Online Algorithms with Unreliable Guidance

Julien Dallot
TU Berlin, Germany

Yuval Emek
Technion, Israel

Yuval Gil
Reykjavik University, Iceland

Maciej Pacut
TU Berlin, Germany

Stefan Schmid
TU Berlin and Weizenbaum Institute, Germany

Abstract

This paper introduces *online algorithms with unreliable guidance (OAG)*, a model for ML-augmented online decision-making that cleanly separates the predictive and algorithmic components, thus offering a single, well-defined analysis framework that depends only on the problem at hand. Formulated through the lens of request-answer games, the OAG model brings multiple concepts (predictions from the answer space, guide, anytime competitiveness) which enable learning-augmented algorithms to be analyzed independently of predictor-specific choices—such as prediction semantics, error functions, or probing strategies—that would otherwise restrict the algorithm's generality and applicability. The clean framework of the OAG model allows to build the first generic compiler, the *drop-or-trust-blindly (DTB) compiler*, that turns almost any standard, prediction-free online algorithm into a learning-augmented one. Although simple, we show that the DTB compiler produces new learning-augmented algorithms with strong consistency-robustness guarantees for three classic online problems: we achieve new trade-offs for *bipartite matching* with adversarial arrival order, and obtain optimal solutions for *caching* and *uniform metrical task systems*.

2012 ACM Subject Classification Theory of computation → Machine learning theory; Theory of computation → Online learning algorithms

Keywords and phrases Learning-Augmented Algorithms, Online Algorithms, Online Bipartite Matching

Funding The research of J. Dallot and S. Schmid is supported by the German Research Foundation (DFG), Schwerpunktprogramm SPP 35 2378, ReNO-2 (511099228), 2025-2029. The research of Y. Emek is partially supported by the Israel Science Foundation (ISF), grant 730/24.

1 Introduction

Introduced in the influential ICML 2018 paper of Lykouris and Vassilvitskii [32], Learning-Augmented online algorithms refer to online algorithms which accept an additional argument, the *prediction*; the goal is to outperform traditional online algorithms in case of accurate predictions and still provide provable performance guarantees regardless of prediction accuracy. At their core, learning-augmented algorithms seek for the proper use of "guessing machines" — typically an ML predictor — to solve concrete problems, especially when the predictor is a complete black-box and the prediction is therefore unreliable by nature. Since their introduction, this framework gathered significant attention and was applied to multiple problems — the literature on learning-augmnented online algorithms is too extensive to list here; see [30] for a comprehensive record.

Over the years, multiple models have been proposed to describe and analyze learning-augmented algorithms. In essence, those models seek to optimize the same three goals:

(1) *consistency*, the competitive ratio when the predictions are perfect; (2) *robustness*, the competitive ratio when the predictions are arbitrarily bad; and (3) *smoothness*, the function describing the decline in the competitive ratio as predictions' quality degrades. While the community agrees on these three metrics (*what to measure*), their concrete implementations (*how to measure*) remain debated and differ across models. The key question is: *how can one measure the accuracy of a prediction while treating the predictor as a black box?* We review below how existing models interact with the predictor and the issues this raises.

Prediction semantics. Before designing a learning-augmented algorithm, one must fix a *prediction semantic* — that is, decide what a prediction represents and how many distinct predictions are possible. This choice is consequential: it can affect the power of a learning-augmented algorithm, as shown in [2] for the line search problem, where three different prediction semantics yield different optimal results. Yet no standard way to choose a prediction semantic exists, and the choice is left to the analyst's judgment. As a result, semantics vary across problems and sometimes even across algorithms for the same problem (e.g., predicting nodes' degrees [1] or the matched node [17] for online bipartite matching, predicting the cache content [4] or the page to evict [32] for caching). This inconsistency makes results for the same problem incomparable.

Error functions. Most existing learning-augmented online algorithms are analyzed using an *error function* [32], which acts as an intermediary metric to assess the quality of a prediction given a problem instance. Initially motivated by the loss function optimized during ML training, the error function is now widely used for analysis purposes, especially to derive crucial *smoothness* results, i.e., the competitive ratio for specific error values.

As of today, no consensus exists on how to choose an error function for a given problem; in practice, the choice of how to assess errors is largely left to the analyzer's judgment and justified by intuitive considerations. Consequently, a wide variety of error functions appear in the literature: some compare predictions against aspects of the problem instance [8, 33], others against an optimal offline algorithm [1, 6], and different functions are sometimes used for similar problems (e.g., for bipartite matching, sanctioning any error [17] and counting the number of matches that differ from the optimal solution [5], or for online caching, the absolute distance between real and predicted requests for each page [32] and the size of the xor between optimal and predicted cache contents [4]).

This lack of a common ground poses multiple issues: it prevents adequate comparisons between results or transfers of algorithmic ideas, leaves room for analysis-driven error functions that artificially advantage certain algorithms or do not properly account for realistic errors, and prevents the emergence of assumption-free impossibility results on the smoothness metric.

Predictor probing. In part to address the limitations of error functions, *quantitative* models were introduced, notably ϵ-Accurate Predictions [25] and Infused Advice [22]. Those models replace the error function with a *probability of failure* inspired by the success rate metric of ML predictors: each prediction is independently good or bad decided with coin tosses of a fixed probability. While more systematic and both problem- and predictor-agnostic, these models share two restrictive assumptions. First, bad predictions are random rather than adversarial[1], which cannot model structured errors such as mislabeling or representation

[1] As assumed in [25] for the caching problem.

errors. Second, the failure probability is fixed over the entire input sequence, which is unrealistic when predictor accuracy varies over time. Together, these assumptions incentivize *probing* strategies [25] that sample predictions to estimate the failure rate, and then exploit it — a strategy that breaks down if accuracy changes mid-sequence.

The starting point of this paper is that existing learning-augmented algorithms rely on non-general assumptions about the predictor, restricting the applicability of their guarantees. We therefore ask: *does there exist a model whose formal guarantees depend solely on the problem?*

1.1 The OAG Model

We answer the aforementioned question in the affirmative by introducing a new model, called *online algorithms with unreliable guidance (OAG)*, that stems directly from the formulation of online problems as *request-answer games* [12] and does not involve any "external ingredients" such as prediction semantics, error functions or probing strategies. The OAG model brings three main innovations: a systematic manner to model predictions, a new manner to model the predictor's accuracy with the *guide*, and a new manner to evaluate the algorithm's performance called the *anytime-competitive ratio*; refer to Section 2 for a formal definition of the OAG model.

Modeling predictions with guidance. We propose a universal standard for prediction semantics: predictions should take the form of answers the predictor would give if it were in the algorithm's place. We call this prediction semantic *guidance*. Concretely, this means predictions are drawn directly from the answer space of the problem — a natural choice, since (almost) all useful online problems can be modeled as request-answer games [12].

Modeling errors with a guide. Inspired by the success rate of an ML predictor, we model good and (adversarially) bad predictions with a *guide*. An OAG algorithm for an online problem $\mathcal{P}$ receives, with each incoming request ρ_t, a guidance γ_t taken from the answer space of $\mathcal{P}$. Ideally, γ_t is a *good guidance* that was generated knowing the entire request sequence and aiming to help the algorithm by selecting the optimal answer for ρ_t, however, γ_t may also be a *bad guidance* that selects the answer adversarially knowing the whole input sequence. The choice between the two guidance at time t is made by nature based on an independent β-biased coin toss, where $0 \leq \beta \leq 1$ is a model *bad guidance* parameter, so that γ_t is bad with probability β and good with probability $1 - \beta$. Similarly to the approach of [25] (where β is analogous to $1 - \epsilon$), consistency and robustness are obtained by setting $\beta = 0$ and $\beta = 1$, respectively, whereas smoothness arises naturally as β shifts from 0 to 1.

Modeling predictor's changes with anytime competitiveness. We introduce *anytime-competitiveness*, a metric that bounds performance against the offline optimum *on every subsequence of the input sequence*. Since guarantees hold for all subsequences and all values of β, algorithms have no incentive to probe the predictor: tuning the algorithm's behavior according to an estimated β would only hurt performance on the current interval for other predictor accuracies. This cleanly decouples the predictive and algorithmic parts by separating the acts of estimating the predictor's accuracy from the act of analyzing the algorithms' performance — a third party may still probe and, upon detecting a change in accuracy, adjust the influence of predictions in real time (e.g., by changing the trust parameter τ, see

next Subsection 1.2), with our framework directly supplying the relevant guarantees to take informed decisions.

Figure 1 A request sequence along which the predictor's accuracy varies between request subsequences. Anytime competitiveness ensures that the algorithm has provable performance guarantees for each of those subsequences. A third-party prober that keeps accuracy estimates can detect changes in the predictor's behavior and use anytime competitive guarantees to take an informed decision in real time.

1.2 Drop or Trust Blindly: Augmenting Any Algorithm with Predictions

Beyond its elegant analytical framework, the OAG model opens the gate for a generic method, referred to as the *drop or trust blindly (DTB) compiler*, that transforms, in a blackbox manner, any "standard" (a.k.a. *prediction-free*) online algorithm into a learning-augmented algorithm in the OAG model. In addition to the excellent performances of some of its output learning-augmented algorithms (see Subsection 1.3), the DTB compiler is the first proposal for a systematic manner to generate new learning-augmented algorithms to serve as a canonical reference point.

To see how the DTB compiler works, consider a prediction-free algorithm `Alg` and an incoming request ρ_t and let D_t be the distribution from which `Alg` picks the answer for ρ_t. Assuming that the incoming guidance γ_t belongs to the support of D_t, the OAG algorithm produced by the DTB compiler employs the following simple rule, where $0 < \tau < 1$ is the compiler's *trust parameter*: with probability τ, `Alg`* adopts γ_t blindly; with probability $1-\tau$, `Alg`* picks the answer from D_t (ignoring γ_t). Refer to Section 2 for a formal exposition of the DTB compiler.

On the face of it, the approach taken by the DTB compiler may be seen as "too simple", however, we prove that the resulting OAG algorithms actually provide attractive consistency-robustness guarantees for some classic and extensively studied online problems.

1.3 Our Technical Contributions

In addition to our conceptual contributions, we develop OAG algorithms for classic online problems by applying the DTB compiler to the following well known prediction-free online algorithms: the ranking algorithm of [28] for the *bipartite matching* problem with adversarial arrival order (Section 3); the random marking algorithm of [23] for the *caching* problem (Section 4); and the algorithm of [13] for the *uniform metrical task system* problem (Section 5). We analyze those algorithms in the OAG model and anytime competitiveness guarantees expressed with the bad guidance parameter β and the trust parameter τ. The consistency and robustness of our algorithms are obtained by setting $\beta = 0$ and $\beta = 1$, respectively; see Table 1. By modifying τ, our algorithms can be made more risky or more prudent with respect to trusting the guidance they receive.

Our results for online bipartite matching give the first non-trivial trade-off between consistency and robustness under adversarial arrival order [28]. Due to its many applications (most notably ads allocation), online bipartite matching has been intensively studied in the learning-augmented framework in multiple variants: random arrival order [17, 5], random graph [1], and two-stage arrival [27]. We are the first to derive non-trivial points on the

online problem	consistency	robustness
bipartite matching	$\frac{1-e^{-(1-\tau)}}{1-\tau}$	$\max\left\{\frac{1}{2}, 1-e^{-(1-\tau)}\right\}$
caching (cache size k)	$\min\left\{\frac{2}{\tau}, 2H_k\right\}$	$\min\left\{\frac{2H_k}{1-\tau}, k\right\}$
uniform metrical task system (n states)	$2+2\cdot\min\left\{\frac{1}{\tau}, (1-\tau)H_n\right\}$	$2+2\cdot\min\left\{\frac{1}{1-\tau}H_n, n-1\right\}$

Table 1 The consistency and robustness guarantees of our OAG algorithms, expressed in terms of the trust parameter $0 < \tau < 1$.

consistency-robustness Pareto frontier for the original version of the problem. For the online caching and uniform metrical task system problems, our algorithms (with constant $0 < \tau < 1$) admit asymptotically optimal constant consistency and logarithmic robustness, while being arguably simpler than the existing algorithms that have such guarantees (see, e.g.,[32]).

1.4 Model limitations

We see two limitations of the OAG model. First, it still assumes a probabilistic predictor (independent coin tosses per request) while the model was designed to avoid predictor-specific assumptions. Second, it only captures quantitative errors and cannot express qualitative ones as error functions do. Despite their drawbacks, error functions offer finer-grained error characterization — particularly for "one-shot" problems (e.g., ski rental [29]) where quantitative approaches give deceptively simple guarantees; the OAG model is best suited for problems with long input sequences.

2 Model

Request-Answer Games. Consider an online minimization (resp., maximization) problem $\mathcal{P}$ defined over a *request* space $\mathcal{R}$ and an *answer* space $\mathcal{A}$. An instance of $\mathcal{P}$ is given by a finite request sequence $\rho \in \mathcal{R}^*$; a solution for ρ is an answer sequence $\sigma \in \mathcal{A}^*$ satisfying $|\sigma| = |\rho|$. The quality of the solutions is measured by a *cost function* $f_{\mathcal{P}} : \bigcup_{\ell \geq 0} \mathcal{R}^\ell \times \mathcal{A}^\ell \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ (resp., a *payoff function* $f_{\mathcal{P}} : \bigcup_{\ell \geq 0} \mathcal{R}^\ell \times \mathcal{A}^\ell \to \mathbb{R}_{\geq 0} \cup \{-\infty\}$) that determines the cost (resp., payoff) $f_{\mathcal{P}}(\rho, \sigma)$ associated with applying the solution $\sigma \in \mathcal{A}^\ell$ to the request sequence $\rho \in \mathcal{R}^\ell$.

On an input request sequence $\rho = (\rho_1, \ldots, \rho_\ell) \in \mathcal{R}^\ell$, a (randomized) *online algorithm* $\texttt{Alg}$ for $\mathcal{P}$ constructs the solution $\sigma = (\sigma_1, \ldots, \sigma_\ell) \in \mathcal{A}^\ell$ in an online fashion so that request ρ_t is revealed to $\texttt{Alg}$ at (discrete) time $t = 1, \ldots, \ell$ and in response, $\texttt{Alg}$ (probabilistically) decides on the answer σ_t irrevocably. Formally, an online algorithm $\texttt{Alg}$ is a family $\texttt{Alg} = \{\texttt{Alg}_t\}_{t \geq 1}$ of functions

$$\texttt{Alg}_t : \mathcal{R}^{t-1} \times \mathcal{A}^{t-1} \times \mathcal{R} \times \{0,1\}^\infty \to \mathcal{A}$$

that construct the answer $\sigma_t = \texttt{Alg}_t\left((\rho_1, \ldots, \rho_{t-1}), (\sigma_1, \ldots, \sigma_{t-1}), \rho_t, r\right) \in \mathcal{A}$ at time t based on the request history $(\rho_1, \ldots, \rho_{t-1}) \in \mathcal{R}^{t-1}$, the answer history $(\sigma_1, \ldots, \sigma_{t-1}) \in \mathcal{A}^{t-1}$, the incoming request $\rho_t \in \mathcal{R}$, and the current random coin tosses $r \in \{0,1\}^\infty$.[2]

Anytime Competitiveness. The gold standard for evaluating the performance of online algorithms is *competitive analysis* that compares the cost/payoff of the algorithm on the

[2] For the sake of simplifying some of the discussions in the sequel, our formulation assumes (without loss of generality) that the online algorithm has no access to past coin tosses.

entire request sequence to that of an optimal offline algorithm. In this paper, we enhance this classic notion and introduce the notion of *anytime competitive analysis*, requiring that the competitiveness of the online algorithm holds for any time interval of the request sequence.

To this end, consider a request sequence $\rho = (\rho_1, \ldots, \rho_\ell) \in \mathcal{R}^\ell$ and times $0 \leq t^0 < t^1 \leq \ell$. Let $S(\rho, t^0) \subseteq \mathcal{A}^{t^0}$ be the set of answer sequences $\sigma^0 = (\sigma_1^0, \ldots, \sigma_{t^0}^0)$ such that when $\texttt{Alg}$ runs on ρ, it constructs σ^0 at times $t = 1, \ldots, t^0$ with a positive probability (that is, $S(\rho, t^0)$ is the support of the probability distribution from which the answer prefix of $\texttt{Alg}$ is picked when $\texttt{Alg}$ runs on ρ).

Fix some answer sequence $\sigma^0 = (\sigma_1^0, \ldots, \sigma_{t^0}^0) \in S(\rho, t^0)$ and let $(\sigma_{t^0+1}, \ldots, \sigma_{t^1}) \in \mathcal{A}^{t^1-t^0}$ be the answer sequence constructed by $\texttt{Alg}$ when it runs on ρ at times $t = t^0 + 1, \ldots, t^1$, conditioned on the event that $\texttt{Alg}$ constructs the request sequence σ^0 at times $t = 1, \ldots, t^0$; notice that $(\sigma_{t^0+1}, \ldots, \sigma_{t^1})$ is a random variable that depends on the coin tosses of $\texttt{Alg}$ at times $t = t^0 + 1, \ldots, t^1$. Define the cost (resp., payoff) of $\texttt{Alg}$ on ρ during the time interval $(t^0, t^1]$ given the answer prefix σ^0, denoted by $\texttt{Alg}(\rho, t^0, t^1, \sigma^0)$, as

$$\texttt{Alg}\left(\rho, t^0, t^1, \sigma^0\right) = \mathbb{E}\left(f_{\mathcal{P}}\left(\rho, \left(\sigma_1^0, \ldots, \sigma_{t^0}^0 \sigma_{t^0+1}, \ldots, \sigma_{t^1}\right)\right)\right) - f_{\mathcal{P}}\left((\rho_1, \ldots, \rho_{t^0}), \sigma^0\right) .$$

Let $(\sigma_{t^0+1}^*, \ldots, \sigma_{t^1}^*) \in \mathcal{A}^{t^1-t^0}$ be a request sequence that minimizes (resp., maximizes) $f_{\mathcal{P}}\left(\rho, (\sigma_1^0, \ldots, \sigma_{t^0}^0, \sigma_{t^0+1}^*, \ldots, \sigma_{t^1}^*)\right)$, that is, a request sequence constructed by an omnipotent optimal algorithm that knows the entire request sequence ρ in advance (i.e., an optimal *offline algorithm*), given the answer prefix σ^0. Denote

$$\texttt{Opt}\left(\rho, t^0, t^1, \sigma^0\right) = f_{\mathcal{P}}\left(\rho, \left(\sigma_1^0, \ldots, \sigma_{t^0}^0 \sigma_{t^0+1}^*, \ldots, \sigma_{t^1}^*\right)\right) - f_{\mathcal{P}}\left((\rho_1, \ldots, \rho_{t^0}), \sigma^0\right) .$$

Online algorithm $\texttt{Alg}$ is said to be *anytime c-competitive* if there exists a constant d such that for every integer $\ell \geq 0$, request sequence $\rho = (\rho_1, \ldots, \rho_\ell) \in \mathcal{R}^\ell$, times $0 \leq t^0 < t^1 \leq \ell$, and answer sequence $\sigma^0 = (\sigma_1^0, \ldots, \sigma_{t^0}^0) \in S(\rho, t^0)$, it is guaranteed that $\texttt{Alg}(\rho, t^0, t^1, \sigma^0) \leq c \cdot \texttt{Opt}(\rho, t^0, t^1, \sigma^0) + d$ (resp., $\texttt{Alg}(\rho, t^0, t^1, \sigma^0) \geq c \cdot \texttt{Opt}(\rho, t^0, t^1, \sigma^0) - d$).[3]

The OAG Model. An *online algorithm with unreliable guidance (OAG)* $\texttt{Alg}$ for $\mathcal{P}$ is an online algorithm augmented with a *guidance sequence* $\gamma = (\gamma_1, \ldots, \gamma_\ell) \in \mathcal{A}^\ell$, where ℓ is the length of the request sequence, that ideally provides the optimal answers to $\texttt{Alg}$, but should be regarded with caution as it is not fully trustworthy. Formally, an OAG algorithm $\texttt{Alg}$ is a family $\texttt{Alg} = \{\texttt{Alg}_t\}_{t \geq 1}$ of functions

$$\texttt{Alg}_t : \mathcal{R}^{t-1} \times \mathcal{A}^{t-1} \times \mathcal{R} \times \mathcal{A} \times \{0,1\}^\infty \to \mathcal{A}$$

that construct the answer $\sigma_t = \texttt{Alg}_t\left((\rho_1, \ldots, \rho_{t-1}), (\sigma_1, \ldots, \sigma_{t-1}), \rho_t, \gamma_t, r\right) \in \mathcal{A}$ at time t based on the request history $(\rho_1, \ldots, \rho_{t-1}) \in \mathcal{R}^{t-1}$, the answer history $(\sigma_1, \ldots, \sigma_{t-1}) \in \mathcal{A}^{t-1}$, the incoming request $\rho_t \in \mathcal{R}$, the incoming guidance $\gamma^t \in \mathcal{A}$, and the current random coin tosses $r \in \{0,1\}^\infty$.

Ideally, the guidance sequence γ is constructed by a *guide* $\mathcal{G}$ that aims to minimize $\texttt{Alg}$'s cost (resp., maximize $\texttt{Alg}$'s payoff). Formally, the guide is a family $\mathcal{G} = \{\mathcal{G}_{\ell,t}\}_{\ell \geq 1, 1 \leq t \leq \ell}$ of functions

$$\mathcal{G}_{\ell,t} : \mathcal{R}^\ell \times \mathcal{A}^{t-1} \to \mathcal{A}$$

[3] Notice that the classic definition of competitiveness is obtained as a special case of anytime competitiveness by fixing $t^0 = 0$ and $t^1 = \ell$.

that generate a guidance $\gamma_t^{\mathsf{g}} = \mathcal{G}_{\ell,t}(\rho, (\sigma_1, \ldots, \sigma_{t-1})) \in \mathcal{A}$ at time t based on the entire request sequence $\rho \in \mathcal{R}^\ell$ and the answer history $(\sigma_1, \ldots, \sigma_{t-1}) \in \mathcal{A}^{t-1}$. The crux of the OAG model is that at each time $1 \leq t \leq \ell$, the guidance generated by the guide is replaced with an adversarially corrupted guidance $\gamma_t^{\mathsf{c}} \in \mathcal{A}$ independently with probability β, where $0 \leq \beta \leq 1$ is a model *bad guidance* parameter; that is, the guidance γ_t provided to $\texttt{Alg}$ is set to γ_t^{g} with probability $1 - \beta$, and to γ_t^{c} with probability β.

To make the model formulation precise, given a function $c : [0, 1] \to \mathbb{R}_{>0}$, an OAG algorithm $\texttt{Alg}$ is said to be anytime $c(\beta)$-competitive if there exists a guide $\mathcal{G}$ such that for every bad guidance parameter $0 \leq \beta \leq 1$ and for every corrupted guidance sequence $(\gamma_1^{\mathsf{c}}, \ldots, \gamma_\ell^{\mathsf{c}}) \in \mathcal{A}^\ell$, the algorithm is guaranteed to be anytime $c(\beta)$-competitive assuming that $\gamma_t \leftarrow \gamma_t^{\mathsf{g}}$ with probability $1 - \beta$ and $\gamma_t \leftarrow \gamma_t^{\mathsf{c}}$ with probability β for each time $1 \leq t \leq \ell$, independently. The function $c = c(\beta)$ that optimizes the anytime competitiveness of $\texttt{Alg}$ for each $0 \leq \beta \leq 1$ is referred to as $\texttt{Alg}$'s *smoothness*, whereas $c(0)$ and $c(1)$ are referred to as $\texttt{Alg}$'s *consistency* and *robustness*, respectively.

The DTB Compiler. In this paper, we restrict our attention to a class of OAG algorithms that are derived from (prediction-free) online algorithms via a blackbox transformation. This transformation, referred to as the *drop or trust blindly (DTB) compiler*, is parameterized by a *trust parameter* $0 \leq \tau \leq 1$ (we may occasionally restrict to non-extreme values of τ for simplicity). Given an online algorithm $\texttt{Alg} = \{\texttt{Alg}_t\}_{t \geq 1}$, the OAG algorithm $\texttt{Alg}^* = \{\texttt{Alg}_t^*\}_{t \geq 1}$ obtained from $\texttt{Alg}$ by applying the DTB compiler with trust parameter τ is defined as follows.

Consider time $t \geq 1$ and fix some request history $(\rho_1, \ldots, \rho_{t-1}) \in \mathcal{R}^{t-1}$, answer history $(\sigma_1, \ldots, \sigma_{t-1}) \in \mathcal{A}^{t-1}$, and incoming request $\rho_t \in \mathcal{R}$. Let $D_t \in \Delta(\mathcal{A})$ be the probability distribution from which $\texttt{Alg}_t((\rho_1, \ldots, \rho_{t-1}), (\sigma_1, \ldots, \sigma_{t-1}), \rho_t, r)$ is picked and let $V_t = \text{support}(D_t)$ be the set of answers in the support of D_t, referred to as *valid answers*.[4] Let γ_t be the incoming guidance and let φ_{trust} be the outcome of a fresh Bernoulli trial with success probability τ. Denoting the answer of $\texttt{Alg}^*$ at time t by σ_t, the function $\sigma_t = \texttt{Alg}_t^*((\rho_1, \ldots, \rho_{t-1}), (\sigma_1, \ldots, \sigma_{t-1}), \rho_t, \gamma_t, r)$ is constructed by setting $\sigma_t = \gamma_t$ if $\gamma_t \in V_t$ and $\varphi_{\text{trust}} = \text{success}$; and by picking σ_t from D_t otherwise (i.e., if $\gamma_t \notin V_t$ or $\varphi_{\text{trust}} = \text{failure}$).

3 Online Bipartite Matching

An instance of the online bipartite matching problem consists of a bipartite graph $G = (U, V, E)$ and a permutation π of the nodes of U. The nodes in U arrive one by one in the order π, potentially adversarial. Each time a node $u \in U$ arrives, its incident edges are revealed. The online algorithm must then decide to match u right away and in an irrevocable manner with one of its neighbors in V that was not already matched. The goal is to maximize the size of the obtained matching.

The online bipartite matching problem was first solved in 1990 by Karp, Vazirani and Vazirani [28] who presented the well-known Ranking algorithm and proved it achieves a competitive ratio always greater than $1 - e^{-1}$ against an oblivious adversary. They also showed that this competitive ratio is optimal up to lower-order factors in the number of nodes.

[4] Depending on $\texttt{Alg}$, we may wish to change the description of the DTB compiler so that the valid action set V_t includes all answers $\sigma \in \mathcal{A}$ that are legal for the incoming request ρ_t in the sense that they do not lead to ∞ cost (resp., $-\infty$ payoff). This is without loss of generality as $\texttt{Alg}$ can be modified so that $\text{support}(D_t)$ consists of all such legal answers while changing the overall cost (resp., payoff) by a negligible amount.

To the best of our knowledge, and despite multiple works on related problems [1, 17, 5, 27], there exists no learning-augmented algorithm for the online bipartite matching problem.

This section presents the DTB variant of the Ranking algorithm, Ranking-DTB, whose pseudo-code can be found in appendix (Algorithm 1). We derive an anytime-competitive ratio for Ranking-DTB which interpolates smoothly around the optimal, prediction-free competitive ratio of $1 - e^{-1}$ and depends on the values of β and τ:

▶ **Theorem 1.** *Ranking-DTB is* $\max\left\{\frac{1}{2},\ (1-\beta\tau)\cdot\frac{1-e^{-(1-\tau)}}{1-\tau}\right\}$*-competitive anytime in the OAG model.*

Proof. Proof in Appendix (Subsection B.6). ◀

From now on, we call M^* a maximum matching (the optimal offline solution); let $x \in U \cup V$, we call $m^*(u) \in V$ the match of x according to M^*. To fix a guide, we assume that a good guidance suggests to match according to M^*. When $u \in U$ is revealed, we call $N(u)$ the set of neighbors of u that were not already matched by the considered algorithm. The proof of Theorem 1 is inspired by the work of Birnbaum and Mathieu [9].

The perfect matching hypothesis. As in prior work [28, 9], we restrict to graphs admitting a *perfect matching*: this ensures positive matching probabilities and fixes the optimal payoff at n (for a graph with $2n$ nodes). In the prediction-free setting this is natural since Ranking's worst cases are perfect-matching graphs. This assumption is no more true in the OAG model — bad guidance could in principle make other graphs harder for Ranking-DTB—but our first claim (Lemma 2) shows that perfect-matching graphs remain worst cases provided that bad guidance plays optimally to minimize our anytime competitive ratio. We assume henceforth that G admits a perfect matching, with $n = |V| = |U|$.

▶ **Lemma 2** (Node Removal). *Fix a graph G, an arrival order π. Let $\mathcal{G}^{bad}$ be the bad function that minimizes the expected competitive ratio of Ranking-DTB in this context. Assume there exists a node $x \in U \cup V$ that is not matched by the maximum matching M^* and define G' the graph G where x was removed. Then there exists a bad guidance so that the expected competitive ratio of Ranking-DTB on G' is no greater that the expected competitive ratio on G.*

Proof. Proof in Appendix (Subsection B.2). ◀

Independence under the guide's influence. The original Ranking analysis [28] exploits correlations across decisions: since all nodes share the same ranking, the probability that a node is unmatched can be bounded via the expected number of nodes matched so far: matching few nodes increases the chances to match more nodes later. Lemma 6 adapts this argument to the OAG model.

Establishing this requires a key independence property in order to contain the guide's influence. To see this, let $t \in [\![n]\!]$, let $v \in V$ be the (random) node of rank t. Assume v is not matched. We call $u = m^*(v)$ the optimal match of v, and $R_{t-1} \subseteq U$ the set of nodes already matched to some $v' \in V$ of rank less than t. Since by assumption the ranking algorithm (not the guide) matched u, we have $u \in R_{t-1}$—otherwise u would have been matched to v. Deriving Lemma 6 would require u and R_{t-1} to be independent, which unfortunately fails in general [9].

Both [28, 9] resolve this by replacing u with a node drawn uniformly at random, which is independent of R_{t-1}. This requires first verifying that permuting node ranks leaves the

distribution of output matchings unchanged—which holds in the OAG model, since relabeling ranks carries the guide's behavior with it. We introduce the following notation for *replaced ranking*:

▶ **Definition 3** (Replaced Ranking). *Let $t \in [\![n]\!]$ and let σ be a ranking of the nodes in V. We define $\sigma_{t,i}$ as the ranking σ where the node with rank t was removed and put back in with rank i.*

We then show the core of our proof: even under the guide's influence, transforming a ranking into a replaced ranking incurs at most a difference of one alternating chain between the two output matchings.

Unique alternating chain despite the guide. We want to show that, for any bad guidance, replacing a node in a ranking always has a small effect on the output matching; more precisely, executing Ranking-DTB on a replaced ranking outputs a matching that differs by at most one alternating chain. We show that, despite the influence of the guide, a similar claim holds in the OAG model. Intuitively we show that, (1) an alternating chain cannot start when Ranking-DTB uses the guidance to make its decisions — the guide has no access to the algorithms' random seed — and (2) if an alternating chain has already started in the past, we use the fact that both executions of Ranking-DTB (with and without replacement in the ranking) receive the same guidance, implying that the guide either does not interfere with the chain (if the suggested match is possible in both executions) or else makes the chain longer (if the suggested match is possible in only one execution, hence following the guidance implies to match with a node at the end of an alternating chain).

▶ **Lemma 4** (Unique Alternating Chain). *Let $t, i \in [\![n]\!]$. Let σ be a ranking and let M be the output matching of Ranking-DTB using ranking σ. We call Ranking-DTB$_{t,i}$ the same algorithm as Ranking-DTB except it uses the replaced ranking $\sigma_{t,i}$, let $M_{t,i}$ be the output matching. Then assuming that M and $M_{t,i}$ are not equal and that M does not match some node $v \in V$, then M and $M_{t,i}$ differ by one single alternating chain starting at v.*

Proof. Proof in Appendix (Subsection B.3). ◀

Once we established the uniqueness of the alternating chain, we can efficiently isolate the influence of the guide from our Ranking logic and deduce our independence result in Lemma 5. Intuitively this states that, provided that the ranking (not the guide) is used, the probability that a node v with rank t is not matched is upper-bounded by the probability that any node of U (i is free of choice, thus the desired independence property) is in R_t using the replaced ranking $\sigma_{t,i}$.

▶ **Lemma 5** (Higher Rank). *Let G, π be a problem instance, let γ be a guidance sequence, let σ be a ranking. Let $u \in U$ and let $v = m^*(u)$, let t be the rank of v in σ. Assuming that the ranking (not the guide) is used to match u then, if v is not matched under σ, for all $i \in [\![n]\!]$, u is matched under $\sigma_{t,i}$ to a node with rank at most t in $\sigma_{t,i}$.*

Proof. Proof in Appendix (Subsection B.4). ◀

We finally derive our induction lemma which directly implies the main claim (Theorem 1) using an inductive reasoning. The fact that an alternating chain is unique *whatever the bad guidance* allowed us to decouple the losses in payoff generated by the guidance (left term $\beta\tau$) from those generated by the Ranking algorithm (right term $\frac{1-\tau}{n} \cdot \sum_{1 \le s \le t} x_s$) using simple conditional probabilities.

▶ **Lemma 6** (Induction Lemma). *Let $t \in [\![n]\!]$. Given a problem instance and a prediction, let x_t be the probability that the node of rank t is matched. It holds:*

$$1 - x_t \quad \leq \quad \beta\tau \; + \; \frac{1-\tau}{n} \cdot \sum_{1 \leq s \leq t} x_s$$

Proof. Proof in Appendix (Subsection B.5). ◀

4 Online Caching

The online caching problem considers a set of pages and a memory (cache) which can store up to k pages. A sequence of pages (requests) arrives in an online manner. Each time a page arrives, it must be *cached*, that is, the page must be fetched into the cache in case it was not cached already. An algorithm which solves the caching problem must maintain the cache over time so that each page is cached when it was requested and ensure that the cache never contains more than k pages. Fetching a page costs 1 and the goal of the algorithm is to minimize the overall cost.

The online caching problem [23] is the most studied online problem in the learning-augmented framework. The seminal paper from Lykouris and Vassilvitskii [31] already focused on randomized caching and gave (asymptotic) optimal robustness and consistency bounds. Most notable follow-up works solved more general variants such as the weighted case [6, 26] while others focused on improving the smoothness bound [34, 35].

In this section, we present the DTB variant of the well-known Random Mark algorithm [23] to solve the online caching problem. We call our algorithm Marking-DTB (Algorithm 2, pseudocode in appendix C.1) and show it achieves the (asymptotic) optimal trade-off between consistency and robustness ($\beta = 0$ and $\beta = 1$, respectively) while its competitive ratio smoothly degrades as the predictive errors become greater (Theorem 7):

▶ **Theorem 7.** *Marking-DTB (Algorithm 2) is* $\min\left\{\frac{2}{\tau(1-\beta)}, \frac{2H_k}{1-\tau\beta}, k\right\}$*-competitive anytime in the OAG model with an additive constant of* $2k$*.*

Proof. Proof in Appendix (Subsection C.4). ◀

Efficiently solving the caching problem amounts to evicting the pages that will be requested the furthest in time. Hence, we assume that a good guidance suggests to evict such an (unmarked) page; here guidance overlaps with literature standars [32].

blame chains. Our proof works on *blame chains*. A blame chain is a sequence of pages $r(1) \dots r(n)$ that were evicted by Marking-DTBsuch that, for all $i \in [\![n-1]\!]$, $r(i+1)$ was evicted because $r(i)$ was requested. In addition, all pages of a blame chain are pages that should not have been evicted, except the last one. As a result, each page that is not the last incurs a non-optimal cost for Marking-DTB.

Our proof reasons on the *length of blame chains*. For $i \in [0, n]$, let X_i be the random variable equal to the remaining number of pages in the blame chain starting from page $r(n-i)$ (conditioned on the event that $r(n-i)$ is part of the blame chain). We now state the core of our proof with Lemma 8 which states that, whatever its strategy, *a bad guidance can only shorten the expected remaining length of a blame chain*:

▶ **Lemma 8.** *Let $i, s \in [0, n]$ be two nodes such that i and s are successive nodes in the blame chain in case the bad guidance was used. It holds:*

$$\mathbb{E}[X_i] \geq \mathbb{E}[X_s].$$

Proof. Proof in Appendix (Subsection C.2). ◀

This result directly enable us to contain the nuisance of the bad guidance. We comply with the anytime competitive ratio by simply upper-bounding the costs of the first and last phases of the considered interval by k. Using induction reasonings, we can now derive a result on the expected length of a blame chain and obtain our main claim:

▶ **Lemma 9.** *Assuming that $\beta\tau < 1$, a blame chain has an expected length of at most $H_k/(1-\beta\tau)$.*

Proof. Proof in Appendix (Subsection C.3). ◀

5 Uniform Metrical Task Systems

In metrical task systems (MTS), a metric space (S, d) is given, where S is a set of n *states* and $d : S^2 \to \mathbb{R}_{\geq 0}$ is a metric *distance* function. A *task* is a function $r : S \to \mathbb{R}_{\geq 0}$ that associates each state $s \in S$ with a *processing cost* $r(s)$. For a sequence $\mathcal{R} = r_1, \ldots, r_m$ of m tasks, we define a *solution* as a sequence $\mathcal{A} = s_1, \ldots, s_m$ of states $s_i \in S$, where the solution $\mathcal{A}$ is said to *serve* the task r_i at state s_i. We define the *cost* of a solution $\mathcal{A}$ naturally as the sum between the total processing cost and the total transition cost incurred by $\mathcal{A}$, i.e., $\texttt{cost}(\mathcal{A}) = \sum_{i\in[m]} r_i(s_i) + \sum_{i\in[m-1]} d(s_i, s_{i+1})$. In the online problem, the tasks r_i are given one by one in an online manner and a solution needs to serve task r_i immediately as it arrives.

In this section, we focus on MTS on the uniform metric, i.e., a metric where d satisfies $d(s, s') = 1$ for every pair of distinct states $s, s' \in S$. We will present an algorithm MTS-DTB that extends the classical online algorithm of Borodin, Linial, and Saks [13] to the OAG model.

Before describing MTS-DTB, we recall the notions of phases and saturation from [13]. Towards that goal, we extend the definition of processing cost to *continuous* time. Concretely, for each discrete time i, we naturally define the processing cost of task r_i in a (continuous) time interval $[t, t'] \subseteq [i, i+1]$ as $(t'-t) \cdot r_i(s)$ for each state $s \in S$. Consider an MTS instance $\mathcal{R} = r_1, \ldots, r_m$. We define a partition of the (continuous) interval $[1, m+1]$ into sub-intervals $[t_0 = 1, t_1], [t_1, t_2], \ldots, [t_{\ell-1}, t_\ell = m+1]$ referred to as *phases*. Each phase begins with all states being *unsaturated*. Consider some phase $p = [t_j, t_{j+1}]$. A state s is said to be *saturated* for p at time $t \in p$ if the total processing cost associated with s during the time interval $[t_j, t]$ is at least 1. The phase ends immediately after all states become saturated. Observe that upon the arrival of a task r_i at (discrete) time i, the algorithm can determine which states will become saturated for the current phase by time $i+1$.

The algorithm. Consider the task r_i arriving at time i. Let p be the current phase, and let s be the current state of MTS-DTB. We denote by V_i the set of *valid* states in time i and initialize $V_i = \emptyset$. If s does not become saturated for p by time $i+1$, then MTS-DTB sets $V_i = \{s\}$ (i.e., MTS-DTB stays in state s). Otherwise, if p does not end by time $i+1$, then MTS-DTB sets V_i to be the set of all states that are not saturated at time $i+1$ (notice that it is guaranteed that $V_i \neq \emptyset$ since p does not end by time $i+1$). Finally, if p ends by time $i+1$, then MTS-DTB sets $V_i = \{s_{\min}\}$ where $s_{\min}$ is a state that minimizes the processing cost r_i. Let γ_i be the guidance given to MTS-DTB at time i. If $\gamma_i \notin V_i$, then MTS-DTB simply chooses a state uniformly at random from V_i. Suppose now that $\gamma_i \notin V_i$. Then, MTS-DTB moves to γ_i with probability τ; and moves to a state chosen uniformly at random from V_i otherwise.

This completes the description of MTS-DTB. We now analyze its performance. For a trust parameter τ and a bad guidance parameter β, we establish the following.

▶ **Theorem 10.** *MTS-DTB is* $2 \cdot \min\{\frac{1}{\tau(1-\beta)} + 1, \frac{1-\tau}{(1-\tau\beta)^2} H_n + 1, n\}$*-competitive anytime in the OAG model.*

Proof. Proof in Appendix (Subsection D.1). ◀

6 Conclusion

This paper presents advances in our understanding of the usage of machine-learned predictions to reliably solve a given task. We show that any online algorithms can be adapted in a systematic, canonical manner to integrate input predictions. While general, this transformation naturally provides the usual desired properties of learning-augmented algorithms: consistency, robustness, and smoothness. We analyze this transformation on three well-studied problems and show they match, or even beat, state-of-the-art performances. We envision that this framework will be used as a reference point in the future to evaluate learning-augmented algorithms.

References

1 Anders Aamand, Justin Y. Chen, and Piotr Indyk. (optimal) online bipartite matching with degree information. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

2 Spyros Angelopoulos. Online search with a hint. *Inf. Comput.*, 295(Part B):105091, 2023. URL: https://doi.org/10.1016/j.ic.2023.105091, doi:10.1016/J.IC.2023.105091.

3 Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. Online computation with untrusted advice. *Journal of Computer and System Sciences*, 144:103545, 2024.

4 Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM transactions on algorithms*, 19(2):1–34, 2023.

5 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

6 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-augmented weighted paging. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–89. SIAM, 2022.

7 MohammadHossein Bateni, Prathamesh Dharangutte, Rajesh Jayaram, and Chen Wang. Metric clustering and MST with strong and weak distance oracles. In Shipra Agrawal and Aaron Roth, editors, *The Thirty Seventh Annual Conference on Learning Theory, June 30 - July 3, 2023, Edmonton, Canada*, volume 247 of *Proceedings of Machine Learning Research*, pages 498–550. PMLR, 2024.

8 Naama Ben-David, Muhammad Ayaz Dzulfikar, Faith Ellen, and Seth Gilbert. Byzantine agreement with predictions. In Alkida Balliu and Fabian Kuhn, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*, pages 3–14. ACM, 2025. doi:10.1145/3732772.3733518.

9 Benjamin E. Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008. doi:10.1145/1360443.1360462.

10 Lucas Boczkowski, Uriel Feige, Amos Korman, and Yoav Rodeh. Navigating in trees with permanently noisy advice. *ACM Trans. Algorithms*, 17(2):15:1–15:27, 2021.

11 Lucas Boczkowski, Uriel Feige, Amos Korman, and Yoav Rodeh. The query complexity of searching trees with permanently noisy advice. *ACM Trans. Algorithms*, 21(2):18:1–18:30, 2025.

12 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, USA, 1998.

13 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

14 Joan Boyar, Lene M Favrholdt, Christian Kudahl, Kim S Larsen, and Jesper W Mikkelsen. Online algorithms with advice: A survey. *ACM Computing Surveys (CSUR)*, 50(2):1–34, 2017.

15 Vladimir Braverman, Prathamesh Dharangutte, Vihan Shah, and Chen Wang. Learning-augmented maximum independent set. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, London School of Economics, London, UK, August 28-30, 2024*, volume 317 of *LIPIcs*, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

16 Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. Online algorithms with advice: The tape model. *Information and Computation*, 254:59–83, 2017. doi:10.1016/j.ic.2017.03.001.

17 Davin Choo, Themistoklis Gouleakis, Chun Kai Ling, and Arnab Bhattacharyya. Online bipartite matching with imperfect advice. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

18 Nicolas Christianson, Junxuan Shen, and Adam Wierman. Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems. In *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, pages 9377–9399, 2023.

19 Vincent Cohen-Addad, Tommaso d'Orsi, Anupam Gupta, Euiwoong Lee, and Debmalya Panigrahi. Learning-augmented approximation algorithms for maximum cut and related problems. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

20 Marek Eliás, Haim Kaplan, Yishay Mansour, and Shay Moran. Learning-augmented algorithms with explicit predictors. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

21 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011. doi:10.1016/J.TCS.2010.08.007.

22 Yuval Emek, Yuval Gil, Maciej Pacut, and Stefan Schmid. Online algorithms with randomly infused advice. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, Amsterdam, The Netherlands, September 4-6, 2023*, volume 274 of *LIPIcs*, pages 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

23 Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

24 Hendrik Fichtenberger, Michael Kapralov, Ekaterina Kochetkova, Silvio Lattanzi, Davide Mazzali, and Weronika Wrzos-Kaminska. Spectral clustering with side information. *CoRR*, abs/2511.17326, 2025. arXiv:2511.17326, doi:10.48550/ARXIV.2511.17326.

25 Anupam Gupta, Debmalya Panigrahi, Bernardo Subercaseaux, and Kevin Sun. Augmenting online algorithms with ε-accurate predictions. In Alice H. Oh, Alekh Agarwal,

Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

26 Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. *ACM Transactions on Algorithms (TALG)*, 18(4):1–27, 2022.

27 Billy Jin and Will Ma. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

28 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358. ACM, 1990. doi:10.1145/100216.100262.

29 Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ml predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

30 Alexander Lindermayr and Nicole Megow. ALPS - algorithms with predictions, 2022. Accessed: 24-Jan-2026.

31 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *35th International Conference on Machine Learning (ICML)*, 2018.

32 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

33 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9684–9693, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

34 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1834–1845. SIAM, 2020.

35 Alexander Wei. Better and simpler learning-augmented online caching. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, Virtual Conference, August 17-19, 2020*, volume 176 of *LIPIcs*, pages 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.APPROX/RANDOM.2020.60.

A Related Works

A.1 Model Comparison

Beyond the aforementioned online algorithms with predictions model of [32] and ϵ-accurate predictions model of [25], the literature on online decision making includes various other models that also bear certain relations to our OAG model. In particular, the popular *online algorithms with advice* model, introduced by Emek et al. [21] and Böckenhauer et al. [16] (see [14] for a survey), considers online algorithms that receive advice from a fully trustworthy oracle, aiming to minimize the size of the advice. Angelopoulos et al. [3] introduced a generalization of the online algorithms with advice model in which the advice may come from an adversarial source, with the objective of optimizing (the equivalent of) the algorithm's consistency and robustness, while still aiming for small advice. There are also various works on algorithms with "noisy advice", i.e., the advice is generated by a trusted oracle, however, it is perturbed by some random noise. The problems studied within this framework include maximum independent set [15], max cut [19], searching in a tree [10, 11], and spectral clustering [24].

Slightly further away from the current work, the *randomly infused advice* model of Emek et al. [22] performs beyond worst-case analysis of online algorithms by "infusing" an advice generated by a (trusted) oracle into the online algorithm's random bits; this is related to our OAG model as the advice is infused for each incoming request independently based on a (biased) random coin toss. Elias et al. [20] take an opposite approach from ours and integrate the predictor in the learning problem, so it is no longer a blackbox and can also learn from the input. Finally, Bateni et al. [7] study a setting with queries that can be directed to either an expensive trusted oracle or to a cheaper untrusted oracle.

A.2 Problem-Specific Related Work

The online bipartite matching problem [28] has been intensively studied by the learning-augmented community. So far, the existing literature focused on problem variants such as the random arrival model, initiated in [5], and later improved in [17], the random graph model [1] or the two-stage arrival model [27]. In this paper, we focus on the standard variant as introduced in [28] where both the graph and the nodes' arrivals are adversarial.

The online caching problem [23] is probably the most studied online problem in the learning-augmented framework. The seminal paper from Lykouris and Vassilvitskii [31] already focused on randomized caching and gave (asymptotic) optimal robustness and consistency bounds. Most notable follow-up works solved more general variants such as the weighted case [6, 26] while others focused on improving the smoothness bound [34, 35]. While our consistency and robustness bounds match the state-of-the-art, our smoothness measure is however not comparable with those existing works.

The online metrical task system problem with uniform costs was first introduced and optimaly solved by [13]. It later received the attention of the learning-augmented community in [18] who presented optimal robustness and consistency trade-offs for the problem in its general version.

B Proofs and Algorithms for Online Bipartite Matching

B.1 Algorithm's Pseudocode for Online Bipartite Matching

Algorithm 1 Ranking-DTB

Input: the set V, trust parameter $\tau \in [0,1]$
$\sigma \leftarrow$ a permutation on V chosen uniformly at random
when $u \in U$ and $N(u) \subseteq V$ are revealed **do:**
  if $|N(u)| > 0$ **then**
    $r \leftarrow$ a random number in $[0,1]$
    $g \leftarrow$ the node the guide suggests to match with u
    if $r \leq \tau$ and $g \in N(u)$ **then**
      Match u with g
    else
      Match u with $v \in N(u)$ with lowest rank in σ
    end if
  end if

B.2 Proof of Lemma 2

▶ **Lemma 2** (Node Removal). *Fix a graph G, an arrival order π. Let $\mathcal{G}^{bad}$ be the bad function that minimizes the expected competitive ratio of Ranking-DTB in this context. Assume there exists a node $x \in U \cup V$ that is not matched by the maximum matching M^* and define G' the graph G where x was removed. Then there exists a bad guidance so that the expected competitive ratio of Ranking-DTB on G' is no greater that the expected competitive ratio on G.*

Proof. Fix a ranking σ and fix a guidance sequence γ where the guide thinks the problem instance is (G, π). We consider two runs: Ranking-DTB(τ, G, π, σ) and Ranking-DTB(τ, G', π, σ) under the exact same guidance sequence γ, call M and M' their respective output matchings. We will show that M' has a size no greater than M. If M and M' have equal sizes then the subclaim holds. Otherwise, let C be an alternating chain between M and M'. We will now show that x is an extremity of C. We define the node u such that

"u is the node in $U \cap C$ with earliest arrival time" (1)

We now distinguish between two cases and show that x is an extremity of C in both: either (1) u is matched in both M and M' or (2) u is matched in one of them and not in the other. We first consider case (1). Then call $v \in V$ and $v' \in V$ the nodes such that $\{u, v\} \in M$ and $\{u, v'\} \in M'$, it holds $v \neq v'$ by assumption. One of v or v' was not available to match for one of the matchings M and M'. Assuming that both v and v' are nodes of G' (i.e., $v \neq x$ and $v' \neq x$), we obtain that one of v or v' was already matched earlier, contradicting the assumption that u is the node of $C \cap U$ with earliest arrival time. We deduce that $v = x$, proving the claim in case (1).
We now consider case (2) where u is matched in only one of M or M' but not in the other. Assuming that u is a node in both G and G', then $N(u)$ was empty at the time u was revealed for the matching that did not match u; calling $v \in V$ the node that is matched with u in the other matching, v was therefore matched earlier which contradicts the assumption that u is the node with earliest arrival time in C. We deduce that u is not in both G and G', therefore $u = x$ and x is an extremity of C.
As x is an extremity of C, we deduce that $|M' \cap C| \leq |M \cap C|$. Finally, we proved that x is an extremity of any alternating chain which implies that there is at most one alternating chain, proving that $|M'| \leq |M|$.

We just showed that for a constant ranking σ and constant guidance sequence γ, the output matching for G' is no greater than for G. Hence for a constant guidance sequence γ, the expected competitive ratio for G' is no greater than for G — notice the size of a maximum matching is the same for G and G'. Finally notice that, going from G to G', the good guidance (which we assume blindly suggests to match like M^*) does not change and the bad guidance either does not change (as we assumed so far) or incurs a payoff even lower, the claim follows. ◀

B.3 Proof of Lemma 4

▶ **Lemma 4** (Unique Alternating Chain). *Let $t, i \in [\![n]\!]$. Let σ be a ranking and let M be the output matching of Ranking-DTB using ranking σ. We call Ranking-DTB$_{t,i}$ the same algorithm as Ranking-DTB except it uses the replaced ranking $\sigma_{t,i}$, let $M_{t,i}$ be the output matching. Then assuming that M and $M_{t,i}$ are not equal and that M does not match some node $v \in V$, then M and $M_{t,i}$ differ by one single alternating chain starting at v.*

Proof. Assume that M and $M_{t,i}$ are not equal and that M does not match v. We first show that the first alternating chain (chronologically) starts with v. Let u be the first node of U in the arrival order π that is matched differently in M and $M_{t,i}$. If the algorithms were following the guidance to match u then they would both match u to the same node as the matchings so far are the same — note that the guide gave the same guidance for both algorithms since they do not have direct access to the rankings. The algorithms therefore do not follow the guidance at the time u arrives, instead they both match according to their respective rankings. Ranking-DTB$_{t,i}$ thus has v as its candidate with lowest rank in $N(u)$ since it would match u to the same node as Ranking-DTB otherwise: $M_{t,i}$ therefore matches u with v. As M does not match v, v is at an end of an alternating chain, proving the subclaim.

We then show that no more than one alternating chain can appear. Let $u \in U$ and assume that an alternating chain has already started when u is revealed. In the case where both algorithm's executions use the ranking to match u, either both executions match u to the same node, not interfering with the alternating chain, or u is matched in two different ways (possibly matched or not matched) which implies that, in one execution, u is matched to a node that is already part of the alternating chain— except v, all nodes in V have the same pairwise ordering in both rankings, thus only a node that is matched in one ranking but not in the other (hence, the end of an alternating chain) may incur different decisions. Otherwise in the case where both algorithm's executions use the guidance to match u, either both executions match u to the same node (as the guidance is the same for both executions), the guidance is impossible to follow in one of the executions (same case as before: the other execution matches u with a node part of the alternating chain) or, finally, the guidance is impossible to follow in both executions, both falling back to the ranking (handled by the previous case). ◀

B.4 Proof of Lemma 5

▶ **Lemma 5** (Higher Rank). *Let G, π be a problem instance, let γ be a guidance sequence, let σ be a ranking. Let $u \in U$ and let $v = m^*(u)$, let t be the rank of v in σ. Assuming that the ranking (not the guide) is used to match u then, if v is not matched under σ, for all $i \in [\![n]\!]$, u is matched under $\sigma_{t,i}$ to a node with rank at most t in $\sigma_{t,i}$.*

Proof. Let M be the matching output under σ, let $M_{t,i}$ be the matching output under $\sigma_{t,i}$. First we show that, in the output matching under σ, u is matched to a node with rank lower

than t. u cannot be matched to a node with rank t as it is v which we assumed is not part of the output matching, otherwise u cannot be matched to a node with rank greater than t as v is available, our first subclaim holds.
Then, we show the desired claim. Let $u_1, u_2 \dots u_l$ be the nodes in U in reveal order that are matched (or not matched) in different manners under σ and $\sigma_{t,i}$. If u is not listed in $u_1 \dots u_l$ then the wanted claim directly holds: u is matched to a node with rank lower than t in σ and the rank of that node increases by at most one in $\sigma_{t,i}$. In the remainder, we therefore assume that u is listed in $u_1 \dots u_l$. We show that for each $i \in [1, l]$, at the reveal time of u_i, the set of matching candidates for u_i under $\sigma_{t,i}$ strictly contains the set of matching candidates minus v for u_i under σ. We start with the case $i = 1$. When u_1 is revealed, the current matching is the same under σ and $\sigma_{t,i}$ and the set of matching candidates for u_1 is the same in both. v is a neighbor of u_1 otherwise u_1 would be matched similarly under both rankings and the case $i = 1$ holds.
We now deal with the case $i \geq 2$. As stated in Lemma 4, $u_1 \dots u_{i-1}$ are the nodes of U making up the unique alternating chain when u_i is revealed; that chain starts with an edge of $M_{t,i}$ and ends with an edge of M linked to u_{i-1}. All the internal nodes of that chain are both part of M and $M_{t,i}$. For u_i, the beginning of the chain is a candidate node under σ and the end of the chain is a candidate option under $\sigma_{t,i}$. Removing v from the candidate set under σ hence gives the wanted subclaim.
We just proved that, removing the option to match v under σ, all the nodes of U that are part of the unique alternating chain (assuming it exists) have a set of matching candidates strictly greater under $\sigma_{t,i}$ than under σ (in the sense of inclusion). Hence, assuming that v is not matched under σ, u is matched under $\sigma_{t,i}$ to a node with rank at most $t - 1$ in σ and so at most t in $\sigma_{t,i}$. ◀

B.5 Proof of Lemma 6

▶ **Lemma 6** (Induction Lemma). *Let $t \in [\![n]\!]$. Given a problem instance and a prediction, let x_t be the probability that the node of rank t is matched. It holds:*

$$1 - x_t \quad \leq \quad \beta\tau \; + \; \frac{1-\tau}{n} \cdot \sum_{1 \leq s \leq t} x_s$$

Proof. Let $v \in V$ be the random variable equal to the node with rank t, let $u = m^*(v)$. Using total probabilities, we distinguish between three cases: when Ranking-DTB decides how to match u, it uses either the good guidance, the bad guidance or the ranking. If the good guidance is used, v is therefore matched with probability one — either Ranking-DTB matches u with v, or v was already matched before. In case the bad guidance happens when u is matched, we simply upper-bound the probability that v is matched by 1, hence the $\beta\tau$ term on the right member of our desired claim.
The rest of the proof seeks an upper-bound on the probability that v is matched in case the random ranking is used when u is matched. We consider the following alternative process: pick a node $v' \in V$ uniformly at random, remove it from the ranking σ and put it back in with rank t, call σ' the newly obtained ranking. We call $u' = m^*(v')$ and Ranking-DTB$'$ the algorithm running with ranking σ'. Clearly, the probability that v' is not matched equals $1 - x_t$. By Lemma 5, the event that v' is not matched under ranking σ' implies that u' is matched under ranking σ with a node of rank no greater than t. Notice that u' is independent of σ, so the probability that u' is matched with a node of rank no greater than t is $\frac{1}{n} \cdot \sum_{1 \leq s \leq t} x_s$, giving the desired result. ◀

B.6 Proof of Theorem 1

▶ **Theorem 1.** *Ranking-DTB is* $\max\left\{\frac{1}{2},\ (1-\beta\tau)\cdot\frac{1-e^{-(1-\tau)}}{1-\tau}\right\}$*-competitive anytime in the OAG model.*

Proof. First, notice that Ranking-DTB always outputs a matching that is maximal (i.e., it cannot be augmented greedily) which guarantees a competitive ratio greater than $1/2$. The rest of the proof shows that the competitive ratio is no lower than $(1-\beta\tau)\cdot(1-e^{-(1-\tau)})/(1-\tau)$. For all $i \in [\![n]\!]$ we define $S_i = \sum_{1\le s\le i} x_i$. Like in Lemma 6, we define x_t as the probability the node with rank t in σ is matched (note that σ is a random variable). To start, we prove by induction that, for all $i \in [\![n]\!]$ it holds:

$$S_t \ge (1-\beta\tau)\cdot\sum_{s=1}^{t}\left(1-\frac{1-\tau}{n+1-\tau}\right)^s$$

The base case for $t=1$ is obtained simply by rearranging the terms of Lemma 6 for $t=1$. By induction, we assume the result holds for t and show it for $t+1$. Adding S_{t+1} on both sides of Lemma 6 for $t+1$:

$$\begin{aligned}
&1+S_t \le \left(1+\frac{1-\tau}{n}\right)\cdot S_{t+1}+\beta\tau\\
&\implies S_{t+1} \ge \left(1-\frac{1-\tau}{n+1-\tau}\right)\cdot\left[(1-\beta\tau)+S_t\right]\\
&\ge (1-\beta\tau)\cdot\sum_{s=1}^{t+1}\left(1-\frac{1-\tau}{n+1-\tau}\right)^s
\end{aligned}$$

which proves the subclaim by induction. We therefore have that

$$\begin{aligned}
\sum_{s=1}^{n} x_s \;=\; S_n &\ge (1-\beta\tau)\cdot\sum_{s=1}^{n}\left(1-\frac{1-\tau}{n+1-\tau}\right)^s\\
&\ge (1-\beta\tau)\cdot\sum_{s=1}^{n}\left(1-\frac{1-\tau}{n}\right)^s\\
&= (1-\beta\tau)\cdot n\cdot\frac{1-(1-(1-\tau)/n)^{n+1}}{1-\tau}\\
&\ge (1-\beta\tau)\cdot n\cdot\frac{1-e^{-(1-\tau)}}{1-\tau}
\end{aligned}$$

Noticing that $\sum_{s=1}^{n} x_s$ is the expected profit of Ranking-DTB and that an optimal algorithm has profit n ends the proof. ◀

C Proofs and Algorithm for Online Caching

C.1 Pseudocode of Marking-DTB

C.2 Proof of Lemma 8

▶ **Lemma 8.** *Let $i, s \in [0, n]$ be two nodes such that i and s are successive nodes in the blame chain in case the bad guidance was used. It holds:*

$$\mathbb{E}[X_i] \ge \mathbb{E}[X_s].$$

Algorithm 2 Marking-DTB

Input: cache size k, initial cache content, trust parameter $\tau \in [0,1]$
At the beginning, no pages are marked
when a request to a page x arrives **do:**
  if the requested page is not cached **then**
    if the cache is full **then**
      $r \leftarrow$ a random number in $[0,1]$
      $g \leftarrow$ the page the guide suggests to evict
      if $r \leq \tau$ and g is cached and non-marked **then**
        Evict g
      else
        Evict a non-marked page uniformly at random
      end if
    end if
  end if
  Mark x
  if there are k marked pages **then**
    Unmark all pages
  end if

Proof. We prove this by strong induction on $i = 0, 1 \dots n$. The claim holds for $i = 0$. Let $i \in [0, n]$, we assume that the induction hypothesis holds for all $0, 1 \dots i-1$ and show it also holds for i. Let $r(n+1-s)$ be the next page after $r(n+1-i)$ on the blame chain if the bad guidance was used. It holds:

$$\mathbb{E}[X_i] = 1 + \beta\tau\mathbb{E}[X_s] + \frac{1-\tau}{i} \cdot \sum_{l=1}^{i-1} \mathbb{E}[X_l].$$

Using the same equality for s and then the induction hypothesis we obtain:

$$\mathbb{E}[X_s] \leq 1 + \beta\tau\mathbb{E}[X_s] + \frac{1-\tau}{s} \cdot \sum_{l=1}^{s-1} \mathbb{E}[X_l]$$

Moreover it holds:

$$\frac{1}{i} \cdot \sum_{l=1}^{i-1} \mathbb{E}[X_l] \geq \frac{1}{s} \cdot \sum_{l=1}^{s-1} \mathbb{E}[X_l]$$

as the averaged terms on the left sum are either contained in the right sum or greater (induction hypothesis). Gathering the three previous inequalities gives the desired claim. ◀

C.3 Proof of Lemma 9

▶ **Lemma 9.** *Assuming that $\beta\tau < 1$, a blame chain has an expected length of at most $H_k/(1-\beta\tau)$.*

Proof. Consider a blame chain $r(0), r(i_1) \dots r(n+1)$. Let $p \in [0, n]$ and let $r(s)$ be the next page after $r(p)$ in the blame chain in case the bad guidance was used. It holds:

$$\mathbb{E}[X_p] = 1 + \beta\tau\mathbb{E}[X_s] + \frac{1-\tau}{p} \sum_{i=1}^{p-1} \mathbb{E}[X_i].$$

Using Lemma 8 gives:

$$(1-\beta\tau)\cdot\mathbb{E}[X_p] \leq 1 + \frac{1-\beta\tau}{p}\sum_{i=1}^{p-1}\mathbb{E}[X_i]$$

We now assume by strong induction that $\mathbb{E}[X_i] \leq H_i/(1-\beta\tau)$ for all $i \in [1, p-1]$ — the base case for $i = 1$ clearly holds. Dividing the previous inequality by $(1-\beta\tau)$:

$$\begin{aligned}
&\mathbb{E}[X_p] \leq \frac{1}{1-\beta\tau} + \frac{1}{p}\sum_{i=1}^{p-1}\frac{H_i}{1-\beta\tau} \\
\implies &\mathbb{E}[X_p] \leq \frac{1}{1-\beta\tau} + \frac{1}{p\cdot(1-\beta\tau)}\sum_{i=1}^{p-1}\frac{p-i}{i} \\
\implies &\mathbb{E}[X_p] \leq \frac{1}{1-\beta\tau} + \frac{1}{1-\beta\tau}\cdot H_{p-1} - \frac{p-1}{p}\cdot\frac{1}{1-\beta\tau} \\
\implies &\mathbb{E}[X_p] \leq \frac{H_p}{1-\beta\tau}
\end{aligned}$$

It therefore holds that $\mathbb{E}[X_n] \leq H_n/(1-\beta\tau) \leq H_k/(1-\beta\tau)$ and the claim follows. ◀

C.4 Proof of Theorem 7

▶ **Theorem 7.** *Marking-DTB (Algorithm 2) is* $\min\left\{\frac{2}{\tau(1-\beta)}, \frac{2H_k}{1-\tau\beta}, k\right\}$*-competitive anytime in the OAG model with an additive constant of* $2k$*.*

Proof. We assume that a good guidance suggests to evict the unmarked, cached page that will be requested latest. Let σ be the sequence of requested pages. We define a *phase* as a subsequence of consecutive requested pages that contains no more than k different pages. We greedily divide σ into phases: starting from the first request, take the longest possible phase, and so on until the end of σ. Let P be a phase, we define:

- A *clean* page is a page that is requested during P but was not cached at the beginning of P. Define C the number of clean pages.
- A *returning* page is a page that is requested during P and was cached at the beginning of P. Define R the number of returning pages.
- A *vanishing* page is a page that is not requested during P and was cached at the beginning of P. There are C vanishing pages.

First, notice that Marking-DTB pays no more than k during P, since it pays at most one for each page kind thanks to the marking mechanism.
Then, we prove that the expected number of page faults is upper bounded by $C/(\tau(1-\beta))$. With probability $\tau(1-\beta)$, the algorithm uses a good guidance and Marking-DTB evicts a vanishing page. At any given time of P, the number of evictions until the next vanishing page is evicted follows a geometric law of parameter $\tau(1-\beta)$. As there are C vanishing pages, the expected number of evictions until all vanishing pages are out of the cache is $C/(\tau(1-\beta))$, which is an upper bound on the expected cost of Marking-DTB during phase P.
Finally, there are C blame chains evolving in parallel, each with an expected length of at most $H_k/(1-\beta\tau)$ by Lemma 9. The expected number of page faults is therefore no greater than $C\cdot H_k/(1-\beta\tau)$. Meanwhile, an optimal offline algorithm pays an amortized cost of at least $C/2$ per phase (well-known proof from [23]) and the claim holds. ◀

D Proofs for Online Metrical Task System

D.1 Proof of Theorem 10

▶ **Theorem 10.** *MTS-DTB is* $2 \cdot \min\{\frac{1}{\tau(1-\beta)} + 1, \frac{1-\tau}{(1-\tau\beta)^2}H_n + 1, n\}$*-competitive anytime in the OAG model.*

Proof. Consider an arbitrary time interval $[t, t']$ and recall that our goal is to show that MTS-DTB is $2 \cdot \min\{\frac{1}{\tau(1-\beta)} + 1, \frac{1-\tau}{(1-\tau\beta)^2}H_n + 1, n\}$-competitive in this interval. Let p and p' be the phases in which t and t' lie, respectively. Observe that the cost paid by MTS-DTB between time t and the end of p as well as the cost between the beginning of p' and t' can be attributed to the additive constant. Thus, to complete our analysis, it is left to analyze the competitive ratio in a single (complete) phase.

We first observe that any offline algorithm must pay a cost of at least 1 on every phase. Indeed, if an algorithm transitions during a phase, then it pays at least 1 in transition cost. Otherwise, it resided in the same state throughout the phase. By definition, a phase continues until all states become saturated. Thus, any offline algorithm must pay at least 1 in processing cost.

It is therefore sufficient to provide an upper bound on the expected cost of MTS-DTB in a single phase p. Notice that by design, MTS-DTB incurs a cost of at most 1 in processing cost for every state it visits during p. This is because MTS-DTB stays in a state only until it becomes saturated. In particular, MTS-DTB incurs a total processing cost of 2 on the first and last states it visits during p. For each intermediate state, MTS-DTB incurs 1 in transition cost and at most 1 in processing cost. Therefore, to complete the analysis it is left to show that the expected number of *transitions* during p is bounded by $\min\{\frac{1}{\tau(1-\beta)}, \frac{1-\tau}{(1-\tau\beta)^2}H_n, n-1\}$.

We start by showing that the number of transitions during p is at most $n-1$. To see that, notice that by construction, whenever MTS-DTB transitions, it moves to an unsaturated state for p. Thus, in any transition, the number of unsaturated states decreases. It follows that the number of transitions during p is at most $n-1$.

We now show that the expected number of transitions MTS-DTB makes during p is upper bounded by $1/(\tau(1-\beta))$. To that end, we consider a guide that selects a state s^* with longest time until saturation. Observe that if MTS-DTB transitions to s^* during p, its next transition is only in the next phase. Recall that the probability of receiving non-corrupted guidance at each time i is $\tau(1-\beta)$. Thus, the expected number of transitions during p is at most $1/(\tau(1-\beta))$.

Finally, we need to show that the expected number of transitions in a phase is upper bounded by $\frac{1-\tau}{(1-\tau\beta)^2}H_n$. We say that a transition round is *adversarial* if a corrupted guidance is given in that round. Let X_1 be a random variable that counts the number of adversarial rounds until the first non-adversarial round. For each $i > 1$, let X_i be a random variable that counts the number of adversarial rounds between the $(i-1)$-th and the i-th non-adversarial rounds. Let N be a random variable that counts the total number of non-adversarial rounds in the phase.

The goal is now to bound the sum $X_1 + \cdots + X_N$ which serves as an upper bound on the number of transitions during the phase. To that end, we note that we can assume w.l.o.g. that N is determined strictly by the randomness of the algorithm. This is because non-adversarial rounds in which a non-corrupted guidance is given can only decrease the value of N (and thus, the sum). Based on this assumption, we get that N and the X_i variables are determined based on two independent sources of randomness: the former by the algorithm's randomness, whereas the latter depends on the random assignment of bad guidance. In particular, N is independent of the random variables X_i. Thus, it follows from

Wald's identity that $\mathbb{E}[X_1 + \cdots + X_N] = \mathbb{E}[X_1] \cdot \mathbb{E}[N]$ (here, notice that $\mathbb{E}[X_1] = \mathbb{E}[X_i]$ for all i since the X_is are identically distributed). Since X_1 is a geometric variable with success probability $1 - \tau\beta$, we get $\mathbb{E}[X_1] \cdot \mathbb{E}[N] = (1/(1 - \tau\beta))\mathbb{E}[N]$.

It is left to show that $\mathbb{E}[N] \leq \frac{(1-\tau)}{1-\beta\tau} H_n$. Denote by ν_k the number of non-adversarial rounds in the phase given that there are k unsaturated states and notice that $N = \nu_n$. Clearly, $\nu_1 = 1$. For $k > 1$, first note that given that the round is non-adversarial, the probability of moving from k unsaturated states to a state that leaves $k - 1$ unsaturated states (i.e., the state with nearest saturation time) is at most $(1/k)(1 - \tau)$. Now, conditioning on the round being non-adversarial we get a probability bound of $\frac{(1-\tau)}{1-\beta\tau}(1/k)$. Based on that, we get the recursive formula $\nu_k \leq \nu_{k-1} + \frac{(1-\tau)}{1-\beta\tau}(1/k)$. Developing the recurrence for n, we get $\nu_n \leq \frac{(1-\tau)}{1-\beta\tau} H_n$ which completes the analysis. ◀