Improved Learning-Augmented Algorithms for the Multi-Option Ski Rental Problem via Best-Possible Competitive Analysis

Yongho Shin [1] **Changyeol Lee** [1] **Gukryeol Lee** [1] **Hyung-Chan An** [1]

Abstract

In this paper, we present improved learning-augmented algorithms for the multi-option ski rental problem. Learning-augmented algorithms take ML predictions as an added part of the input and incorporates these predictions in solving the given problem. Due to their unique strength that combines the power of ML predictions with rigorous performance guarantees, they have been extensively studied in the context of online optimization problems. Even though ski rental problems are one of the canonical problems in the field of online optimization, only deterministic algorithms were previously known for multi-option ski rental, with or without learning augmentation. We present the first randomized learning-augmented algorithm for this problem, surpassing previous performance guarantees given by deterministic algorithms. Our learning-augmented algorithm is based on a new, provably best-possible randomized competitive algorithm for the problem. Our results are further complemented by lower bounds for deterministic and randomized algorithms, and computational experiments evaluating our algorithms' performance improvements.

1. Introduction

A *learning-augmented algorithm* takes an ML prediction as an added part of the input and incorporates this prediction in solving the given problem. One major advantage of these algorithms is that they can benefit from the powerful ML predictions while yielding provable performance guarantees at the same time. These guarantees often surpass those given by classical algorithms without predictions, and they are obtained with no assumptions on how the ML predictions are generated, keeping their black-box natures. The recent success of learning-augmented algorithms is particularly remarkable in the field of *online optimization*, where we are given the input in an online manner over multiple timesteps and forced to make irrevocable decisions at each timestep. It is not at all surprising that learning-augmented algorithms are useful in this setting, because the challenge in designing a classical competitive algorithm is usually in avoiding the worst-case without any knowledge on the future input, where ML predictions can serve as a substitute for this knowledge. The success of learning-augmented algorithms in online optimization is evidenced by the seminal work of Lykouris and Vassilvitskii (2021) and subsequent studies including (Kumar et al., 2018; Bamas et al., 2020; Lattanzi et al., 2020) for example.

Ski rental problems are one of the canonical problems in online optimization that have been extensively studied, with or without learning augmentation. They succinctly capture the key nature of online optimization, and many algorithmic techniques in online optimization have been devised from their studies. For learning-augmented algorithms as well, ski rental problems naturally have been serving as an important testbed (Kumar et al., 2018; Gollapudi & Panigrahi, 2019; Bamas et al., 2020; Wei & Zhang, 2020).

Whilst the ski rental problem emerges in a variety of applications (Fleischer, 2001; Karlin et al., 2001; Meyerson, 2005), it is perhaps easiest to state it as a simple problem of renting skis (hence the name) as follows. In the *multi-option ski rental problem*, we are given a set of n renting options: for $i = 1, \cdots, n$, we have the option of renting skis for $d_i \in \mathbb{Z}_+ \cup \{\infty\}$ days at cost $c_i \in \mathbb{Q}_+$. Let T be the number of days we will go skiing, but it depends on, say, the weather, so we do not know T in advance; we will learn that a specific day was indeed the last day of skiing only at the end of that day. The objective of this problem is to ensure that we rent skis for the whole T days, while minimizing the total renting cost. Traditionally, the *rent-or-buy* case, where we only have two options $(d_1, c_1) = (1, 1)$ and $(d_2, c_2) = (\infty, B)$, was extensively studied, both under the classical competitive analysis setting without learning augmentation (Karlin et al., 1988; 1994; Buchbinder et al., 2007) and the learning-augmented settings (Kumar et al.,

[1]Department of Computer Science, Yonsei University, Seoul, South Korea. Correspondence to: Hyung-Chan An <hyung-chan.an@yonsei.ac.kr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2018; Gollapudi & Panigrahi, 2019; Angelopoulos et al., 2020; Bamas et al., 2020; Banerjee, 2020; Wei & Zhang, 2020). The general multi-option problem has been also studied in both settings (Zhang et al., 2011; Ai et al., 2014; Wang et al., 2020; Anand et al., 2021).

In this paper, we present the first *randomized* learning-augmented algorithm for the multi-option ski rental problem. Previously, Anand et al. (2021) gave a deterministic learning-augmented algorithm for the same problem. The performance guarantee of their algorithm is stated under the standard consistency-robustness scheme: their algorithm is $(1+\lambda)$-consistent and $(5+5/\lambda)$-robust, where λ is the trade-off parameter (or the "level of confidence") that dictates how much the algorithm would "trust" or "ignore" the prediction and determines the performance guarantees.[1] Their learning-augmented algorithm was obtained by modifying their classical competitive algorithm to use the given ML prediction (Anand et al., 2021); hence, in order to attain an improved learning-augmented algorithm, it is natural to delve in devising a more competitive algorithm for the classical problem.

Section 3 is a warm-up where we first consider deterministic algorithms. We present a competitive algorithm without learning augmentation in Section 3.1, which is very similar to Anand et al.'s algorithm and even has the same competitive ratio. We nonetheless present this algorithm as the subtle difference turns out to be useful in obtaining the improved learning-augmented algorithm presented in Section 3.2. To obtain this improvement, we allow our learning-augmented algorithm to "ignore" the prediction. The previous algorithm (Anand et al., 2021) internally computes the optimal solution assuming the (scaled) prediction is accurate, and insists on including this in the algorithm's output as a means of guaranteeing the consistency. On the other hand, our algorithm is capable of choosing not to do so when the trade-off parameter says the ML prediction is not too reliable. While this may sound natural, it is a new characteristic of our algorithm that leads to the improvement in the performance trade-off. This characteristic is shared also by our randomized learning-augmented algorithm.

In Section 4, we show how randomization can improve our algorithms. Section 4.1 presents an e-competitive algorithm for the multi-option ski rental problem. In the doubling scheme employed by Anand et al. (2021), one can adversarially construct a malicious instance by calculating the doubling budgets the algorithm will use. We can prevent the adversary from this exploitation by randomizing the budgets;

[1] Intuitively speaking, consistency is a performance guarantee that is valid only when the ML prediction is accurate, whereas robustness is always valid. Both are measured as a worst-case ratio of the algorithm's output to the true optimum with hindsight; see Section 2 for the formal definitions.

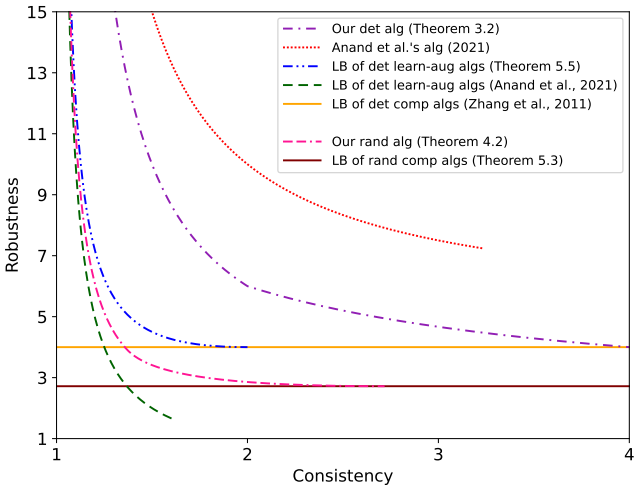

Figure 1. The consistency-robustness trade-off of learning-augmented algorithms.

our algorithm reveals that only a small amount of randomness suffices to obtain an e-competitive algorithm, which is provably the best possible (cf. Section 5.2). In Section 4.2, we base on this e-competitive algorithm to improve the performance trade-off of our learning-augmented algorithm. Our learning-augmented algorithm is $\chi(\lambda)$-consistent and (e^λ/λ)-robust, where

$$\chi(\lambda) := \begin{cases} 1+\lambda, & \text{if } \lambda < \frac{1}{e}, \\ (e+1)\lambda - \ln\lambda - 1, & \text{if } \lambda \geq \frac{1}{e}, \end{cases}$$

improving over the previous algorithms.

The parameter λ here governs the trade-off between the enhancement in the algorithm's performance when the prediction is accurate and the minimum performance guarantee independent of the prediction. For example, if we are willing to tolerate (in the unfortunate case when the prediction is inaccurate) a solution with a cost (in expectation) nearly five times higher than the optimum, we can set $\lambda = 0.26$ so that we can obtain solutions whose costs are no more than 26% worse than the optimum when the prediction is accurate. Figure 1 depicts the trade-off presented by learning-augmented algorithms.

In Section 5, we present lower bounds. We first propose an auxiliary problem called the *button problem* that is more amenable to lower-bound arguments, which we then reduce to the multi-option ski rental problem. In Section 5.2, we show that our algorithm in Section 4.1 is the best possible: i.e., for all constant $\rho < e$, there does not exist a randomized ρ-competitive algorithm for the multi-option ski rental problem. To prove this, we carefully formulate a linear program (LP) that bounds the competitive ratio of any randomized

algorithms for a given instance. We then obtain the desired lower bound by analytically constructing feasible solutions to the dual of this LP. Section 5.3 then begins with showing a lower bound of 4 on the competitive ratio of deterministic algorithms. Although Zhang et al. (2011) already gives the same lower bound of 4, we still present this proof to extend it into a lower bound for deterministic learning-augmented algorithms in Section 5.3.

Lastly, in Section 6, we experimentally evaluate the performance of our learning-augmented algorithms. We conduct computational experiments to measure the performance of our algorithms under a similar setting to (Kumar et al., 2018), to demonstrate the performance improvements our algorithms bring.

Related Work Recently, learning-augmented algorithms have been studied for a broad range of traditional optimization problems. Many studies introduce ML predictions to online optimization problems in particular, including caching (Rohatgi, 2020; Lykouris & Vassilvitskii, 2021; Im et al., 2022), matching (Antoniadis et al., 2020b; Lavastida et al., 2021), and graph/metric problems (Antoniadis et al., 2020a; Azar et al., 2022; Jiang et al., 2022) for example. We refer interested readers to the survey of Mitzenmacher and Vassilvitskii (2022) for a more thorough review.

Being a traditional online optimization problem itself, the ski rental problem is no exception and is widely studied. Karlin et al. (1994) presented a randomized $(e/(e-1))$-competitive algorithm for the rent-or-buy problem, which is the best possible. Zhang et al. (2011) presented a deterministic 4-competitive algorithm for the multi-option ski rental problem under the decreasing marginal cost assumption. The problem has also been studied in a variety of settings, including the multi-shop ski rental problem (Ai et al., 2014), the Banchard problem (Fleischer, 2001), the dynamic TCP acknowledgement problem (Karlin et al., 2001), and the parking permit problem (Meyerson, 2005) for example.

2. Preliminaries

In the *multi-option ski rental problem without learning augmentation*, we are given a set of n renting options: each option i covers $d_i \in \mathbb{Z}_+ \cup \{\infty\}$ days at cost $c_i \in \mathbb{Q}_+$. A solution to this problem is a sequence of options. In this problem, at the beginning of each skiing day, if the day is not yet covered by our solution, we must choose an option and add it to the "current" output solution. Let T be the (finite) number of skiing days, which is revealed only at the end of day T. The goal of the problem is to cover these T days by a solution of minimum cost. We say an algorithm is γ-*competitive* if $\mathbb{E}[\mathrm{SOL}] \leq \gamma \cdot \mathrm{OPT}(T)$, where SOL denotes the cost of the algorithm's output and $\mathrm{OPT}(T)$ denotes the cost of an optimal solution that covers T days. Note that $\mathrm{OPT}(T)$ is the *offline* optimal cost with the omniscient knowledge of the number of skiing days T.

In the *learning-augmented multi-option ski rental problem*, we are additionally given a prediction $\widehat{T}$ on the number of skiing days T. The performance of a learning-augmented algorithm is measured by the standard consistency-robustness analysis. That is, we say that an algorithm is χ-*consistent* if the algorithm satisfies $\mathbb{E}[\mathrm{SOL}] \leq \chi \cdot \mathrm{OPT}(T)$ when the prediction is accurate (i.e., $T = \widehat{T}$), and the algorithm is ρ-*robust* if $\mathbb{E}[\mathrm{SOL}] \leq \rho \cdot \mathrm{OPT}(T)$ for all T regardless how accurate the prediction $\widehat{T}$ is.

Given two solutions S_1 and S_2, we say that we *append* S_2 to S_1 when we add the options in S_2 to the end of S_1. For example, if S_1 is to choose option 1 and S_2 is to choose option 2, option 3, and option 3, adding S_2 to S_1 yields a solution that chooses option 1, option 2, option 3, and option 3.

For each $t \in \mathbb{Z}_+$, let $\mathcal{O}(t)$ be a minimum-cost solution that covers (at least) t days. Ties are broken arbitrarily. Recall that $\mathrm{OPT}(t)$ denotes the cost of $\mathcal{O}(t)$. We will assume that $\mathrm{OPT}(1) \geq 1$ without loss of generality: otherwise, we can divide the cost of every option by $\mathrm{OPT}(1)$. In this paper, algorithms operate by repeatedly appending an optimal solution $\mathcal{O}(t)$ for some t to the "current" solution. Note that this $\mathcal{O}(t)$ can be computed for any t without knowing the true number of skiing days T: a standard dynamic programming technique can be used to obtain $\mathcal{O}(t)$.

3. Deterministic Algorithms

In this section, we present our deterministic algorithms for the multi-option ski rental problem. We begin with a 4-competitive algorithm in Section 3.1. Although this algorithm is very similar to Anand et al.'s algorithm (2021), we still present our algorithm here because it leads to an improved learning-augmented algorithm, as will be presented in Section 3.2.

For simplicity of presentation, we will describe the algorithm's execution as if it never terminates, or in other words, there always comes another new skiing day; however, the actual algorithm is to terminate as soon as it learns that the last day has been reached.

For any $j \geq 1$, let $B(j)$ be a solution covering the most number of days among those whose cost does not exceed j, i.e., $B(j) := \mathcal{O}(t^\star)$, where $t^\star := \max\{t \in \mathbb{Z}_+ \cup \{\infty\} \mid \mathrm{OPT}(t) \leq j\}$. In other words, $B(j)$ is the "best" thing to do within a budget of j.

3.1. Competitive Algorithm

The algorithm runs in several *iterations*. For each iteration i (for $i = 1, 2, \cdots$), let SOL_i be the total (starting from the very first iteration) cost of our solution at the end of the i-th iteration. In the first iteration, we append $\mathcal{O}(1)$ to our solution. We thus have $\mathrm{SOL}_1 := \mathrm{OPT}(1)$. Let $\tau_1 := 1$. In each later iteration $i \geq 2$, we append $B(\mathrm{SOL}_{i-1})$ to our solution. Let τ_i be the number of days *newly* covered in iteration i, or in other words, the number of days covered by $B(\mathrm{SOL}_{i-1})$. Remark that, if $B(\mathrm{SOL}_{i-1})$ includes a buy option (i.e., an option j with $d_j = \infty$), τ_i becomes ∞ and no further iterations exist.

We note that the difference from Anand et al.'s algorithm (2021) is the fact that we append $B(\mathrm{SOL}_{i-1})$ instead of $B(2\,\mathrm{OPT}(\tau_{i-1}))$ at each iteration i, as is inspired by Zhang et al. (2011).

Theorem 3.1. *This algorithm is 4-competitive.*

The proof of the theorem is deferred to Appendix A.

3.2. Learning-Augmented Algorithm

In this subsection, we describe our deterministic learning-augmented algorithm. We are given the prediction $\widehat{T}$ on T and the level of confidence $\lambda \in [0, 1]$. The algorithm consists of (at most) three *phases* as follows.

First Ignore Phase We enter this phase at the very beginning if $\mathrm{OPT}(1) \leq \lambda\,\mathrm{OPT}(\widehat{T})$. Otherwise, we directly enter the respect phase. In this phase, we simply run the previous 4-competitive algorithm. Let $i^\star$ be the first iteration where the total cost incurred by the competitive algorithm exceeds $\lambda\,\mathrm{OPT}(\widehat{T})$, i.e.,

$$\mathrm{SOL}_{i^\star - 1} \leq \lambda\,\mathrm{OPT}(\widehat{T}) < \mathrm{SOL}_{i^\star}\,. \tag{1}$$

Note that there always exists such $i^\star \geq 2$ since we enter this phase only if $\mathrm{OPT}(1) \leq \lambda\,\mathrm{OPT}(\widehat{T})$. After processing the iteration $i^\star$, we move on to the respect phase or the second ignore phase depending on $\mathrm{SOL}_{i^\star}$. If $\mathrm{SOL}_{i^\star} \leq \mathrm{OPT}(\widehat{T})$, we enter the respect phase; otherwise, we enter the second ignore phase.

Respect Phase Intuitively, this phase is where the algorithm respects the prediction. In this phase, we append $\mathcal{O}(\widehat{T})$ and then move on to the second ignore phase.

Second Ignore Phase In this phase, we run the 4-competitive algorithm with a slight modification as follows.

First, let SOL'_0 be the total price incurred so far. Therefore, if the preceding phase was the respect phase, we have $\mathrm{SOL}'_0 := \mathrm{SOL}_{i^\star} + \mathrm{OPT}(\widehat{T})$ (or $\mathrm{SOL}'_0 := \mathrm{OPT}(\widehat{T})$ if the first ignore phase was skipped). On the other hand, if the preceding phase was the first ignore phase, we have $\mathrm{SOL}'_0 := \mathrm{SOL}_{i^\star}$.

We then choose τ'_0 so that it becomes a lower bound on the number of days covered so far. If the preceding phase was the respect phase, we choose $\tau'_0 := \widehat{T}$. If the immediately preceding phase was the first ignore phase, we choose $\tau'_0 := \tau_{i^\star}$.

Now, for each iteration $i \geq 1$, we append $B(\mathrm{SOL}'_{i-1})$ into our solution and let τ'_i be the number of days covered by $B(\mathrm{SOL}'_{i-1})$. This is the end of the algorithm description.

Following is the main theorem for this learning-augmented algorithm. We defer the proof to Appendix A.

Theorem 3.2. *The algorithm is a deterministic* $\max\{1 + 2\lambda, 4\lambda\}$*-consistent* $\left(2 + \frac{2}{\lambda}\right)$*-robust algorithm.*

4. Randomized Algorithms

In this section, we give randomized algorithms for the multi-option ski rental problem. We present our e-competitive algorithm first and then our learning-augmented algorithm which is, for any given $\lambda \in [0, 1]$, $\chi(\lambda)$-consistent and (e^λ/λ)-robust, where

$$\chi(\lambda) := \begin{cases} 1 + \lambda, & \text{if } \lambda < \frac{1}{e}, \\ (e+1)\lambda - \ln\lambda - 1, & \text{if } \lambda \geq \frac{1}{e}. \end{cases}$$

As in the previous section, we will describe the algorithm's execution as if it never terminates. Recall also that, for any $j \geq 1$, $B(j)$ denotes a solution covering the most number of days among those whose cost does not exceed j. If $j < \mathrm{OPT}(1)$, let $B(j) := \emptyset$ be an empty solution.

4.1. Competitive Algorithm

The algorithm begins with sampling $\alpha \in [1, e)$ from a distribution whose probability density function is $f(\alpha) := 1/\alpha$. The algorithm then runs in *phases*: in phase i (for $i = 0, 1, \cdots$), we append $B(\alpha e^i)$ to our current solution.

Theorem 4.1. *The given algorithm is a randomized* e*-competitive algorithm.*

The proof of this theorem is deferred to Appendix B.1.

4.2. Learning-Augmented Algorithm

Now we present our randomized learning-augmented algorithm. Recall that we are given a prediction $\widehat{T}$ and the level of confidence $\lambda \in [0, 1]$.

Assumptions We need several assumptions to describe the algorithm. First, let us assume that $\mathrm{OPT}(\widehat{T}) = e^k$ for some integer k. This assumption is without loss of generality since, if we have $e^{k-1} < \mathrm{OPT}(\widehat{T}) < e^k$, we

may multiply the cost of every option by $\frac{e^k}{\mathrm{OPT}(\widehat{T})}$. We also assume $\lambda \in (0,1)$; we will consider the cases for $\lambda \in \{0,1\}$ at the end of this section. Finally, let us assume that $\lambda\,\mathrm{OPT}(\widehat{T}) \geq e$. Observe that we can easily insist this assumption by multiplying the cost of every option by an appropriate power of e.

In what follows, we may write $\lambda := e^{-q-r}$ for some $q \in \{0,1,2,\cdots,k-1\}$ and $r \in (0,1]$. Here we note the range of r: it is strictly positive. For example, if $\lambda = e^{-i}$ for some integer i, we regard this λ as of $q = i-1$ and $r = 1$.

Algorithm Description Let $\mathcal{A}$ be the algorithm presented in Section 4.1. We run $\mathcal{A}$ and run the same phases as $\mathcal{A}$. On each phase i, we append the same solution as $\mathcal{A}$ with one following exception: If $\alpha e^i \in [\lambda\,\mathrm{OPT}(\widehat{T}), \mathrm{OPT}(\widehat{T})) = [e^{k-q-r}, e^k)$, we append $\mathcal{O}(\widehat{T})$ instead of $B(\alpha e^i)$ at this phase. If $\mathcal{O}(\widehat{T})$ has already been appended in a previous phase, we simply do nothing instead of appending it once more.

Theorem 4.2. *For $\lambda \in (0,1)$, this algorithm is $\chi(\lambda)$-consistent and (e^λ/λ)-robust, where χ is defined as follows:*

$$\chi(\lambda) := \begin{cases} 1+\lambda, & \text{if } \lambda < \frac{1}{e}, \\ (e+1)\lambda - \ln\lambda - 1, & \text{if } \lambda \geq \frac{1}{e}. \end{cases}$$

Consistency Analysis Let us begin with showing that our algorithm is $\chi(\lambda)$-consistent. Let SOL be the total cost that our algorithm incurs until the end. For each phase $i = 0,1,\cdots,k-q-2$, the algorithm appends $B(\alpha e^i)$ as the same as $\mathcal{A}$. Therefore, up to phase $(k-q-2)$, our algorithm also incurs in expectation at most

$$\int_1^e \left(\sum_{i=0}^{k-q-2} \alpha e^i\right) f(\alpha)d\alpha \leq e^{k-q-1}. \tag{2}$$

Let us assume for now that $\lambda < e^{-1}$, i.e., $q \geq 1$. Suppose the algorithm enters phase $(k-q-1)$. If $\alpha e^{k-q-1} \geq e^{k-q-r}$ (or simply $\alpha \geq e^{1-r}$), the algorithm appends $\mathcal{O}(\widehat{T})$ and terminates then. Otherwise if $\alpha < e^{1-r}$, the algorithm appends $B(\alpha e^{k-q-1})$ and may proceed to the next phase. In phase $(k-q)$, the algorithm appends $\mathcal{O}(\widehat{T})$ and terminates since $\alpha e^{k-q} \in [e^{k-q-r}, e^k)$. Together with Equation (2), the total expected cost of our algorithm can be bounded by

$$\begin{aligned} \mathbb{E}[\mathrm{SOL}] &\leq e^{k-q-1} + \int_1^{e^{1-r}} \alpha e^{k-q-1} f(\alpha)d\alpha + e^k \\ &\leq e^{k-q-r} + e^k \\ &= (1+\lambda)\cdot \mathrm{OPT}(\widehat{T}). \end{aligned} \tag{3}$$

We now turn to the case where $\lambda \leq e^{-1}$, i.e., $q = 0$. Here we have one distinction from the previous case; when the algorithm enters phase $(k-1)$, if $\alpha < e^{1-r}$, the algorithm not only appends $B(\alpha e^{k-1})$, but may also proceed to phase k and append $B(\alpha e^k)$ since $\alpha e^k \geq \mathrm{OPT}(\widehat{T})$. Therefore, again by Equation (2),

$$\begin{aligned} \mathbb{E}[\mathrm{SOL}] &\leq e^{k-1} + \int_1^{e^{1-r}} (\alpha e^{k-1} + \alpha e^k) f(\alpha)d\alpha \\ &\quad + \int_{e^{1-r}}^{e} e^k f(\alpha)d\alpha \\ &\leq e^{k+1-r} + e^{k-r} + e^k(r-1) \\ &= ((e+1)\lambda - \ln\lambda - 1)\,\mathrm{OPT}(\widehat{T}), \end{aligned} \tag{4}$$

where the last equality holds since $\lambda = e^{-r}$.

Robustness Analysis Let us now show that our algorithm is (e^λ/λ)-robust. Here we break down into several cases as follows depending on $\mathrm{OPT}(T)$.

Case 1. $\mathrm{OPT}(T) < e^{k-q-2}$**.** Note that, in this case, the algorithm executes the same as $\mathcal{A}$, yielding $\mathbb{E}[\mathrm{SOL}] \leq e\,\mathrm{OPT}(T)$ by Theorem 4.1.

Case 2. $e^{k-q-2} \leq \mathrm{OPT}(T) < e^{k-q-1}$**.** Let $\mathrm{OPT}(T) = \beta e^{k-q-2}$ for some $\beta \in [1,e)$. Observe that, up to phase $(k-q-2)$, the algorithm appends the same solution as $\mathcal{A}$. If $\alpha \geq \beta$ in $\mathcal{A}$, the algorithm terminates at phase $(k-q-2)$. Otherwise, the algorithm may proceed to phase $(k-q-1)$.

Suppose $\beta < e^{1-r}$. Observe that, if the algorithm enters phase $(k-q-1)$, we have $\alpha e^{k-q-1} < \lambda\,\mathrm{OPT}(\widehat{T})$, implying that the algorithm executes the same as $\mathcal{A}$. We thus have $\mathbb{E}[\mathrm{SOL}] \leq e\,\mathrm{OPT}(T)$ again by Theorem 4.1.

Let us now assume that $\beta \geq e^{1-r}$. Observe that, no matter whether $q \geq 1$ or $q = 0$, if $\alpha < e^{1-r}$, the algorithm appends $B(\alpha e^{k-q-1})$ by following $\mathcal{A}$ and terminates; otherwise if $e^{1-r} \leq \alpha < \beta$, the algorithm appends $\mathcal{O}(\widehat{T})$ and terminates. We therefore have

$$\begin{aligned} \mathbb{E}[\mathrm{SOL}] &\leq \int_1^e \left(\sum_{i=0}^{k-q-2} \alpha e^i\right) f(\alpha)d\alpha \\ &\quad + \int_1^{e^{1-r}} \alpha e^{k-q-1} f(\alpha)d\alpha \\ &\quad + \int_{e^{1-r}}^{\beta} e^k f(\alpha)d\alpha \\ &= \left(e^{k-q-1} - 1\right) + (e^{1-r} - 1)e^{k-q-1} \\ &\quad + (\ln\beta + r - 1)e^k \\ &\leq e^{k-q-r} + (\ln\beta + r - 1)e^k. \end{aligned}$$

Now we can upper bound the robustness ratio for this case

as follows:

$$\frac{\mathbb{E}[\mathrm{SOL}]}{\mathrm{OPT}(T)} \le \frac{e^{k-q-r} + (\ln\beta + r - 1)e^k}{\beta e^{k-q-2}} = \frac{e^{2-r}}{\beta} + \frac{(\ln\beta + r - 1)e^{q+2}}{\beta} = \frac{e^{2-r}}{\beta} + \frac{(\ln\beta + r - 1)e^{2-r}}{\lambda\beta},$$

where the last equality comes from the definition of $\lambda = e^{-q-r}$. The following claim completes the proof for this case.

Claim 4.3. *We have* $h(\beta) := \frac{e^{2-r}}{\beta} + \frac{(\ln\beta + r - 1)e^{2-r}}{\lambda\beta} \le \frac{e^\lambda}{\lambda}$ *for every* $\beta > 0$.

The claim can be shown by a simple calculus, which is deferred to Appendix B.2.

Case 3. $e^{k-q-1} \le \mathrm{OPT}(T) < \lambda\,\mathrm{OPT}(\widehat{T}) = e^{k-q-r}$**.** Let $\mathrm{OPT}(T) = \beta e^{k-q-1}$ for some $\beta \in [1, e^{1-r})$. Let us assume for now that $q \ge 1$. Note that the execution of our algorithm can be described as the following steps.

(a) For each phase $i = 0, 1, \cdots, k-q-2$, the algorithm appends the same solution as $\mathcal{A}$.

(b) On phase $(k-q-1)$, if $\alpha < e^{1-r}$, the algorithm follows $\mathcal{A}$ by appending $B(\alpha e^{k-q-1})$. Otherwise, the algorithm appends $\mathcal{O}(\widehat{T})$. If $\alpha \ge \beta$, the algorithm immediately terminates then.

(c) If $\alpha < \beta < e^{1-r}$, the algorithm enters phase $(k-q)$ and appends $\mathcal{O}(\widehat{T})$ since $\alpha e^{k-q} \in [e^{k-q-r}, e^k)$ for $q \ge 1$. The algorithm then terminates.

From this description, we can bound the total expected cost incurred by the algorithm as follows:

$$\mathbb{E}[\mathrm{SOL}] \le \int_1^e \sum_{i=0}^{k-q-2} \alpha e^i f(\alpha)d\alpha + \int_1^{e^{1-r}} \alpha e^{k-q-1} f(\alpha)d\alpha + \int_{e^{1-r}}^e e^k f(\alpha)d\alpha + \int_1^\beta e^k f(\alpha)d\alpha \le e^{k-q-r} + e^k \cdot (\ln\beta + r),$$

yielding that the robustness ratio for this case can be bounded from above by

$$\frac{e^{k-q-r} + e^k \cdot (\ln\beta + r)}{\beta \cdot e^{k-q-1}} = \frac{e^{1-r}}{\beta} + \frac{e^{1-r}(\ln\beta + r)}{\lambda\beta},$$

where the equality holds since $\lambda = e^{-q-r}$ by definition. We claim that the right-hand side can be further bounded by $\frac{e^\lambda}{\lambda}$, completing the proof for this case. The proof for this claim can be found in Appendix B.2.

Claim 4.4. *We have* $h(\beta) := \frac{e^{1-r}}{\beta} + \frac{e^{1-r}(\ln\beta + r)}{\lambda\beta} \le \frac{e^\lambda}{\lambda}$ *for every* $\beta > 0$.

Now we turn to the case where $q = 0$, i.e., $\mathrm{OPT}(T) = \beta \cdot e^{k-1}$ and $\lambda = e^{-r}$. Observe that there exists one difference on the execution from that of the case where $q \ge 1$; in Step (c), the algorithm appends $B(\alpha \cdot e^k)$ instead of $\mathrm{OPT}(\widehat{T})$ since the algorithm now enters phase k and hence $\alpha \cdot e^k \notin [e^{k-q-r}, e^k)$. Note that the total expected price for this case can be bounded by

$$\mathbb{E}[\mathrm{SOL}] \le \int_1^e \sum_{i=0}^{k-2} \alpha e^i f(\alpha)d\alpha + \int_1^{e^{1-r}} \alpha e^{k-1} f(\alpha)d\alpha + \int_{e^{1-r}}^e e^k f(\alpha)d\alpha + \int_1^\beta \alpha \cdot e^k f(\alpha)d\alpha \le e^{k-r} + e^k \cdot (\beta + r - 1),$$

resulting in the following upper bound for the robustness ratio for this case:

$$\frac{e^{k-r} + e^k \cdot (\beta + r - 1)}{\beta \cdot e^{k-1}} = e \cdot \left(1 + \frac{\lambda + r - 1}{\beta}\right) \le e \cdot (\lambda + r),$$

where the inequality can be derived from the fact that $\beta \ge 1$. The following claim completes the proof for this case. Recall that $\lambda = e^{-r}$, and hence $r = -\ln\lambda$, for this case.

Claim 4.5. *We have* $e(x - \ln x) \le \frac{e^x}{x}$ *for every* $x > 0$.

We defer the proof of this claim to Appendix B.2.

It remains to show the cases where $\mathrm{OPT}(T) \ge \lambda\,\mathrm{OPT}(\widehat{T})$. We again defer the proof for these remaining cases to Appendix B.2.

Final Remark In our analysis, we assume that $\lambda \in (0, 1)$. However, we can see that Theorem 4.2 still holds when $\lambda = 0$ or $\lambda = 1$. In fact, if $\lambda = 0$, this algorithm may correspond to appending $\mathcal{O}(\widehat{T})$ at the very beginning. Observe that this is 1-consistent, yet ∞-robust. On the other hand, if $\lambda = 1$, the algorithm is exactly the same as $\mathcal{A}$ itself, implying that the algorithm is e-consistent and e-robust.

5. Lower Bounds

In this section, we present lower bounds for the multi-option ski rental problem.

5.1. Auxiliary Problem

We define an auxiliary problem which we call the *button problem*. In this problem, we are given an ordered list of m buttons where some buttons are designated as *targets*. We know that the "targetness" of the buttons is monotone, i.e., there exists some $J \le m$ such that buttons 1 to $(J-1)$ are not targets, but buttons J to m are all targets. However, we

do not know in advance the first target button (i.e., button J). To sense whether button j is a target, we must click it paying $b_j \in \mathbb{Q}_+$ as the price; we then learn whether this button is a target or not. The prices of the buttons are all given at the beginning. We also assume that the prices are nondecreasing, i.e., $b_1 \leq b_2 \leq \cdots \leq b_m$. The algorithm clicks buttons until it clicks a target button, and the natural objective is to minimize the total price.

We say that an algorithm for the button problem is γ-competitive if $\mathbb{E}[\mathrm{SOL}] \leq \gamma \cdot b_J$, where SOL is the total price that the algorithm incurs until it clicks a target button. In the learning-augmented version of the problem, the algorithm is given a prediction $\widehat{J}$ on the first target button J. We say that an algorithm for the button problem is χ-consistent if $\mathbb{E}[\mathrm{SOL}] \leq \chi \cdot b_{\widehat{J}}$ when the prediction is accurate (i.e., $\widehat{J} = J$), and the algorithm is ρ-robust if $\mathbb{E}[\mathrm{SOL}] \leq \rho \cdot b_J$ for all J regardless how accurate the prediction $\widehat{J}$ is.

Lemma 5.1 states that a lower bound for the button problem immediately extends to give (almost) the same lower bound for the multi-option ski rental problem.

Lemma 5.1. *Suppose there exists a randomized χ-consistent ρ-robust algorithm for the multi-option ski rental problem with $1 \leq \chi \leq \rho$. Then, for all constant $\varepsilon \in (0, 1)$, there exists a randomized $(\chi + \varepsilon)$-consistent $(\rho + \varepsilon)$-robust algorithm for the button problem.*

Let us present the reduction algorithm from the multi-option ski rental problem to the button problem. Let $\mathcal{A}$ be the χ-consistent ρ-robust algorithm for the multi-option ski rental problem. Without loss of generality, we can assume that the prices $\{b_j\}_{j=1,\cdots,m}$ are all integers; otherwise, we enforce this assumption by simply multiplying the prices by a common denominator. We thus have $b_1 \geq 1$.

Consider the following algorithm for the button problem. Let $C := \lceil \rho/\varepsilon \rceil \cdot b_m$. Note that $C \geq 2$. The algorithm constructs an instance for the ski rental problem as follows. Let the number of rental options n be b_m; set $(d_i, c_i) := (C^i, i)$ for $i = 1, \cdots, n$, and $\widehat{T} := C^{b_{\widehat{J}}}$. The algorithm internally runs $\mathcal{A}$ on this constructed instance. Whenever $\mathcal{A}$ chooses an option, say, option i, we click the *last* button whose cost does not exceed $c_i = i$. (If there does not exist such a button, we do nothing.) If the clicked button turns out to be a target button, we report to $\mathcal{A}$ that the last skiing day has been reached and terminate the whole algorithm.

However, we might never be able to click a target button with only this procedure since $\mathcal{A}$ may choose only "cheap" options. In order to ensure that the algorithm always terminates, we add the following condition. Once the total cost incurred by $\mathcal{A}$ so far becomes at least C, we click the last button m and terminate. We say the algorithm is *forced* to terminate in this case. Let SOL be the cost incurred by the constructed algorithm. Note that SOL is no greater than the cost incurred by $\mathcal{A}$ unless the constructed algorithm is forced to terminate. This is the end of the reduction from the multi-option ski rental problem to the button problem.

The proof of Lemma 5.1 then follows from the next lemma. We defer its proof to Appendix C.1

Lemma 5.2. *This reduction algorithm is a randomized $(\chi + \varepsilon)$-consistent $(\rho + \varepsilon)$-robust algorithm for the button problem.*

5.2. Competitive Ratio of Randomized Algorithms

For randomized algorithms, we obtain the following lower bound on the competitive ratio. Together with Theorem 4.1, this shows that the algorithm presented in Section 4.1 is indeed the best possible.

Theorem 5.3. *For all constant $\varepsilon > 0$, no randomized algorithm can achieve a competitive ratio of $e - \varepsilon$.*

Here we give a proof sketch of this theorem. For any instance $\{b_j\}_{j=1,\cdots,m}$, we can formulate the following LP whose value constitutes a lower bound on the competitive ratio for any randomized algorithm for the button problem.

$$
\begin{aligned}
&\min \gamma \\
&\text{s.t. } \textstyle\sum_{j=1}^{m} x_j = 1, \\
&\quad \textstyle\sum_{j=t+1}^{m} y_{t,j} = x_t + \sum_{j=1}^{t-1} y_{j,t}, \\
&\qquad \forall t = 1, \cdots, m-1 \\
&\quad \textstyle\sum_{j=1}^{m} b_j \cdot \left(x_j + \sum_{t=1}^{\min(J,j)-1} y_{t,j} \right) \leq \gamma \cdot b_J, \\
&\qquad \forall J = 1, \cdots, m, \\
&\quad x_j \geq 0, \quad \forall j = 1, \cdots, m, \\
&\quad y_{t,j} \geq 0, \quad \forall t = 1, \cdots, m-1, \forall j = t+1, \cdots, m.
\end{aligned}
$$

Next, we carefully construct a family of instances and obtain the dual of this LP with respect to this family. We then analytically identify a dual feasible solution whose value converges to $(e - \varepsilon)$. We can then complete the proof of Theorem 5.3 by the weak duality. The full proof can be found in Appendix C.2.

5.3. Trade-off Between Consistency and Robustness of Deterministic Algorithms

For deterministic algorithms, we consider a variant of the regular button problem, called the (K, β)-continuum button problem, where the buttons are given as a continuum on $[1, m]$ with a price function $b : [1, m] \to \mathbb{R}_+$ satisfying that b is a nondecreasing K-Lipschitz continuous function and $b(1) = \beta > 0$. The following lemma justifies that we can instead consider this variant to obtain a trade-off between consistency and robustness.

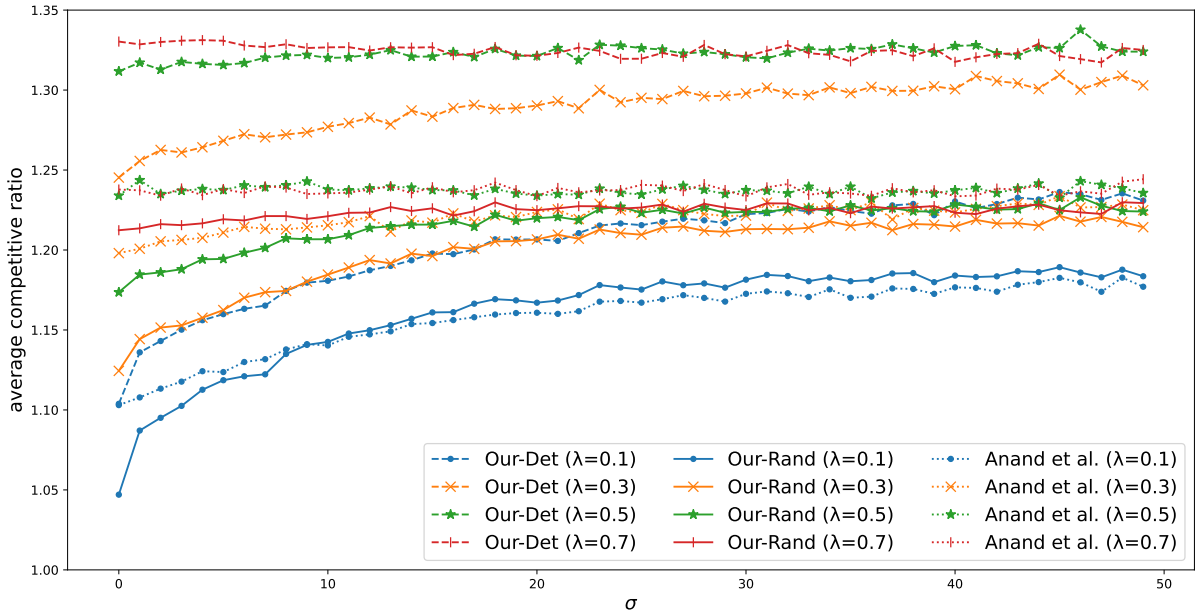

Figure 2. Average competitive ratio achieved by Our-Det (dashed line), Our-Rand (solid line), and Anand et al. (dotted line) are shown as a function of σ. Different colors/markers correspond to different values of the trade-off parameter λ.

Lemma 5.4. *Let $f : (0,1) \to \mathbb{R}_+$ be a continuous function. Suppose that, for all $\varepsilon > 0$ and $\lambda \in (0,1)$, the robustness of any deterministic $(1+\lambda)$-consistent algorithm for the (K,β)-continuum button problem must be strictly greater than $f(\lambda) - \varepsilon$. Then, for all $\varepsilon > 0$ and $\lambda \in (0,1)$, the robustness of any deterministic $(1+\lambda)$-consistent algorithm for the regular button problem must also be strictly greater than $f(\lambda) - \varepsilon$.*

We first show that no deterministic algorithm for the continuum button problem can achieve a competitive ratio better than 4; the proof of this statement can be found in Appendix C.3.1. Indeed, Zhang et al. (2011) provided the same lower bound on the competitive ratio of deterministic algorithms, but we still present our proof since it is useful in obtaining the following trade-off between consistency and robustness for deterministic learning-augmented algorithms.

Theorem 5.5. *For all constants $\lambda \in (0,1)$ and $\varepsilon > 0$, the robustness of any deterministic $(1+\lambda)$-consistent algorithm must be greater than $2 + \lambda + 1/\lambda - \varepsilon$.*

The full proof of this theorem is deferred to Appendix C.3.2.

6. Computational Evaluation

Experiment Setup In this section, we computationally measure how the average competitive ratios of three learning-augmented algorithms—our deterministic algorithm from Section 3.2 (Our-Det), our randomized algorithm from Section 4.2 (Our-Rand), and Anand et al.'s algorithm (Anand et al.)—behave as the prediction $\hat{T}$ drifts away from T. For the rent-or-buy (i.e., two-option) ski rental problem, Kumar et al. (2018) measured the competitive ratio of their algorithm by adding to T a Gaussian error in order to obtain prediction $\hat{T}$. They adjust the standard deviation of the Gaussian distribution to let $\hat{T}$ drift away from T, and conduct experiments for different values of the trade-off parameter λ. Our experiments are organized in a similar way: we measure each combination of $\lambda \in \{0.1, 0.3, 0.5, 0.7\}$ and $\sigma \in \{0, 1, \cdots, 50\}$ for all three algorithms; the average competitive ratio of each combination is plotted in Figure 2.

For a fixed (λ, σ), we generate $10,000$ independent random instances. Unlike Kumar et al. (2018), our experiments need multiple renting options. The rental period and cost of each option is randomly generated; the number of skiing days T is randomly chosen with taking the maximum range of rental period d^{max} into consideration; the prediction error η is sampled from $N(0, \sigma)$, with the appropriate clipping and rounding operations[2].

[2]The number of renting options is set as $n := 15$ We sample n numbers from $\{1, 2, \cdots, d^{max} := 50\}$ uniformly at random without replacements; let $d_1, \cdots, d_n$ be the sampled numbers, ordered so that $d_1 < \cdots < d_n$. We next sample n values from $(0,1)$ uniformly at random independently; let $r_1, \cdots, r_n$ be the sampled values such that $r_1 \leq \cdots \leq r_n$. We could have just define $c_i := r_i \cdot d_i$ for all i but this can cause $c_j > c_{j-1}$ for some j which makes option 1 to option $j-1$ useless. In order

Discussion As can be seen from Figure 2, Our-Rand generally outperforms the other two deterministic algorithms. In particular, when σ is small, Our-Rand performs much better than the other two. When $\lambda = 0.7$, the plotted lines start to look more "flat", showing that the behavior of all three algorithms are less affected by the prediction $\widehat{T}$. On the other hand, the impact of good predictions was much more dramatic when $\lambda = 0.1$. An interesting observation is that Our-Det performs slightly worse than Anand et al. We believe that this is because Anand et al. tends to be more aggressive in using longer rental options compared to Our-Det.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1H1A209312211). This research was supported by the Yonsei Signature Research Cluster Program of 2023 (2023-22-0008). Part of this work was done while the authors were visiting Cornell University.

References

Ai, L., Wu, X., Huang, L., Huang, L., Tang, P., and Li, J. The multi-shop ski rental problem. In *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 463–475, 2014.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. A regression approach to learning-augmented online algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 30504–30517, 2021.

Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., and Renault, M. P. Online computation with untrusted advice. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pp. 52:1–52:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Antoniadis, A., Coester, C., Elias, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning (ICML)*, pp. 345–355. PMLR, 2020a.

Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7933–7944, 2020b.

Azar, Y., Panigrahi, D., and Touitou, N. Online graph algorithms with predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 35–66. SIAM, 2022.

Bamas, E., Maggiori, A., and Svensson, O. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 20083–20094, 2020.

Banerjee, S. Improving online rent-or-buy algorithms with sequential decision making and ML predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 21072–21080, 2020.

Buchbinder, N., Jain, K., and Naor, J. Online primal-dual algorithms for maximizing ad-auctions revenue. In *15th Annual European Symposium on Algorithms (ESA)*, pp. 253–264. Springer, 2007.

Fleischer, R. On the Bahncard problem. *Theoretical Computer Science*, 268(1):161–174, 2001.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In *International Conference on Machine Learning (ICML)*, pp. 2319–2327. PMLR, 2019.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In *International Conference on Machine Learning (ICML)*, pp. 9588–9601. PMLR, 2022.

Jiang, S. H., Liu, E., Lyu, Y., Tang, Z. G., and Zhang, Y. Online facility location with predictions. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. Competitive snoopy caching. *Algorithmica*, 3: 79–119, 1988.

Karlin, A. R., Manasse, M. S., McGeoch, L. A., and Owicki, S. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

Karlin, A. R., Kenyon, C., and Randall, D. Dynamic TCP acknowledgement and other stories about $e/(e-1)$. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 502–509, 2001.

to prevent this situation, we inductively construct $c_1, \cdots, c_n$ as follows. Let $c_1 := r_1 \cdot d_1$. For $i = 2, \cdots, n$, if $r_i \cdot d_i < c_{i-1}$, we multiply $r_i, \cdots, r_n$ by $\frac{c_{i-1}}{r_i \cdot d_i}$ and add a small constant 10^{-4}; otherwise, we do nothing. We then set $c_i := r_i \cdot d_i$. This ensures that both $\{d_i\}_{i=1,\cdots,n}$ and $\{c_i\}_{i=1,\cdots,n}$ are nondecreasing. Let $\{(d_i, c_i)\}_{i=1,\cdots,n}$ be the set of renting options. The number of skiing days T is sampled from $\{1, \cdots, 10 \cdot d^{max}\}$ uniformly at random. Lastly, we sample error $\eta \sim N(0, \sigma)$ and let $\widehat{T} := \max\{\lfloor T + \eta \rceil, 1\}$ be the prediction on T.

Kumar, R., Purohit, M., and Svitkina, Z. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Online scheduling via learned weights. In *Proceedings of the 2020 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1859–1877. SIAM, 2020.

Lavastida, T., Moseley, B., Ravi, R., and Xu, C. Learnable and instance-robust predictions for online matching, flows and load balancing. In *29th Annual European Symposium on Algorithms (ESA)*, volume 204, pp. 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

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

Meyerson, A. The parking permit problem. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 274–282. IEEE, 2005.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.

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

Wang, S., Li, J., and Wang, S. Online algorithms for multi-shop ski rental with machine learned advice. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 8150–8160, 2020.

Wei, A. and Zhang, F. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 8042–8053, 2020.

Zhang, G., Poon, C. K., and Xu, Y. The ski-rental problem with multiple discount options. *Information Processing Letters*, 111(18):903–906, 2011.

A. Deferred Proofs in Section 3

To prove Theorems 3.1 and 3.2, we need the following observation for the deterministic competitive algorithm presented in Section 3.1.

Observation A.1. *For each iteration $i \geq 2$, we have* $\mathrm{OPT}(\tau_i) \leq \mathrm{SOL}_{i-1} \leq \mathrm{OPT}(\tau_i + 1)$ *and* $\mathrm{SOL}_i \leq 2\,\mathrm{SOL}_{i-1}$.

Proof. The first series of inequalities follows from the construction. (Note that the only reason why $\mathrm{SOL}_{i-1} \leq \mathrm{OPT}(\tau_i+1)$ is not strict is due to the possibility that $\tau_i = \infty$.) The second series of inequalities follows from the fact that, in iteration $i \geq 2$, the algorithm appends $B(\mathrm{SOL}_{i-1})$ to the solution. □

We are now ready to prove the two main theorems.

Theorem A.2 (Theorem 3.1 restated)**.** *This algorithm is 4-competitive.*

Proof. If $T = \tau_1 = 1$, the algorithm runs optimally. If $\tau_1 + 1 \leq T \leq \tau_2$, note that $\mathrm{SOL}_2 \leq 2\,\mathrm{SOL}_1 = 2\,\mathrm{OPT}(1) \leq 2\,\mathrm{OPT}(T)$. For $i \geq 3$, if $\tau_{i-1} + 1 \leq T \leq \tau_i$, the algorithm terminates at iteration i or earlier. We have

$$\mathrm{SOL}_i \leq 2\,\mathrm{SOL}_{i-1} \leq 4\,\mathrm{SOL}_{i-2} \leq 4\,\mathrm{OPT}(\tau_{i-1} + 1) \leq 4\,\mathrm{OPT}(T)$$

from Observation A.1. □

Theorem A.3 (Theorem 3.2 restated)**.** *The algorithm is a deterministic* $\max\{1 + 2\lambda, 4\lambda\}$*-consistent* $\left(2 + \frac{2}{\lambda}\right)$*-robust algorithm.*

Proof. Let us first show the consistency of the algorithm. If the algorithm starts in the respect phase, it is easy to see that the algorithm is optimal. Thus, we will assume from now on that the algorithm starts in the first ignore phase. If the algorithm moves on to the respect phase, it will terminate at the respect phase. The total cost is exactly $\mathrm{SOL}_{i^\star} + \mathrm{OPT}(\widehat{T})$. By Observation A.1 and Equation (1), this cost is at most $(1 + 2\lambda)\,\mathrm{OPT}(\widehat{T})$. If the algorithm terminates during the first ignore phase, imagine we continue the execution of the algorithm until at least the first phase is completed. The total cost in this case is at most $\mathrm{SOL}_{i^\star}$, and the consistency follows from $\mathrm{SOL}_{i^\star} \leq 2\lambda\,\mathrm{OPT}(\widehat{T})$.

Now consider the case where the algorithm directly enters the second ignore phase after the first ignore phase. This happens when $\mathrm{SOL}_{i^\star} > \mathrm{OPT}(\widehat{T})$. Observe that $B(\mathrm{SOL}'_0) = B(\mathrm{SOL}_{i^\star})$ covers at least $\widehat{T}$ days, implying that the algorithm must terminate after the first iteration, during which we append $B(\mathrm{SOL}'_0)$. The total cost therefore can be bounded by

$$\mathrm{SOL}'_1 = 2\,\mathrm{SOL}_{i^\star} \leq 4\,\mathrm{SOL}_{i^\star - 1} \leq 4\lambda\,\mathrm{OPT}(\widehat{T})$$

from Observation A.1 and Equation (1).

We now turn to proving the robustness. Note that if the algorithm does not enter the respect phase, the algorithm is 4-robust since the execution of this algorithm is exactly the same as that of our 4-competitive algorithm from Section 3.1. Therefore, it suffices to consider the case that the algorithm indeed enters the respect phase.

Let us first consider the case where the algorithm starts in the first ignore phase and terminates in the respect phase. From our assumption that the algorithm enters the respect phase, we have $T > \tau_{i^\star}$. Observe that

$$\lambda\,\mathrm{OPT}(\widehat{T}) < \mathrm{SOL}_{i^\star} \leq 2\,\mathrm{SOL}_{i^\star - 1} \leq 2\,\mathrm{OPT}(\tau_{i^\star} + 1) \leq 2\,\mathrm{OPT}(T),$$

where the first inequality follows from Equation (1) and the rest from Observation A.1 and $T \geq \tau_{i^\star} + 1$. This implies $\mathrm{OPT}(\widehat{T}) \leq \frac{2}{\lambda}\,\mathrm{OPT}(T)$. Therefore, the total cost is at most $\mathrm{SOL}_{i^\star} + \mathrm{OPT}(\widehat{T}) \leq \left(2 + \frac{2}{\lambda}\right)\mathrm{OPT}(T)$.

We now consider the case where the algorithm starts in the first phase, moves on to the respect phase, and terminates in the second ignore phase. Suppose that $\tau'_{i-1} + 1 \leq T \leq \tau'_i$ for some $i \geq 1$. Note that the algorithm terminates at iteration i or earlier then. Since the algorithm does not terminate in the respect phase, we have $T > \widehat{T} = \tau'_0$. If $i = 1$, observe that

$$\mathrm{SOL}'_1 \leq 2\,\mathrm{SOL}'_0 \leq 4\,\mathrm{OPT}(\widehat{T}) \leq 4\,\mathrm{OPT}(\tau'_0 + 1) \leq 4\,\mathrm{OPT}(T),$$

where the second inequality follows from the fact that the algorithm enters the respect phase only if $\mathrm{SOL}_{i^\star} \le \mathrm{OPT}(\widehat{T})$. If $i \ge 2$, note that

$$\mathrm{SOL}'_i \le 4\,\mathrm{SOL}'_{i-2} \le 4\,\mathrm{OPT}(\tau'_{i-1}+1) \le 4\,\mathrm{OPT}(T), \tag{5}$$

where the first two inequalities follow from Observation A.1.

Finally, consider the case where the algorithm starts in the respect phase. This happens only if $\mathrm{OPT}(1) > \lambda\,\mathrm{OPT}(\widehat{T})$. If $T \le \widehat{T}$, observe that the algorithm terminates in the respect phase and the total cost incurred by the algorithm is at most

$$\mathrm{OPT}(\widehat{T}) < \frac{1}{\lambda}\,\mathrm{OPT}(1) \le \frac{1}{\lambda}\,\mathrm{OPT}(T).$$

If $T > \widehat{T}$, the algorithm terminates in the second ignore phases. Suppose that $\tau'_{i-1}+1 \le T \le \tau'_i$ for some i. The algorithm then terminates at iteration i or earlier. If $i = 1$, we have

$$\mathrm{SOL}'_1 \le 2\,\mathrm{SOL}'_0 = 2\,\mathrm{OPT}(\widehat{T}) \le 2\,\mathrm{OPT}(\tau'_0+1) \le 2\,\mathrm{OPT}(T).$$

If $i \ge 2$, Equation (5) again holds, completing the proof. □

B. Deferred Proofs in Section 4

B.1. Deferred Proof in Section 4.1

Theorem B.1 (Theorem 4.1 restated)**.** *The algorithm is a randomized e-competitive algorithm.*

Proof. Let $\mathrm{OPT}(T) = \beta e^{i^\star}$ for some $i^\star \in \mathbb{Z}_{\ge 0}$ and $\beta \in [1, e)$. Suppose that the algorithm enters phase $i^\star$; otherwise, the algorithm only incurs less. If $\alpha \ge \beta$, observe that the algorithm terminates at this phase since $B(\alpha e^{i^\star})$ covers T days by the definition of B and that $\alpha e^{i^\star} \ge \mathrm{OPT}(T)$. On the other hand, if we sample α such that $\alpha < \beta$, the algorithm may enter the next phase $(i^\star+1)$. Note that the algorithm terminates then since $\alpha e^{i^\star+1} \ge \mathrm{OPT}(T)$. Recall that, in each phase i, the algorithm incurs at most αe^i. We therefore have

$$\begin{aligned}
\mathbb{E}[\mathrm{SOL}] &\le \int_1^\beta \left(\sum_{i=0}^{i^\star+1} \alpha e^i\right) f(\alpha)d\alpha + \int_\beta^e \left(\sum_{i=0}^{i^\star} \alpha e^i\right) f(\alpha)d\alpha \\
&= \sum_{i=0}^{i^\star} e^i \cdot \int_1^e \alpha f(\alpha)d\alpha + e^{i^\star+1} \cdot \int_1^\beta \alpha f(\alpha)d\alpha \\
&= \frac{e^{i^\star+1}-1}{e-1} \cdot (e-1) + e^{i^\star+1} \cdot (\beta-1) \\
&= (e^{i^\star+1}-1) + e^{i^\star+1} \cdot (\beta-1) \\
&= \beta \cdot e^{i^\star+1} - 1 \le e \cdot \mathrm{OPT}(T).
\end{aligned}$$

□

B.2. Deferred Proof in Section 4.2

Theorem B.2 (Theorem 4.2 restated)**.** *For $\lambda \in (0,1)$, this algorithm is $\chi(\lambda)$-consistent and (e^λ/λ)-robust, where χ is defined as follows:*

$$\chi(\lambda) := \begin{cases} 1+\lambda, & \text{if } \lambda < \frac{1}{e}, \\ (e+1)\lambda - \ln\lambda - 1, & \text{if } \lambda \ge \frac{1}{e}. \end{cases}$$

Robustness Analysis We continue to prove the robustness of our randomized learning-augmented algorithm. Let us first prove Claims 4.3, 4.4, and 4.5.

Claim B.3 (Claim 4.3 restated)**.** *We have*

$$h(\beta) := \frac{e^{2-r}}{\beta} + \frac{(\ln\beta + r - 1)e^{2-r}}{\lambda\beta} \le \frac{e^\lambda}{\lambda}$$

for every $\beta > 0$.

Proof. Let us obtain the partial derivative of $h(\beta)$ with respect to β as follows:

$$\frac{\partial h(\beta)}{\partial \beta} = \frac{e^{2-r}}{\lambda}\left(\frac{1-\ln\beta}{\beta^2} - \frac{\lambda+r-1}{\beta^2}\right) = \frac{e^{2-r}}{\lambda\beta^2}\cdot(2-\lambda-r-\ln\beta)\,,$$

implying that $h(\beta)$ achieves the maximum value of $\frac{e^\lambda}{\lambda}$ at $\beta = e^{2-\lambda-r}$. □

Claim B.4 (Claim 4.4 restated)**.** *We have*

$$h(\beta) := \frac{e^{1-r}}{\beta} + \frac{e^{1-r}(\ln\beta + r)}{\lambda\beta} \le \frac{e^\lambda}{\lambda}$$

for every $\beta > 0$.

Proof. When we calculate the partial derivative of $h(\beta)$ with respect to β, we have

$$\frac{\partial h(\beta)}{\partial \beta} = \frac{e^{1-r}}{\lambda}\left(\frac{1-\ln\beta}{\beta^2} - \frac{\lambda+r}{\beta^2}\right) = \frac{e^{1-r}}{\lambda\beta^2}\cdot(1-\lambda-r-\ln\beta)\,,$$

leading to that $h(\beta)$ achieves the maximum value of $\frac{e^\lambda}{\lambda}$ at $\beta = e^{1-\lambda-r}$ □

Claim B.5 (Claim 4.5 restated)**.** *We have*

$$e(x - \ln x) \le \frac{e^x}{x}$$

for every $x > 0$.

Proof. Let $h(x) := \frac{e^x}{x} - ex + e\ln x$. It suffices to show that $h(x) \ge 0$ for every $x > 0$. Observe that

$$h'(x) = \frac{e^x(x-1)}{x^2} - e + \frac{e}{x} = \frac{(x-1)(e^x - ex)}{x^2},$$

implying that $h(x)$ has the minimum value of 0 at $x = 1$. □

It remains to analyze the cases where $\mathrm{OPT}(T) \ge e^{k-q-1}$.

Case 4. $\lambda\,\mathrm{OPT}(\widehat{T}) = e^{k-q-r} \le \mathrm{OPT}(T) < \mathrm{OPT}(\widehat{T}) = e^k$**.** Remark that, in this case, the algorithm incurs in expectation at most the cost of the case when the prediction is accurate, i.e., $T = \widehat{T}$. Therefore, the bounds we obtained in the consistency analysis can be also used in this case. For $q \ge 1$, by Equation (3), we can obtain an upper bound for the robustness ratio for this case as follows:

$$\frac{\mathbb{E}[\mathrm{SOL}]}{\mathrm{OPT}(T)} \le \frac{e^k + e^{k-q-r}}{e^{k-q-r}} = 1 + \frac{1}{\lambda} \le \frac{e^\lambda}{\lambda},$$

where the equality comes from the definition of $\lambda = e^{-q-r}$ and the last inequality holds since $1 + x \le e^x$ for every x.

If $q = 0$, by Equation (4), the robustness ratio can be bounded by

$$\frac{\mathbb{E}[\mathrm{SOL}]}{\mathrm{OPT}(T)} \le \frac{e^{k-r} + e^{k+1-r} + e^k(r-1)}{e^{k-r}} = 1 + e + \frac{r-1}{\lambda},$$

where we derive the equality by the definition of $\lambda = e^{-r}$. Here we claim that the right-hand side can still be bounded by $\frac{e^\lambda}{\lambda}$ as desired for λ in this case. Recall that $\lambda \in \left[\frac{1}{e}, 1\right)$ and $r = -\ln\lambda$.

Claim B.6. *For* $x \in \left[\frac{1}{e}, 1\right]$*, we have*

$$1 + e - \frac{\ln x + 1}{x} \le \frac{e^x}{x}.$$

Proof. Let $h(x) := e^x - (1+e)x + \ln x + 1$. It suffices to show that $h(x) \geq 0$ for $x \in \left[\frac{1}{e}, 1\right]$. Taking the derivative, we can obtain

$$h'(x) = e^x - (1+e) + \frac{1}{x} = \left(e^x + \frac{1}{x}\right) - (e+1).$$

Note that $h'(x)$ is also convex over $x > 0$. Moreover, we have

$$h'\left(\frac{1}{e}\right) = e^{e^{-1}} - 1 > 0,$$

together with $h'(1) = 0$. We can thus conclude that the minimum of $h(x)$ over $\left[\frac{1}{e}, 1\right]$ must be either $h\left(\frac{1}{e}\right)$ or $h(1)$. Observe that $h(1) = 0$ and

$$h\left(\frac{1}{e}\right) = e^{e^{-1}} - \frac{1+e}{e} + \ln\left(\frac{1}{e}\right) + 1 = e^{e^{-1}} - \left(1 + \frac{1}{e}\right) \geq 0,$$

where the inequality follows from that $e^x \geq 1 + x$ for every x. □

Case 5. $\mathrm{OPT}(T) > \mathrm{OPT}(\widehat{T}) = e^k$**.** Let $\mathrm{OPT}(T) := \beta \cdot e^{k+p}$ for some $p \in \mathbb{Z}_{\geq 0}$ and $\beta \in [1, e)$. Let us first consider the case where $q \geq 1$. Observe that, until the algorithm reaches phase k, the algorithm incurs in expectation at most $e^k + e^{k-q-r}$ by Equation (3). Note also that, from phase k, we again follow $\mathcal{A}$. In total, we have

$$\begin{aligned}\mathbb{E}[\mathrm{SOL}] &\leq e^k + e^{k-q-r} + \int_1^e \left(\sum_{i=k}^{k+p} \alpha \cdot e^i\right) f(\alpha) d\alpha + \int_1^\beta \alpha \cdot e^{k+p+1} f(\alpha) d\alpha \\ &= \beta \cdot e^{k+p+1} + e^{k-q-r},\end{aligned}$$

implying that the robustness ratio for this case can be bounded by

$$\frac{\beta \cdot e^{k+p+1} + e^{k-q-r}}{\beta \cdot e^{k+p}} = e + \frac{\lambda}{\beta e^p} \leq e + \lambda,$$

where the equality follows from the definition of $\lambda = e^{-q-r}$ and the inequality can be derived by the fact that $\beta \geq 1$ and $p \geq 0$. Now the proof can be completed by the following claim. Recall that $\lambda < \frac{1}{e}$ in this case.

Claim B.7. *For $x \in \left(0, \frac{1}{e}\right]$, we have*

$$x + e \leq \frac{e^x}{x}.$$

Proof. Note that $\frac{e^x}{x}$ is decreasing over $x \in (0, 1]$ while $x + e$ is increasing over $x \in \mathbb{R}$. Direct calculation gives

$$\frac{e^{e^{-1}}}{e^{-1}} > e + \frac{1}{e}$$

as claimed. □

Finally, let us assume that $q = 0$, i.e., $\lambda = e^{-r}$. Observe that, in this case, the algorithm executes almost identical to $\mathcal{A}$ with only exception that, on phase $(k-1)$, the algorithm appends $\mathcal{O}(\widehat{T})$ instead of $B(\alpha \cdot e^{k-1})$ if $\alpha \geq e^{1-r}$. Therefore, by adjusting the proof of Theorem 4.1 accordingly, we can derive

$$\begin{aligned}\mathbb{E}[\mathrm{SOL}] &\leq \int_1^e \left(\sum_{i=0}^{k-2} \alpha \cdot e^i\right) f(\alpha) d\alpha + \int_1^{e^{1-r}} \alpha \cdot e^{k-1} f(\alpha) d\alpha + \int_{e^{1-r}}^e e^k f(\alpha) d\alpha \\ &\quad + \int_1^e \left(\sum_{i=k}^{k+p} \alpha \cdot e^i\right) f(\alpha) d\alpha + \int_1^\beta \alpha \cdot e^{k+p+1} f(\alpha) d\alpha \\ &\leq e^{k-r} + r \cdot e^k + \beta \cdot e^{k+p+1} - e^k \\ &= \beta \cdot e^{k+p+1} + (\lambda + r - 1) \cdot e^k.\end{aligned}$$

We can thus obtain an upper bound of the robustness ratio for this case as follows:

$$\frac{\mathbb{E}[\mathrm{SOL}]}{\mathrm{OPT}(T)} \leq e + \frac{\lambda + r - 1}{\beta e^p} \leq e + \lambda + r - 1,$$

where the last inequality is due to the fact that $\beta \geq 1$ and $p \geq 0$. Note that the next claim completes the proof for this case. Recall that $r = -\ln \lambda$.

Claim B.8. *We have*

$$x - \ln x + e - 1 \leq \frac{e^x}{x}$$

for any $x > 0$.

Proof. Let $h(x) := \frac{e^x}{x} - x + \ln x$. It suffices to show that $h(x) \geq e - 1$ for all $x > 0$. Note that

$$h'(x) = \frac{e^x(x-1)}{x^2} - 1 + \frac{1}{x} = \frac{(e^x - x)(x-1)}{x^2},$$

implying that the minimum value of $h(x)$ for $x > 0$ is $h(1) = e - 1$. □

C. Deferred Proofs in Section 5

C.1. Deferred Proof in Section 5.1

Lemma C.1 (Lemma 5.2 restated)**.** *This reduction algorithm is a randomized* $(\chi + \varepsilon)$*-consistent* $(\rho + \varepsilon)$*-robust algorithm for the button problem.*

Proof. Let us first show that this algorithm is $(\rho + \varepsilon)$-robust. Let k be the price of the first target button, i.e., $k = b_J$. Now imagine the execution of $\mathcal{A}$ with $T = C^k$. It is easy to see that $\mathrm{OPT}(T) = k = b_J$, where $\mathrm{OPT}(T)$ is the cost of an optimal solution to cover T days. Observe that, if $k = 1$, the reduction algorithm immediately terminates after $\mathcal{A}$ chooses any option. On the other hand, the first option chosen by $\mathcal{A}$ must be of cost at most ρ in expectation, since $\mathcal{A}$ must be ρ-competitive even when $T = 1$ (and $\mathrm{OPT}(T) = 1$). This implies the ρ-robustness of the constructed algorithm for the button problem. Thus, we assume from now on that $k \geq 2$. This implies $n = b_m \geq b_J \geq 2$.

We claim that, if $\mathcal{A}$ covers T days only with options 1 to $(k-1)$, the total cost $c(\mathcal{A})$ incurred by $\mathcal{A}$ is at least C. To see this fact, let a_i be the number of times that $\mathcal{A}$ chooses option i, for $i = 1, \cdots, k-1$. As the output of $\mathcal{A}$ covers T days, we have

$$\sum_{i=1}^{k-1} C^i \cdot a_i \geq T = C^k,$$

yielding

$$\sum_{i=1}^{k-1} \frac{k-1}{C^{k-1-i}} \cdot a_i \geq C \cdot (k-1).$$

We can now show our claim as follows:

$$c(\mathcal{A}) = \sum_{i=1}^{k-1} i \cdot a_i \geq \sum_{i=1}^{k-1} \frac{k-1}{C^{k-1-i}} \cdot a_i \geq C \cdot (k-1) \geq C,$$

where the first inequality holds since $h(x) := (k-1)/C^{k-1-x}$ is a convex function satisfying $h(1) = (k-1)/C^{k-2} \leq 1$ and $h(k-1) = k-1$. Recall that $k \geq 2$ and $C \geq 2$.

The above claim implies that, if $c(\mathcal{A}) < C$, $\mathcal{A}$ chose an option whose cost is at least $k = b_J$. Therefore, the constructed algorithm will not be forced to terminate in this case, and we have

$$\mathrm{SOL} \leq c(\mathcal{A}). \tag{6}$$

On the other hand, if $c(\mathcal{A}) \geq C$, the constructed algorithm is forced to terminate with an additional price of

$$b_m = n \leq \frac{\varepsilon}{\rho} \cdot C \leq \frac{\varepsilon}{\rho} \cdot c(\mathcal{A}), \tag{7}$$

yielding

$$\mathrm{SOL} \leq \left(1 + \frac{\varepsilon}{\rho}\right) c(\mathcal{A}). \tag{8}$$

Equations (6) and (8) together implies that $\mathrm{SOL} \leq \left(1 + \frac{\varepsilon}{\rho}\right) \cdot c(\mathcal{A})$ holds whether or not $c(\mathcal{A}) < C$. Hence, we have

$$\mathbb{E}[\mathrm{SOL}] \leq \left(1 + \frac{\varepsilon}{\rho}\right) \cdot \mathbb{E}[c(\mathcal{A})] \leq (\rho + \varepsilon) \cdot b_J, \tag{9}$$

where the second inequality is derived from that $\mathcal{A}$ is ρ-robust.

To see that the constructed algorithm is $(\chi + \varepsilon)$-consistent, note that $b_m \leq (\varepsilon/\chi) \cdot c(\mathcal{A})$ from Equation (7) since $\chi \leq \rho$. By adjusting Equation (9) with the fact that $\mathcal{A}$ is χ-consistent for $\widehat{T} = C^{b_{\widehat{\jmath}}}$, we can obtain $\mathbb{E}[\mathrm{SOL}] \leq (\chi + \varepsilon) \cdot b_{\widehat{\jmath}}$ as desired.

Finally, let us remark that the above argument only holds when ρ is finite. Suppose $\mathcal{A}$ is non-robust (i.e., $\rho = \infty$). In this case, we do not need to consider the robustness of the constructed algorithm either. By setting $C := \lceil \chi/\varepsilon \rceil \cdot b_m$, we can see that the above argument for the consistency still follows. □

C.2. Deferred Proof in Section 5.2

Theorem C.2 (Theorem 5.3 restated)**.** *For all constant $\varepsilon > 0$, no randomized algorithm can achieve a competitive ratio of $e - \varepsilon$.*

Proof. By Lemma 5.1, it suffices to exhibit a family of instances for the button problem that makes any algorithm have a competitive ratio at least $(e - \varepsilon)$ for the given constant $\varepsilon > 0$.

Now we introduce a set of parameters that define an instance. It is important in which order we choose these parameters, but we will discuss this at the end of the proof. For now, let δ be a sufficiently large number; intuitively speaking, δ corresponds to the granularity of button prices. Parameter c satisfies $c/\delta \geq -\ln(e - \varepsilon) - \ln\ln(e/(e - \varepsilon))$. An integer m is chosen so that $m > c$ and $m/\delta \geq \ln(e/\varepsilon)$. Consider the following instance of the button problem defined by these parameters: the number of buttons is m and their prices are $b_j := e^{j/\delta}$ for $j = 1, \cdots, m$.

Fix any algorithm for the button problem. Without loss of generality, we can assume that the indices of the buttons clicked by the algorithm strictly increases. That is, if the algorithm clicks a button j at some point, it will click button $j' > j$ in the following rounds. Suppose for the moment that only the last button m is a target: i.e, $J = m$. For $j = 1, \cdots, m$, let x_j be the marginal probability that the algorithm clicks button j in the first round. For $t = 1, \cdots, m-1$ and $j = t+1, \cdots, m$, let $y_{t,j}$ be the marginal probability that the algorithm clicks button t in some round and then clicks button j in the immediately following round.

Observe that x_j does not depend on J: the algorithm chooses the first button without any knowledge on J anyways. Similarly, for all $t < J$, $y_{t,j}$ does not depend on J either. Intuitively speaking, for any $J_1 < J_2$, the "prefix" of the execution of the algorithm for $J = J_2$ is the same as that for $J = J_1$ due to the algorithm's lack of knowledge on J.

We now write an LP where these x's and y's are variables and the constraints specify the properties that must be satisfied by the algorithm. We can write these constraints assuming $J = m$, since we can retrieve the marginal probabilities for the cases where $J \neq m$ simply by taking a "prefix". The value of the following LP is a lower bound on the competitive ratio of

the algorithm.

$$
\begin{aligned}
&\text{minimize } \gamma \\
&\text{subject to } \sum_{j=1}^{m} x_j = 1, \\
&\qquad \sum_{j=t+1}^{m} y_{t,j} = x_t + \sum_{j=1}^{t-1} y_{j,t}, && \forall t = 1, \cdots, m-1 \\
&\qquad \sum_{j=1}^{m} b_j \cdot \left(x_j + \sum_{t=1}^{\min(J,j)-1} y_{t,j} \right) \leq \gamma \cdot b_J, && \forall J = 1, \cdots, m, \\
&\qquad x_j \geq 0, && \forall j = 1, \cdots, m, \\
&\qquad y_{t,j} \geq 0, && \forall t = 1, \cdots, m-1, \forall j = t+1, \cdots, m.
\end{aligned}
$$

Note that the first constraint must be satisfied because x's must form a probability distribution. Note that both sides of the second constraint is a way of writing the marginal probability that the algorithm clicks button t (recall that we assume $J = m$). Therefore, these equalities must be satisfied. Note that $x_j + \sum_{t=1}^{\min(J',j)-1} y_{t,j}$ is the marginal probability that button j is clicked when $J = J'$: we use $\min(J', j) - 1$ to ensure that we take an appropriate prefix. This shows that the third constraints must be satisfied as long as γ is no smaller than the true competitive ratio.

In order to compute the worst-case value of this LP, we consider its dual, shown below:

$$
\begin{aligned}
&\text{maximize } w \\
&\text{subject to } \sum_{j=1}^{m} b_j v_j = 1, \\
&\qquad w \leq u_t + b_t \sum_{j=1}^{m} v_j, && \forall t = 1, \cdots, m-1 \\
&\qquad w \leq b_m \sum_{j=1}^{m} v_j, && \text{(D1)} \\
&\qquad u_s - u_t \leq b_t \sum_{j=s+1}^{m} v_j, && \forall s = 1, \cdots, m-2, \forall t = s+1, \cdots, m-1, \\
&\qquad u_s \leq b_m \sum_{j=s+1}^{m} v_j, && \forall s = 1, \cdots, m-1, \\
&\qquad w \in \mathbb{R}, \\
&\qquad u_t \in \mathbb{R}, && \forall t = 1, \cdots, m-1, \\
&\qquad v_j \geq 0, && \forall j = 1, \cdots, m.
\end{aligned}
$$

We want to find a feasible solution to (D1) whose value is close to $e - \varepsilon$. To this end, let us consider the following auxiliary

LP.

$$
\begin{aligned}
&\text{maximize } w \\
&\text{subject to } w \le u_t + b_t \sum_{j=1}^{m} v_j, && \forall t = 1, \cdots, m, \\
&\quad u_s - u_t \le b_t \sum_{j=s+1}^{m} v_j, && \forall s = 1, \cdots, m-1, \forall t = s+1, \cdots, m, \\
&\quad u_m = 0, \\
&\quad w \in \mathbb{R}, \\
&\quad u_t \in \mathbb{R}, && \forall t = 1, \cdots, m, \\
&\quad v_j \ge 0, && \forall j = 1, \cdots, m.
\end{aligned} \tag{D2}
$$

Observe that, once we obtain a feasible solution to (D2), we can easily construct a feasible solution to (D1) by dividing every variable by $\sum_{j=1}^{m} b_j v_j$.

Let us construct a feasible solution to (D2) as follows: for all $j = 1, \cdots, m$,

$$
\begin{aligned}
v_j &:= \int_{(j-1)/\delta}^{j/\delta} e^{-z} dz, \\
u_j &:= \frac{e-\varepsilon}{\delta} (m - c - j)_+, \text{ and} \\
w &:= \min_{t=1,\cdots,m} \left\{ u_t + e^{t/\delta} \sum_{j=1}^{m} v_j \right\},
\end{aligned}
$$

where $(\cdot)_+ := \max\{\cdot, 0\}$. Observe that the first set of constraints of (D2) is satisfied by the choice of w. It is easy to see that the third and fourth sets are also satisfied. The next lemma shows that the solution satisfies the second set of constraints.

Lemma C.3. *If $c/\delta \ge -\ln(e-\varepsilon) - \ln\ln(e/(e-\varepsilon))$, we have*

$$
u_s - u_t \le b_t \sum_{j=s+1}^{m} v_j
$$

for any $s < t$.

Proof. Let us first consider the case where $s < t \le m - c$. Observe that

$$
\begin{aligned}
u_s - u_t &= (e-\varepsilon) \cdot \frac{t-s}{\delta} \text{ and} \\
b_t \sum_{j=s+1}^{m} v_j &= e^{t/\delta} \int_{s/\delta}^{m/\delta} e^{-z} dz = e^{(t-s)/\delta} - e^{(t-m)/\delta}.
\end{aligned}
$$

We will show that

$$
(e-\varepsilon) \cdot \frac{t-s}{\delta} \le e^{(t-s)/\delta} - e^{-c/\delta} \le e^{(t-s)/\delta} - e^{(t-m)/\delta},
$$

where the last inequality follows from $t \le m - c$, showing the lemma for this case. By substituting $z := (t-s)/\delta$ and rearranging the terms, it suffices to show that, for any $z > 0$,

$$
e^z - (e-\varepsilon) \cdot z \ge e^{-c/\delta}.
$$

By taking the derivative of the left-hand side with respect to z, we can easily see that the left-hand side is minimized when $z = \ln(e-\varepsilon)$. Observe that

$$
e^{-c/\delta} \le (e-\varepsilon) - (e-\varepsilon)\ln(e-\varepsilon)
$$

due to the condition of the lemma, showing the claim.

For $s < m - c \leq t$, we have

$$u_s - u_t = (e - \varepsilon) \cdot \frac{m - c - s}{\delta} \text{ and}$$
$$b_t \sum_{j=s+1}^{m} v_j = e^{t/\delta} \left(e^{-s/\delta} - e^{-m/\delta} \right).$$

Note that, for a fixed s, $b_t \sum_{j=s+1}^{m} v_j$ is minimized when $t = m - c$ while $u_s - u_t$ is unaffected by t. Therefore, it suffices to show that the lemma follows when $t = m - c$. This is already proven by the previous case.

Finally, for $m - c \leq s < t$, we have $u_s - u_t = 0$. □

The following lemma gives a lower bound on w.

Lemma C.4. *If $m/\delta \geq \ln(e/\varepsilon)$, we have*

$$h(t) := u_t + e^{t/\delta} \sum_{j=1}^{m} v_j \geq \frac{e - \varepsilon}{\delta}(m - c)$$

for any t.

Proof. Recall that, for $t \geq m - c$, $u_t = 0$. Hence, it suffices to consider $t \leq m - c$. We have

$$\begin{aligned} h(t) &= \frac{e - \varepsilon}{\delta}(m - c - t) + e^{t/\delta} \int_0^{m/\delta} e^{-z} dz \\ &= \frac{e - \varepsilon}{\delta}(m - c - t) + e^{t/\delta} \left(1 - e^{-m/\delta} \right). \end{aligned}$$

By taking the partial derivative of $h(t)$ with respect to t, we obtain

$$\frac{\partial h(t)}{\partial t} = -\frac{e - \varepsilon}{\delta} + \frac{e^{t/\delta}}{\delta} \cdot \left(1 - e^{-m/\delta} \right).$$

This implies that $h(t)$ is minimized when $t = t^\star$, where $t^\star$ satisfies

$$e^{t^\star/\delta} = \frac{e - \varepsilon}{1 - e^{-m/\delta}} \leq e,$$

where the inequality follows from $m/\delta \geq \ln(e/\varepsilon)$. Here we can also observe that $t^\star \leq \delta$. We now have

$$h(t^\star) = \frac{e - \varepsilon}{\delta}(m - c - t^\star) + (e - \varepsilon) \geq \frac{e - \varepsilon}{\delta}(m - c),$$

where the inequality can be derived from $t^\star \leq \delta$. □

By these lemmas, we can conclude that (w, u, v) is a feasible solution to (D2) whose objective value is at least $\frac{e-\varepsilon}{\delta}(m - c)$. We have argued that, by normalizing every variable by $\sum_{j=1}^{m} b_j v_j$, we can derive a feasible solution to (D1). The next lemma indicates that $\sum_{j=1}^{m} b_j v_j$ is sufficiently small.

Lemma C.5. *We have $\sum_{j=1}^{m} b_j v_j \leq (m/\delta) \cdot e^{1/\delta}$.*

Proof. Observe that

$$\begin{aligned}\sum_{j=1}^{m} b_j v_j &= \sum_{j=1}^{m} e^{j/\delta} \int_{(j-1)/\delta}^{j/\delta} e^{-z} dz \\ &= e^{1/\delta} \sum_{j=1}^{m} e^{(j-1)/\delta} \int_{(j-1)/\delta}^{j/\delta} e^{-z} dz \\ &\leq e^{1/\delta} \sum_{j=1}^{m} \int_{(j-1)/\delta}^{j/\delta} e^{z} \cdot e^{-z} dz \\ &= \frac{m}{\delta} \cdot e^{1/\delta}.\end{aligned}$$

□

Therefore, we can obtain a solution feasible to (D1) whose objective value is at least

$$\frac{(e-\varepsilon)\cdot(m/\delta - c/\delta)}{(m/\delta)\cdot e^{1/\delta}}. \tag{10}$$

Given ε, we first fix c/δ as some constant satisfying $c/\delta \geq -\ln(e-\varepsilon) - \ln\ln(e/(e-\varepsilon))$. Then we choose δ to be sufficiently large, which in turn determines c. After this, we choose m to be sufficiently large. This shows that Equation (10) can be made arbitrarily close to $(e-\varepsilon)$. Now the desired conclusion follows from the weak duality of LP, completing the proof of Theorem 5.3. □

C.3. Deferred Proofs in Section 5.3

Lemma C.6. *Let $f : (0,1) \to \mathbb{R}_+$ be a continuous function. Suppose that, for all $\varepsilon > 0$ and $\lambda \in (0,1)$, the robustness of any deterministic $(1+\lambda)$-consistent algorithm for the (K,β)-continuum button problem must be strictly greater than $f(\lambda) - \varepsilon$. Then, for all $\varepsilon > 0$ and $\lambda \in (0,1)$, the robustness of any deterministic $(1+\lambda)$-consistent algorithm for the regular button problem must also be strictly greater than $f(\lambda) - \varepsilon$.*

Proof. Suppose towards contradiction that, for some $\lambda \in (0,1)$ and $\varepsilon > 0$, there exists a deterministic $(1+\lambda)$-consistent $(f(\lambda)-\varepsilon)$-robust algorithm $\mathcal{A}$ for the regular button problem. Without loss of generality, assume that $\varepsilon < f(\lambda)$. By choosing a sufficiently small $\varepsilon' > 0$, we can satisfy $\varepsilon' < \varepsilon/3$, $\lambda + \varepsilon' \in (0,1)$, and $f(\lambda+\varepsilon') > f(\lambda) - \varepsilon/3$. Let $F := \max\{2, \sup_{\lambda\in(0,1)} f(\lambda)\}$. We choose $\delta \in (0, (\varepsilon'\beta)/(FK)]$ so that $(m-1)/\delta$ becomes an integer.

Now consider the following algorithm for the (K,β)-continuum button problem. Let b be the given price function and $\widehat{J} \in [1,m]$ be the prediction. The algorithm constructs an instance of the regular button problem by creating $((m-1)/\delta+1)$ buttons whose values are $b((i-1)\delta+1)$ for $i = 1, \cdots, (m-1)/\delta+1$. The algorithm then internally executes $\mathcal{A}$ on this constructed instance, to which a prediction of $\tilde{J} := \left\lceil (\widehat{J}-1)/\delta \right\rceil + 1$ is given. Each time $\mathcal{A}$ clicks a button, say b_k, the algorithm clicks the corresponding button $(k-1)\delta+1$ in the continuum problem. When the algorithm clicks a target, it reports to $\mathcal{A}$ that the last button was a target and terminates. Note that the total cost incurred by the algorithm is exactly equal to that by $\mathcal{A}$.

We claim that this algorithm is $(1+\lambda+\varepsilon')$-consistent and $(f(\lambda)-(2\varepsilon)/3)$-robust for the (K,β)-continuum button problem. Since $f(\lambda)-(2\varepsilon)/3 < f(\lambda+\varepsilon')-\varepsilon/3$, this leads to contradiction. It remains to verify the consistency and robustness.

Suppose we run the above algorithm when the first target J of the continuum problem is equal to $\widehat{J}$. Note that the algorithm terminates when $\mathcal{A}$ clicks a button with index $\tilde{J}$ or higher. This means that the prediction $\tilde{J}$ given to $\mathcal{A}$ is also accurate (for the regular button problem). From the $(1+\lambda)$-consistency of $\mathcal{A}$, the total cost incurred by the algorithm is at most

$$\begin{aligned}(1+\lambda)b_{\tilde{J}} &= (1+\lambda)b\left(\left\lceil (\widehat{J}-1)/\delta \right\rceil \cdot \delta + 1\right) \\ &\leq (1+\lambda)(b(\widehat{J}) + K\delta) \\ &\leq (1+\lambda+\varepsilon')b(\widehat{J}),\end{aligned}$$

where the first inequality follows from the K-Lipschitz continuity of b and the second from the monotonicity of b and the choice of ε'. This shows the $(1+\lambda+\varepsilon')$-consistency of the algorithm.

Now we verify the robustness. Let J be the first target button of the continuum problem. The algorithm terminates when $\mathcal{A}$ clicks a button with index $\lceil (J-1)/\delta \rceil + 1$ or higher. Hence, from the $(f(\lambda)-\varepsilon)$-robustness of $\mathcal{A}$, the total cost incurred by the algorithm is at most

$$\begin{aligned}(f(\lambda)-\varepsilon)b\left(\lceil (J-1)/\delta \rceil \cdot \delta + 1\right) &\leq (f(\lambda)-\varepsilon)(b(J)+K\delta) \\ &\leq (f(\lambda)-\varepsilon+\varepsilon')b(J),\end{aligned}$$

where the first inequality again follows from the K-Lipschitz continuity of b and the second from the monotonicity of b and the choice of ε'. Note that $f(\lambda)-\varepsilon+\varepsilon' < f(\lambda)-(2\varepsilon)/3$, showing the desired robustness of the algorithm. □

C.3.1. Lower Bound on Competitivenss

Before presenting our trade-off between consistency and robustness for deterministic learning-augmented algorithms, let us first show that no deterministic algorithm for the (K,β)-continuum button problem can have a competitive ratio strictly better than 4. Let m be a sufficiently large number and $b(j)=j$ for $j \in [1,m]$. Note that b is 1-Lipschitz continuous with $b(1)=1>0$.

Fix any deterministic algorithm for the continuum button problem. Let n be the number of buttons the algorithm clicks; for any $i=1,\cdots,n$, let x_i be the button that the algorithm clicks in the i-th round. Note that, by the choice of b, the price of this button is also x_i. Moreover, we can without loss of generality assume that $x_1 < x_2 < \cdots < x_n$.

In order for the algorithm to be competitive, it should be competitive for any $J \in \{1, x_1+\varepsilon, x_2+\varepsilon, \cdots, x_{n-1}+\varepsilon\}$, where ε is an infinitesimal. We therefore have the following infinite sequence of constraints as follows: for any integer $i=0,1,2,\cdots,n-1$,

$$\sum_{j=1}^{i+1} x_j \leq \gamma x_i, \tag{11}$$

where γ denotes the competitive ratio of the algorithm and $x_0 := 1$.

Let $\{a_i\}$ be an infinite sequence defined recursively as follows:

$$a_1 = \gamma - 1 \text{ and } a_i = \gamma - \frac{\gamma}{a_{i-1}} \text{ for } i \geq 2.$$

Note that this infinite sequence is depends only on γ.

We show by induction that, for any $i=1,\cdots,n-1$, we have

$$x_{i+1} \leq a_i x_i \text{ and } \sum_{j=1}^{i+1} x_j \geq \frac{\gamma}{a_i} \cdot x_{i+1}. \tag{12}$$

For $i=1$, it is easy to see that $x_2 \leq (\gamma-1)x_1$ due to Equation (11) with $i=1$. Moreover,

$$x_1 + x_2 \geq \frac{1}{\gamma-1} \cdot x_2 + x_2 = \frac{\gamma}{a_1} \cdot x_2,$$

where the first inequality comes from $x_2 \leq a_1 x_1$, showing the second half of the statement. For $i \geq 2$, we have

$$x_{i+1} \leq \gamma x_i - \sum_{j=1}^{i} x_j \leq \left(\gamma - \frac{\gamma}{a_{i-1}}\right) x_i = a_i x_i,$$

where the first inequality comes from Equation (11) and the second is derived from the induction hypothesis. We also have

$$\begin{aligned}\sum_{j=1}^{i+1} x_j &\geq \frac{\gamma}{a_{i-1}} \cdot x_i + x_{i+1} \\ &\geq \frac{\gamma}{a_{i-1}} \cdot \frac{1}{a_i} \cdot x_{i+1} + x_{i+1} \\ &= \left(\frac{\gamma}{a_{i-1}} \cdot \frac{1}{\gamma - \frac{\gamma}{a_{i-1}}} + 1 \right) x_{i+1} \\ &= \frac{\gamma}{a_i} \cdot x_{i+1},\end{aligned}$$

from the recurrence relation of $\{a_i\}_i$. This completes the proof of the claim.

The next two lemmas exhibit useful properties of $\{a_i\}_i$. These are crucial in proving the lower bound on competitive ratio for deterministic algorithms.

Lemma C.7. *If the sequence $\{a_i\}_i$ is convergent, we have $\gamma \geq 4$.*

Proof. Let $\alpha := \lim_{i\to\infty} a_i$. From the recurrence relation, we can obtain

$$\alpha = \gamma - \frac{\gamma}{\alpha} \iff \frac{\alpha^2 - \gamma\alpha + \gamma}{\alpha} = 0.$$

In order to have a real solution, we should have $\gamma^2 - 4\gamma = \gamma(\gamma - 4) \geq 0$. □

Lemma C.8. *If $1 \leq \gamma < 4$, there exists a nonpositive element in the sequence $\{a_i\}_i$.*

Proof. Suppose towards contradiction that $\{a_i\}_i$ is all positive. Observe that, for any $i \geq 1$,

$$a_i - a_{i+1} = a_i - \left(\gamma - \frac{\gamma}{a_i}\right) = \frac{a_i^2 - \gamma a_i + \gamma}{a_i} > 0,$$

where the inequality holds since $1 \leq \gamma < 4$, implying that the sequence is strictly decreasing. This implies that $\{a_i\}_i$, leading to contradiction from Lemma C.7. □

By choosing a sufficiently large m, we can make n be arbitrarily large. By Lemma C.8, if $\gamma < 4$, there exists a nonpositive element in $\{a_i\}$. However, in that case, the algorithm cannot satisfy Equation (12). Hence, it is required to have $\gamma \geq 4$, completing the proof.

C.3.2. Trade-off between Consistency and Robustness

We are now ready to prove Theorem 5.5. We are given a constant $\lambda \in (0, 1)$. Let us bring the same instance where m is a sufficiently large number and $b(j) = j$ for $j \in [1, m]$. Let the prediction $\widehat{J} \leq m$ be also sufficiently large. Fix any deterministic algorithm that utilizes the prediction $\widehat{J}$. As we want to guarantee the algorithm to be $(1 + \lambda)$-consistent, the algorithm must have incurred at most $\lambda \cdot \widehat{J}$ before it clicks button with index $\widehat{J}$ or higher. Let n be the number of buttons the algorithm has clicked before it clicks button with index $\widehat{J}$ or higher. We thus obtain the following constraint on the algorithm:

$$\sum_{j=1}^{n} x_j \leq \lambda \cdot \widehat{J}, \tag{13}$$

where x_j again denotes the price of the button clicked at the j-th round as in the previous section.

Meanwhile, we also want the algorithm to be robust. Therefore, the algorithm should satisfy Equation (11) for $i = 1, 2, \cdots, n-1$, where $\gamma \geq 4$ now represents the robustness ratio, along with an additional constraint capturing the situation when $J = x_n + \varepsilon$ while the algorithm incurs at least $\widehat{J}$ after clicking button x_n. In other words, we have

$$\sum_{i=1}^{n} x_i + \widehat{J} \leq \gamma x_n. \tag{14}$$

Observe that, from Equations (13) and (14), we have

$$\left(1+\frac{1}{\lambda}\right)\cdot\sum_{j=1}^{n}x_j \le \gamma x_n.$$

Combining this inequality with Equation (12), we can obtain

$$\left(1+\frac{1}{\lambda}\right)\cdot\frac{\gamma}{a_{n-1}}x_n \le \gamma x_n,$$

resulting in that

$$1+\frac{1}{\lambda} \le a_{n-1}.$$

When m and $\widehat{J}$ is sufficiently large, we also have a sufficiently large n. We can thus replace a_{n-1} with $\alpha := \lim_{i\to\infty} a_i$ in the above inequality. From the proof of Lemma C.7, it is easy to see that $\alpha = \frac{\gamma+\sqrt{\gamma^2-4\gamma}}{2}$. We can thus obtain, for $\lambda \in (0,1)$,

$$\begin{aligned}
1+\frac{1}{\lambda} \le \frac{\gamma+\sqrt{\gamma^2-4\gamma}}{2} &\implies 2+\frac{2}{\lambda}-\gamma \le \sqrt{\gamma^2-4\gamma} \\
&\implies \left(2+\frac{2}{\lambda}-\gamma\right)^2 \le \gamma^2-4\gamma \\
&\implies \frac{4}{\lambda^2}+\frac{4}{\lambda}(2-\gamma)+4 \le 0 \\
&\implies \gamma \ge 2+\lambda+\frac{1}{\lambda}.
\end{aligned}$$

Together with Lemma 5.4, this completes the proof of Theorem 5.5.