Complexity Classes for Online Problems with and without Predictions*

Magnus Berg Joan Boyar
Lene M. Favrholdt Kim S. Larsen

University of Southern Denmark

{magbp,joan,lenem,kslarsen}@imada.sdu.dk

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, focusing on binary predictions for minimization problems. Based on the most generic hard online problem 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 predictions apply directly to all hard problems for each class in the hierarchy based on the canonical pair of error measures.

*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

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 is 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. As a starting point, we consider minimization problems with binary predictions. 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 [36] (see [12, 29]), 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 [1]. 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 [25]. 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 *asymmetric string guessing* [14, 32, 33], a generic hard online problem, where each request is simply a prompt for the algorithm to guess a bit. String guessing has played a fundamental rôle in what is often referred to as advice complexity [10, 19, 22, 26], where online algorithms have access to oracle-produced information about the instance which, in our context, can be considered infallible predictions. 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 wrong guesses of 0.

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 [34], the W-hierarchy [20], 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 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, 22], with different objectives. The closest related work is in [14], 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 the framework enabling a complexity theory for online algorithms with predictions. Using this framework, we prove hardness and class membership results for several problems, including showing 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 is, for algorithms without predictions.

2 Preliminaries

In this paper, we consider online problems with predictions, where algorithms have to make an irrevocable decision for each request, by outputting a bit.[1] For any such problem, P, we let $\mathcal{I}_P$ be the collection of instances of P, and we let 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 $r = \langle r_1, r_2, \dots, r_n \rangle$ for P. The bitstring x is an encoding of 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, 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 OPT_P.

[1]The only exception to this is Paging with Discard Predictions, where the output is a page (see Section 7).

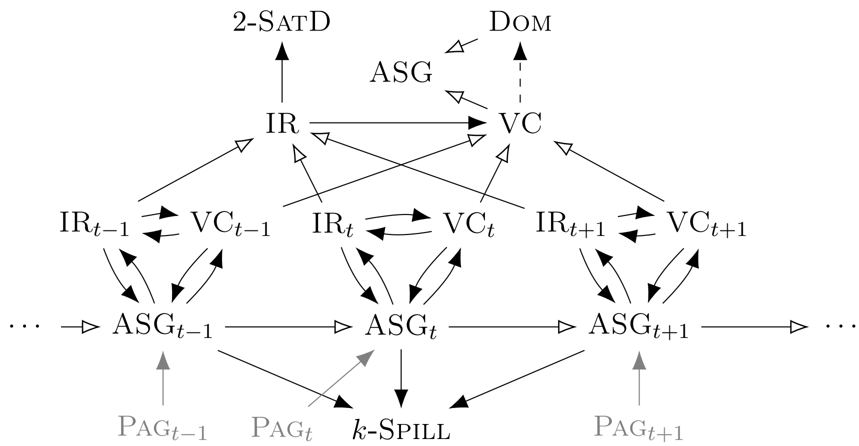

Figure 1: A hardness graph based on our complexity hierarchy. The problems shown are defined in Definitions 3, 28, 41, 32, 34, 38, and 47. Given two problems P and Q, we write $P \to Q$ to indicate that Q is as hard as P (see Definition 11). If the arrowhead is only outlined, P is not as hard as Q. If the arrow is dashed, P is as hard as Q in the weak sense (see Definition 44). 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 2), and the remaining arrows hold with respect to all pairs of *insertion monotone* error measures (see Definition 23).

If P is a graph problem, we consider 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.

Given an algorithm, Alg, and an instance, $I \in \mathcal{I}_P$, we let $\text{Alg}[I]$ be Alg's solution to I, and $\text{Alg}(I)$ be the cost of $\text{Alg}[I]$. Further, given a map $\kappa \colon \mathcal{I}_P \to \mathbb{R}$, we say that κ is *sublinear in* Opt, or $\kappa \in o(\text{Opt})$, if

$$\forall \delta > 0 \colon \exists b_\delta \colon \forall I \in \mathcal{I}_P \colon \kappa(I) < \delta \cdot \text{Opt}(I) + b_\delta. \tag{1}$$

In Lemma 53 in Appendix A, we show that if $\kappa, \kappa' \in o(\text{Opt})$ and $k \geqslant 0$, then $\kappa + \kappa' \in o(\text{Opt})$ and $k \cdot \kappa \in o(\text{Opt})$, where $(\kappa + \kappa')(I) = \kappa(I) + \kappa'(I)$ and $(k \cdot \kappa)(I) = k \cdot \kappa(I)$.

We use the following definition of competitiveness: An algorithm, Alg, for an online minimization problem without predictions, P, is *c-competitive* if there exists a map $\kappa \in o(\textsc{Opt}_P)$, called the *additive term*, such that for all instances $I = (x, r) \in \mathcal{I}_P$,

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

This is similar to what is often referred to as the asymptotic competitive ratio, where one usually requires that $\limsup_{|I|\to\infty} \frac{\textsc{Alg}(I)}{\textsc{Opt}(I)} \leqslant c$. Our formulation is analogous to the definition in [12, 29], except that they require the additive term to be a constant. Observe that for any constant, $k \in \mathbb{R}$, $k \in o(\textsc{Opt})$.

Next, 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[2], η_0 and η_1, where η_b is a function of the incorrectly predicted bits, where the prediction is b. We assume that $0 \leqslant \eta_b(I) < \infty$, for all instances I.

Definition 1 Let (η_0, η_1) be a pair of error measures, let P be an online minimization problem with predictions, and let $\textsc{Alg}$ be a deterministic online algorithm for P. If there exist three maps $\alpha, \beta, \gamma \colon \mathcal{I}_P \to [0, \infty)$ and a map $\kappa \in o(\textsc{Opt})$ such that for all $I \in \mathcal{I}_P$,

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

then $\textsc{Alg}$ is (α, β, γ)*-competitive with respect to* (η_0, η_1). When (η_0, η_1) is clear from the context, we write that $\textsc{Alg}$ is (α, β, γ)-competitive, and when $\kappa(I) \leqslant 0$, for all $I \in \mathcal{I}_P$, then $\textsc{Alg}$ is *strictly* (α, β, γ)*-competitive.* □

Observe that the above definition does not require α, β, and γ to be constants, though it is desirable that they are, especially α which is the consistency. In particular, α, β, and γ are allowed to be functions of the instance, $I \in \mathcal{I}_P$, for example of n. In our notation, α, β, and γ's dependency on I is, however, kept implicit. Further, observe that any α-competitive algorithm without predictions is $(\alpha, 0, 0)$-competitive with respect to any pair of error measures (η_0, η_1).

We define a pair of error measures that is equivalent to the pair of error measures from [4], where μ_b is the number of wrong predictions of $b \in \{0, 1\}$.

[2] 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, Theorem 6 implies that $\textsc{FtP}$ is $(1, 1)$-competitive with respect to $\mathrm{Ham}_{t-1,1}(x, \hat{x}) = (t-1) \cdot \mu_0(x, \hat{x}) + \mu_1(x, \hat{x})$. However, combining the two error measures into one, we lose some detail such as the tradeoffs between α, β, and γ given in Theorem 51.

Definition 2 For any instance $I = (x, \hat{x}, r)$, (μ_0, μ_1) is given by

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

□

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 [14].

Definition 3 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, 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 $t \in \mathbb{Z}^+$,

$$\textsc{Alg}(I) = \sum_{i=1}^{n} \left(y_i + t \cdot x_i \cdot (1 - y_i) \right), \tag{2}$$

where y_i is $\textsc{Alg}$'s i'th guess. When $t = \infty$, we abbreviate ASG_t by ASG and rewrite the objective function as:

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

□

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 the competitiveness of online algorithms for ASG_t, with and without predictions. The following observation is well-known:

Observation 4 ([14]) For any algorithm, $\textsc{Alg}$, for ASG without predictions, there is no function, f, such that $\textsc{Alg}$ 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 5 Let $t \in \mathbb{Z}^+$ and let $0 < \varepsilon < 1$. Then, for ASG_t, the following hold.

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

(ii) There is no $(t - \varepsilon)$-competitive deterministic algorithm without predictions.

(iii) For any pair of error measures (η_0, η_1), there is no $(t-\varepsilon, 0, 0)$-competitive deterministic ASG_t algorithm with predictions with respect to (η_0, η_1).

Proof **Towards (i):** Let ALG be the algorithm that always guesses 0. Then, for any instance $I = (x, r) \in \mathcal{I}_{\text{ASG}_t}$,

$$\text{ALG}(I) = \sum_{i=1}^{n} t \cdot x_i = t \cdot \sum_{i=1}^{n} x_i = t \cdot \text{OPT}(I).$$

Towards (ii): Assume there exists a constant $0 < \varepsilon < 1$, and a deterministic online algorithm without predictions, ALG, such that ALG is $(t - \varepsilon)$-competitive. Then, there exists an additive term, $\kappa \in o(\text{OPT})$, such that, for all $I = (x, r) \in \mathcal{I}_{\text{ASG}_t}$,

$$\text{ALG}(I) \leqslant (t - \varepsilon) \cdot \text{OPT}(I) + \kappa(I). \tag{3}$$

Since $\kappa \in o(\text{OPT})$, by Equation (1),

$$\forall \delta > 0 \colon \exists b_\delta \colon \forall I \in \mathcal{I}_{\text{ASG}_t} \colon \kappa(I) < \delta \cdot \text{OPT}(I) + b_\delta.$$

Fix $\delta < \varepsilon$, and determine b_δ.

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

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

where y_j is ALG's j'th guess when run on I^n, for $j = 1, 2, \ldots, n$. Since ALG is deterministic, the collection $\{I^n\}_{n \in \mathbb{Z}^+}$ is well-defined.

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

$$\textsc{Alg}(I^n) = t \cdot \textsc{Opt}(I^n) + \sum_{i=1}^{n} (1 - x_i^n). \tag{4}$$

Combining Equations (3) and (4) we get that

$$t \cdot \textsc{Opt}(I^n) + \sum_{i=1}^{n} (1 - x_i^n) \leqslant (t - \varepsilon) \cdot \textsc{Opt}(I^n) + \kappa(I^n).$$

Hence,

$$\varepsilon \cdot \textsc{Opt}(I^n) + \sum_{i=1}^{n} (1 - x_i^n) \leqslant \kappa(I^n).$$

Since $\textsc{Opt}(I^n) = \sum_{i=1}^{n} x_i^n$ and $\varepsilon < 1$,

$$\varepsilon \cdot \sum_{i=1}^{n} (x_i^n + (1 - x_i^n)) < \varepsilon \cdot \textsc{Opt}(I^n) + \sum_{i=1}^{n} (1 - x_i^n),$$

and so

$$\varepsilon \cdot n = \varepsilon \cdot \sum_{i=1}^{n} (x_i^n + (1 - x_i^n)) < \kappa(I^n). \tag{5}$$

Since $\kappa \in o(\textsc{Opt})$, we get that

$$\varepsilon \cdot n < \kappa(I^n) < \delta \cdot \textsc{Opt}(I^n) + b_\delta \leqslant \delta \cdot n + b_\delta$$

subtract $\delta \cdot n$, then

$$(\varepsilon - \delta) \cdot n < b_\delta$$

Since $\delta < \varepsilon$, we get a contradiction by taking the limit as $n \to \infty$.

Towards (iii): This is a direct consequence of (ii). □

Next, we consider ASG_t with predictions. 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 6 Let $t \in \mathbb{Z}^+$. Then,

(a) for any instance $I \in \mathcal{I}_{\mathrm{ASG}_t}$,

$$\textsc{FtP}(I) = \textsc{Opt}(I) + (t-1) \cdot \mu_0(I) + \mu_1(I),$$

(b) for any $\alpha, \beta \in \mathbb{R}^+$ with $\alpha \geqslant 1$ and $\alpha + \beta \geqslant t$, FtP is a strictly $(\alpha, \beta, 1)$-competitive algorithm for ASG_t with respect to (μ_0, μ_1).

Proof **Towards (a):** Let $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$ with $n = |I|$. Then,

$$\begin{aligned}
\textsc{FtP}(I) &= \sum_{i=1}^{n} (\hat{x}_i + t \cdot x_i \cdot (1 - \hat{x}_i)) \\
&= \sum_{i=1}^{n} ((x_i + (1 - x_i)) \cdot \hat{x}_i) + t \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)) \\
&= \sum_{i=1}^{n} (x_i \cdot \hat{x}_i) + t \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)) + \sum_{i=1}^{n} ((1 - x_i) \cdot \hat{x}_i) \\
&= \sum_{i=1}^{n} x_i + (t-1) \cdot \sum_{i=1}^{n} (x_i \cdot (1 - \hat{x}_i)) + \sum_{i=1}^{n} ((1 - x_i) \cdot \hat{x}_i) \\
&= \textsc{Opt}(I) + (t-1) \cdot \mu_0(I) + \mu_1(I).
\end{aligned}$$

Towards (b): Determine $\alpha', \beta' \in \mathbb{R}^+$, such that $1 \leqslant \alpha' \leqslant \alpha$, $\beta' \leqslant \beta$, and $\alpha' + \beta' = t$. Observe that such α' and β' always exists, since $\alpha + \beta \geqslant t$. Then, for any $I \in \mathcal{I}_{\mathrm{ASG}_t}$, we have that

$$\begin{aligned}
\textsc{FtP}(I) = &\sum_{i=1}^{n} (\hat{x}_i + t \cdot x_i \cdot (1 - \hat{x}_i)) \\
&= \alpha' \cdot \sum_{i=1}^{n} \hat{x}_i \cdot (x_i + (1 - x_i)) + t \cdot \sum_{i=1}^{n} x_i \cdot (1 - \hat{x}_i) \\
&\quad - (\alpha' - 1) \cdot \sum_{i=1}^{n} \hat{x}_i \cdot (x_i + (1 - x_i)) \\
&= \alpha' \cdot \sum_{i=1}^{n} \hat{x}_i \cdot x_i + \alpha' \cdot \sum_{i=1}^{n} \hat{x}_i \cdot (1 - x_i) + t \cdot \sum_{i=1}^{n} x_i \cdot (1 - \hat{x}_i) \\
&\quad - (\alpha' - 1) \cdot \sum_{i=1}^{n} \hat{x}_i \cdot x_i - (\alpha' - 1) \cdot \sum_{i=1}^{n} \hat{x}_i \cdot (1 - x_i).
\end{aligned}$$

Since $\alpha' \geqslant 1$, we have that

$$\begin{aligned}
\textsc{FtP}(I) &\leqslant \alpha' \cdot \sum_{i=1}^{n} \hat{x}_i \cdot x_i + t \cdot \sum_{i=1}^{n} x_i \cdot (1-\hat{x}_i) + \sum_{i=1}^{n} \hat{x}_i \cdot (1-x_i) \\
&= \alpha' \cdot \sum_{i=1}^{n} \hat{x}_i \cdot x_i + t \cdot \sum_{i=1}^{n} x_i \cdot (1-\hat{x}_i) + \sum_{i=1}^{n} \hat{x}_i \cdot (1-x_i) \\
&\quad + \alpha' \cdot \sum_{i=1}^{n} x_i \cdot (1-\hat{x}_i) - \alpha' \cdot \sum_{i=1}^{n} x_i \cdot (1-\hat{x}_i) \\
&= \alpha' \cdot \sum_{i=1}^{n} x_i + (t-\alpha') \cdot \sum_{i=1}^{n} x_i \cdot (1-\hat{x}_i) + \sum_{i=1}^{n} \hat{x}_i \cdot (1-x_i) \\
&= \alpha' \cdot \textsc{Opt}(I) + \beta' \cdot \mu_0(I) + \mu_1(I) \\
&\leqslant \alpha \cdot \textsc{Opt}(I) + \beta \cdot \mu_0(I) + \mu_1(I).
\end{aligned}$$

□

In the following, for any error measures η, we assume that $\infty \cdot \eta(I) = 0$, whenever $\eta(I) = 0$.

Theorem 7 FtP is a strictly $(1, \infty, 1)$-competitive algorithm for ASG with respect to (μ_0, μ_1).

Proof Adapt the proof from Theorem 6.(a), using that $\sum_{i=1}^{n} x_i\hat{x}_i \leqslant \sum_{i=1}^{n} x_i$. □

Observe that the competitive ratio $(1, \infty, 1)$ only provides a guarantee when $\eta_1(I) = 0$.

In the following corollary, we extend a lower bound on Paging with Discard Predictions by Antoniadis et al. [4] to, among other problems, ASG_t. For completeness, we state the Theorem here, though in a weak form and without proof. We strengthen the statement in Theorem 51 in Section 7, where it is restated and proven in a more general form.

Theorem 8 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) $\alpha + (t-1) \cdot \gamma \geqslant t$.

4 A Hierarchy of Complexity Classes

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

4.1 Relative Hardness and Reductions

Labeling one problem, Q, as hard as another problem, P, usually means that no algorithm for Q can be better than a best possible algorithm for P. However, since there is no total order on the (α, β, γ)-triples capturing competitiveness, one cannot determine which algorithm is "best possible" when measuring the quality of algorithms by their competitiveness. Thus, we require the existence of an (α, β, γ)-competitive Pareto-optimal algorithm for Q to imply the existence of an (α, β, γ)-competitive algorithm for P: Recall that an (α, β, γ)-competitive algorithm is called *Pareto-optimal* for 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.

Next, we give a negative result on the competitiveness of algorithms for ASG_t with respect to (μ_0, μ_1), which, together with Theorems 5, 6, and 8 gives a complete classification of all Pareto-optimal algorithms for ASG_t.

Lemma 9 Let $\textsc{Alg}$ 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 $\textsc{Alg}$ is (α, β, γ)-competitive where $\alpha = t - \varepsilon$ and $\gamma < 1$, for some $\varepsilon > 0$.

Consider the instance $I = (x, \hat{x})$, where $\hat{x} = \langle 1^n \rangle$ and $x_i = 1 - y_i$, for $i = 1, 2, \ldots, n$. Observe that x is well-defined since $\textsc{Alg}$ is deterministic.

Now, let Y_0 and Y_1 be the number of times that $\textsc{Alg}$ sets $y_i = 0$ and $y_i = 1$, respectively, and observe that $n = Y_0 + Y_1$. Then, by definition of x, we have

$$\textsc{Opt}(I) = Y_0 \quad \text{and} \quad \textsc{Alg}(I) = Y_1 + t \cdot Y_0 = n + (t-1) \cdot Y_0.$$

Since $\textsc{Alg}$ is $(t - \varepsilon, \beta, \gamma)$-competitive with additive constant $\kappa \in \mathbb{R}$, we also have that

$$\begin{aligned} \textsc{Alg}(I) &\leqslant (t - \varepsilon) \cdot \sum_{i=1}^{n} x_i + \beta \cdot \sum_{i=1}^{n} x_i \cdot (1 - \hat{x}_i) + \gamma \cdot \sum_{i=1}^{n} (1 - x_i) \cdot \hat{x}_i + \kappa \\ &= (t - \varepsilon) \cdot Y_0 + \gamma \cdot Y_1 + \kappa. \end{aligned}$$

Hence,

$$n + (t-1) \cdot Y_0 \leqslant (t-\varepsilon) \cdot Y_0 + \gamma \cdot Y_1 + \kappa,$$

which holds if and only if

$$\varepsilon \cdot Y_0 + (1-\gamma) \cdot Y_1 - \kappa \leqslant 0.$$

Finally, since $1-\gamma > 0$ as $\gamma < 1$ and $\varepsilon > 0$, we get that

$$\begin{aligned} \min\{\varepsilon, 1-\gamma\} \cdot n - \kappa &= \min\{\varepsilon, 1-\gamma\} \cdot (Y_0 + Y_1) - \kappa \\ &\leqslant \varepsilon \cdot Y_0 + (1-\gamma) \cdot Y_1 - \kappa. \end{aligned}$$

However, $\lim_{n\to\infty} \min\{\varepsilon, 1-\gamma\} \cdot n - \kappa = \infty$, so we have a contradiction. □

Theorem 10 Let ALG be an (α, β, γ)-competitive algorithm for ASG_t. Then, ALG is Pareto-optimal if and only if

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

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

Proof (a) ⇒ Pareto-optimal; Assume that ALG is $(t, 0, 0)$-competitive. Since neither the second nor the third entry can be improved and there cannot exist a $(t-\varepsilon, 0, 0)$-competitive algorithm for ASG_t by Theorem 5.(iii), then ALG is Pareto-optimal.

(b) ⇒ Pareto-optimal: Since $\alpha + \beta = t$, one cannot improve either α or β without making the other larger, by Theorem 8. Finally, since $\alpha < t$ then, by Lemma 9, one cannot improve γ without increasing α.

Pareto-optimal ⇒ (a) or (b): Let ALG be an (α, β, γ)-competitive Pareto-optimal algorithm. We consider two cases:

Case $\alpha \geqslant t$: Assume towards contradiction that $\alpha > t$, $\beta > 0$, or $\gamma > 0$. By Theorem 5, there exists an $(t, 0, 0)$-algorithm, which improves on at least one of the parameters capturing the competitiveness of ALG, without making the others worse. Hence, ALG cannot be Pareto-optimal, a contradiction.

Case $\alpha < t$: Since $\alpha < t$, then $\beta \geqslant t - \alpha$, by Theorem 8, and $\gamma \geqslant 1$ by Lemma 9. □

Definition 11 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 *(at least) 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 Pareto-optimal 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 also say that P is *no harder than* 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 12 The as-hard-as relation is reflexive and transitive.

Proof **Reflexivity:** Proving reflexivity translates to proving that the existence of an (α, β, γ)-competitive Pareto-optimal 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 binary predictions with error measures (η_0, η_1), (ξ_0, ξ_1) and (φ_0, φ_1), and assume that Q is as hard as W and that W is as hard as P. Let $\text{ALG}_Q \in \mathcal{A}_Q$ be an (α, β, γ)-competitive Pareto-optimal algorithm for Q with respect to (φ_0, φ_1). Since Q is as hard as W with respect to (φ_0, φ_1) and (ξ_0, ξ_1), the existence of ALG_Q implies the existence of an (α, β, γ)-competitive algorithm with respect to (ξ_0, ξ_1) for W, say ALG_W. If ALG_W is Pareto-optimal, then since W is as hard as P with respect to (ξ_0, ξ_1) and (η_0, η_1), then the existence of 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 is as hard as P with respect to (φ_0, φ_1) and (η_0, η_1). If, on the other hand, ALG_W is not Pareto-optimal, then, by definition of Pareto-optimality there exists an $(\alpha', \beta', \gamma')$-competitive Pareto-optimal algorithm, ALG'_W, for W with respect to (ξ_0, ξ_1), where $\alpha' \leqslant \alpha$, $\beta' \leqslant \beta$, and $\gamma' \leqslant \gamma$. Since ALG'_W is Pareto-optimal, and W is as hard as P with respect to (ξ_0, ξ_1) and (η_0, η_1), then there also exists an $(\alpha', \beta', \gamma')$-competitive algorithm for P with respect to (η_0, η_1), say ALG'_P. Since $\alpha' \leqslant \alpha$, $\beta' \leqslant \beta$, and $\gamma' \leqslant \gamma$, then ALG'_P is also (α, β, γ)-competitive, and so Q is as hard as 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 Pareto-optimal algorithms for Q map to (α, β, γ)-competitive algorithms for P. In this paper, we use a restricted type of reduction, defined in Definition 13 using the following notation. For any online minimization problem, P, we let $\mathcal{A}_P$ be the set of online algorithms for P.

Definition 13 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 $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ be a tuple consisting of two maps, $\rho_{\text{A}} \colon \mathcal{A}_Q \to \mathcal{A}_P$ and $\rho_{\text{R}} \colon \mathcal{A}_Q \times \mathcal{I}_P \to \mathcal{I}_Q$. 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 there exists a map $k_{\text{A}} \in o(\text{OPT}_P)$, called the *reduction term* of ρ, such that for each instance $I_P \in \mathcal{I}_P$ and each algorithm $\text{ALG}_Q \in \mathcal{A}_Q$, letting $\text{ALG}_P = \rho_{\text{A}}(\text{ALG}_Q)$ and $I_Q = \rho_{\text{R}}(\text{ALG}_Q, I_P)$,

(O1) $\text{ALG}_P(I_P) \leqslant \text{ALG}_Q(I_Q) + k_{\text{A}}(I_P)$,

(O2) $\text{OPT}_Q(I_Q) \leqslant \text{OPT}_P(I_P)$,

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

(O4) $\varphi_1(I_Q) \leqslant \eta_1(I_P)$,

For brevity, if such a tuple, ρ, exists, and the pairs (η_0, η_1) and (φ_0, φ_1) are clear from the context, then we say that ρ is a *strict online reduction* from P to Q, and, by abuse of notation, write $\rho \colon P \xrightarrow{\text{red}} Q$. □

Lemma 14 Let $\rho \colon P \xrightarrow{\text{red}} Q$ be an online reduction as in Definition 13, 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 Let $\kappa_Q \in o(\text{OPT}_Q)$ be the additive term of ALG_Q. Then, for any instance $I_P \in \mathcal{I}_P$ of P, letting $I_Q = \rho_{\text{R}}(\text{ALG}, I_P)$, we have that

$$\begin{aligned}
\text{ALG}_P(I_P) &\leqslant \text{ALG}_Q(I_Q) + k_{\text{A}}(I_P), \text{ by (O1)} \\
&\leqslant \alpha \cdot \text{OPT}_Q(I_Q) + \beta \cdot \varphi_0(I_Q) + \gamma \cdot \varphi_1(I_Q) \\
&\quad + \kappa_Q(I_Q) + k_{\text{A}}(I_P) \\
&\leqslant \alpha \cdot \text{OPT}_P(I_P) + \beta \cdot \eta_0(I_P) + \gamma \cdot \eta_1(I_P) \\
&\quad + \kappa_Q(I_Q) + k_{\text{A}}(I_P), \text{ by (O2), (O3), and (O4)} \\
&= \alpha \cdot \text{OPT}_P(I_P) + \beta \cdot \eta_0(I_P) + \gamma \cdot \eta_1(I_P) + \kappa_P(I_P),
\end{aligned}$$

where $\kappa_P = k_{\text{A}} + \kappa_Q \in o(\text{OPT}_P)$, since $\kappa_Q \in o(\text{OPT}_Q)$ and $k_{\text{A}} \in o(\text{OPT}_P)$ (see Lemma 54 in Appendix A). Thus, ALG_P is (α, β, γ)-competitive with respect to (η_0, η_1). □

Observe that Lemma 14 implies that strict online reductions serve the desired purpose of reductions: If there exists a strict online reduction $\rho \colon P \xrightarrow{\text{red}} Q$, then Q is as hard as 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 Defining 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 binary predictions that are no harder than ASG_t with respect to (η_0, η_1):

Definition 15 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 ASG_t is as hard as P,
- P is $\mathcal{C}^t_{\eta_0,\eta_1}$-*hard*, if P is as hard as 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.

□

Thus, 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.

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:

Lemma 16 Let $t \in \mathbb{Z}^+ \cup \{\infty\}$ and let (η_0, η_1) be any pair of error measures.

(a) If $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, and P is as hard as Q, then $Q \in \mathcal{C}^t_{\eta_0,\eta_1}$.

(b) If P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, and Q is as hard as P, then Q is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof Towards (a): Since $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, ASG_t is as hard as P. Since P is as hard as Q, transitivity implies that ASG_t is as hard as Q, and thus $Q \in \mathcal{C}^t_{\eta_0,\eta_1}$.

Towards (b): Since P is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard, then P is as hard as ASG_t. Since Q is as hard as P, transitivity implies that Q is as hard as ASG_t, and thus we conclude that Q is $\mathcal{C}^t_{\eta_0,\eta_1}$-hard. □

This lemma implies results concerning special cases of a problem:

Corollary 17 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 $\mathcal{I}_{P_{\text{sub}}} \subseteq \mathcal{I}_P$.

- If $P \in \mathcal{C}^t_{\eta_0,\eta_1}$, then $P_{\text{sub}} \in \mathcal{C}^t_{\eta_0,\eta_1}$, and
- 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 $\rho\colon P_{\text{sub}} \xrightarrow{\text{red}} P$, obtained by setting $\rho_{\text{A}}(\text{ALG}) = \text{ALG}$ and $\rho_{\text{R}}(\text{ALG}, I) = I$, for all algorithms $\text{ALG} \in \mathcal{A}_P$, and all instances $I \in \mathcal{I}_{P_{\text{sub}}}$. Hence, this is a consequence of Lemma 16. □

Observe that Lemma 16 implies that any $\mathcal{C}^t_{\eta_0,\eta_1}$-complete problem can be used as the base problem when defining $\mathcal{C}^t_{\eta_0,\eta_1}$. For instance, as shown in Lemmas 29 and 30, 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 analysed. 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 is as hard as 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.

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.

Lemma 18 For $t \in \mathbb{Z}^+$ and any pair of error measures, (η_0, η_1),

(i) ASG_{t+1} is as hard as ASG_t, and

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

Proof **Towards (i):** We give an online reduction from ASG_t to ASG_{t+1}. Let $\text{ALG} \in \mathcal{A}_{\text{ASG}_{t+1}}$ and set $\rho_{\text{A}}(\text{ALG}) = \text{ALG}'$ and $\rho_{\text{R}}(\text{ALG}, I) = I$ for any $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{ASG}_t}$, where ALG' is the algorithm which always guesses the same as ALG. Further, ALG' also reveals x to ALG, when it learns the contents of x.

Let $I = (x, \hat{x}, r)$ be any instance of ASG_t. We verify that this reduction satisfies conditions (O1)–(O4) from Definition 13. Since $\rho_{\text{R}}(\text{ALG}, I) = I$, the conditions (O2)–(O4) are immediate for any pair of error measures (η_0, η_1). Towards (O1), denote by $y_1, y_2, \ldots, y_n$ the bits guessed by ALG, and therefore also by ALG′. Then,

$$\begin{aligned}\text{ALG}(I) - \text{ALG}'(I) &= \sum_{i=1}^{n} (y_i + (t+1) \cdot (1 - y_i) \cdot x_i) \\ &\quad - \sum_{i=1}^{n} (y_i + t \cdot (1 - y_i) \cdot x_i) \\ &= \sum_{i=1}^{n} (1 - y_i) \cdot x_i \geqslant 0.\end{aligned}$$

(O1) follows.

Towards (ii): Assume that ASG_t is as hard as ASG_{t+1}. Then, for any (α, β, γ)-competitive Pareto-optimal algorithm for ASG_t we get an (α, β, γ)-competitive algorithm for ASG_{t+1}. By Theorem 5, the algorithm that always guesses 0 is $(t, 0, 0)$-competitive for ASG_t, and since no algorithm for ASG_t can be $(t - \varepsilon, 0, 0)$-competitive by Theorem 5.(iii), this algorithm is Pareto-optimal. Hence, since ASG_t is as hard as ASG_{t+1} by assumption, there exist a $(t, 0, 0)$-competitive algorithm for ASG_{t+1}, which contradicts Theorem 5.(iii). □

Lemma 19 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 Let $t \in \mathbb{Z}^+$ be arbitrary, (η_0, η_1) be any pair of error measures, and $P \in \mathcal{C}^t_{\eta_0,\eta_1}$ be any problem. Since $P \in \mathcal{C}^t_{\eta_0,\eta_1}$ then ASG_t is as hard as P. Since ASG_{t+1} is as hard as ASG_t by Lemma 18, transitivity implies that ASG_{t+1} is as hard as P, and so $P \in \mathcal{C}^t_{\eta_0,\eta_1}$. Hence, $\mathcal{C}^t_{\eta_0,\eta_1} \subseteq \mathcal{C}^{t+1}_{\eta_0,\eta_1}$. To see that $\mathcal{C}^t_{\eta_0,\eta_1} \subsetneq \mathcal{C}^{t+1}_{\eta_0,\eta_1}$, observe that $\text{ASG}_{t+1} \in \mathcal{C}^{t+1}_{\eta_0,\eta_1}$ and $\text{ASG}_{t+1} \notin \mathcal{C}^t_{\eta_0,\eta_1}$, by Lemma 18. □

Lemma 20 Let $t \in \mathbb{Z}^+$ be arbitrary, and let (η_0, η_1) be any pair of error measures. Then, for any $\mathcal{C}^{t+1}_{\eta_0,\eta_1}$-hard problem, P, $P \notin \mathcal{C}^t_{\eta_0,\eta_1}$.

Proof Assume towards contradictions that P is $\mathcal{C}^{t+1}_{\eta_0,\eta_1}$-hard and $P \in \mathcal{C}^t_{\eta_0,\eta_1}$. Then, P is as hard as ASG_{t+1} and ASG_t is as hard as P. Thus, due to transitivity, ASG_t is as hard as ASG_{t+1}, contradicting Lemma 18. □

By similar arguments as above, we get the following:

Lemma 21 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
- for any $\mathcal{C}_{\eta_0,\eta_1}$-hard problem, P, $P \notin \mathcal{C}^t_{\eta_0,\eta_1}$.

Theorem 22 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, $\mathcal{C}^t_{Z_0,Z_1}$, based on the pair of error measures (Z_0, Z_1), given by $Z_0(I) = Z_1(I) = 0$, 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

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

for all instances $I \in \mathcal{I}_P$, and so Alg is an α-competitive purely online algorithm for P. Hence, we obtain a similar complexity theory for purely online algorithms.

Remark All results in this paper involving problems and complexity classes also hold for the same problems and classes without predictions.

Remark In Section 7 we discuss a strategy for proving lower bounds for all $\mathcal{C}^t_{\eta_0,\eta_1}$-hard problems. Using the same strategy, the lower bound from Theorem 5.(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.

5 A Template for Online Reductions via Simulation

In this section we introduce a template for creating online reductions from ASG_t to P, implying $\mathcal{C}^t_{\eta_0,\eta_1}$-hardness of P with respect to a collection of pairs of error measures (η_0, η_1). To this end, we define a property of pairs of error measures that is sufficient for the existence of these reductions.

Definition 23 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. □

Thus, an error measure is called *insertion monotone* if the insertion of a finite number of correctly predicted bits into the instance does not increase the error. Clearly μ_b and Z_b are insertion monotone for all $b \in \{0, 1\}$. We provide a non-exhaustive list of pairs of insertion monotone error measures in Appendix B.

We proceed by presenting the construction of the reduction template.

5.1 The Reduction Template

Let P be any problem, and (η_0, η_1) be any pair of insertion monotone error measures. We introduce the template for creating strict online reductions of the form

$$\rho\colon \mathrm{ASG}_t \xrightarrow{\text{red}} P.$$

First, we define a method for creating an algorithm for ASG_t for each possible algorithm, $\mathrm{ALG}_P \in \mathcal{A}_P$, for P.

The algorithm for ASG_t which we associate to ALG_P is $\mathrm{REDALG}(\mathrm{ALG}_P)$, defined in Algorithm 1. Formally, $\rho_{\mathrm{A}}(\cdot) = \mathrm{REDALG}(\cdot)$. For this reduction template, the construction of ρ_{A} and ρ_{R} are closely related, and so the details of REDALG will become clear as we define ρ_{R} in depth.

Next, we define the translation of instances of ASG_t into instances of P, given ALG_P.

The translation part of our reduction template requires proving the existence of two things: a notion of *challenge request*, and a collection of *blocks*.

The idea is that it should be as hard to compute the true bit of a challenge request, as to correctly guess a bit. Whenever $\mathrm{REDALG}(\mathrm{ALG}_P)$ has to guess the next bit of the instance of ASG_t, it will create a new challenge request for P, and then guess the same as ALG_P outputs on the challenge request. ALG_P will get the same prediction for the i'th challenge request as $\mathrm{REDALG}(\mathrm{ALG}_P)$ gets for its i'th guess.

Algorithm 1 RedAlg

Input: An algorithm $\textsc{Alg}_P \in \mathcal{A}_P$, and an instance $(x, \hat{x}, r)$ for ASG_t
Output: An instance for P and an algorithm for ASG_t
$i \leftarrow 1$
while receiving prompts for guessing bits **do**
    Give $\textsc{Alg}_P$ the challenge request c_i and the prediction $\hat{x}'_i = \hat{x}_i$
    Let y'_i be $\textsc{Alg}_P$'s output for c_i
    Guess/output $y_i = y'_i$
    $i \leftarrow i + 1$
Receive $x = x_1, x_2, \ldots, x_n$ ▷ Finishes the ASG_t instance
for $j = 1, 2, \ldots, n$ **do**
    Give all requests in the block $B_P(x_j, y'_j, j)$

After all bits have been guessed, and $\textsc{Alg}_P$ learns the true contents of x, we include $|x|$ blocks of requests for P. The i'th block should "clean up" after $\textsc{Alg}_P$'s response to the i'th challenge request. In particular, the blocks should ensure that the cost of $\textsc{Alg}_P$ is close to the cost of $\textsc{RedAlg}(\textsc{Alg}_P)$, that the cost of $\textsc{Opt}_P$ is close to the cost of $\textsc{Opt}_{\text{ASG}_t}$, and that the bit string encoding $\textsc{Opt}_P$'s solution actually corresponds to an optimal solution of the instance.

Definition 24 A *challenge request* is a request for P, for which the true bit can be either 0 or 1. □

Recall from RedAlg that x_i is the i'th true bit from the ASG_t instance, and y'_i is $\textsc{Alg}_P$'s answer to the i'th challenge request.

Definition 25 Given two bits $x_i, y'_i \in \{0,1\}$ and an integer $i \in \mathbb{Z}^+$, the *block* $B_P(x_i, y'_i, i)$, abbreviated $B(x_i, y'_i)$, is a sequence of correctly predicted requests, $r^{x_i,y'_i} = \langle r_1^{x_i,y'_i}, r_2^{x_i,y'_i}, \ldots, r_k^{x_i,y'_i} \rangle$, for P. We let b^{x_i,y'_i} denote the sequence of bits associated to the requests in $B(x_i, y'_i)$, such that, for each $j = 1, 2, \ldots, k$, $b_j^{x_i,y'_i}$ is both the true and the predicted bit for $r_j^{x_i,y'_i}$. □

Definition 26 Let P be an online problem, and suppose that there exists a collection of challenge requests, $\{c'_i\}_{i \in \mathbb{Z}^+}$, and a collection of blocks, $B(x_i, y'_i)$, with $x_i, y'_i \in \{0,1\}$, for P. Then, the *candidate strict online reduction for* P, is the tuple $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ consisting of two maps $\rho_{\text{A}} \colon \mathcal{A}_P \to \mathcal{A}_{\text{ASG}_t}$ and $\rho_{\text{R}} \colon \mathcal{A}_P \times \mathcal{I}_{\text{ASG}_t} \to \mathcal{I}_P$, where, for any algorithm $\textsc{Alg}_P \in \mathcal{A}_P$, and any instance $I = (x, \hat{x}, r)$, with $|I| = n$, of ASG_t,

- $\rho_{\text{A}}(\textsc{Alg}_P) = \textsc{RedAlg}(\textsc{Alg}_P)$, and

- $\rho_{\text{R}}(\text{ALG}_P, I) = I'$,

where $I' = (x', \hat{x}', r')$ is given by

- $x' = \langle x_1, x_2, \ldots, x_n, b^{x_1,y'_1}, b^{x_2,y'_2}, \ldots, b^{x_n,y'_n}\rangle$,
- $\hat{x}' = \langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n, b^{x_1,y'_1}, b^{x_2,y'_2}, \ldots, b^{x_n,y'_n}\rangle$, and
- $r' = \langle c_1, c_2, \ldots, c_n, r^{x_1,y'_1}, r^{x_2,y'_2}, \ldots, r^{x_n,y'_n}\rangle$.

□

By construction, the number of prediction errors (in either direction) in I' is exactly the same as in I. Hence, it is easy to relate the error for the instance of P to the error of the instance of ASG_t, when the pair of error measures, with respect to which we measure that quality of predictions, are insertion monotone.

Theorem 27 Consider an online problem, P, let (η_0, η_1) be a pair of insertion monotone error measures, and let $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ be a candidate strict online reduction for P. If, for any $\text{ALG}_P \in \mathcal{A}_P$ and any instance $I \in \mathcal{I}_{\text{ASG}_t}$,

(i) $\text{REDALG}(\text{ALG}_P)(I) \leqslant \text{ALG}_P(I')$,

(ii) $\text{OPT}_P(I') \leqslant \text{OPT}_{\text{ASG}_t}(I)$, and

(iii) x' encodes $\text{OPT}_P[I']$,

Then, the candidate online reduction $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ is a strict online reduction from ASG_t to P.

Proof To see that ρ is an online reduction from ASG_t to P preserving competitiveness with respect to (η_0, η_1), we verify conditions (O1)–(O4) from Definition 13.

Firstly, observe that Conditions (i) and (ii) directly imply Conditions (O1) and (O2) with $k_{\text{A}} = 0$.

Lastly, Conditions (O3) and (O4) follow by (iii). In particular, since x' encodes $\text{OPT}_P[I']$, and since the number of prediction errors in I' and in I is equal, then, by Definition 26, we have that $\eta_b(I') \leqslant \eta_b(I)$, for $b \in \{0, 1\}$, since η_b is insertion monotone. □

We use the reduction template multiple times to prove $\mathcal{C}^t_{\eta_0,\eta_1}$-hardness. In particular, we use the template to show hardness of Vertex Cover (Lemma 29), k-Minimum-Spill (Theorem 33) and Interval Rejection (Lemma 35).

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

Throughout, given a graph $G = (V, E)$, we let $\deg\colon V \to \mathbb{N}$ be the map which associates to each vertex $v \in V$ its degree, $\deg(v)$, and we let $\Delta(G) = \max_{v \in V}\{\deg(v)\}$.

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.

Other work on Vertex Cover includes the following. In the purely online vertex-arrival model, there exists a t-competitive algorithm for t-Bounded Degree Vertex Cover, and an impossibility result showing that one cannot create a $(t - \varepsilon)$-competitive online algorithm for t-Bounded Degree Vertex Cover, for any $\varepsilon > 0$ [17]. In the offline setting, Minimum Vertex Cover is MAX-SNP-hard [34], APX-complete [35, 18], and NP-complete [24]. Lastly, Online Vertex-Arrival Vertex Cover is AOC-complete [14].

We consider an online variant of Vertex Cover with predictions:

Definition 28 *Online t-Bounded Degree Vertex Cover with Predictions* (VC_t) is a vertex-arrival problem, where input graphs, G, satisfy that $\Delta(G) \leqslant t$. An algorithm, $\textsc{Alg}$, outputs $y_i = 1$ to include v_i in its vertex cover, and $\{v_i \mid x_i = 1\}$ is an optimal vertex cover. Given an instance $I \in \mathcal{I}_{\mathrm{VC}_t}$,

$$\textsc{Alg}(I) = \begin{cases} \sum_{i=1}^n y_i, & \text{if } \textsc{Alg}\text{'s output is a vertex cover,} \\ \infty, & \text{otherwise.} \end{cases}$$

□

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

Lemma 29 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 $\rho\colon \mathrm{ASG}_t \to \mathrm{VC}_t$, using the reduction template from Section 5 (see Figure 2 for an example).

Consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{ASG}_t}$ and any $\textsc{Alg}' \in \mathcal{A}_{\mathrm{VC}_t}$. Each challenge request, c_i, is an isolated vertex, v_i, with $x'_i = x_i$ and $\hat{x}'_i = \hat{x}_i$. Let y'_i be

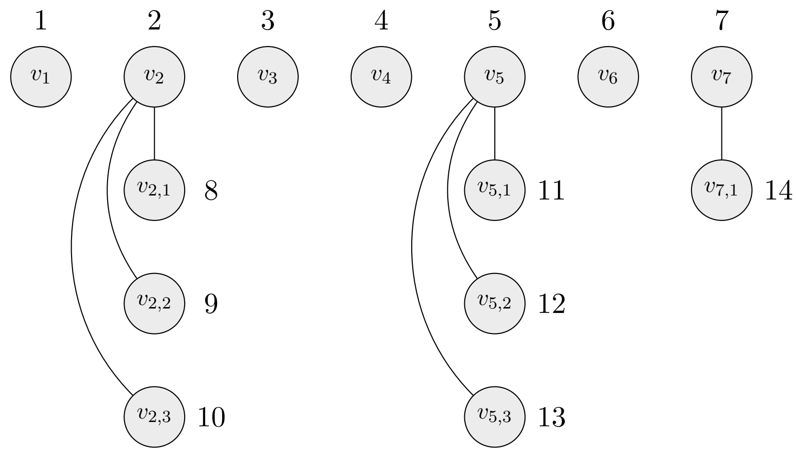

Figure 2: The reduction graph with $t = 3$, $x = 0100101$, and $y' = 1011001(\_\_\_)(\_\_\_)(\_)$, where $x_2 = x_5 = x_7 = y_7 = 1$ and $y_2 = y_5 = 0$. The first seven bits of y' are the VC_t algorithm's response to the challenge requests. The algorithm's response to the requests of the three nonempty blocks are are simply shown as '_', since they do not influence the definition of the graph. The number next to each vertex corresponds to the order in which these vertices are revealed.

the output of ALG$'$ on c_i. The ith block, $B(x_i, y'_i)$, is constructed as follows, with all true and predicted bits equal to 0.

- 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.

Let $\rho = (\rho_{\mathrm{A}}, \rho_{\mathrm{R}})$ be the candidate strict online reduction from Definition 26, and let ALG be the algorithm for ASG_t the produces the same output as ALG$'$ does on the challenge requests, i.e. $y_i = y'_i$, $1 \leqslant i \leqslant n$. It is not hard to check that ρ satisfies (i)–(iii) from Theorem 27.

Towards (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 the ith request. If $x_i = y'_i = 1$, both algorithms have a cost of 1 on the ith request. 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 $B(x_i, y'_i)$.

Towards (ii) and (iii): The set $\{v_i \mid x'_i = 1\}$ is an optimal solution to the instance, I', of VC_t created. Thus, $\mathrm{Opt}_{\mathrm{VC}_t}(I') = \mathrm{Opt}_{\mathrm{ASG}_t}(I)$. □

Algorithm 2

Input: An algorithm, $\mathrm{Alg}' \in \mathcal{A}_{\mathrm{ASG}_t}$, and an instance $(x, \hat{x}, r) \in \mathcal{I}_{\mathrm{VC}_t}$
Output: An instance of ASG_t and an algorithm for VC_t
while receiving requests r_i **do**
  Get prediction $\hat{x}_i$
  Ask Alg$'$ to guess the next bit given $\hat{x}_i$, and let y_i be its output
  $b_i \leftarrow$ `false`
  for each newly revealed edge e **do**
    Let v be such that $e = (v_i, v)$ ▷ v_i is the vertex in r_i
    if v is not included in the vertex cover **then**
      $b_i \leftarrow$ `true`
  if $b_i =$ `true` **then**
    Output 1 ▷ To avoid uncovered edges
  else
    Output y_i
Compute $\mathrm{Opt}_{\mathrm{VC}_t}((x, \hat{x}, r))$ and reveal x to Alg

Lemma 30 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, $\rho\colon \mathrm{VC}_t \xrightarrow{\text{red}} \mathrm{ASG}_t$, with reduction term $k_{\mathrm{A}} = 0$ as follows. Consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\mathrm{VC}_t}$ and any $\mathrm{Alg}' \in \mathcal{A}_{\mathrm{ASG}_t}$. We give 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$'$. 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 $y_i = 1$. Otherwise, it outputs $y_i = y'_i$. We give psuedo-code for Alg in Algorithm 2. 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 $\hat{x}' = \hat{x}$, (O3) and (O4) are trivially satisfied for any pair of error measures. Moreover, since $x' = x$, $\mathrm{Opt}_{\mathrm{VC}_t}(I) = \mathrm{Opt}_{\mathrm{ASG}_t}(I')$, and so (O2) is also satisfied. Hence, it only remains to check Condition (O1):

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'. 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 $y_j = y'_j = 0$. Moreover, $x_i = 0$ implies that $x_j = 1$, since x encodes a valid 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, (O1) is satisfied. □

Our results about VC_t and VC are summarized in Theorem 31. Note that Items (i) and (iv) follow directly from Lemmas 29 and 30.

Theorem 31 For any $t \in \mathbb{Z}^+$ and any pairs of insertion monotone error measures (η_0, η_1),

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

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

(iii) $\text{VC} \notin \mathcal{C}^t_{\eta_0,\eta_1}$.

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

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

(v) $\text{VC} \in \mathcal{C}_{\eta_0,\eta_1}$, and

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

Proof **Towards (i):** This is a direct consequence of Lemmas 29 and 30.

Towards (ii): Since VC_t is a subproblem of VC, Corollary 17 and Lemma 29 implies that VC is as hard as ASG_t for all $t \in \mathbb{Z}^+$.

Towards (iii): Suppose that ASG_t is as hard as VC for some $t \in \mathbb{Z}^+$. By (ii) VC is as hard as ASG_{t+1} and so, by transitivity (see Lemma 12), ASG_t is as hard as ASG_{t+1}, which contradicts Lemma 18.(ii).

Towards (iv): This is a direct consequences of Lemma 30.

Towards (v): We prove the existence of a strict online reduction $\rho\colon \text{VC} \xrightarrow{\text{red}} \text{ASG}$. To this end, let $\text{ALG}' \in \mathcal{A}_{\text{ASG}}$ be any algorithm, and $I = (x, \hat{x}, r)$ be any instance of VC. Let $I' = (x', \hat{x}', r') = \rho_{\text{R}}(\text{ALG}, I)$, and $\text{ALG} = \rho_{\text{A}}(\text{ALG})$, where ALG always outputs the same as ALG', and, when no more vertices arrive, ALG computes $\text{OPT}_{\text{VC}_t}[I]$ and reveals x to ALG'. By construction, Conditions (O2)–(O4) are trivially satisfied, and so it only remains to check Condition (O1).

To this end, observe that $\text{ALG}(I) = \text{ALG}'(I')$, if, an only if, whenever ALG creates an infeasible solution, then $\text{ALG}'(I') = \infty$. 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 ALG'. However, since the edge (v_i, v_j) is contained in the underlying graph of I, either v_i or v_j has been accepted by OPT_{VC}, and so either $x_i = 1$ or $x_j = 1$. Hence, ALG' has guessed 0 on a true 1, implying that $\text{ALG}'(I') = \infty$.

Towards (vi): Let ACC be the following online algorithm for VC. When receiving a vertex, v, if v has no neighbours, reject v, else, accept v. We claim that ACC is $(n-1, 0, 0)$-competitive, where $n = |V|$. To this end, consider any graph, G. If $\text{OPT}(G) = 0$, then G contains no edges, and so ACC never accepts a vertex, in which case $\text{ACC}(G) = 0$. If, on the other hand, $\text{OPT}(G) \geqslant 1$, then $\text{ACC}(G) \leqslant n-1$ since ACC never accepts the first vertex, and so $\frac{\text{ACC}(G)}{\text{OPT}(G)} \leqslant n-1$. Hence, ACC, is $(n-1, 0, 0)$-competitive. Now, let ALG be an $(\alpha, 0, 0)$-competitve Pareto-optimal algorithm for VC, where $\alpha \leqslant n-1$. If VC is as hard as ASG, then the existence of ALG implies the existence of an $(\alpha, 0, 0)$-competitive algorithm for ASG, which in turn implies the existence of an $(n-1, 0, 0)$-competitive algorithm for ASG. This contradicts Observation 4. □

6.2 *k*-Minimum-Spill

Given a graph $G = (V, E)$, the objective of k-Minimum-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 rôle. 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 32 *Online k-Minimum-Spill with Predictions* (k-SPILL) is a vertex-arrival problem. An algorithm, ALG, outputs $y_i = 1$ to mark v_i as a spill and thus add it to V_1, and $\{v_i \mid x_i = 1\}$ is an optimal solution. Given an instance $(x, \hat{x}, r) \in \mathcal{I}_{k\text{-SPILL}}$,

$$\text{ALG}(x, \hat{x}, r) = \begin{cases} \sum_{i=1}^{n} y_i, & \text{if } V \setminus V_1 \text{ is } k\text{-colorable}, \\ \infty, & \text{otherwise.} \end{cases}$$

Throughout, we set $V_0 = V \setminus V_1$. □

For a fixed number of colors, minimizing spill is equivalent to maximizing the number of colored vertices and this problem is NP-complete [3]. A multi-objective variant of this problem was studied in [23]. As discussed below, 1-SPILL is equivalent to VC.

Theorem 33 For all $k, t \in \mathbb{Z}^+$, and all pairs of insertion monotone error measures (η_0, η_1), k-SPILL 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 31.

For k-SPILL with $k \geqslant 2$, we use the reduction template to prove the existence of a reduction ρ: $\text{ASG}_t \xrightarrow{\text{red}} k\text{-SPILL}$, with respect to (η_0, η_1) (see Figure 3 for an example). To this end, consider any $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{ASG}_t}$ and any $\text{ALG}' \in \mathcal{A}_{k\text{-SPILL}}$ and let $I' = (x', \hat{x}', r')$ be the k-SPILL-instance created in the reduction.

For each x_i, the corresponding challenge request, c_i, with $x'_i = x_i$ and $\hat{x}'_i = \hat{x}_i$. Let y'_i be the response of ALG' on c_i.

For each x_i, the corresponding block, $B(x_i, y'_i)$ is defined as follows. All true and predicted bits are 0. The last vertex, f_i, of the block is called a *final*

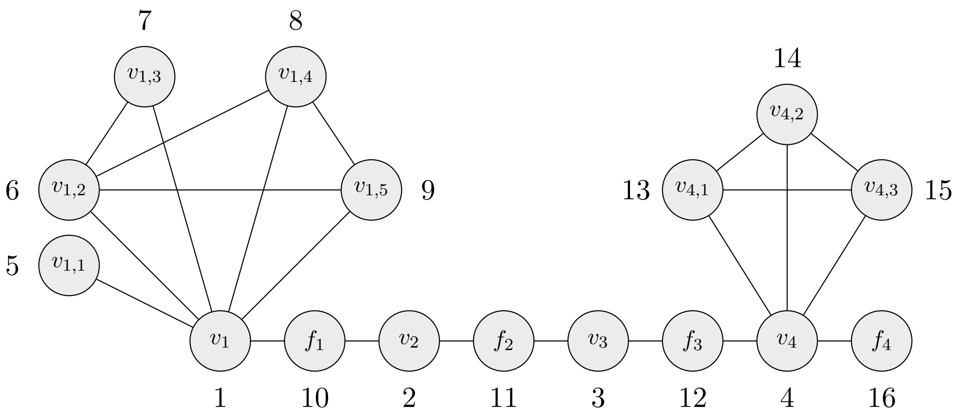

Figure 3: Example of reduction graph with $t = 3$, $k = 3$, $x = 1001$ and $y' = 0011(10101\_)(\_)(\_)(\_\_\_\_)$. The first four bits of y' are the k-Spill algorithm's response to the four challenge requests, and the bits in parenthesis correspond to its responses to the four blocks. Bits that do not influence the definition of the graph are simply shown as '_'. The number next to each vertex corresponds to the order in which these vertices are revealed.

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 $B(x_i, y'_i)$ has a single request containing the final vertex, f_i.
- If $x_i = y'_i = 1$, then $B(x_i, y'_i)$ contains $k + 1$ requests:
  - 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 the following requests:
  (a) Let $j = 1$. Until Alg$'$ has added t new vertices to V_1:
    - A vertex, $v_{i,j}$, connected to v_i and to each vertex in $\bigcup_{l=1}^{j-1} v_{i,l} \setminus V_1$. If Alg$'$ has added more than k vertices to $V \setminus V_1$ in this block, only include the edge $(v_{i,l}, v_{i,j})$ to the first k of these vertices and go to (c) (this happens only if Alg$'$ creates an infeasible solution). Otherwise, increment j.

  (b) If $\{v_i\} \cup \bigcup_j \{v_{i,j}\}$ does not contain a $(k + 1)$-clique:

– For each $j = 1, 2, \ldots, k$, a vertex, $v_{i,j}$, connected to v_i and to $v_{i,l}$, $l < j$.

(c) The final vertex, f_i.

Algorithm 3 Optimal k-coloring of G'

```
Sort the vertices in the order that they are presented to Alg
V'_0 = {v_i ∈ V | x'_i = 0}
Let G'_0 = (V'_0, E'_0) ⊆ G, be the graph induced by V'_0
for all v ∈ V do
   if v is a challenge request then
      if v ∈ V'_0 then
         Give v color λ_1
   else if v is a final request then
      Give v color λ_2                                  ▷ Exists since k ⩾ 2
   else                                    ▷ v is a non-final request in a block
      C ← {λ_1, λ_2, ..., λ_k}
      for all edges e ∈ E'_0 incident to v do
         Let u be such that e = (u, v)
         if u has received a color then
            Let λ_ℓ be that color
            C ← C \ {λ_ℓ}
      Give v an arbitrary color from C
```

Having defined a notion of challenge requests and blocks, we let $\rho = (\rho_{\mathrm{A}}, \rho_{\mathrm{R}})$ be the candidate strict online reduction from ASG_t to k-Spill (see Definition 26). It only remains to check that $\rho_{\mathrm{R}}(\textsc{Alg}', I)$ is a valid instance of k-Spill, for all $I \in \mathcal{I}_{\mathrm{ASG}_t}$, and that (i)–(iii) from Theorem 27 are satisfied.

First, we check that Condition (iii) is satisfied. For ease of notation, let G be the graph defined by I', and let $G'_0 = (V'_0, E'_0) \subset G$ be the subgraph of G induced by the vertices in G whose associated bit in x' is 0. We show that x' is an optimal solution in two steps:

(a) We create a valid k-coloring of the vertices in G'_0.

(b) We show that one cannot create a valid k-coloring of a larger subgraph of G than G'_0.

Towards (a): We create a valid k-coloring of G'_0 in Algorithm 3, and color the graph from Figure 3 in Figure 4. We let $\lambda_1, \lambda_2, \ldots, \lambda_k$ denote the available colors.

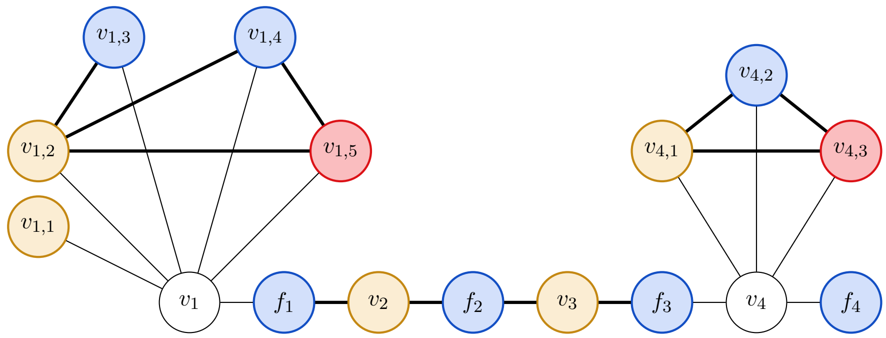

Figure 4: Example of an optimal coloring of the reduction graph from Figure 3, created using Algorithm 3. The vertices in $V \setminus V_0'$, are white. All vertices in V_0', are colored either λ_1, λ_2, or λ_3. The edges in E_0' are thick, and the edges in $E \setminus E_0'$ are thin.

The challenge requests in G_0 receive color λ_1, and the final vertices all receive color λ_2. The challenge requests and final requests form a path, $v_1, f_1, v_2, f_2, \dots, v_n, f_n$, and the colors of the vertices in this path alternate between λ_1 and λ_2, except when a challenge request does not receive a color because it is not part of G_0.

Observe that within G_0, all non-final vertices in a block are only adjacent to other non-final vertices of the same block. Hence, to finish the coloring of G_0, we can consider the non-final vertices of each block separately. Since there are no non-final vertices in the blocks $B(x_i, y_i')$ with $x_i = 0$, we only need to consider blocks with $x_i = 1$.

When $x_i = y_i' = 1$, the block $B(x_i, y_i')$ forms a $(k+1)$-clique together with the vertex v_i of the challenge request c_i. Since v_i is not contained in G_0, the remaining k vertices of the clique can be colored with k colors.

Constructing $B(x_i, y_i')$ with $x_i = 1$ and $y_i' = 0$, larger and larger cliques are formed using the vertices that ALG$'$ has decided to color. However, we never create a clique of size more than $k+1$, and each clique contains the vertex v_i of the challenge request c_i which is not contained in G_0. Hence, at the arrival of each vertex, v, in $B(x_i, y_i')$, v is connected to at most $k-1$ other vertices in G_0. Thus, coloring the vertices in the order of arrival, k colors suffice.

Observe that the only vertices not colored are the challenge requests whose true bit is 1. Hence, the cost of this solution is $\sum_{i=1}^{n} x_i$.

Towards (b): Since each challenge request with true bit 1 introduces a new $(k+1)$-clique into the instance, at least one vertex has to be deemed not colorable for each challenge request with true bit 1. Hence, $\text{OPT}_{k\text{-SPILL}}(I') \geqslant \sum_{i=1}^{n} x_i$, implying that the above solution is optimal. This shows that (iii) from Theorem 27 is satisfied.

Since $\text{OPT}_{\text{ASG}_t}(I) = \sum_{i=1}^{n} x_i$, for any instance, I, of ASG_t, the above also implies that $\text{OPT}_{k\text{-SPILL}}(\rho_{\text{R}}(\text{ALG}', I)) = \text{OPT}(I)$ for all instances $I \in \mathcal{I}_{\text{ASG}_t}$ and all $\text{ALG}' \in \mathcal{A}_{k\text{-SPILL}}$. Hence, (ii) from Theorem 27 is satisfied.

Now, it only remains to check Condition (i). Recall that the ASG_t algorithm, $\text{ALG} = \text{REDALG}(\text{ALG}')$, outputs $y_i = y_i'$, for each $i = 1, 2, \ldots, n$. We verify that, for each r_i, $1 \leqslant i \leqslant n$, the cost incurred by ALG is bounded by the total cost incurred by ALG' on c_i and $B(x_i, y_i')$. To this end, we consider any combination x_i and y_i. Fir the first three combinations considered, the cost of ALG on r_i is bounded be the cost of ALG' on c_i, meaning that the cost of ALG' on $B(x_i, y_i')$ is irrelevant.

Case $x_i = y_i = 0$: In this case, ALG incurs a cost of 0 on r_i. Clearly, this cannot be larger than the cost of ALG' on c_i.

If $x_i = 0$ and $y_i = 1$: In this case, ALG' and ALG both have a cost of 1.

If $x_i = y_i = 1$: Since ALG' decides not color v_i, it incurs a cost of 1, and since ALG guesses 1, it also incurs cost 1.

If $x_i = 1$ and $y_i = 0$: In this case, ALG guesses 0 on a true 1 and incurs a cost of t. At the same time, ALG' deem v_i colorable, and so it does not incur any cost on the challenge request v_i. By construction of $B(x_i, y_i')$, ALG' is forced to incur cost at least t, or create an infeasible solution, only using that it decided to color v_i.

Since for each request, r_i, the cost of ALG is no larger than that of ALG' on c_i and $B(x_i, y_i')$, (ii) from Theorem 27 is satisfied. □

6.3 Interval Rejection

Given a collection of intervals $\mathcal{S}$, 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 size 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 [28].

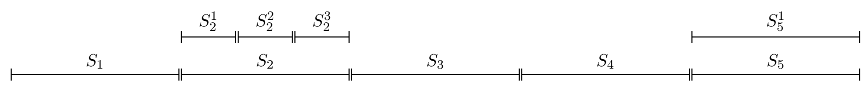

Figure 5: Example reduction for IR_t, with $t = 3$, $x = 01001$, and $y' = 00011(\_\_\_)(\_)$. The first five bits of y' are the IR_t algorithm's responses to the five challenge request, and the bits in parenthesis correspond to its responses to the two blocks. Bits that do not influence the definition of the graph are simply shown as '_'.

Definition 34 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, S, overlaps at most t other intervals in the instance, I. An algorithm, ALG, outputs $y_i = 1$ to include S_i into $\mathcal{S}'$, and $\{S_i \mid x_i = 1\}$ is an optimal solution. Given an instance $I \in \mathcal{I}_{\mathrm{IR}_t}$,

$$\textsc{Alg}(I) = \begin{cases} \sum_{i=1}^{n} y_i, & \text{if no two intervals in } \mathcal{S} \setminus \mathcal{S}' \text{ overlap,} \\ \infty, & \text{otherwise.} \end{cases}$$

□

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

Lemma 35 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 IR_t is as hard as ASG_t by giving a strict online reduction ρ: $\mathrm{ASG}_t \xrightarrow{\text{red}} \mathrm{IR}_t$. This reduction is essentially identical to the one from ASG_t to VC_t from Lemma 29, 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 $S_i^j = ((i-1) \cdot t + j, (i-1) \cdot t + j + 1)$. See Figure 5 for an example reduction. One may observe that this is an interval representation of the interval graph created in the reduction from Lemma 29, assuming that the VC_t algorithm outputs the same bits as the IR_t algorithm. The remainder of the analysis resembles that from Lemma 29. □

Lemma 36 For any $t \in \mathbb{Z}^+ \cup \{\infty\}$, and any pair of error measures (η_0, η_1), $\mathrm{IR}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$.

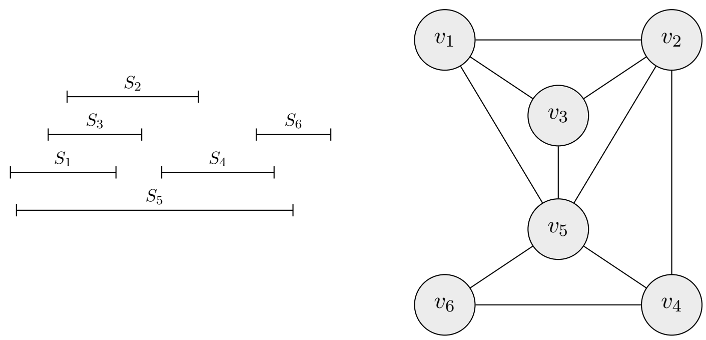

Figure 6: An example reduction from IR_t to VC_t, for $t \geqslant 5$. On the left is the intervals in the instance I for IR_t, and on the right is the underlying graph of the instance $\rho_{\text{R}}(\text{ALG}, I)$, for any algorithm $\text{ALG} \in \mathcal{A}_{\text{VC}_t}$.

Proof Let (η_0, η_1) be any pair of error measures. We prove existence of a reduction, $\rho\colon \text{IR}_t \xrightarrow{\text{red}} \text{VC}_t$, with reduction term $k_{\text{A}} = 0$. This, together with Lemmas 16 and 30, shows that $\text{IR}_t \in \mathcal{C}^t_{\eta_0,\eta_1}$. We show an example of this reduction in Figure 6.

To this end, let $\text{ALG}' \in \mathcal{A}_{\text{VC}_t}$ be any algorithm, and let $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{IR}_t}$ be any instance of IR_t. We define $\text{ALG} = \rho_{\text{A}}(\text{ALG}')$ and $I' = (x', \hat{x}', r') = \rho_{\text{R}}(\text{ALG}', I)$ as follows: When ALG receives a request, r_i, containing an interval, S_i, we request a vertex, v_i, together with all edges of the form (v_j, v_i), for $j < i$, such that $S_j \cap S_i \neq \emptyset$, with true bit $x'_i = x_i$ and predicted bit $\hat{x}'_i = \hat{x}_i$. Then, ALG outputs the same for S_i as ALG' does for v_i. We give pseudocode for ALG in Algorithm 4.

By construction, $x = x'$ and $\hat{x} = \hat{x}'$, and so (O3) and (O4) from Definition 13 are satisfied.

Since each interval overlaps at most t other intervals, all vertices in the underlying graph of I' has degree at most t. Hence, to see that I' is a valid instance, it only remains to check that x' encodes an optimal vertex cover of the underlying graph.

To this end, observe that the intervals in I constitute an interval representation of the underlying graph, G', of I'. Since a collection of non-overlapping intervals from I corresponds to an independent set in G', and since x encodes an optimal solution to I, the set $\{v_i \mid x'_i = 0\}$ is an optimal independent set of G'. Hence, $\{v_i \mid x'_i = 1\}$, the complement of $\{v_i \mid x'_i = 0\}$, encodes an optimal vertex cover of G'.

This shows that I' is a valid instance of VC_t, that $\text{OPT}_{\text{IR}_t}(I) = \text{OPT}_{\text{VC}_t}(I')$, and (O2) from Definition 13 is satisfied.

Hence, it remains to check (O1) to verify that ρ is an online reduction. We give the pseudocode for ALG in Algorithm 4. By constriction, ALG

Algorithm 4

Input: an instance, I, for IR_t and an algorithm, ALG$'$, for VC_t
Output: an instance, I', for VC_t and an algorithm, ALG, for IR_t
while receiving requests r_i, containing the interval S_i **do**
  Get prediction $\hat{x}_i$
  $E_i \leftarrow \emptyset$
  for $j = 1, 2, \ldots, i-1$ **do**
    if $S_j \cap S_i \neq \emptyset$ **then**
      $E_i \leftarrow E_i \cup \{(v_j, v_i)\}$
  Request a new vertex v_i together with E_i, with predicted bit $\hat{x}_i$, and let y_i be ALG's output
  Output y_i

produces the output $\mathcal{S}' = \{S_i \mid y_i' = 1\}$ and ALG$'$ produced the output $V_A' = \{v_i \mid y_i = 1\}$. By construction, $|\mathcal{S}'| = |V_A'|$. Hence, the cost of ALG on instance I is equal to the cost of ALG$'$ on instance I', for all $I \in \mathcal{I}_{\text{IR}_t}$, if and only if whenever ALG creates an infeasible solution, then so does ALG$'$. Hence, suppose that ALG creates an infeasible solution on instance I. Then, there exist two intervals $S_i, S_j \in \mathcal{S} \setminus \mathcal{S}'$ such that $S_i \cap S_j \neq \emptyset$. Since $S_i, S_j \notin \mathcal{S}'$, then $y_i = y_j = 0$, and so $v_i, v_j \notin V_A'$. Since $S_i \cap S_j \neq \emptyset$, the edge (v_i, v_j) is contained in E', and so V_A' is not a vertex cover.

This verifies (O1), and thus finishes the proof. □

Out results about IR_t and IR are summarized in Theorem 37. Observe that Items (i) and (iv) are direct consequences of Lemmas 35 and 36.

Theorem 37 For any $t \in \mathbb{Z}^+$ and 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

(iii) IR $\notin \mathcal{C}^t_{\eta_0,\eta_1}$.

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

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

(v) $\text{IR} \in \mathcal{C}_{\eta_0,\eta_1}$, and

(vi) IR is not $\mathcal{C}_{\eta_0,\eta_1}$-hard.

Proof **Towards (i):** This is a direct consequence of Lemmas 35 and 36.

Towards (ii): Since IR_t is a subproblem of IR, this is a direct consequence of Corollary 17 and Lemma 35.

Towards (iii): Assume that $\text{IR} \in \mathcal{C}^t_{\eta_0,\eta_1}$ for some $t \in \mathbb{Z}^+$ and some (η_0, η_1). Then ASG_t is as hard as IR. Since IR is as hard as ASG_{t+1} by (ii), transitivity implies that ASG_t is as hard as ASG_{t+1}, which contradicts Lemma 18.(ii).

Towards (iv): This is a direct consequences of Lemma 36.

Towards (v): Observe that VC is as hard as IR by Lemma 36. Since $\text{VC} \in \mathcal{C}_{\eta_0,\eta_1}$ by Theorem 31.(v), we conclude that $\text{IR} \in \mathcal{C}_{\eta_0,\eta_1}$ by transitivity.

Towards (vi): Assume that IR is $\mathcal{C}_{\eta_0,\eta_1}$-hard. Then, IR is as hard as ASG. Since VC is as hard as IR, transitivity of the as-hard-as relation (see Lemma 12) implies that VC is as hard as ASG. This contradicts Theorem 31.(vi). □

6.4 Minimum 2-SAT Deletion

Given a collection of variables, V, and a 2-CNF-SAT formula φ, an algorithm for Minimum 2-SAT Deletion finds a subset of clauses in φ to delete, in order to make φ satisfiable [30, 15]. Equivalently, the algorithm must assign a truth value to all variables, while minimizing the number of unsatisfied clauses in φ.

This problem was first posted by Mahajan and Raman in [30], who also proved that Minimum 2-SAT Deletion is NP-complete and W[1]-hard. Later, Chlebík and Chlebíkova proved that it is also APX-hard [15].

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

Definition 38 A request, r_i, for *Online Minimum 2-SAT Deletion with Predictions* (2-SatD), contains a variable, v_i, and the full contents of all clauses 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 $v_i = \texttt{true}$, and x_i encodes an optimal truth-value of the variable v_i. Given an instance $I = (x, \hat{x}, r)$,

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

□

In this section, we prove the existence of a reduction $\rho\colon \text{IR} \to$ 2-SatD, that is based on the following idea. When an algorithm for IR receives an interval S, we create a new request v_S for 2-SatD, which is revealed together with the clauses:

$$(\overline{v_S} \vee \overline{v_S}) \wedge \left(\bigwedge_{\tilde{S}: \tilde{S} \cap S \neq \emptyset} (v_S \vee v_{\tilde{S}}) \right),$$

where $\tilde{S}$ is any previously revealed interval. We refer to the clauses of the form $(v_S \vee v_{\tilde{S}})$ as *collision clauses*.

Observe that the clause $(\overline{v_S} \vee \overline{v_S})$ tempts an algorithm for 2-SatD to set $v_i = \texttt{false}$, and so output 0, corresponding to not placing S in $\mathcal{S}'$. The collision clauses are intended to detect collisions. For any two intervals S and $\tilde{S}$ with $S \cap \tilde{S} \neq \emptyset$, we include the collision clause $(v_S \vee v_{\tilde{S}})$ which is satisfied if, and only if, either $v_S = \texttt{true}$ or $v_{\tilde{S}} = \texttt{true}$, corresponding to placing at least one of S and $\tilde{S}$ in $\mathcal{S}'$.

Theorem 39 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 Let (η_0, η_1) be any pair of error measures. We prove the existence of a reduction $\rho\colon \text{IR} \xrightarrow{\text{red}}$ 2-SatD with reduction term $k_{\text{A}} = 0$. This, together with Lemmas 12 and 35 establishes the hardness of 2-SatD with respect to any pair of insertion monotone error measures.

For any instance $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{IR}}$ and any $\textsc{Alg}' \in \mathcal{A}_{\text{2-SatD}}$, letting $\textsc{Alg} = \rho_{\text{A}}(\textsc{Alg}')$ and $I' = (x', \hat{x}', r') = \rho_{\text{R}}(\textsc{Alg}', I)$, we show three things:

(a) $\textsc{Alg}(I) \leqslant \textsc{Alg}'(I')$,

(b) $\textsc{Opt}_{\text{IR}}(I) = \textsc{Opt}_{\text{2-SatD}}(I')$, and

(c) $x' = x$ and $\hat{x}' = \hat{x}$.

Observe that (a) implies Condition (O1) with $k_{\text{A}} = 0$, (b) implies Conditions (O2), and (c) implies Conditions (O3) and (O4).

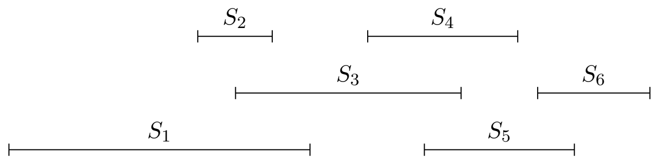

Figure 7: An example reduction from IR to 2-SatD. The above instance of IR gives rise to the following set of variables and CNF-formula: $V = \{v_{S_1}, v_{S_2}, v_{S_3}, v_{S_4}, v_{S_5}, v_{S_6}\}$, and $\varphi = (\overline{v_{S_1}} \vee \overline{v_{S_1}}) \wedge (\overline{v_{S_2}} \vee \overline{v_{S_2}}) \wedge (v_{S_1} \vee v_{S_2}) \wedge (\overline{v_{S_3}} \vee \overline{v_{S_3}}) \wedge (v_{S_1} \vee v_{S_3}) \wedge (v_{S_2} \vee v_{S_3}) \wedge (\overline{v_{S_4}} \vee \overline{v_{S_4}}) \wedge (v_{S_3} \vee v_{S_4}) \wedge (\overline{v_{S_5}} \vee \overline{v_{S_5}}) \wedge (v_{S_3} \vee v_{S_5}) \wedge (v_{S_4} \vee v_{S_5}) \wedge (\overline{v_{S_6}} \vee \overline{v_{S_6}}) \wedge (v_{S_5} \vee v_{S_6})$.

The construction of Alg and I' are closely related (see Figure 7 for an example reduction). The pseudocode for Alg is given in Algorithm 5.

For any $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{IR}}$ and any $\text{Alg}' \in \mathcal{A}_{\text{2-SatD}}$, the instance $I' = (x', \hat{x}', r') = \rho_{\text{R}}(\text{Alg}', I)$ is given by:

- $x' = x$.
- $\hat{x}' = \hat{x}$.
- $r' = \langle r'_1, r'_2, \ldots, r'_n \rangle$, where r'_i contains the variable v_{S_i}, together with the clauses:

$$\varphi_{S_i} = (\overline{v_{S_i}} \vee \overline{v_{S_i}}) \wedge \left(\bigwedge_{S_j \colon j<i \colon S_j \cap S_i \neq \emptyset} (v_{S_i} \vee v_{S_j}) \right).$$

We verify that x' encodes an optimal solution to the 2-SatD instance, such that I' is a valid instance, as part of (b). Observe that (c) is clear from the definition of I', and so the error of I' is identical to the error of I.

All intervals that an algorithm for IR places in $\mathcal{S}'$ are referred to as being *rejected* by the algorithm. If an interval, S, was not rejected then we say that S was *accepted* by the algorithm.

Towards (a): We refer to an interval S which has been rejected by $\text{Alg} = \rho_{\text{A}}(\text{Alg}')$ in Line 18 of Algorithm 5 even though $v_S = \texttt{false}$ as a *conflict interval.*

We show that

$$\text{Alg}(I) \leqslant \text{Alg}'(I'), \tag{6}$$

Algorithm 5

Input: An algorithm $\text{ALG}' \in \mathcal{A}_{\text{2-SATD}}$, and an IR instance $I = (x, \hat{x}, r)$
Output: An instance of 2-SATD and an algorithm for IR
$\varphi \leftarrow \emptyset$ ▷ An empty CNF-formula for ALG
while receiving request r_i containing the interval S_i **do**
  Get prediction $\hat{x}_i$
  Make a new variable v_{S_i} for the 2-SATD instance
  $O \leftarrow \emptyset$ ▷ Set of overlapping intervals
  for $j = 1, 2, \dots, i$ **do**
    if $S_j \cap S_i \neq \emptyset$ **then**
      $O \leftarrow O \cup \{S_j\}$
  $\varphi \leftarrow \varphi \wedge (\overline{v_{S_i}} \vee \overline{v_{S_i}}) \wedge \left(\bigwedge_{S \in O} (v_{S_i} \vee v_S)\right)$
  Request v_{S_i} with prediction $\hat{x}_i$, and updated CNF-formula φ, and let y_i' be the output of ALG′
  $b \leftarrow$ `false`
  for $S \in O$ **do**
    if S has been accepted **then**
      $b \leftarrow$ `true`
  if $b =$ `true` **then**
    Output 1 ▷ Reject S_i
  else
    Output y_i

by showing that the cost that ALG′ incurs is larger than the cost that ALG incurs for each newly requested interval S in I. To this end, suppose that S has just been revealed. We continue by splitting into three cases:

(A) S does not intersect any other interval.

(B) S intersects some non-conflict intervals.

(C) S intersects at least one conflict interval.

Case (A): If S does not intersect any other interval then ALG accepts S if and only if ALG′ sets $v_S =$ `false`. If $v_S =$ `false`, then neither ALG nor ALG′ incur any cost, as ALG accepts S, and ALG′ satisfies the only newly revealed clause $(\overline{v_S} \vee \overline{v_S})$. If, on the other hand, $v_S =$ `true`, then ALG incurs cost 1 from rejecting S, and ALG′ incurs cost 1 for not satisfying $(\overline{v_S} \vee \overline{v_S})$.

Case (B): Denote by $\{S_1, S_2, \dots, S_k\}$ the collection of intervals satisfying that $S \cap S_i \neq \emptyset$, for $i = 1, 2, \dots, k$. We assume that S_i is a non-conflict interval, for all $i = 1, 2, \dots, k$, implying that S_i has been rejected by ALG if, and only if, $v_{S_i} = \texttt{true}$.

If $v_{S_i} = \texttt{true}$ for all $i = 1, 2, \dots, k$, then ALG incurs the same cost as ALG′. In particular, since $v_{S_i} = \texttt{true}$ for all $i = 1, 2, \dots, k$, we have that $(v_S \vee v_{S_i}) = \texttt{true}$ for all $i = 1, 2, \dots, k$, no matter the truth value of v_S. Hence, ALG and ALG′ incur the same cost by a similar analysis as in Case (A), since ALG outputs the same bit as ALG′.

On the other hand, assume that there exists some $i \in \{1, 2, \dots, k\}$ such that $v_{S_i} = \texttt{false}$. Since there are no conflict intervals, we know that S_i has been accepted by ALG. Observe that no matter the truth value of v_S, ALG will reject S, by Line 18 in Algorithm 5, and so incur cost 1. Hence it remains to verify that ALG′ also incur cost at least 1. If ALG′ sets $v_S = \texttt{false}$, then, for each interval S_i with $v_{S_i} = \texttt{false}$, we have that $(v_{S_i} \vee v_S) = \texttt{false}$. Since there is at least one of these, ALG′ incurs cost at least 1. If, on the other hand, ALG′ sets $v_S = \texttt{true}$, then $(\overline{v_S} \vee \overline{v_S}) = \texttt{false}$, and so ALG′ incurs cost 1.

Case (C): Denote by $\{S_1, S_2, \dots, S_k\}$ the collection of intervals satisfying that $S \cap S_i \neq \emptyset$, for $i = 1, 2, \dots, k$. We assume that there exists a subset $\{S_{i_1}, S_{i_2}, \dots, S_{i_\ell}\} \subset \{S_1, S_2, \dots, S_k\}$, for some $1 \leqslant \ell \leqslant k$, of conflict intervals. By definition of conflict intervals, observe that for each S_{i_j}, $v_{S_{i_j}} = \texttt{false}$, but S_{i_j} has been rejected by ALG.

In case all intervals S_i, for $i = 1, 2, \dots, k$, have been rejected by ALG, ALG will output the same as ALG′. If ALG′ sets $v_S = \texttt{false}$, then ALG accepts S, and so ALG does not incur any cost. At the same time, ALG′ satisfies $(\overline{v_S} \vee \overline{v_S})$, but there may exist collision clauses that are not satisfied. If, on the other hand, ALG′ sets $v_S = \texttt{true}$, then ALG rejects S, and so it incurs cost 1. In this case, ALG′ cannot satisfy $(\overline{v_S} \vee \overline{v_S})$, and so it incurs cost at least one.

On the other hand, if there exists at least one interval S_i such that ALG has accepted S_i, it rejects S, by Line 18 in Algorithm 5. In this case, ALG incurs cost 1, and so we check that ALG′ also incurs cost at least 1.

To this end, suppose that ALG′ sets $v_S = \texttt{true}$. Then, it can not satisfy $(\overline{v_S} \vee \overline{v_S})$ and therefore incurs cost at least 1.

On the other hand, if ALG′ sets $v_S = \texttt{false}$, then $(v_S \vee v_{S_{i_j}})$ will be unsatisfied for each $j = 1, 2, \dots, \ell$. Since $\ell \geqslant 1$, ALG′ incurs cost at least 1.

This proves that Equation (6) is satisfied, which finishes the analysis of (a).

Towards (b): We show that

$$\textsc{Opt}_{\mathrm{IR}}(I) = \textsc{Opt}_{2\text{-}\textsc{SatD}}(I'), \tag{7}$$

and that x' encodes an optimal solution to the instance I'.

We do this by creating a solution, O, to the instance I', based on the optimal solution to the instance I, and show that

(1) $\mathrm{cost}(O) = \textsc{Opt}_{\mathrm{IR}}(I)$.

(2) any solution O' with $\mathrm{cost}(O') < \mathrm{cost}(O)$ translates to a solution O'_{IR} such that $\mathrm{cost}(O'_{\mathrm{IR}}) < \textsc{Opt}_{\mathrm{IR}}(I)$.

Construction of *O*: Consider any interval S. If S is accepted by $\textsc{Opt}_{\mathrm{IR}}$, set $v_S =$ `false` and otherwise set $v_S =$ `true`.

Towards (1): Since $\textsc{Opt}_{\mathrm{IR}}$ does not have intersecting intervals that are both accepted, all collision clauses are satisfied. Hence, $\mathrm{cost}(O)$ equals to the number of clauses of the form $(\overline{v_S} \vee \overline{v_S})$ that are not satisfied. The number of these corresponds exactly to the number intervals that have been rejected by $\textsc{Opt}_{\mathrm{IR}}$. Hence, $\mathrm{cost}(O) = \textsc{Opt}_{\mathrm{IR}}(I)$.

Towards (2): Observe that for any instance I of IR, $\mathrm{cost}(O) = \textsc{Opt}_{\mathrm{IR}}(I) \leqslant n-1$, where n is the number of intervals, as $\textsc{Opt}_{\mathrm{IR}}$ can always accept at least one interval.

Assume, that there exists a solution O' to the instance I' such that $\mathrm{cost}(O') < \mathrm{cost}(O)$. Let k be the number of collision clauses that are not satisfied by O'. Since $\mathrm{cost}(O') < \mathrm{cost}(O)$, it follows that $k < \mathrm{cost}(O) < n$.

Until all collision clauses are satisfied, modify O' as follows. Let $C = (v_{S_i} \vee v_{S_j})$ be any unsatisfied collision clause. Since C is unsatisfied, then $v_{S_i} = v_{S_j} =$ `false`. Set $v_{S_i} =$ `true` instead, such that C becomes satisfied, and $(\overline{v_{S_i}} \vee \overline{v_{S_i}})$ becomes unsatisfied. Observe that this change cannot make the cost of the solution increase, as v_{S_i} only occurs negated in the clause $(\overline{v_{S_i}} \vee \overline{v_{S_i}})$. It may, however, make the cost smaller, in case there are other unsatisfied collision clauses that contain v_{S_i}. Since each unsatisfied collision clause features two variables that are both `false`, this modified solution is well-defined.

After at most k repetitions, all collision clauses that were in O' will be satisfied, and we therefore have a subset of at most $\mathrm{cost}(O')$ unsatisfied clauses of the form $(\overline{v_S} \vee \overline{v_{S}})$. Since all collision clauses are satisfied, these at most $\text{cost}(O')$ variables imply a solution, O'_{IR}, to IR in which no two accepted intervals intersect, satisfying that

$$\text{cost}(O'_{\text{IR}}) \leqslant \text{cost}(O') < \text{cost}(O) = \textsc{Opt}_{\text{IR}}(I).$$

This finishes (2).

Having established (1) and (2), we have shown that x' encodes an optimal solution to I', making I' a valid instance. Further, this shows that Equation (7) is satisfied, finishing the analysis of (b).

Having established (A)–(C), $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ is an online reduction from IR to 2-SatD preserving competitiveness with respect to any pair of error measures (η_0, η_1). □

Corollary 40 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, ASG_t is as hard as 2-SatD and, by Theorem 39, ASG_{t+1} is as hard as 2-SatD. By transitivity, ASG_t is as hard as ASG_{t+1}, contradicting Lemma 18.(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 [27], though he referred to it as the set cover problem. Previous work has concluded that Dominating Set is W[2]-complete [21] and APX-hard [16]. Further, Online Dominating Set is studied in [13] and shown to be AOC-complete in [14].

We study an online variant of Dominating Set with predictions.

Definition 41 *Online Dominating Set with Predictions* (Dom) is a vertex-arrival problem. An algorithm, Alg, outputs $y_i = 1$ to accept v_i into its dominating set, and $\{v_i \mid x_i = 1\}$ is an optimal dominating set. Given an instance $I \in \mathcal{I}_{\textsc{Dom}}$,

$$\textsc{Alg}(I) = \begin{cases} \sum_{i=1}^{n} y_i, & \text{if } \textsc{Alg}\text{'s output is a dominating set,} \\ \infty, & \text{otherwise.} \end{cases}$$

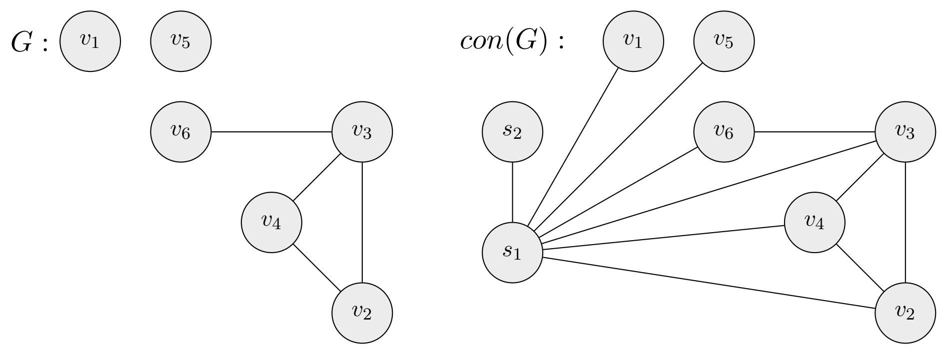

Figure 8: Connectified graph.

□

In our reduction from VC to DOM, we need that the instance of VC is a connected graph. Unlike in an NP-completeness reduction, from VC to DOM we cannot detect that a vertex is isolated immediately, and simply not include it in an online reduction. In the following, we give a method for "connectifying" a (possibly) disconnected graph, while controlling the size of the optimal vertex cover. This is an easy procedure that has been considered before [2]. However, we have not found it in refereed works, so for completeness, we give the argument below. Formally, we let *con* be the map which associates to a graph $G = (V, E)$ a connected graph $con(G) = (V', E')$, 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 8 for an example of $con(G)$.

Lemma 42 Let G be any graph.

(i) If G contains a vertex cover of size k, then $con(G)$ contains a vertex cover of size $k + 1$.

(ii) If $con(G)$ contains a vertex cover of size $k+1$, then G contains a vertex cover of size at most k.

Proof Towards (i): Assume that G contains a vertex cover, C, of size k. Since $E' = E \cup \{(s_1, s_2)\} \cup \{(v, s_1) \mid v \in V\}$, it follows that all edges that may not currently be covered have s_1 as one of their endpoints. Hence $C \cup \{s_1\}$ is a vertex cover of $con(G)$, that has size $k + 1$.

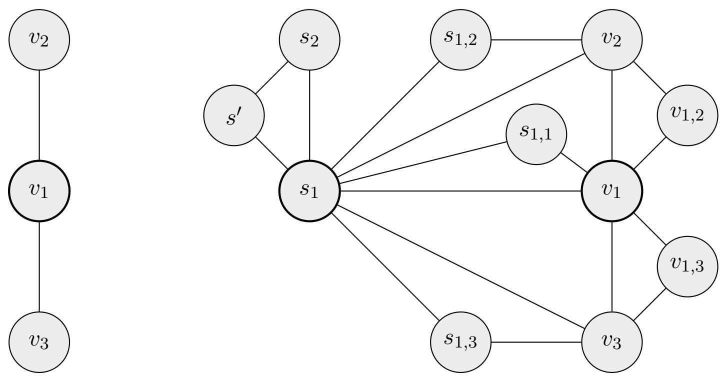

Figure 9: Example reduction for dominating set. The vertices with thicker boundary constitute an optimal vertex cover in the left graph, and the corresponding optimal dominating set in the right graph.

Towards (ii): Suppose that $con(G)$ has a vertex cover, C', of size $k+1$. Let $e \in E$ be any edge. Since e's endpoints are in V, and thus not s_1 nor s_2, the vertex that covers e is contained in $C' \setminus \{s_1, s_2\}$. Hence, $C' \setminus \{s_1, s_2\}$ is a vertex cover for G. Further, C' contains either s_1 or s_2, since $(s_1, s_2) \in E'$. Hence, $C' \setminus \{s_1, s_2\}$ is a vertex cover of G, and $|C' \setminus \{s_1, s_2\}| \leqslant |C'| - 1 = k$. □

Lemma 43 Let C be an optimal vertex cover for G. Then $C \cup \{s_1\}$ is an optimal vertex cover for $con(G)$.

Proof Let $k = |C|$. Then, by Lemma 42, $C \cup \{s_1\}$ is a valid vertex cover for $con(G)$, of size $k+1$. Assume, towards contradiction, that $con(G)$ contains a vertex cover, C', of size at most k. Then, by Lemma 42, G contains a vertex cover of size at most $k-1 < k = |C|$, contradicting the optimality of C. □

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

To describe our most central result on DOM, we introduce a slightly weaker notion of hardness:

Definition 44 Let $\mathcal{C}$ be a complexity class of online problems with binary predictions, and let Q be a $\mathcal{C}$-complete problem. Consider an online problem, P, with binary predictions. If any (α, β, γ)-competitive Pareto-optimal algorithm for P, with $\alpha \in o(\text{OPT}_P)$, implies the existence of an (α, β, γ)-competitive algorithm for Q, then P is said to be *weakly $\mathcal{C}$-hard.* □

Theorem 45 Let $t \in \mathbb{Z}^+$, let (η_0, η_1) be any pair of insertion monotone error measures. Then DOM is weakly $\mathcal{C}^t_{\eta_0,\eta_1}$-hard.

Proof We show that there exists a tuple $\rho = (\rho_{\text{A}}, \rho_{\text{R}})$ similar to an online reduction, that satisfies Conditions (O1), (O3), and (O4) from Definition 13, but not (O2). Instead of (O2), ρ satisfies that for any instance $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{VC}}$ of VC and any algorithm $\text{ALG}' \in \mathcal{A}_{\text{DOM}}$ for DOM,

$$\text{OPT}_{\text{DOM}}(\rho_{\text{R}}(\text{ALG}', I)) \leqslant \text{OPT}_{\text{VC}}(I) + 1. \tag{O2'}$$

Hence, the reduction we create here does not preserve the competitiveness of any algorithm for DOM, but only for the algorithms with $\alpha \in o(\text{OPT}_{\text{DOM}})$. To see that this reduction does preserve the competitiveness of algorithms with $\alpha \in o(\text{OPT}_{\text{DOM}})$, let $\text{ALG}' \in \mathcal{A}_{\text{DOM}}$ be any (α, β, γ)-competitive algorithm for DOM with additive term $\kappa \in o(\text{OPT}_{\text{DOM}})$. Then, assuming the existence of the above reduction ρ satisfying (O1), (O2'), (O3), and (O4), and letting $\text{ALG} = \rho_{\text{A}}(\text{ALG}')$ and $I' = \rho_{\text{R}}(\text{ALG}', I)$ for any $I \in \mathcal{I}_{\text{VC}}$,

$$\begin{aligned}
\text{ALG}(I) &\leqslant \text{ALG}'(I') + k_{\text{A}}(I), \text{ by (O1)} \\
&\leqslant \alpha \cdot \text{OPT}_{\text{DOM}}(I') + \beta \cdot \varphi_0(I') + \gamma \cdot \varphi_1(I') + \kappa(I') + k_{\text{A}}(I) \\
&\leqslant \alpha \cdot (\text{OPT}_{\text{VC}}(I) + 1) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) \\
&\quad + \kappa(I') + k_{\text{A}}(I), \text{ by (O2), (O3), and (O4)} \\
&= \alpha \cdot \text{OPT}_{\text{VC}}(I) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) + \alpha + \kappa(I') + k_{\text{A}}(I).
\end{aligned}$$

Hence, if $\alpha \in o(\text{OPT}_{\text{DOM}})$, we find that $\alpha + \kappa + k_{\text{A}} \in o(\text{OPT}_{\text{VC}})$ (by similar arguments as in Lemma 54 in Appendix A), implying that ALG is (α, β, γ)-competitive.

To prove the existence of ρ, pick any instance I for VC, with associated graph $G = (V, E)$, and then create an instance I', with associated graph G', for DOM based on $con(G)$ as follows:

- Reveal the vertex s_1, with true and predicted bit 1.
- Reveal the vertex s_2 together with the edge (s_1, s_2), with true and predicted bit 0.
- Reveal the vertex s' together with the edges (s_1, s') and (s_2, s'), with true and predicted bit 0.
- When receiving a request v_i as part of the VC instance:
  (1) Reveal a new vertex v_i together with all edges $(v_j, v_i) \in E$, for $j < i$, and the edge (s_1, v_i). The true bit for v_i is x_i, and the predicted bit for v_i is $\hat{x}_i$.

(2) Reveal a new vertex $s_{1,i}$, together with the edges $(s_1, s_{1,i})$ and $(s_{1,i}, v_i)$, with true and predicted bit 0.

(3) For each edge $(v_j, v_i) \in E$, for $j < i$, reveal a new vertex $v_{i,j}$ together with the edges $(v_i, v_{i,j})$ and $(v_j, v_{i,j})$, with true and predicted bit 0.

See Figure 9 for an example. Observe that if G contains n vertices and m edges, then G' contains $2n+m+3$ vertices and $3(n+m+1)$ edges. In short, we create the graph $con(G)$ based on G, and then transform $con(G)$ into G' using the well-known reduction from vertex cover to dominating set [31] that is often used when proving that dominating set is NP-complete.

By Lemma 42, if x encodes an optimal vertex cover in G, then adding a 1 as the true bit of s_1 and a 0 as the true bit of s_2, and not changing the true bit of any other vertex, this new bit string encodes an optimal vertex cover in $con(G)$, by Lemma 43. By [31] an optimal vertex cover in $con(G)$ encodes an optimal dominating set in G'.

Hence, $\text{OPT}_{\text{DOM}}(I') = \text{OPT}_{\text{VC}}(I)+1$, for any instance $I \in \mathcal{I}_{\text{VC}}$. This implies that Condition (O2') is satisfied. Further, since we only add correctly predicted requests, and (η_0, η_1) is a pair of insertion monotone error measures, Conditions (O3) and (O4) are also satisfied. Hence, it only remains to check Condition (O1). To this end, let $\text{ALG}' \in \mathcal{A}_{\text{DOM}}$ be any algorithm. Then, the strategy of $\rho_{\text{A}}(\text{ALG}')$ is given in Algorithm 6.

By construction of $\text{ALG} = \rho_{\text{A}}(\text{ALG}')$, it accepts the vertex v_i if and only if ALG' has accepted one of the vertices that have been revealed to ALG' in Lines 8–11 in Algorithm 6, by the check in Lines 12–15 in Algorithm 6. That is, ALG accepts v_i if and only if one of the vertices that has been revealed after v_i and before v_{i+1} as part of the instance of DOM has been accepted by ALG', making ALG' incur cost at least one as well. Observe that $v_{i,j}$, for each $(v_i, v_j) \in E$ with $j < i$, is revealed after v_i and before v_{i+1}. Hence, the cost that ALG incurs is bounded by the cost that ALG' incurs if and only if whenever ALG creates an infeasible solution, then so does ALG'.

To this end, assume that there exists an edge $e = (v_j, v_i) \in E$ such that neither v_i nor v_j has been accepted by ALG. In this case, neither v_i, v_j, nor $v_{i,j}$ has been accepted by ALG'. Since the only vertices adjacent to $v_{i,j}$ in G' are v_i and v_j, then $v_{i,j}$ is not dominated, and so ALG' has created an infeasible solution.

This verifies Condition (O1), and so concludes the proof. □

Algorithm 6

Input: An algorithm, $\text{ALG}' \in \mathcal{A}_{\text{DOM}}$, and a VC instance $I = (x, \hat{x}, r)$
Output: An instance of DOM and an algorithm for VC
Request s_1, with true and predicted bit 1
Request s_2, together with (s_1, s_2), with true and predicted bit 0
Request s', together with (s_1, s') and (s_2, s'), with true and predicted bit 0
while receiving requests v_i **do**
  Get prediction $\hat{x}_i$
  Request the vertex v_i together with all edges $(v_j, v_i) \in E$, for $j < i$, and let y'_{v_i} be the output of ALG'
  Request the vertex $s_{1,i}$ together with the edges $(v_i, s_{1,i})$ and $(s_1, s_{1,i})$, with true and predicted bit 0, and let $y'_{s_{1,i}}$ be the response of ALG'
  for $(v_j, v_i) \in E$ with $j < i$ **do**
    Request the vertex $v_{i,j}$ together with the edges $(v_i, v_{i,j})$ and $(v_j, v_{i,j})$, with true and predicted bit 0, and let $y'_{v_{i,j}}$ be the response of ALG'
  if there exists a true bit in $\{y_{v_i}, y_{s_i}\} \cup \{y_{v_{i,j}} \mid (v_j, v_i) \in E \text{ with } j < i\}$ **then**
    Output 1
  else
    Output 0

We summarize our results about DOM in Theorem 46:

Theorem 46 Let $t \in \mathbb{Z}^+$. For all pairs of insertion monotone error measures (η_0, η_1),

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

For all pairs of error measures (η_0, η_1),

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

(iii) $\text{DOM} \in \mathcal{C}_{\eta_0,\eta_1}$.

Proof **Towards (i):** This follows by Theorem 45.

Towards (ii): The algorithm, $\text{ACC} \in \mathcal{A}_{\text{DOM}}$, that accepts all vertices is $(n, 0, 0)$-competitive, as OPT_{DOM} has to accept at least one vertex to create a dominating set. Now, adapt the proof of Theorem 31.(vi).

Towards (iii): Adapt the setup from the proof of Theorem 31.(v), with the following addition. Observe that 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, ALG $\in \mathcal{A}_{\text{ASG}}$ has guessed 0 on a true 1, and so ALG$(x, \hat{x}) = \infty$. $\square$

7 Establishing Upper and Lower Bounds for Problems Related to $\mathcal{C}^t_{\mu_0,\mu_1}$

In this section, we show a method for proving lower bounds on the competitiveness of all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems. In [4], Antoniadis et al. prove strong lower bounds for online algorithms with predictions for Paging with Discard Predictions (PAG$_t$), a binary prediction scheme, with respect to (μ_0, μ_1). In the next subsection, we recall the definition of the problem PAG$_t$ from [4], and prove the existence of a strict online reduction from PAG$_t$ to ASG$_t$, which extends the lower bounds from [4] to all $\mathcal{C}^t_{\mu_0,\mu_1}$-hard problems and upper bounds from ASG$_t$ to PAG$_t$.

7.1 A Reduction from Paging to ASG$_t$

In Paging, we have a *universe*, U, of N pages, and a *cache* of a fixed size, $k < N$. 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 *miss*), 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 misses. 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 miss, 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, 36]. We study a paging problem with succinct predictions [4]:

Definition 47 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. Following [4], if the page in r_i is never requested again, we set $x_i = 0$ if LFD keeps the page in cache until all requests has been seen, and $x_i = 1$ otherwise. □

In [4], Antoniadis et al. introduces an unnamed algorithm, that we call FlushWhenAll0s, for Pag_k whose strategy 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 FlushWhenAll0s with respect to the pair of error measures (μ_0, μ_1) defined in Definition 2 and use the algorithm to establish a relation between Paging and ASG_t:

Theorem 48 ([4]) For any instance, $I = (x, \hat{x}, r)$ of Pag_k,

$$\text{FlushWhenAll0s}(I) \leqslant \text{Opt}_{\text{Pag}_k}(I) + (k-1) \cdot \mu_0(I) + \mu_1(I).$$

That is, FlushWhenAll0s is strictly $(1, k-1, 1)$-competitive for Pag_k with respect to (μ_0, μ_1).

We use FlushWhenAll0s to establish a relation between Paging and ASG_t:

Theorem 49 For all $t \in \mathbb{Z}^+$, $\text{Pag}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$.

Proof We define a strict online reduction $\rho \colon \text{Pag}_t \xrightarrow{\text{red}} \text{ASG}_t$ with $k_{\text{A}} = 0$. To this end, consider any instance, $I = (x, \hat{x}, r) \in \mathcal{I}_{\text{Pag}_t}$ and any algorithm $\text{Alg}' \in \mathcal{A}_{\text{ASG}_t}$. We construct an instance, $I' = (x', \hat{x}', r') \in \text{ASG}_t$, 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 (O3) and (O4) are both satisfied.

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

$$\text{Opt}_{\text{Pag}_t}(I) = t + \sum_{i=1}^{n} x_i. \tag{8}$$

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

Towards (O1): Firstly, by definition of I', and letting y' be the output of $\text{ALG}'(I')$,

$$\text{ALG}'(I') \geqslant \sum_{i=1}^{n} \left(y'_i + t \cdot x_i \cdot (1 - y'_i)\right) + t. \tag{9}$$

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

We give pseudocode for ρ given $\text{ALG}' \in \mathcal{A}_{\text{ASG}_t}$ and $I \in \mathcal{I}_{\text{PAG}_t}$ in Algorithm 7. Formally, $\text{ALG} = \rho_{\text{A}}(\text{ALG}')$ runs the algorithm FLUSHWHENALL0S from [4] where it uses y'_i as the prediction for r_i. Hence, letting $y'(n) = \langle y'_1, y'_2, \ldots, y'_n \rangle$ and $I_{y'} = (x, y'(n), r)$,, we have that $\text{ALG}(I) = \text{FLUSHWHENALL0S}(I_{y'})$. Therefore,

$$\begin{aligned}
&\text{ALG}(I)\\
\leqslant\ & \text{OPT}_{\text{PAG}_t}(I_{y'}) + (t-1) \cdot \mu_0(I_{y'}) + \mu_1(I_{y'}), \text{ by Theorem 48}\\
=\ & t + \sum_{i=1}^{n} x_i + (t-1) \cdot \sum_{i=1}^{n} (1 - y'_i) \cdot x_i + \sum_{i=1}^{n} y'_i \cdot (1 - x_i), \text{ by (8)}\\
=\ & t + \sum_{i=1}^{n} \left(t \cdot x_i \cdot (1 - y'_i) + y'_i\right)\\
\leqslant\ & \text{ALG}'(I'), \text{ by (9)}.
\end{aligned}$$

□

Since $\text{PAG}_t \in \mathcal{C}^t_{\mu_0,\mu_1}$, 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. However, in [4] the additive terms are restricted to constants rather than sublinear terms. The next lemma handles this difference.

Lemma 50 Let P be any online minimization problem with binary predictions and let (η_0, η_1) be a pair of error measures. If $\text{ALG} \in \mathcal{A}_P$ is (α, β, γ)-competitive with respect to (η_0, η_1) then, for all $0 < \delta < \alpha$, there exists $b_\delta \in \mathbb{R}$ such that ALG is $(\alpha + \delta, \beta, \gamma)$-competitive with respect to (η_0, η_1) with additive term b_δ.

Algorithm 7

Input: an instance $(x, \hat{x}, r) \in \mathcal{I}_{\text{PAG}_t}$ and an algorithm, $\text{ALG}' \in \mathcal{A}_{\text{ASG}_t}$
Output: an instance, $(x', \hat{x}', r')$, of ASG_t and a paging strategy for r
while receiving requests r_i **do**
  Get prediction $\hat{x}_i$
  Ask ALG' to guess the next bit given $\hat{x}'_i \leftarrow \hat{x}_i$, and let y'_i be its output

  if r_i is not in cache **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 $i \leftarrow 1$ **to** t **do**
  Ask ALG' to guess the next bit given $\hat{x}'_{n+i} = 1$
Compute $x \leftarrow \text{LFD}(r)$ and reveal x' which is x appended by t 1s to ALG

Proof Since ALG is (α, β, γ)-competitive, there exists $\kappa \in o(\text{OPT}_P)$ such that for all $I \in \mathcal{I}_P$,

$$\text{ALG}(I) \leqslant \alpha \cdot \text{OPT}_P(I) + \eta_0(I) + \eta_1(I) + \kappa(I).$$

Since $\kappa \in o(\text{OPT}_P)$, then

$$\forall \delta > 0 \colon \exists b_\delta \colon \forall I \in \mathcal{I}_P \colon \kappa(I) < \delta \cdot \text{OPT}_P(I) + b_\delta.$$

Let $\delta > 0$, and compute $b_\delta \in \mathbb{R}$. Then, for any $I \in \mathcal{I}_P$,

$$\begin{aligned}\text{ALG}(I) &\leqslant \alpha \cdot \text{OPT}_P(I) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) + \kappa(I)\\ &< \alpha \cdot \text{OPT}_P(I) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) + \delta \cdot \text{OPT}_P(I) + b_\delta\\ &= (\alpha + \delta) \cdot \text{OPT}_P(I) + \beta \cdot \eta_0(I) + \gamma \cdot \eta_1(I) + b_\delta.\end{aligned}$$

Hence, ALG is $(\alpha + \delta, \beta, \gamma)$-competitive with the additive term in $\mathbb{R}$. □

Theorem 51 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) $\alpha + \beta \geqslant t$,

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

Proof By Theorem 49, ASG_t is as hard as PAG_t. Since P is $\mathcal{C}^t_{\mu_0,\mu_1}$-hard, we have that P is as hard as ASG_t. Hence, by transitivity of the as-hard-as relation, P is as hard as 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 $\alpha + \beta < t$. If ALG is not Pareto-optimal, then there exists an $(\alpha', \beta', \gamma')$-competitive Pareto-optimal algorithm, $\text{ALG}' \in \mathcal{A}_P$, with $\alpha' \leqslant \alpha$, $\beta' \leqslant \beta$, and $\gamma' \leqslant \gamma$.

Since P is as hard as PAG_t, and ALG' is Pareto-optimal, we get an $(\alpha', \beta', \gamma')$-competitive algorithm for PAG_t with $\alpha' + \beta' \leqslant \alpha + \beta < t$, say $\text{ALG}'_{\text{PAG}_t}$. Since the additive term of $\text{ALG}'_{\text{PAG}_t}$ may be sublinear in $\text{OPT}_{\text{PAG}_t}$, pick $\delta > 0$ small enough such that $\alpha + \beta + \delta < t$. Then, by Lemma 50, $\text{ALG}'_{\text{PAG}_t}$ is $(\alpha' + \delta, \beta', \gamma')$-competitive with an additive constant. 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], thus proving (ii). □

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

Theorem 52 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 49 there exists a strict online reduction $\rho\colon \text{PAG}_t \xrightarrow{\text{red}} \text{ASG}_t$. Moreover, by Theorem 6.(b), 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 $\alpha + \beta \geqslant t$, say $\text{ALG}_{\alpha,\beta}$. Finally, by Lemma 14, the existence of $\text{ALG}_{\alpha,\beta}$ together with the existence of ρ, implies the existence of an $(\alpha, \beta, 1)$-competitive algorithm for PAG_t, $\rho_{\text{A}}(\text{ALG}_{\alpha,\beta})$.

□

8 Concluding Remarks and Future Work

We have defined complexity classes for online minimization problems with and without binary 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 Binary Predictions.

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 binary predictions are considered and several maximization problems are shown to be members, hard and complete for $\mathcal{C}^t_{\mu_0,\mu_1}$ [8].

Possible directions for future work include considering randomization and changing the hardness measure from competitiveness to, e.g., relative worst order or random order.

References

[1] Algorithms with predictions. https://algorithms-with-predictions.github.io/. Accessed: 2024-01-19.

[2] Reduction from vertex cover to dominating set. https://cs.stackexchange.com/questions/117567/reduction-from-vertex-cover-to-dominating-set. Accessed: 2024-01-19.

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

[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. PMLR, 2023.

[5] Andrew W. Appel. *Modern Compiler Implementation in C*. Cambridge University Press, 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, Berlin, Heidelberg, 1999.

[7] Laszlo A. Belady. A study of replacement algorithms for virtual-storage computer. *IBM Syst. J.*, 5:78–101, 1966.

[8] Magnus Berg. Comparing the hardness of online minimization and maximization problems with predictions, 2024. arXiv:2409.12694.

[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 Královic, Richard Královic, and Tobias Mömke. On the advice complexity of online problems. In *Algorithms and Computation, 20th International Symposium, ISAAC*, volume 5878, pages 331–340. Springer, 2009.

[11] Allan Borodin, Joan Boyar, Kim S. Larsen, and Denis Pankratov. Advice complexity of priority algorithms. *Theory of Comuting 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] Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholdt, Michal Kotrbčík, and Kim S. Larsen. Online Dominating Set. *Algorithmica*, 81(5):1938–1964, 2019.

[14] 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.

[15] Miroslav Chlebík and Janka Chlebíkova. On approxmiation hardness of the minimum 2SAT-deletion problem. *Discrete Applied Mathematics*, 155:172–179, 2007.

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

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

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

[19] Stefan Dobrev, Rastislav Královic, and Dana Pardubská. Measuring the problem-relevant information in input. *RAIRO Theor. Informatics Appl.*, 43:585–613, 2009.

[20] Rodney G. Downey and Michael R. Fellows. *Parametrized Complexity*. Springer, New York, NY, USA, 1999.

[21] Rodney G. Downey and Micheal R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on Computing*, 24:873–921, 1995.

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

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

[24] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, USA, 1990.

[25] Monika Henzinger, Barna Saha, Martin P. Seybold, and Christopher Ye. On the complexity of algorithms with predictions for dynamic graph problems, 2023. arXiv:2307.16771.

[26] Juraj Hromkovic, Rastislav Královic, and Richard Královic. Information complexity of online problems. In *Mathematical Foundations of Computer Science 2010 (MFCS)*, volume 6281, pages 24–36. Springer, 2010.

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

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

[29] Dennis Komm. *An Introduction to Online Computation: Determinism, Randomization, Advice*. Springer Cham, Switzerland, 2016.

[30] Meena Mahajan and Venkatesh Raman. Parametrizing above guaranteed values: Maxsat and maxcut. *Journal of Algorithms*, 31:335–354, 1999.

[31] Udi Manber. *Introduction to algorithms - a creative approach*. Addison-Wesley Longman Publishing Co., Inc., Bosten, MA, USA, 1989.

[32] Jesper W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation, 2015. arXiv:1511.05886.

[33] 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 2016)*, pages 39:1–39:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

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

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

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

A Some Results on $o(\textsc{Opt})$

Lemma 53 Let P be an online minimization problem, $\kappa, \kappa' \in o(\textsc{Opt})$, and $k \geqslant 0$. Then,

(i) $\kappa + \kappa' \in o(\textsc{Opt})$, and

(ii) $k \cdot \kappa \in o(\textsc{Opt})$,

where $(\kappa + \kappa')(I) = \kappa(I) + \kappa'(I)$ and $(k \cdot \kappa)(I) = k \cdot \kappa(I)$.

Proof Since $\kappa \in o(\textsc{Opt})$, we have that

$$\forall \delta > 0 \colon \exists b_\delta \colon \forall I \in \mathcal{I}_P \colon \kappa(I) < \delta \cdot \textsc{Opt}(I) + b_\delta. \tag{10}$$

Since $\kappa' \in o(\textsc{Opt})$, we have that

$$\forall \delta' > 0 \colon \exists b_{\delta'} \colon \forall I \in \mathcal{I}_P \colon \kappa'(I) < \delta' \cdot \textsc{Opt}(I) + b_{\delta'}. \tag{11}$$

Towards (i): We have to show that

$$\forall \delta_0 > 0 \colon \exists b_{\delta_0} \colon \forall I \in \mathcal{I}_P \colon (\kappa + \kappa')(I) < \delta_0 \cdot \textsc{Opt}(I) + b_{\delta_0}.$$

Let $\delta_0 > 0$, and set $\delta = \delta' = \frac{\delta_0}{2}$. Then there exists two constants b_δ and $b_{\delta'}$. Set $b_{\delta_0} = b_\delta + b_{\delta'}$, then

$$\begin{aligned} (\kappa + \kappa')(I) &= \kappa(I) + \kappa'(I) \\ &\leqslant \frac{\delta_0}{2} \cdot \textsc{Opt}(I) + b_\delta + \frac{\delta_0}{2} \cdot \textsc{Opt}(I) + b_{\delta'} \\ &= \delta_0 \cdot \textsc{Opt}(I) + b_{\delta_0}. \end{aligned}$$

Towards (ii): We have to show that

$$\forall \delta_1 > 0 \colon \exists b_{\delta_1} \colon \forall I \in \mathcal{I}_P \colon (k \cdot \kappa)(I) < \delta_1 \cdot \textsc{Opt}(I) + b_{\delta_1}.$$

Let $\delta_1 > 0$, set $\delta = \frac{\delta_1}{k}$, and let $b_{\delta_1} = k \cdot b_\delta$. Then, for any $I \in \mathcal{I}_P$,

$$\begin{aligned} (k \cdot \kappa)(I) = k \cdot \kappa(I) &< k \cdot (\delta \cdot \textsc{Opt}(I) + b_\delta) \\ &= k \cdot \frac{\delta_1}{k} \cdot \textsc{Opt}(I) + b_{\delta_1} = \delta_1 \cdot \textsc{Opt}(I) + b_{\delta_1}. \end{aligned}$$

Hence, $k \cdot \kappa \in o(\textsc{Opt})$. □

Lemma 54 Let P and Q be online minimization problems, and let $\rho \colon P \xrightarrow{\text{red}} Q$ be a strict online reduction with reduction term k_{A}. If $\kappa_Q \in o(\textsc{Opt}_Q)$, then $\kappa_P \in o(\textsc{Opt}_P)$, where, for any $\textsc{Alg} \in \mathcal{A}_Q$ we let $\kappa_P(I_P) = \kappa_Q(\rho_{\text{R}}(\textsc{Alg}, I_P))$.

Proof Since $\kappa_Q \in o(\text{OPT}_Q)$, then

$$\forall \delta_Q > 0 \colon \exists b_{\delta_Q} \colon \forall I_Q \in \mathcal{I}_Q \colon \kappa_Q(I_Q) < \delta_Q \cdot \text{OPT}_Q(I_Q) + b_{\delta_Q}.$$

Since $k_{\text{O}} \in o(\text{OPT}_P)$, then

$$\forall \delta_P > 0 \colon \exists b_{\delta_P} \colon \forall I_P \in \mathcal{I}_P \colon k_{\text{O}}(I_P) < \delta_P \cdot \text{OPT}_P(I_P) + b_{\delta_P}.$$

Since $\rho \colon P \xrightarrow{\text{red}} Q$ is a strict online reduction, (O2) implies that

$$\text{OPT}_Q(\rho_{\text{R}}(\text{ALG}, I_P)) \leqslant \text{OPT}_P(I_P),$$

for all $\text{ALG} \in \mathcal{A}_Q$ and all $I_P \in \mathcal{I}_P$.

To see that $\kappa_P \in o(\text{OPT}_P)$, we have to verify that

$$\forall \delta > 0 \colon \exists b_\delta \colon \forall I_P \in \mathcal{I}_P \colon \kappa_P(I_P) < \delta \cdot \text{OPT}_P(I_P) + b_\delta.$$

For any $\delta > 0$, let $\delta_Q = \delta$, and let $b_\delta = b_{\delta_Q}$. Then, letting $I_Q = \rho_{\text{R}}(\text{ALG}, I_P)$ for any $\text{ALG} \in \mathcal{A}_Q$ and any $I_P \in \mathcal{I}_P$, we get that

$$\kappa_P(I_P) = \kappa_Q(I_Q) \leqslant \delta_Q \cdot \text{OPT}_Q(I_Q) + b_{\delta_Q} \leqslant \delta \cdot \text{OPT}_P(I_P) + b_\delta.$$

□

B A List of Insertion Monotone Error Measures

For completeness, we include a non-exhaustive list of pairs of insertion monotone error measures.

(IM1) (μ_0, μ_1) from Definition 2.

(IM2) (Z_0, Z_1) from Section 4.2.

(IM3) A pair of error measures, $(\mu_0^{\text{rel}}, \mu_1^{\text{rel}})$, that are an adaptation of the error measures from Definition 2, taking into account the sequence length, given by

$$\mu_b^{\text{rel}}(x, \hat{x}) = \frac{\mu_b(x, \hat{x})}{n}.$$

(IM4) A weighted variant of (μ_0, μ_1), denoted (μ_0^w, μ_1^w), taking into account how far into the sequence the error occurs, weighting it such that the prediction errors are weighted higher if they occur earlier:

$$\mu_0^w(x, \hat{x}) = \sum_{i=1}^{n} \left(\frac{n-i}{n} \cdot ((1 - x_i) \cdot \hat{x}_i) \right)$$
$$\text{and } \mu_1^w(x, \hat{x}) = \sum_{i=1}^{n} \left(\frac{n-i}{n} \cdot (x_i \cdot (1 - \hat{x}_i)) \right).$$

(IM5) A pair of error measures, $(\mathcal{L}_0^p, \mathcal{L}_1^p)$, where each error measure is an adaptation of the $\mathcal{L}^p$-measure, for $p \in [1, \infty)$, given by:

$$\mathcal{L}_0^p(x, \hat{x}) = \sqrt[p]{\sum_{i=1}^{n} (1 - x_i) \cdot |x_i - \hat{x}_i|^p}$$
$$\text{and } \mathcal{L}_1^p(x, \hat{x}) = \sqrt[p]{\sum_{i=1}^{n} x_i \cdot |x_i - \hat{x}_i|^p}.$$

(IM6) A pair of error measures $(\mathcal{L}_0^\infty, \mathcal{L}_1^\infty)$, where each error measure is an adaptation of the $\mathcal{L}^\infty$-measure, given by

$$\mathcal{L}_0^\infty(x, \hat{x}) = \sup_i \{(1 - x_i) \cdot \hat{x}_i\}$$
$$\text{and } \mathcal{L}_1^\infty(x, \hat{x}) = \sup_i \{x_i \cdot (1 - \hat{x}_i)\}.$$

Beyond the above list, the sum of insertion monotone error measures is insertion monotone, the product of two non-negative insertion monotone error measures is insertion monotone, a constant times an insertion monotone error measure is insertion monotone.