I]$ and reveals it to $\textsc{Alg}'$ in the form of the bit string $x' = x$.

By construction, Conditions (O2) and (O3) from Definition 18 are trivially satisfied, and so it only remains to check Condition (O1). Since Alg outputs the same as $\textsc{Alg}'$, we only need to argue that as long as $\textsc{Alg}'$ produces a feasible solution, Alg does so too. Hence, suppose that Alg has created

an infeasible solution to instance I. Then, there exists an uncovered edge (v_i, v_j). Hence, $y_i = y_j = 0$, where y_i and y_j are the ith and jth guesses made by $\textsc{Alg}'$. However, since the edge (v_i, v_j) is contained in the underlying graph of I, at least one of v_i and v_j has been accepted by $\textsc{Opt}_{\text{VC}}$, and so $x_i = 1$ or $x_j = 1$. Hence, $\textsc{Alg}'$ has guessed 0 on a true 1, implying that $\textsc{Alg}'$ produced an infeasible solution.

To prove that VC is not $\mathcal{C}_{\eta_0,\eta_1}$-hard, let $\textsc{Acc}$ be the following online algorithm for VC. When receiving a vertex, v, if v has no neighbors, reject v; otherwise, accept v. We claim that $\textsc{Acc}$ is $(n-1, 0, 0)$-competitive, where $n = |V|$. To this end, consider any graph, G. If $\textsc{Opt}(G) = 0$, then G contains no edges, and so $\textsc{Acc}$ never accepts a vertex, in which case $\textsc{Acc}(G) = 0$. If, on the other hand, $\textsc{Opt}(G) \geqslant 1$, then $\textsc{Acc}(G) \leqslant n - 1$, since $\textsc{Acc}$ never accepts the first vertex. Hence, $\textsc{Acc}$, is $(n-1, 0, 0)$-competitive. If $\text{VC} \geq_{\text{o}} \text{ASG}$, then this implies the existence of an $(n-1, 0, 0)$-competitive algorithm for ASG, contradicting Observation 7. □

6.2 Interval Rejection

Given a collection of intervals $\mathcal{S}$ on the line, an Interval Rejection algorithm finds a subset $\mathcal{S}' \subseteq \mathcal{S}$ of intervals such that no two intervals in $\mathcal{S} \setminus \mathcal{S}'$ overlap. The cost of the solution is given by the cardinality of $\mathcal{S}'$, and the goal is to minimize this cost.

In the offline setting, Interval Rejection is solvable in polynomial time, by a greedy algorithm [35].

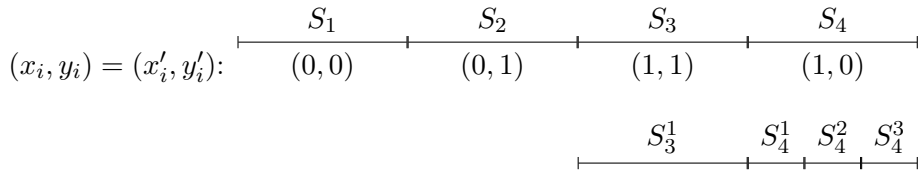
Definition 40 A request, r_i, for *Online t-Bounded Overlap Interval Rejection with Predictions* (IR_t) is an interval, S_i. Instances for IR_t satisfy that any requested interval overlaps at most t other intervals in the instance.

An algorithm, $\textsc{Alg}$, outputs $y_i = 1$ to accept S_i and $y_i = 0$ to reject it. For the algorithm's solution to be feasible, the intervals in $\{S_i \mid y_i = 0\}$ must be pairwise nonoverlapping. Given an instance $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{IR}_t}$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} y_i.$$

The set $\{S_i \mid x_i = 0\}$ is an optimal solution. □

We also consider Online Interval Rejection with Predictions (IR), where there is no bound on the number of overlaps.



$\textsc{Opt} = \textsc{Opt}'$:	0	0	1	1
$\textsc{Alg}$:	0	1	1	t
$\textsc{Alg}'$:	0	1	$\geqslant 1$	t

Figure 3: An IR_3 instance constructed from an ASG_3 instance as described in the proof of Lemma 41. For each challenge request, the pair (x_i, y'_i) is shown below S_i. Below the IR_3 instance, the cost of each challenge request and its corresponding block requests is shown for $\textsc{Opt}$, $\textsc{Opt}'$, $\textsc{Alg}$, and $\textsc{Alg}'$.

Lemma 41 For any $t \in \mathbb{Z}^+$ and any pair of insertion monotone error measures, (η_0, η_1), IR_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We prove that $\mathrm{IR}_t \geq_{\mathrm{o}} \mathrm{ASG}_t$ by giving a strict online reduction $\rho\colon \mathrm{ASG}_t \xrightarrow{r} \mathrm{IR}_t$. This reduction is essentially identical to the one from ASG_t to VC_t from the proof of Lemma 35, since the graph produced there is an interval graph, but we present it for completeness. In particular, the i'th challenge request is $S_i = ((i-1)\cdot t, i \cdot t)$, the blocks $B(0,0)$ and $B(0,1)$ are empty, the block $B(1,1)$ contains a single request to $S_i^1 = S_i$, and the block $B(1,0)$ contains t requests, S_i^j, for $j = 1, 2, \ldots, t$, where $S_i^j = ((i-1)\cdot t + j - 1, (i-1)\cdot t + j)$. See Figure 3 for an example reduction. Observe that this is an interval representation of the interval graph created in the reduction from Lemma 35, assuming that the VC_t algorithm outputs the same bits as the IR_t algorithm. The remainder of the analysis is analogous to that of Lemma 35. □

Now, we turn to showing membership by reducing to VC_t.

Lemma 42 For any $t \in \mathbb{Z}^+ \cup \{\infty\}$ and any pair of error measures, (η_0, η_1), $\mathrm{IR}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$.

Proof Let (η_0, η_1) be any pair of error measures. We define a reduction, $\rho\colon \mathrm{IR}_t \xrightarrow{r} \mathrm{VC}_t$. This, together with Theorem 23(i) and Lemma 38, shows that $\mathrm{IR}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$. Figure 4 shows an example of this reduction. We will see

that the reduction resembles a template reduction, with all blocks empty, in that $x' = x$, $\hat{x}' = \hat{x}$, and $y' = y$.

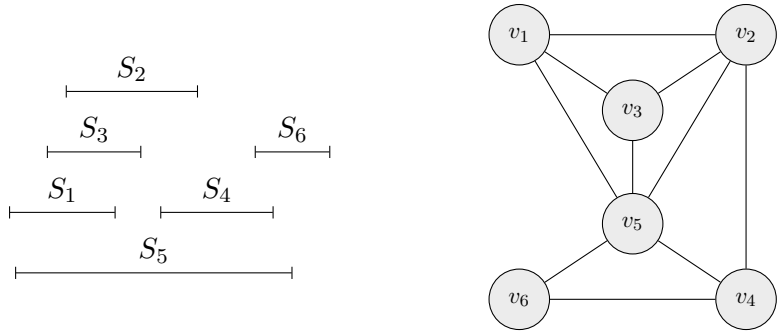


Figure 4: An example reduction from IR_t to VC_t, for $t \geqslant 5$. On the left are the intervals in the instance I of IR_t, and on the right is the graph of the instance $\rho_{\mathrm{I}}(\textsc{Alg}', I)$, for any algorithm $\textsc{Alg}' \in \mathcal{A}_{\mathrm{VC}_t}$.

Consider any algorithm, $\textsc{Alg}'$, for VC_t and any instance, $I = (x, \hat{x}, r)$, of IR_t. We define $\textsc{Alg} = \rho_{\mathrm{A}}(\textsc{Alg}')$ and $I' = (x', \hat{x}', r') = \rho_{\mathrm{I}}(\textsc{Alg}', I)$ as follows. When $\textsc{Alg}$ receives a request containing an interval, S_i, we request a vertex, v_i, together with all edges of $\{(v_j, v_i) \mid j < i \text{ and } S_j \cap S_i \neq \emptyset\}$, with prediction $\hat{x}'_i = \hat{x}_i$. Since each interval of I overlaps at most t other intervals, I' is a valid instance of VC_t. For each S_i, $\textsc{Alg}$ outputs the same as $\textsc{Alg}'$ does for v_i.

Note that the graph, G, of I' is the interval graph of the intervals of I. Thus, $\langle y_1, y_2, \ldots, y_{|r|} \rangle$ is a feasible solution to I, if and only if $\{v_i \mid y_i = 0\}$ is an independent set in G. Since the complement of an independent set is a vertex cover, $x' = x$ is an optimal solution to I' and $\textsc{Alg}(I) = \textsc{Alg}'(I')$, proving Conditions (O2) and (O1) of Definition 18. Combining $x' = x$ with $\hat{x}' = \hat{x}$, we also get that Condition (O3) is satisfied. □

Our results about IR_t and IR are summarized in Theorem 43.

Theorem 43 Consider any $t \in \mathbb{Z}^+$.

For any pair of insertion monotone error measures, (η_0, η_1),

(i) IR_t is $\mathcal{C}^t_{\eta_0,\eta_1}$-complete,

(ii) IR is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, and $\mathrm{IR} \notin \mathcal{C}^t_{\eta_0,\eta_1}$.

For any pair of error measures, (η_0, η_1),

(iii) $\mathrm{IR}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$,

(iv) $\mathrm{IR} \in \mathcal{C}_{\eta_0,\eta_1}$, and IR is not $\mathcal{C}_{\eta_0,\eta_1}$-hard.

Proof We prove each item separately.

Towards (i): This is a direct consequence of Lemmas 41 and 42.

Towards (ii): Since IR_t is a subproblem of IR, the $\mathcal{C}^t_{\eta_0,\eta_1}$-hardness of IR is a direct consequence of Corollary 24(ii) and Lemma 41.

Assume towards contradiction that $\mathrm{IR} \in \mathcal{C}^t_{\eta_0,\eta_1}$. Then $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{IR}$. On the other hand, $\mathrm{IR} \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$, so transitivity implies that $\mathrm{ASG}_t \geq_{\mathrm{o}} \mathrm{ASG}_{t+1}$, which contradicts Lemma 27(ii).

Towards (iii): This is a direct consequences of Lemma 42.

Towards (iv): By Lemma 42, $\mathrm{IR} \leq_{\mathrm{o}} \mathrm{VC}$, and by Theorem 39(viii), $\mathrm{VC} \in \mathcal{C}_{\eta_0,\eta_1}$. Hence, by transitivity, $\mathrm{IR} \in \mathcal{C}_{\eta_0,\eta_1}$.

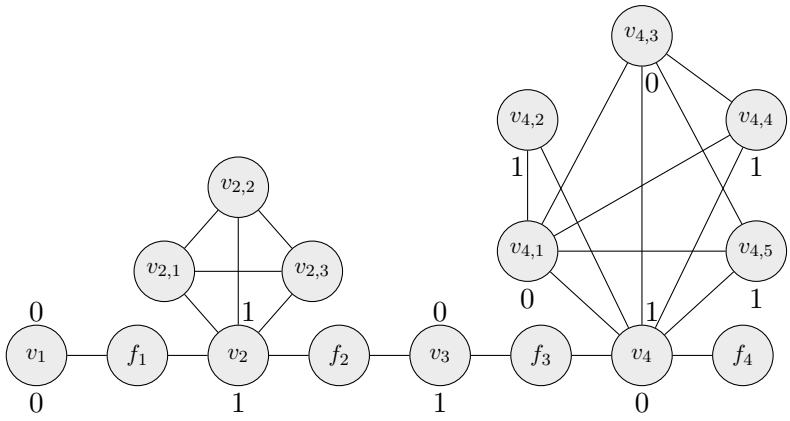
Assume towards contradiction that IR is $\mathcal{C}_{\eta_0,\eta_1}$-hard. Then, $\mathrm{IR} \geq_{\mathrm{o}} \mathrm{ASG}$. Since $\mathrm{VC} \geq_{\mathrm{o}} \mathrm{IR}$, transitivity implies that $\mathrm{VC} \geq_{\mathrm{o}} \mathrm{ASG}$, contradicting Theorem 39(viii). □

6.3 *k*-Spill

Given a graph, $G = (V, E)$, the objective of k-Spill is to select a smallest possible subset $V_1 \subseteq V$ such that the subgraph of G induced by the vertices in $V \setminus V_1$ is k-colorable.

In compiler construction [5], register allocation plays a significant role. It is often implemented by a liveness analysis, followed by a construction of an interference graph, which is then colored. The vertices represent variables (or values) and the edges represent conflicts (values that must be kept at the same point in time). The goal is to place as many of these values as possible in registers. Thus, with a fixed number of registers, this is really coloring with a fixed number of colors. Vertices that cannot be colored are referred to as *spills*. Spilled values must be stored in a more expensive manner. Thus, similar to minimizing faults in Paging, the objective is to minimize spill.

Definition 44 For *Online d-Bounded k-Spill with Predictions* (k-SPILL$_d$), the input is a graph, $G = (V, E)$, where all vertices have degree at most d. For each vertex, v_i, an algorithm, ALG, outputs $y_i = 1$ to mark v_i as a spill or $y_i = 0$ to mark it as colorable.



$\textsc{Opt} = \textsc{Opt}'$:	0	1	0	1
$\textsc{Alg}$:	0	1	1	t
$\textsc{Alg}'$:	0	$\geqslant 1$	1	t

Figure 5: A 3-$\textsc{Spill}_7$ instance constructed from an ASG_3 instance as described in the proof of Theorem 45. For each challenge request, the bits x_i and y'_i are shown above and below v_i, respectively. For the block requests, only the y'_i bits influencing the construction of the graph are shown.

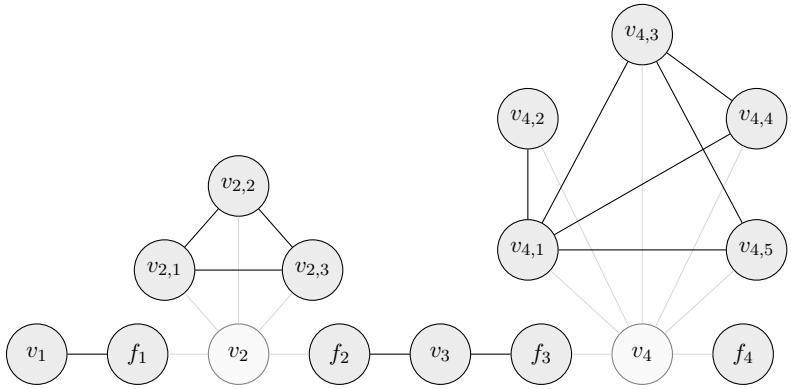


Figure 6: The graph G from Figure 5. The vertices and edges drawn in black are those of the subgraph G_0.

For the algorithm's solution to be feasible, the subgraph induced by $\{v_i \mid y_i = 0\}$ must be k-colorable. Given an instance $I = (x, \hat{x}, r) \in \mathcal{I}_{k\text{-Spill}}$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} y_i.$$

The set $\{v_i \mid x_i = 0\}$ is a maximum set of vertices inducing a k-colorable subgraph. □

We let Online k-Spill with Predictions (k-Spill) denote the version where there is no bound on the degree of vertices.

For a fixed number of colors, minimizing spill is equivalent to maximizing the number of colored vertices and this problem is NP-complete [2]. A multi-objective variant of this problem was studied both online and offline in [30]. As observed in the proof of the following theorem, 1-Spill is equivalent to VC.

Theorem 45 For all $k, t \in \mathbb{Z}^+$, and all pairs of insertion monotone error measures (η_0, η_1), k-Spill$_{t+k+1}$ is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof For any solution, y', to an instance of 1-Spill, the vertices of $\{v_i \in V \mid y'_i = 0\}$ form an independent set. Thus, $\{v_i \in V \mid y'_i = 1\}$ is a vertex cover, so 1-Spill is equivalent to VC. Hence, 1-Spill is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard by Theorem 39(iv).

For $k \geqslant 2$, we give a reduction, $\rho = (\rho_{\text{A}}, \rho_{\text{I}})$, from ASG$_t$ to k-Spill$_{t+k+1}$ with respect to (η_0, η_1), using the reduction template of Algorithm 1 from Section 5 (see Figure 5 for an example). To this end, consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{ASG}_t}$ and any $\textsc{Alg}' \in \mathcal{A}_{k\text{-Spill}_{t+k+1}}$. Let $\textsc{Alg} = \rho_{\text{A}}(\textsc{Alg}')$ and $I' = (x', \hat{x}', r') = \rho_{\text{I}}(\textsc{Alg}', I)$ be the ASG$_t$ algorithm and k-Spill$_{t+k+1}$ instance created in the reduction described below. We denote the graph of I' by $G = (V, E)$ and split V in two sets, $V_0 = \{v_i \in V \mid y_i = 0\}$ and $V_1 = \{v_i \in V \mid y_i = 1\}$.

The ith challenge request is an isolated vertex, v_i, with $\hat{x}'_i = \hat{x}_i$. Let y'_i be the output of $\textsc{Alg}'$ for v_i. Then, Alg outputs $y_i = y'_i$ on the ith request.

The ith block, $B(x_i, y'_i)$, is constructed as follows, with all prediction bits equal to 0. The last vertex, f_i, of the block is called a *final* vertex. If $i < n$, f_i is connected to v_i and v_{i+1}, and if $i = n$, f_i is connected to $v_i = v_n$.

- If $x_i = 0$, then $B(x_i, y'_i)$ contains a single vertex:
  - The final vertex, f_i.

- If $x_i = y'_i = 1$, then $B(x_i, y'_i)$ contains $k+1$ vertices:
  - For each $j = 1, 2, \ldots, k$, a vertex, $v_{i,j}$, connected to v_i and to $v_{i,l}$, $1 \leqslant l < j$.
  - The final vertex, f_i.
- If $x_i = 1$ and $y'_i = 0$, then $B(x_i, y'_i)$ contains between k and $t+k$ vertices:

  Let $j = 1$.
  Until ALG$'$ has added t new vertices to V_1 or k new vertices to V_0:[3]
  - A vertex, $v_{i,j}$, connected to v_i and to each vertex in $V_0 \cap \bigcup_{l=1}^{j-1} v_{i,l}$. Increment j.

  Until $\bigcup_j \{v_{i,j}\}$ contains a k-clique:
  - A vertex, $v_{i,j}$, connected to v_i and to $v_{i,l}$, $l < j$.
    Increment j.

  Finally:
  - The final vertex, f_i.

First, we argue that the maximum degree in the graph G is at most $t+k+1$, Since each challenge request is a neighbor of all corresponding block requests, the maximum degree is obtained by a challenge request. A challenge request, v_i, has degree at most 2, if $x_i = 0$, and at most $k+2$, if $x_i = y'_i = 1$. Now, it only remains to consider the case $x_i = 1$ and $y'_i = 0$. In the first step, where vertices are connected to v_i and to vertices from V_0 in the same block, at most $t+k-1$ vertices are added. In the next step, vertices may be added to ensure a k-clique within the block. If vertices are added in this step, they form a clique with the vertices added to V_0 in the previous step. If the last vertex of the previous step was a V_1-vertex, this vertex will also be part of the clique, so the number of V_0-vertices from that step plus the number of vertices added to ensure a k-clique will be at most $k-1$. If, on the other hand, the last vertex of the previous step was a V_0-vertex, at most $t-1$ V_1-vertices were added in that step. We conclude that the total number of nonfinal vertices in the block is at most $t+k-1$. Hence, v_i is connected to at most $t+k-1$ nonfinal block vertices, one final vertex of its own block, and

[3]The latter happens only if ALG$'$ creates an infeasible solution, incurring an infinite cost.

at most one final vertex of another block, giving a total number of neighbors of at most $t + k + 1$.

Let $X_1 = \{v_i \in V \mid x_i = 1\}$, i.e., let X_1 consist of the challenge requests for which the corresponding block contains more than just a final vertex. In the example of Figure 5, $X_1 = \{v_2, v_4\}$. Let $X_0 = V \setminus X_1$, and note that X_0 contains all block requests. Let G_0 be the subgraph of G induced by X_0. Figure 6 shows the subgraph G_0 of the example graph of Figure 5. We argue that G_0 is a largest possible k-colorable subgraph of G.

To see that G_0 is k-colorable, note that each connected component in G_0 is either a path (induced by challenge requests with $x_i = 0$ and final vertices) or is induced by (a subset of) the nonfinal vertices of a block corresponding to a challenge request with $x_i = 1$. Since the paths can be 2-colored, we focus on the nonfinal block vertices. If $y'_i = 1$, the nonfinal vertices of the block induce a k-clique. If $y'_i = 0$, each nonfinal vertex, at its arrival, is connected to each vertex, $v_{i,j} \in V_0$. Since the block ends if it contains at least k vertices in V_0, each vertex has at most $k-1$ neighbors in the component at its arrival. Thus, if the vertices of G_0 are colored in order of arrival, one of the k colors is available for each new vertex.

To see that G has no k-colorable subgraph with more vertices than G_0, note that each vertex in X_1 together with its block requests induce a subgraph containing a $(k + 1)$-clique.

Since G_0 is a largest possible k-colorable subgraph of G, we conclude that $\textsc{Opt}(I') = |X_1| = \textsc{Opt}(I)$, satisfying Condition (O2) of Definition 18. We also conclude that all block requests are correctly predicted. Since $\hat{x}'_i = \hat{x}_i$, for each challenge request, v_i, and η_0 and η_1 are insertion monotone, this implies that $\eta_0(I') \leqslant \eta_0(I)$ and $\eta_1(I') \leqslant \eta_1(I)$, satisfying Condition (O3).

It only remains to show that Condition (O1) is satisfied. To this end, we prove that, for each request, r_i in I, the cost of $\textsc{Alg}$ on r_i is no larger than the cost of $\textsc{Alg}'$ on v_i and $B(x_i, y'_i)$:

If $y'_i = 1$, $\textsc{Alg}$ guesses 1, incurring a cost of 1. $\textsc{Alg}'$ lets v_i be a spill vertex and thus incurs a cost of at least 1.

If $y'_i = x_i = 0$, $\textsc{Alg}$ correctly guesses 0, incurring a cost of 0.

If $y'_i = 0$ and $x_i = 1$, $\textsc{Alg}$ incorrectly guesses 0 and incurs a cost of t. Unless $\textsc{Alg}'$ produces an infeasible solution, $B(x_i, y'_i)$ contains t vertices of V_1, so $\textsc{Alg}'$ incurs a cost of at least t. □

6.4 2-SAT Deletion

Given a collection of variables, V, and a 2-CNF-SAT formula φ, an algorithm for 2-SAT Deletion finds a subset of clauses in φ to delete, in order to make φ satisfiable [37, 22]. Equivalently, the algorithm must assign a truth value to all variables, while minimizing the number of unsatisfied clauses in φ.

Chlebík and Chlebíkova proved the problem APX-hard [22].

We consider an online variant of 2-SAT Deletion in the context of predictions.

Definition 46 A request, r_i, for *Online 2-SAT Deletion with Predictions* (2-SatD) contains a variable, v_i, and all clauses in the formula of the form $(v_i \vee v_j)$, $(\overline{v}_i \vee v_j)$, $(v_i \vee \overline{v}_j)$, or $(\overline{v}_i \vee \overline{v}_j)$, for all $j < i$. An algorithm, Alg, outputs $y_i = 1$ to set $v_i = \texttt{true}$.

Given an instance $I = (x, \hat{x}, r)$,

$$\textsc{Alg}(I) = |\{C \in \varphi \mid C \text{ is unsatisfied}\}|.$$

The truth assignment given by $v_i = \texttt{true} \Leftrightarrow x_i = 1$ maximizes the number of satisfied clauses. □

Theorem 47 For any pair of insertion monotone error measures, (η_0, η_1), and any $t \in \mathbb{Z}^+$, 2-SatD is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We define a reduction, ρ: IR $\xrightarrow{r}$ 2-SatD. Together with Lemmas 17 and 41, this establishes the hardness of 2-SatD with respect to any pair of insertion monotone error measures, (η_0, η_1). The reduction is given in Algorithm 2 and Figure 7 shows an example of the reduction.

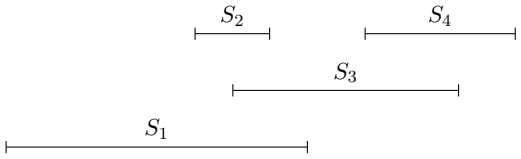


Figure 7: An example reduction from IR to 2-SatD. The above instance of IR gives rise to the CNF-formula $\varphi = (\overline{v_1} \vee \overline{v_1}) \wedge (\overline{v_2} \vee \overline{v_2}) \wedge (v_1 \vee v_2) \wedge (\overline{v_3} \vee \overline{v_3}) \wedge (v_1 \vee v_3) \wedge (v_2 \vee v_3) \wedge (\overline{v_4} \vee \overline{v_4}) \wedge (v_3 \vee v_4)$.

Clauses of the form $(\overline{v}_i \vee \overline{v}_i)$ are called *interval clauses* and clauses of the form $(v_i \vee v_j)$ are called *collision clauses*.

We first prove that, for each request, r_i, the cost of Alg on S_i is no larger than the cost of Alg′ on v_i, thus satisfying Condition (O1) of Definition 18.

Algorithm 2 Reducing IR to 2-SatD

Input: An IR instance, $I = (x, \hat{x}, r)$, and a 2-SatD algorithm, Alg$'$
Output: A 2-SatD instance, $I' = (x', \hat{x}', r')$, and an IR algorithm, Alg

1: $\varphi \leftarrow$ `true` ▷ An empty CNF-formula for Alg$'$
2: **for** each request in r, with interval S_i and prediction $\hat{x}_i$ **do**
3: $O \leftarrow \{S_j \mid j < i \text{ and } S_j \cap S_i \neq \emptyset\}$ ▷ Intervals overlapping S_i
4: Create a new variable, v_i, for I'
5: $\varphi \leftarrow \varphi \wedge (\overline{v}_i \vee \overline{v}_i) \wedge \left(\bigwedge_{S_j \in O} (v_i \vee v_j) \right)$
6: Request v_i with prediction $\hat{x}'_i = \hat{x}_i$ and updated CNF-formula φ
7: **if** O contains an accepted interval **then**
8: Output $y_i = 1$ ▷ Reject S_i
9: **else**
10: Output $y_i = y'_i$, where y'_i is the output of Alg$'$ for v_i

To this end, note that Alg incurs a cost of 1 on S_i, if it outputs $y_i = 1$ to reject the interval, and otherwise, it incurs no cost. Moreover, Alg outputs $y_i = 1$, only if at least one of the following two conditions is fulfilled.

- Alg$'$ outputs $y'_i = 1$ to set $v_i =$ `true` (see Line 10 of Algorithm 2). In this case, the interval clause $(\overline{v}_i \vee \overline{v}_i)$ is not satisfied, and Alg$'$ incurs a cost of 1.
- S_i overlaps an interval, S_j, $j < i$, which Alg accepted by outputting $y_j = 0$ (see Lines 7–8 of Algorithm 2). Note that Line 10 is the only place in Algorithm 2 where Alg may output $y_j = 0$. Thus, we must have $y'_j = 0$, and hence, $v_j =$ `false`. Therefore, if $v_i =$ `false`, the collision clause $(v_j \vee v_i)$ is not satisfied, giving Alg$'$ a cost of 1. On the other hand, if $v_i =$ `true`, the interval clause $(\overline{v}_i \vee \overline{v}_i)$ is not satisfied, and Alg$'$ incurs a cost of 1.

Next, we note that for any feasible solution, z, to I, $z' = z$ is a solution to I' satisfying all collision clauses and all interval clauses corresponding to accepted intervals. Thus, $z' = z$ is a solution to 2-SatD with the same cost as z. This shows that $\textsc{Opt}(I') \leqslant \textsc{Opt}(I)$, satisfying Condition (O2).

To prove that Condition (O3) is satisfied, we show that $x' = x$ is an optimal solution to I'. Since any solution, z', to I' satisfying all collision clauses has the same cost as the solution $z = z'$ for I, $x' = x$ is optimal among the solutions satisfying all collision clauses. Thus, we just need to show that no

solution to I' violating some collision clause has a lower cost than $x' = x$. To this end, we argue that, for any solution, z', to I', there is a solution, u', satisfying all collision clauses and with a cost no higher than that of z'. The solution u' can be constructed in the following way. As long as there is an unsatisfied collision clause, change the truth value of a variable of that clause to `true`. In this way, the collision clause becomes satisfied, no collision clause changes from `true` to `false`, and only one interval clause changes from `true` to `false`, so the cost does not increase. Since the number of satisfied collision clauses increases in each step, the process will terminate, and the resulting solution has a cost no higher than that of z'. Since $\hat{x} = \hat{x}'$, this completes the proof that Condition (O3) is satisfied □

Corollary 48 For all $t \in \mathbb{Z}^+$, and all pairs of insertion monotone error measures (η_0, η_1), 2-SatD $\notin \mathcal{C}^t_{\eta_0,\eta_1}$.

Proof Assume that 2-SatD $\in \mathcal{C}^t_{\eta_0,\eta_1}$ for some t and some pair of error measures (η_0, η_1). Then, $\text{ASG}_t \geq_\text{o}$ 2-SatD and, by Theorem 47, 2-SatD $\geq_\text{o}$ ASG_{t+1}. By transitivity, $\text{ASG}_t \geq_\text{o} \text{ASG}_{t+1}$, contradicting Lemma 27(ii). □

6.5 Dominating Set

Given a graph, $G = (V, E)$, an algorithm for Dominating Set finds a subset $V' \subseteq V$ of vertices such that for all vertices $v \in V$, $v \in V'$ or v is adjacent to a vertex $u \in V'$. The cost of the solution is given by the size of V', and the goal is to minimize this cost.

Dominating Set was one of Karp's first 21 NP-complete problems [33], though he referred to it as the set cover problem. Previous work has concluded that Dominating Set is W[2]-complete [28] and APX-hard [23]. Further, Online Dominating Set is studied in [14] and shown to be AOC-complete in [17].

We study an online variant of Dominating Set with predictions.

Definition 49 The input to *Online Dominating Set with Predictions* (Dom) is a graph, $G = (V, E)$. An algorithm, Alg, outputs $y_i = 1$ to accept v_i into its dominating set.

For the algorithm's solution to be feasible, the accepted vertices must form a dominating set, i.e., each vertex in V must be contained in $\{v_i \mid y_i = 1\}$ or have a neighbor in $\{v_i \mid y_i = 1\}$. Given an instance $I \in \mathcal{I}_\text{Dom}$,

$$\text{Alg}(I) = \sum_{i=1}^{n} y_i.$$

The set $\{v_i \mid x_i = 1\}$ is an optimal dominating set. □

6.5.1 Asymptotic Hardness

To describe our most central result on Dom, we introduce a slightly weaker notion of hardness, considering only algorithms that are $(O(1), \beta, \gamma)$-competitive:

Definition 50 Let P and Q be online problems with predictions. If any (α, β, γ)-competitive algorithm for Q, with $\alpha \in O(1)$, implies the existence of an (α, β, γ)-competitive algorithm for P, then Q is *asymptotically as hard as* P. If P is $\mathcal{C}$-hard for some complexity class, $\mathcal{C}$, and Q is asymptotically as hard as P, then Q is *asymptotically $\mathcal{C}$-hard*. □

For proving asymptotic hardness, we define a new, less restrictive type of online reduction, which motivated the definition of asymptotically as hard as:

Definition 51 An *asymptotic online reduction* is the same as a strict online reduction except that, instead of Condition (O2) from Definition 18, it satisfies the following weaker condition:

$$\textsc{Opt}_Q(I_Q) \leqslant \textsc{Opt}_P(I_P) + b, \text{ for some } b \in O(1) \qquad \text{(O2}')$$

□

We show that, for $(O(1), \beta, \gamma)$-competitive algorithms, asymptotic online reductions can be used the same way as strict online reductions:

Lemma 52 Let $\rho = (\rho_{\text{A}}, \rho_{\text{I}})$ be an asymptotic online reduction from a problem, P, with predictions and error measures (η_0, η_1) to a problem, Q, with predictions and error measures (φ_0, φ_1). Let $\textsc{Alg}_Q \in \mathcal{A}_Q$ be an (α, β, γ)-competitive algorithm for Q with respect to (φ_0, φ_1), with $\alpha \in O(1)$, and let $\textsc{Alg}_P = \rho_{\text{A}}(\textsc{Alg}_Q)$. Then, $\textsc{Alg}_P$ is an (α, β, γ)-competitive algorithm for P with respect to (η_0, η_1).

Proof Consider any instance, $I_P \in \mathcal{I}_P$, and let $I_Q = \rho_{\mathrm{I}}(\textsc{Alg}_Q, I_P)$. Then,

$$\begin{aligned}
\textsc{Alg}_P(I_P) &\leqslant \textsc{Alg}_Q(I_Q) + a, \\
&\quad \text{where } a \in O(1), \text{ by Condition (O1) of Definition 18} \\
&\leqslant \alpha \cdot \textsc{Opt}_Q(I_Q) + \beta \cdot \varphi_0(I_Q) + \gamma \cdot \varphi_1(I_Q) + \kappa + a, \\
&\quad \text{where } \kappa \in O(1), \text{ by Definition 1} \\
&\leqslant \alpha \cdot (\textsc{Opt}_P(I_P) + b) + \beta \cdot \eta_0(I_P) + \gamma \cdot \eta_1(I_P) + \kappa + a, \\
&\quad \text{where } b \in O(1), \text{ by (O2}') \text{ and (O3)} \\
&= \alpha \cdot \textsc{Opt}_P(I_P) + \beta \cdot \eta_0(I_P) + \gamma \cdot \eta_1(I_P) + \alpha \cdot b + \kappa + a, \\
&\quad \text{where } \alpha \cdot b + \kappa + a \in O(1), \text{ since } \alpha \in O(1).
\end{aligned}$$

□

6.5.2 Hardness of Dom

In this section, we give a hardness result for Dom, using a reduction analogous to a well-known reduction from Vertex Cover to Dominating Set [38] used for proving Dominating Set NP-complete.

For any graph, $G = (V, E)$, we let $\mathit{dom}(G) = (V', E')$ be the supergraph of G, where

- $V' = V \cup \{v_{i,j} \mid (v_i, v_j) \in E\}$, and
- $E' = E \cup \{(v_i, v_{i,j}), (v_{i,j}, v_j) \mid (v_i, v_j) \in E\}$.

See Figure 8 for an example of $\mathit{dom}(G)$.

Lemma 53 For any graph, G, with no isolated vertices, any optimal vertex cover for G is an optimal dominating set for $\mathit{dom}(G)$.

Proof Let $G = (V, E)$ and let $G' = (V', E')$ be $\mathit{dom}(G)$.

We first argue that any vertex cover, C, for G is a dominating set for $\mathit{dom}(G)$. Since every edge in E is covered by C and no vertex in V is isolated, each vertex in V is in C or has a neighbor in C. Moreover, since each vertex, $v_{i,j} \in V' \setminus V$ has two neighbors which are the two endpoints of an edge in E, all vertices not in $V' \setminus V$ have a neighbor in C.

Next, we argue that if C is an optimal vertex cover for G, it is an optimal dominating set for $\mathit{dom}(G)$. This follows from the fact that, for each edge $(v_i, v_j) \in E$, there is a vertex, $v_{i,j} \in V'$ that needs to be in the dominating set or have a neighbor (v_i or v_j) in the set. Including v_i or v_j in the set

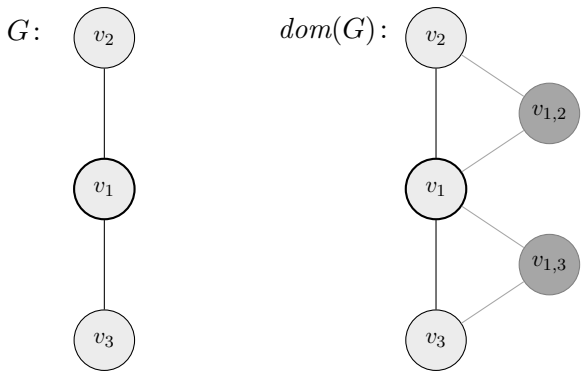


Figure 8: A graph, G, and its supergraph $dom(G)$. The vertices and edges added to G when constructing $dom(G)$ are drawn in dark gray. The vertex v_1 (drawn with a thicker boundary) constitutes an optimal vertex cover for G and an optimal dominating set for $dom(G)$.

is no worse than including $v_{i,j}$, since each of v_i and v_j dominate all three vertices, v_i, v_j, and $v_{i,j}$, and $v_{i,j}$ does not dominate any other vertices than these three. □

Note that Lemma 53 does not hold for graphs with isolated vertices, since each isolated vertex would need to be included in a dominating set, but not in a vertex cover. In an offline reduction, one would simply remove all isolated vertices from the graph, but when receiving vertices online, an algorithm will not be able to tell whether an isolated vertex will stay isolated. Thus, we give a method for "connectifying" a (possibly) disconnected graph, while controlling the size of the optimal vertex cover. This is an easy construction that has been considered before [1]. However, we have not found it in refereed works, so for completeness, we give the argument below.

For any graph, $G = (V, E)$, we let $con(G) = (V', E')$ be the connected supergraph of G, where

- $V' = V \cup \{s_1, s_2\}$, and
- $E' = E \cup \{(s_1, s_2)\} \cup \{(v, s_1) \mid v \in V\}$.

See Figure 9 for an example of $con(G)$.

Observe that given a graph G, we can create $con(G)$ online by first creating vertices s_1 and s_2, and then revealing all future vertices, v_i, as they are being revealed for G, together with the additional edge (s_1, v_i).

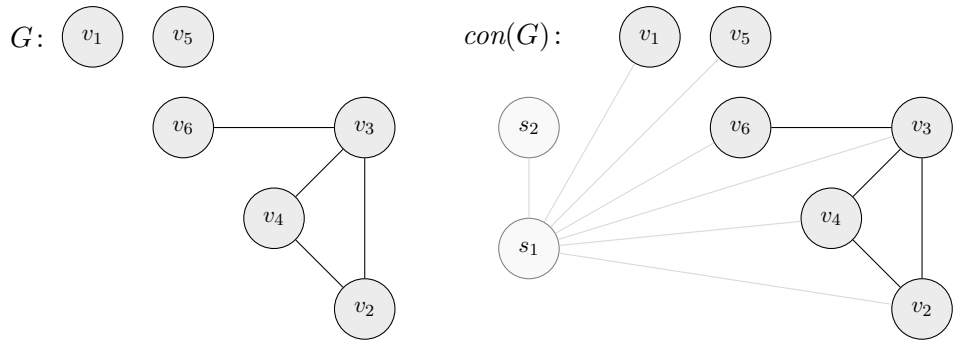


Figure 9: Connectifying a graph. The vertices and edges added to G when constructing $con(G)$ are shown in light gray.

Lemma 54 For any graph, G, if C is an optimal vertex cover for G, then $C \cup \{s_1\}$ is an optimal vertex cover for $con(G)$.

Proof If C is a vertex cover for G, then $C \cup \{s_1\}$ is a vertex cover for $con(G)$ since s_1 covers all edges added to G when creating $con(G)$. Moreover, $C \cup \{s_1\}$ is optimal since (s_1, s_2) is not covered by any vertex in G and neither s_1 nor s_2 covers any edge in G. □

Theorem 55 For any $t \in \mathbb{Z}^+$ and any pair of insertion monotone error measures, (η_0, η_1), the following hold.

(i) DOM on graphs without isolated vertices is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

(ii) DOM is asymptotically $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We give a strict and an asymptotic reduction from VC_t to DOM. In the strict reduction, the VC_t algorithm, ALG, gives $dom(G)$ to the DOM algorithm, ALG′, in an online fashion as it receives the vertices and edges of G. In the asymptotic reduction, ALG gives $dom(con(G))$ to ALG′. The asymptotic reduction is described in Algorithm 3. The part of Algorithm 3 written in black describes the strict reduction, which can be used if the VC_t graph is known to contain no isolated vertices. Figure 10 shows an example of the asymptotic reduction.

Note that for each vertex, v_i, in I, ALG accepts v_i only if ALG′ accepts one of the vertices that ALG gives to it between receiving v_i and v_{i+1}. Therefore, $\text{ALG}(I) \leqslant \text{ALG}'(I')$, satisfying Condition (O1) of Definition 18.

If the strict reduction is used, then, by Lemma 53, $\text{OPT}(I') = \text{OPT}(I)$,

Algorithm 3 Reducing VC to DOM

Input: ALG$' \in \mathcal{A}_{\text{DOM}}$ and $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{VC}}$
Output: ALG $\in \mathcal{A}_{\text{VC}}$ and $I' = (x', \hat{x}', r') \in \mathcal{I}_{\text{DOM}}$

Give ALG$'$ a vertex, s_1, with prediction 1 ▷ Part of $con(G)$
Give ALG$'$ a vertex, s_2, with prediction 0 ▷ Part of $con(G)$
Give ALG$'$ the edge (s_1, s_2) ▷ Part of $con(G)$
Give ALG$'$ a vertex, $s_{1,2}$ with prediction 0 ▷ Part of $dom(con(G))$
Give ALG$'$ the edges $(s_1, s_{1,2})$ and $(s_{1,2}, s_2)$ ▷ Part of $dom(con(G))$
for each vertex, v_i, in I **do** ▷ Step i
    Give ALG$'$ a vertex, v_i, with prediction $\hat{x}_i$ ▷ Part of G
    Give ALG$'$ the edge (s_1, v_i) ▷ Part of $con(G)$
    Give ALG$'$ a vertex, $t_{1,i}$, with prediction 0 ▷ Part of $dom(con(G))$
    Give ALG$'$ the edges $(s_1, t_{1,i})$ and $(t_{1,i}, v_i)$ ▷ Part of $dom(con(G))$
    for each edge (v_j, v_i) with $j < i$ **do**
        Give ALG$'$ the edge (v_j, v_i) ▷ Part of G
        Give ALG$'$ a vertex, $v_{j,i}$, with prediction 0 ▷ Part of $dom(G)$
        Give ALG$'$ the edges $(v_j, v_{j,i})$ and $(v_{i,j}, v_i)$ ▷ Part of $dom(G)$
    if ALG$'$ accepts at least one vertex in Step i **then**
        Accept v_i

satisfying Condition (O2). Otherwise, by Lemmas 53 and 54, $\text{OPT}(I') = \text{OPT}(I) + 1$, satisfying Condition (O2′).

For the strict reduction, Lemma 53 shows that x encodes an optimal dominating set for I'. Since each vertex in $dom(G)$ corresponding to a vertex, v_i, in G have the same prediction as v_i, and all other vertices in $dom(G)$ have prediction 0, this means that Condition (O3) is satisfied. Similarly, for the asymptotic reduction, with the addition that s_1 comes with a prediction of 1 and s_2 with a prediction of 0. Thus, for this reduction, Condition (O3) follows from Lemmas 53 and 54. □

We summarize our results about DOM in Theorem 56:

Theorem 56 Let $t \in \mathbb{Z}^+$.

For any pair of insertion monotone error measures, (η_0, η_1),

(i) DOM is asymptotically $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

For any pair of error measures, (η_0, η_1),

(ii) DOM $\in \mathcal{C}_{\eta_0,\eta_1}$ and DOM is not $\mathcal{C}_{\eta_0,\eta_1}$-hard.

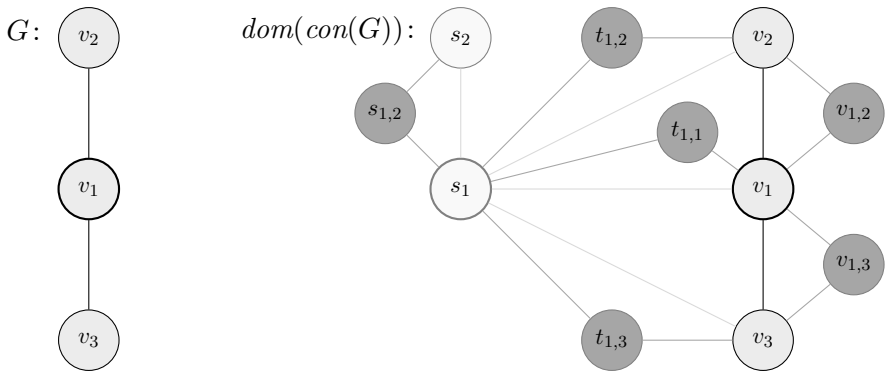


Figure 10: Example reduction for Dominating Set. The vertices and edges added to G when constructing $con(G)$ are drawn in light gray. The vertices and edges added to $con(G)$ when constructing $dom(con(G))$ are drawn in dark gray. In G, $\{v_1\}$ (drawn with a thicker boundary) is an optimal vertex cover. In $dom(con(G))$, $\{s_1, v_1\}$ (drawn with a thicker boundary) is an optimal dominating set.

Proof Item (i) follows directly from Theorem 55.

To see that $\textsc{Dom} \in \mathcal{C}_{\eta_0,\eta_1}$, adapt the setup from the proof of the first part of Theorem 39(viii) with the following addition. Observe that $\textsc{Alg}'$ creates an infeasible solution to Dom if there exists a vertex v_i such that $y_i = 0$, and $y_j = 0$, for all vertices v_j for which (v_i, v_j) is contained in the underlying graph of the instance I of Dom. Since x encodes an optimal dominating set, either $x_i = 1$, or there exists some j for which (v_j, v_i) is contained in the underlying graph of I for which $x_j = 1$, as otherwise v_i is not dominated. Hence, $\textsc{Alg} \in \mathcal{A}_{\text{ASG}}$ has guessed 0 on a true 1, and so $\textsc{Alg}(x, \hat{x}) = \infty$.

To see that Dom is not $\mathcal{C}_{\eta_0,\eta_1}$-hard, note that the algorithm, $\textsc{Acc} \in \mathcal{A}_{\textsc{Dom}}$, that accepts all vertices is $(n, 0, 0)$-competitive, as $\textsc{Opt}_{\textsc{Dom}}$ has to accept at least one vertex to create a dominating set. Now, adapt the proof of the last part of Theorem 39(viii). □

Since allowing the flexibility of the asymptotic online reduction in Definition 51 is very natural and presumably very useful, it seems natural to define complexity classes based on the asymptotically as hard as relation. This would allow the possibility of complete problems using the asymptotic online reductions. Since the relation is reflexive and transitive, the classes, $\mathcal{AC}^t_{\eta_0,\eta_1}$, can be defined exactly as the $\mathcal{C}^t_{\eta_0,\eta_1}$ classes but using the asymptotically as hard as relation. The properties corresponding to those in Theorem 23,

Corollary 24, and Observation 25 also hold for these classes, and the class $\mathcal{AC}^t_{\eta_0,\eta_1}$ contains the class $\mathcal{C}^t_{\eta_0,\eta_1}$. In addition, the classes form a hierarchy with ASG_t being complete for $\mathcal{AC}^t_{\eta_0,\eta_1}$, giving the same separation of classes as in Theorem 31.

7 Paging and $\mathcal{C}^t_{\mu_0,\mu_1}$

In Paging, we have a *universe*, U, of N pages, and a *cache* with room for $k < N$ memory pages. An instance of Paging is a sequence $r = \langle r_1, r_2, \ldots, r_n \rangle$ of requests, where each request holds a page $p \in U$. If p is not in cache (a *fault*), an algorithm has to place p in cache, either by evicting a page from cache to make room for p, or by placing p in an empty slot in the cache, if one exists. The cost of the algorithm is the number of faults. In the following, we assume that the cache is empty at the beginning of the sequence; this affects the analysis by at most an additive constant k.

There is a polynomial time optimal offline Paging algorithm, LFD, which first fills up its cache and then, when there is a page fault, always evicts the page from cache whose next request is furthest in the future [7]. Further, it is well-known that no deterministic online Paging algorithm has a competitive ratio better than k [12, 42].

We study Paging with succinct predictions [4]:

Definition 57 An instance of *Paging with Discard Predictions* (PAG_k) is a triple $I = (x, \hat{x}, r)$, where $r = \langle r_1, r_2, \ldots, r_n \rangle$ is a sequences of pages from U, and $x, \hat{x} \in \{0,1\}^n$ are two bitstrings such that

$$x_i = \begin{cases} 0, & \text{if LFD keeps } r_i \text{ in cache until it is requested again,} \\ 1, & \text{if LFD evicts } r_i \text{ before it is requested again,} \end{cases}$$

and $\hat{x}_i$ predicts the value of x_i. If the page in r_i is never requested again, then $x_i = 0$ if LFD keeps the page in cache until all requests have been seen, and $x_i = 1$ otherwise. □

In this section, we consider the pair (μ_0, μ_1) of error measures. Recall from Definition 3 that μ_0 and μ_1 count the number of wrong predictions of 0 and 1, respectively. We prove that PAG_t is contained in $\mathcal{C}^t_{\mu_0,\mu_1}$, but not $\mathcal{C}^t_{\mu_0,\mu_1}$-hard.

In [4], Antoniadis et al. prove strong lower bounds for online algorithms with predictions for PAG_t. Since $\text{PAG}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$, these bounds extend to all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems. This as well as upper bounds for PAG_t is discussed in Section 7.1.1

7.1 $\textsc{Pag}_t$ is contained in $\mathcal{C}^t_{\mu_0,\mu_1}$

In this section, we give a reduction from $\textsc{Pag}_t$ to ASG_t, using an unnamed $\textsc{Pag}_k$ algorithm from [4] that we call Flush-when-All-Zeros (FwZ). The strategy of FwZ is as follows. Each page in cache has an associated bit which is the prediction bit associated with the latest request for p. Whenever a page not in cache is requested, evict a page from cache whose associated bit is 1, if such a page exists. Otherwise, evict all pages from cache. We restate a positive result from [4] on the competitiveness of FwZ with respect to the pair of error measures (μ_0, μ_1).

Theorem 58 ([4]) For any instance, $I = (x, \hat{x}, r)$ of $\textsc{Pag}_k$,

$$\textsc{FwZ}(I) \leqslant \textsc{Opt}_{\textsc{Pag}_k}(I) + (k-1) \cdot \mu_0(I) + \mu_1(I).$$

That is, FwZ is strictly $(1, k-1, 1)$-competitive for $\textsc{Pag}_k$ with respect to (μ_0, μ_1).

We use this result from [4] to establish a relation between Paging and ASG_t:

Theorem 59 For any $t \in \mathbb{Z}^+$, $\textsc{Pag}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$.

Proof We define a strict online reduction $\rho\colon \textsc{Pag}_t \xrightarrow{r} \mathrm{ASG}_t$. For any instance, $I = (x, \hat{x}, r) \in \mathcal{I}_{\textsc{Pag}_t}$, and any algorithm, $\textsc{Alg}' \in \mathcal{A}_{\mathrm{ASG}_t}$, we construct an instance, $I' = (x', \hat{x}', r') \in \mathrm{ASG}_t$, where x' consists of x followed by t 1's, and $\hat{x}'$ consists of $\hat{x}$ followed by t 1's. In this way, $\mu_b(I) = \mu_b(I')$, for $b \in \{0, 1\}$, so Condition (O3) is satisfied.

Towards Condition (O2): Since the cache is empty from the beginning, $\textsc{Opt}_{\textsc{Pag}_t}$ incurs a cost of 1 on each of the first t requests. After this, $\textsc{Opt}_{\textsc{Pag}_t}$ incurs a cost of 1 for each request with true bit 1, by the definition of discard predictions. Thus,

$$\textsc{Opt}_{\textsc{Pag}_t}(I) = t + \sum_{i=1}^{n} x_i. \tag{6}$$

Moreover, by definition of I', we have that $\textsc{Opt}_{\mathrm{ASG}_t}(I') = t + \sum_{i=1}^{n} x_i = \textsc{Opt}_{\textsc{Pag}_t}(I)$.

Towards Condition (O1): By definition of I', and letting y' be the output of $\textsc{Alg}'(I')$,

$$\textsc{Alg}'(I') \geqslant \sum_{i=1}^{n} \left(y'_i + t \cdot x_i \cdot (1 - y'_i)\right) + t. \tag{7}$$

In particular, $\sum_{i=1}^{n} \left(y'_i + t \cdot x_i \cdot (1 - y'_i)\right)$ is the cost of $\textsc{Alg}'$ on the first n requests, and t is a lower bound on the cost of $\textsc{Alg}'$ on the last t requests.

We give pseudo-code for ρ given $\textsc{Alg}' \in \mathcal{A}_{\mathrm{ASG}_t}$ and $I \in \mathcal{I}_{\mathrm{PAG}_t}$ in Algorithm 4. Formally, $\textsc{Alg} = \rho_{\mathrm{A}}(\textsc{Alg}')$ runs the algorithm FwZ from [4] using y'_i as the prediction for r_i. Hence, letting $y' = \langle y'_1, y'_2, \ldots, y'_n \rangle$ and $I_{y'} = (x, y', r)$, we have that $\textsc{Alg}(I) = \mathrm{FwZ}(I_{y'})$. Therefore,

$$\begin{aligned}
\textsc{Alg}(I) &\leqslant \textsc{Opt}_{\mathrm{PAG}_t}(I_{y'}) + (t-1) \cdot \mu_0(I_{y'}) + \mu_1(I_{y'}), \text{ by Theorem 58} \\
&= t + \sum_{i=1}^{n} \left(x_i + (t-1) \cdot (1 - y'_i) \cdot x_i + y'_i \cdot (1 - x_i)\right), \text{ by (6)} \\
&= t + \sum_{i=1}^{n} \left(t \cdot x_i \cdot (1 - y'_i) + y'_i\right) \\
&\leqslant \textsc{Alg}'(I'), \text{ by (7)}.
\end{aligned}$$

□

Algorithm 4 Reducing PAG_t to ASG_t

Input: An algorithm $\textsc{Alg}' \in \mathcal{A}_{\mathrm{ASG}_t}$ and an instance $(x, \hat{x}, r) \in \mathcal{I}_{\mathrm{PAG}_t}$
Output: An ASG_t instance $I' = (x', \hat{x}', r')$ and a PAG_t algorithm, $\textsc{Alg}$

```
for each request r_i do
    Get prediction x̂_i
    Ask Alg' to guess the next bit given x̂'_i = x̂_i, and let y'_i be its output
    if r_i is a fault then
        if there is a page, p, in cache with associated bit 1 then
            Evict p
        else
            Flush the cache
        Place the page from r_i in cache
    Set the associated bit of r_i to y'_i
for i ← 1 to t do
    Ask Alg' to guess the next bit given x̂'_{n+i} = 1
Compute x ← LFD(r) and reveal x' which is x appended by t 1s to Alg
```

7.1.1 Establishing Bounds with Respect to (μ_0, μ_1)

Now that we have proven that $\text{PAG}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$, we can extend a collection of lower bounds from Paging with Discard Predictions by Antoniadis et al. [4] to all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems:

Theorem 60 Let $t \in \mathbb{Z}^+$, and let P be any $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problem. Then, for any (α, β, γ)-competitive algorithm for P with respect to (μ_0, μ_1),

(i) $\alpha + \beta \geqslant t$ and

(ii) $\alpha + (t-1) \cdot \gamma \geqslant t$.

Proof By Theorem 59, $\text{ASG}_t \geq_\text{o} \text{PAG}_t$. Since P is $\mathcal{C}^t_{\mu_0,\mu_1}$-hard, we have that $P \geq_\text{o} \text{ASG}_t$. Hence, by transitivity of the as-hard-as relation, $P \geq_\text{o} \text{PAG}_t$.

For (i), we assume for the sake of contradiction that there is an (α, β, γ)-competitive algorithm, $\text{ALG} \in \mathcal{A}_P$, with $\alpha + \beta < t$. Since $P \geq_\text{o} \text{PAG}_t$, there exists an (α, β, γ)-competitive algorithm, $\text{ALG}'_{\text{PAG}_t}$, for PAG_t with $\alpha + \beta < t$. This contradicts Theorem 1.7 from [4].

Similarly, assuming an (α, β, γ)-competitive algorithm, $\text{ALG} \in \mathcal{A}_P$, with $\alpha + (t-1) \cdot \gamma < t$, we obtain a contradiction with Theorem 1.7 from [4] again, thus proving (ii). □

Finally, we re-prove an existing positive result for PAG_t (see Remark 3.2 in [4]) but now using that $\text{PAG}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$:

Theorem 61 For any $\alpha, \beta \in \mathbb{R}^+$ with $\alpha \geqslant 1$ and $\alpha + \beta \geqslant t$, there exists an $(\alpha, \beta, 1)$-competitive algorithm for PAG_t with respect to (μ_0, μ_1).

Proof By Theorem 59, any (α, β, γ)-competitive algorithm for ASG_t implies an (α, β, γ)-competitive algorithm for PAG_t. Thus it is sufficient that, by Theorem 9, there exists an $(\alpha, \beta, 1)$-competitive algorithm for ASG_t with respect to (μ_0, μ_1), for any $\alpha, \beta \in \mathbb{R}^+$ with $\alpha \geqslant 1$ and $\alpha + \beta \geqslant t$. □

7.2 PAG_t is not $\mathcal{C}^t_{\mu_0,\mu_1}$-hard

Finally, we show that PAG_t is not $\mathcal{C}^t_{\mu_0,\mu_1}$-complete for $t \geqslant 5$. Since there is much more information available to an algorithm for PAG_t, as opposed to ASG_t, intuitively, it seems that PAG_t should not be as hard as ASG_t, so this is the expected result. We prove this by showing that the lower bound on ASG_t in Lemma 12 does not hold for PAG_t. That lower bound shows that for any (α, β, γ)-competitive algorithm for ASG_t with respect to (μ_0, μ_1), if $\alpha < t$, then $\gamma \geqslant 1$. By Observation 22, we know that this also applies to

any $\mathcal{C}^t_{\mu_0,\mu_1}$-complete problem. We provide a PAG$_t$ algorithm, FBB, that is $(t-\delta, 2t, 1-\varepsilon)$-competitive for some positive δ and ε.

In this section, we say that a page has a prediction of 0 (1), if the latest request to the page had a prediction of 0 (1).

FLUSH-BETWEEN-BLOCKS (FBB) defines consecutive blocks of the input sequence. A block starts at the beginning of the request sequence or with the next request after the previous block ends. Except for the last block, all blocks are complete. When a block ends, the cache is flushed. When evicting a page, FBB considers only pages with a prediction of 1 that have not already been evicted within the current block. Among these pages, it chooses the page that has been in cache longest. A block ends with a request causing a fault when the cache is full and one of the following two conditions hold.

Condition 1 All pages in cache have prediction 0.

Condition 2 Some page has prediction 1 and there are no pages with prediction 1 that have not already been evicted in the block.

As a response to the last request of a block, an arbitrary page is evicted. This decision is not important, since the cache is flushed at the beginning of the next block.

Proposition 62 If a block ends due to Condition 1, the block contains at least one incorrect prediction of 0.

Proof If a block ends due to Condition 1, then just before the last request of the block, the cache is filled with pages whose last prediction was 0. Thus, according to the predictions, LFD has all of them in cache and will keep them there until their next request. This contradicts the fact that the last request of the block is to a page not in cache, so at least one of the predictions must be incorrect. □

The following lemma holds for any block, including the last, possibly incomplete, block.

Lemma 63 For $t \geqslant 3$ and any block, I_b,

$$\text{FBB}(I_b) \leqslant \left(t - \frac{1}{t}\right) \text{LFD}(I_b) + 2t.$$

Proof In any block, FBB only evicts pages that have prediction 1 and have not been evicted earlier in the block, possibly except for the page, p, evicted

as a result of the last request of the block. Even if p is evicted twice, there are at most two faults on it within the block, since the block contains no requests after the second eviction of p. Thus, if there are s distinct pages in I_b, FBB faults at most $2s$ times on I_b. LFD faults at least $s - t$ times. If $s - t \leqslant 0$, then FBB has at most t faults. Otherwise, FBB faults at most $2s$ times and $2s = 2(s-t) + 2t \leqslant 2\text{LFD}(I_b) + 2t < (t - \frac{1}{t})\text{LFD}(I_b) + 2t$, since $t \geqslant 3$. □

Lemma 64 For any complete block, I_b, with no incorrect 0-predictions,

$$\text{LFD}(I_b) \geqslant 2.$$

Proof Let s be the number of distinct pages in the block. Since I_b is a complete block, $s > t$. If $s \geqslant t + 2$, then there are at least two pages among the ones requested in the block that LFD does not initially have in cache, so $\text{LFD}(I_b) \geqslant 2$.

Thus, only the case $s = t + 1$ remains. In this case, we let P denote the $t+1$ pages requested within the block I_b. During the first t faults in I_b, FBB is filling up the cache again after flushing. For each of the remaining requests within the block, exactly one of the $t + 1$ pages is outside the cache. We argue that I_b can be split into two subblocks, I'_b followed by I''_b, such that each subblock contains requests to all $t + 1$ distinct pages in P.

Let r be the first request within the block causing an eviction. At the time r arrives, the cache has been filled and r is a request to the one page in P which is not in cache. Thus, the part of I_b ending with r contains requests to $t + 1$ distinct pages, so we let I'_b denote this part of I_b.

What remains is to prove that the remaining part, I''_b, of I_b also contains requests to all $t+1$ pages in P. Note that none of the pages that are in cache at the beginning of I''_b have been evicted from cache within I_b. We partition P into two subsets, P_0 and P_1, based on the situation at the beginning of I''_b. The set P_0 consists of the pages whose last request came with a 0-prediction and $P_1 = P \backslash P_0$ consists of those whose last request came with a 1-prediction.

Note that by the definition of FBB, the last request, r', of the block is a fault and when r' arrives, all pages in cache with a prediction of 1 have been evicted within I_b. Therefore, all pages in P_1 that are in cache at the arrival of r' have been requested with a 0-prediction or brought into cache within I''_b. If one of the pages in P_1 is the page not in cache, it is the page requested by r'. Thus, all pages in P_1 are requested within I''_b. We now argue that all pages in P_0 must also be requested within I''_b. Since there are no incorrect

0-predictions in I_b and since any paging algorithm, including LFD, can have at most t pages in cache at a time, each page in P_0 must be requested before the last page in P_1 has been requested.

Since I'_b and I''_b each contain requests to $t+1$ distinct pages, LFD must fault at least once in each of the two subblocks. □

Lemma 65 For $t \geqslant 5$ and any complete block, I_b, with no incorrect 0-predictions,

$$\text{FBB}(I_b) \leqslant \left(t - \frac{1}{3t^2}\right) \text{LFD}(I_b) + \left(1 - \frac{1}{3t^2}\right) \mu_1(I_b).$$

Proof Let s be the number of distinct pages in the block. Since I_b is a complete block, $s > t$. By Proposition 62, the block I_b ended due to Condition 2.

Let d be the number of pages FBB faults on twice in the block. Note that for each such page, p, the last request to p before the last fault on p in the block has a prediction of 1. Among the d pages that FBB faults on twice, let d_c (d_w) be the number of pages, p, where the last request to p before the last fault on p had a correct (incorrect) 1-prediction, and note that $d = d_\text{c} + d_\text{w}$.

Thus, FBB faults at most $s + d$ times in the block, and LFD faults at least $s - t$ times. An independent lower bound on LFD is that it faults at least d_c times. Thus, FBB faults at most $s + d_\text{w} + d_\text{c} \leqslant 2\text{LFD}(I_b) + t + d_\text{w}$ times. We now show that for $t \geqslant 5$ and $\varepsilon = \frac{1}{3t^2}$,

$$2\text{LFD}(I_b) + t + d_\text{w} \leqslant (t - \varepsilon)\text{LFD}(I_b) + (1 - \varepsilon)\mu_1(I_b), \tag{8}$$

which will conclude our proof.

We start by establishing

$$t + \varepsilon d_\text{w} \leqslant (t - 2 - \varepsilon)\text{LFD}(I_b). \tag{9}$$

The easier case is $s = t + 1$:

$$\begin{aligned} t + \varepsilon d_\text{w} &\leqslant t + \varepsilon(t+1) \\ &\leqslant 2t - 4 - 2\varepsilon, \text{ for } t \geqslant 5 \text{ and } \varepsilon = \tfrac{1}{3t^2} \\ &\leqslant (t - 2 - \varepsilon)\text{LFD}(I_b), \text{ since, by Lemma 64, } \text{LFD}(I_b) \geqslant 2 \end{aligned}$$

So, now we assume that $s \geqslant t + 2$:

For $t \geqslant 5$ and $\varepsilon = \frac{1}{3t^2}$,

$$2 + \varepsilon t + 2\varepsilon \leqslant (t - 2 - 2\varepsilon)(s - t - 1),$$

since for $s \geqslant t + 2$, the right-hand side is greater than 2.5 and the left-hand side is smaller than 2.5. This is equivalent to

$$t + \varepsilon(s - t - 1 + t + 1) \leqslant (t - 2 - \varepsilon) + (t - 2 - \varepsilon)(s - t - 1). \tag{10}$$

Now,

$$\begin{aligned} t + \varepsilon d_{\mathrm{w}} &\leqslant t + \varepsilon(s - t - 1 + t + 1), \text{ since } d_{\mathrm{w}} \leqslant s \\ &\leqslant (t - 2 - \varepsilon) + (t - 2 - \varepsilon)(s - t - 1), \text{ by (10)} \\ &\leqslant (t - 2 - \varepsilon) + (t - 2 - \varepsilon)(\mathrm{LFD}(I_b) - 1), \text{ since } \mathrm{LFD}(I_b) \geqslant s - t \\ &= (t - 2 - \varepsilon)\mathrm{LFD}(I_b) \end{aligned}$$

Having established (9) for all cases, and adding $2\mathrm{LFD}(I_b) + (1 - \varepsilon)d_{\mathrm{w}}$ to both sides, we get that

$$\begin{aligned} 2\mathrm{LFD}(I_b) + t + d_{\mathrm{w}} &\leqslant 2\mathrm{LFD}(I_b) + (t - 2 - \varepsilon)\mathrm{LFD}(I_b) + (1 - \varepsilon)d_{\mathrm{w}} \\ &\leqslant (t - \varepsilon)\mathrm{LFD}(I_b) + (1 - \varepsilon)\mu_1(I_b), \text{ since } \mu_1(I_b) \geqslant d_{\mathrm{w}} \end{aligned}$$

This establishes (8) and concludes the proof. □

We can now establish that $\textsc{Pag}_t$ is not complete for the class $\mathcal{C}^t_{\mu_0,\mu_1}$.

Theorem 66 For $t \geqslant 5$, the $\textsc{Pag}_t$ algorithm, FBB, is $(t - \frac{1}{3t^2}, 2t, 1 - \frac{1}{3t^2})$-competitive.

Proof We consider the different types of blocks and show that they are each bounded by

$$\left(t - \frac{1}{3t^2}\right)\mathrm{LFD}(I_b) + 2t\mu_0(I_b) + \left(1 - \frac{1}{3t^2}\right)\mu_1(I_b), \tag{11}$$

except a possible last block, where we need an additive term of $2t$. Thus, we can sum up the faults over all blocks to get the result.

If block I_b ends due to Condition 1 where all pages in cache have prediction 0, then there have been at least $t + 1$ different pages requested in the block, and LFD cannot have them all in cache. Thus, there was an incorrect 0-prediction and $\mu_0(I_b) \geqslant 1$. Using Lemma 63, the number of faults by FBB in I_b is bounded by

$$\left(t - \frac{1}{t}\right)\mathrm{LFD}(I_b) + 2t \leqslant \left(t - \frac{1}{3t^2}\right)\mathrm{LFD}(I_b) + 2t\mu_0(I_b).$$

If block I_b ends due to Condition 2, then, by Lemma 65, the number of faults by FBB in I_b is bounded by

$$\left(t - \frac{1}{3t^2}\right) \mathrm{LFD}(I_b) + \left(1 - \frac{1}{3t^2}\right) \mu_1(I_b).$$

For the possible final non-ending block, by Lemma 63, the number of faults by FBB in I_b is bounded by

$$\left(t - \frac{1}{t}\right) \mathrm{LFD}(I_b) + 2t \leqslant \left(t - \frac{1}{3t^2}\right) \mathrm{LFD}(I_b) + 2t.$$

We have seen that for any complete block, the number of faults is bounded by (11). Thus, they can be summed up to give the result, using the $2t$ term from the last block as an additive constant. □

Corollary 67 $\textsc{Pag}_t$ is not $\mathcal{C}^t_{\mu_0,\mu_1}$-hard.

Proof Since ASG_t is complete for the class, it follows by Lemma 12 that any algorithm for a $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problem that has $\alpha < t$, must have $\gamma \geqslant 1$. By Theorem 66, FBB does not fulfill this, so $\textsc{Pag}_t$ is not $\mathcal{C}^t_{\mu_0,\mu_1}$-hard. □

8 Concluding Remarks and Future Work

We have defined complexity classes for online minimization problems with binary predictions and without predictions and proven that they form a strict hierarchy. Further, we showed that our complexity classes have all the structure one expects from complexity classes. We proved membership, hardness, and completeness of multiple problems with respect to various pairs of error measures, using our reduction template as well as other methods. For instance, we have shown completeness of Online t-Bounded Degree Vertex Cover and Online t-Bounded Overlap Interval Rejection. Beyond this, we showed strong lower bounds for all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems, using similar lower bounds from [4] and a reduction from Paging with Discard Predictions to Asymmetric String Guessing with Predictions. We also proved Paging with Discard Predictions to not be $\mathcal{C}^t_{\mu_0,\mu_1}$-hard.

These results were defined and proven using the definition of asymptotic competitivenes that allows for an additive constant. In online algorithms, researchers generally use two slightly different definitions of the asymptotic competitive ratio; either using the style from our paper or defining the ratio as the limit of the ratio of the algorithm's performance to OPT's performance.

The two definitions are equivalent only if an additive $o(\textsc{Opt})$ term is allowed. Since incorporating this more general term adds a significant amount of technicalities to the proofs, we have favored readability and just allow an additive constant. The difference is minor in that if an algorithm is c-competitive with regards to the limit-based definition, it is $(c+\varepsilon)$-competitive in the restricted version for any $\varepsilon > 0$, so taking the infimum over all values results in the same competitive ratios. The results in this paper also hold using a definition of competitivenes that allows for an additive $o(\textsc{Opt})$ term.

Note that our definition of relative hardness also applies to maximization problems. In a very recent follow-up to the arXiv version of our paper, online maximization problems with predictions were considered and several maximization problems shown to be members, hard, and complete for $\mathcal{C}^t_{\mu_0,\mu_1}$ [8].

The most interesting open problem is extending our complexity theory to other kinds of predictions. Placing other online problems with or without predictions into the complexity hierarchy, including determining whether or not k-$\textsc{Spill}_t$ is contained in any of the complexity classes defined, would also be interesting. Other possible directions for future work include considering randomized algorithms as well as other performance measures, for instance using relative worst order analysis [15, 19, 20] or random order analysis [34] instead of competitive analysis.

References

[1] Reduction from vertex cover to dominating set. `https://cs.stackexchange.com/questions/117567/reduction-from-vertex-cover-to-dominating-set`. Accessed: 2025-09-10.

[2] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Boston, MA, USA, 1974.

[3] Algorithms with predictions. `https://algorithms-with-predictions.github.io/`. Accessed: 2026-01-27.

[4] Antonios Antoniadis, Joan Boyar, Marek Eliás, Lene M. Favrholdt, Ruben Hoeksma, Kim S. Larsen, Adam Polak, and Bertrand Simon. Paging with succinct predictions. In *40th International Conference on Machine Learning (ICML)*, volume 202, pages 952–968, Philadelphia, PA, USA, 2023. PMLR.

[5] Andrew W. Appel. *Modern Compiler Implementation in C*. Cambridge University Press, New York, NY, USA, 1998. Reprinted with corrections, 1999; reissued, 2004.

[6] Giorgio Ausiello, Marco Protasi, Alberto Marchetti-Spaccamela, Giorgio Gambosi, Pierluigi Crescenzi, and Viggo Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Heidelberg, Germany, 1999.

[7] Laszlo A. Belady. A study of replacement algorithms for virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[8] Magnus Berg. Comparing the hardness of online minimization and maximization problems with predictions. In *Frontiers of Algorithmics (IJTCS-FAW)*, volume 15828 of *Lecture Notes in Computer Science (LNCS)*, pages 33–48, Heidelberg, Germany, 2025. Springer.

[9] Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Sacha Krug, Jasmin Smula, and Andreas Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. *Theoretical Computer Science*, 554:95–108, 2014.

[10] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. Online algorithms with advice: The tape model. *Information and Computation*, 254(1):59–83, 2017.

[11] Allan Borodin, Joan Boyar, Kim S. Larsen, and Denis Pankratov. Advice complexity of priority algorithms. *Theory of Computing Systems*, 64:593–625, 2020.

[12] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA, 1998.

[13] Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.

[14] Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholdt, Michal Kotrbčík, and Kim S. Larsen. Online dominating set. *Algorithmica*, 81(5):1938–1964, 2019.

[15] Joan Boyar and Lene M. Favrholdt. The relative worst order ratio for on-line algorithms. *ACM Transactions on Algorithms*, 3(2):article 22, (24 pages), 2007.

[16] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online Algorithms with Advice: A Survey. *ACM Computing Surveys*, 50(2):19 (34 pages), 2017.

[17] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. The advice complexity of a class of hard online problems. *Theory of Computing Systems*, 61:1128–1177, 2017.

[18] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. Weighted online problems with advice. *Theory of Computing Systems*, 62:1443–1469, 2018.

[19] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The Relative Worst Order Ratio Applied to Paging. *Journal of Computer and System Sciences*, 73(5):818–843, 2007.

[20] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Relative Worst-Order Analysis: A Survey. *ACM Computing Surveys*, 54(1):1–21, 2021. Article No. 8.

[21] Joan Boyar, Kim S. Larsen, and Denis Pankratov. Advice Complexity of Adaptive Priority Algorithms. *Theoretical Computer Science*, 984:114318 (31 pages), 2024.

[22] Miroslav Chlebík and Janka Chlebíkova. On approximation hardness of the minimum 2SAT-DELETION problem. *Discrete Applied Mathematics*, 155(2):172–179, 2007.

[23] Mirela Damian and Sriram V. Pemmaraju. APX-hardness of domination problems in circle graphs. *Information Processing Letters*, 97(6):231–237, 2006.

[24] Marc Demange and Vangelis Th. Paschos. On-line vertex-covering. *Theoretical Computer Science*, 332(1–3):83–108, 2005.

[25] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[26] Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *RAIRO – Theoretical Informatics and Applications*, 43(3):585–613, 2009.

[27] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, Heidelberg, Germany, 1999.

[28] Rodney G. Downey and Micheal R. Fellows. Fixed-parameter tractibility and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[29] Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.

[30] Leah Epstein, Asaf Levin, and Gerhard Woeginger. Graph coloring with rejection. *Journal of Computer and System Sciences*, 77(2):439–447, 2011.

[31] Monika Henzinger, Barna Saha, Martin P. Seybold, and Christopher Ye. On the complexity of algorithms with predictions for dynamic graph problems. In *15th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:25, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[32] Juraj Hromkovič, Rastislav Královič, and Richard Královič. Information complexity of online problems. In *35th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 6281 of *Lecture Notes in Computer Science (LNCS)*, pages 24–36, Heidelberg, Germany, 2010. Springer.

[33] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, Heidelberg, Germany, 1972. Plenum Press.

[34] Claire Kenyon. Best-fit bin-packing with random order. In *7th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–364, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[35] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing, Boston, MA, USA, 2005.

[36] Dennis Komm. *An Introduction to Online Computation: Determinism, Randomization, Advice*. Texts in Theoretical Computer Science. Springer, Switzerland, 2016.

[37] Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999.

[38] Udi Manber. *Introduction to Algorithms – A Creative Approach*. Addison-Wesley Longman Publishing, Boston, MA, USA, 1989.

[39] Jesper W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Full paper in arXiv:1511.05886 [cs.DS].

[40] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[41] Carla Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14(5):233–235, 1982.

[42] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.