Applied Online Algorithms with Heterogeneous Predictors

Jessica Maghakian [1] **Russell Lee** [2] **Mohammad Hajiesmaili** [2] **Jian Li** [3] **Ramesh Sitaraman** [2 4] **Zhenhua Liu** [1]

Abstract

For many application domains, the integration of machine learning (ML) models into decision making is hindered by the poor explainability and theoretical guarantees of black box models. Although the emerging area of algorithms with predictions offers a way to leverage ML while enjoying worst-case guarantees, existing work usually assumes access to only one predictor. We demonstrate how to more effectively utilize historical datasets and application domain knowledge by intentionally using predictors of *different* quantities. By leveraging the heterogeneity in our predictors, we are able to achieve improved performance, explainability, and computational efficiency over predictor-agnostic methods. Theoretical results are supplemented by large-scale empirical evaluations with production data demonstrating the success of our methods on optimization problems occurring in large distributed computing systems.

1. Introduction

The recent rapid advancements in artificial intelligence (AI) have the potential to reshape and transform industries worldwide. Yet barriers to complete AI adoption still remain. The top concerns of companies include maintaining the integrity of their brand, customer trust, and meeting external regulatory and compliance obligations (IBM, 2022). Black box ML models alone are unable to provide the theoretical guarantees and explainability required for trustworthy AI.

One promising solution comes from the recent area of algorithms with predictions (Mitzenmacher & Vassilvitskii, 2022). By incorporating ML predictions into classical online algorithms, researchers are able to create data-driven algorithms that have theoretical worst-case guarantees. These algorithms aim to optimally trade-off between *consistency* (performance when predictions are accurate) and *robustness* (performance when predictions are noisy). With proper calibration, these algorithms exploit high-quality predictions and default to classical online decision making when predictions are noisy.

A large gap still remains between the current algorithms with predictions framework and practice. The algorithms with predictions literature overwhelmingly consider access to one prediction. Defaulting to the classical online assumption of no information about the future when one prediction is inaccurate neglects the vast domain knowledge of practitioners. Businesses typically have many different mathematical and computational models to forecast quantities of interest that can greatly improve algorithm performance.

Several recent papers (Anand et al., 2022; Dinitz et al., 2022; Wang et al., 2020) consider the setting of online algorithms with multiple predictions. However, these works exclusively consider multiple predictions of the *same* quantity. For example, Dinitz et al. consider a portfolio of predictions generated by different ML models that cover the hyperparameter space. Although the predictors are able to specialize to different scenarios, there is no way to know a priori which predictor will do well on the following problem instance, thus requiring exploration to find the right choice.

We propose an alternative approach of incorporating predictions of *different* quantities. The first predictor we introduce is a *parameter predictor* that learns the correct value of a tunable parameter of an online algorithm. The second predictor is an *input predictor* that predicts the unknown future inputs of the online algorithm in the form of short look-ahead windows. We demonstrate that the two predictors have different theoretical properties and how to exploit these differences for improved performance.

Our Contributions. We initiate the study of algorithms with heterogeneous predictors and provide the first results showing that multiple noisy predictors outperform one, even in worst-case settings. Our approaches have better performance, explainability, and computational efficiency over predictor-agnostic methods, and can even tackle real-world challenges such as COVID-19 related distributional shifts.

[1]Department of Applied Mathematics & Statistics, Stony Brook University, Stony Brook, NY, USA [2]Manning College of Information and Computer Sciences, UMass Amherst, Amherst, MA, USA [3]Department of Electrical and Computer Engineering, SUNY Binghamton, Binghamton, NY, USA [4]Akamai Tech, Cambridge, MA, USA. Correspondence to: Jessica Maghakian <jessica.maghakian@stonybrook.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Background

Our contributions in this paper build on the recent area of online algorithms with predictions (sometimes called algorithms with untrusted advice or learning-augmented algorithms). This field of study aims to unify the strong average performance of ML methods with the worst-case performance guarantees of online algorithms. The area has seen fast progress since the seminal paper of Lykouris & Vassilvitskii (2018), however much of that progress has been driven by theory. This paper aims to move the ideas of algorithms with predictions one step closer to implementation in real-world systems. We initiate the study of online algorithms with heterogeneous predictors to more fully utilize the vast historical datasets and human domain knowledge available in practice. Although we consider rent/buy problems, our framework easily extends to many other settings.

We introduce the terminology parameter predictors and input predictors, but these concepts are present in previous literature. Many online problems can be cleanly characterized by problem-specific parameters, motivating the use of predictions of this parameter. Parameter predictions have been used in settings such as the break-even point of rent-or-buy problems (Purohit et al., 2018; Gollapudi & Panigrahi, 2019; Wang et al., 2020; Wei & Zhang, 2020; Lee et al., 2021), item frequencies in bin packing (Angelopoulos et al., 2022), maximum price in online conversion problems (Sun et al., 2021), and next page requests in caching (Lykouris & Vassilvitskii, 2018; Wei, 2020). In other online problems, predictions of problem inputs are more natural. Such problems include linear quadratic control (Li et al., 2022) and scheduling (Bamas et al., 2020; Bampis et al., 2022). Although parameter and input predictors are familiar, they are seldom used together.

3. Rent/Buy with Constrained Resources

In the traditional ski rental problem, a skier has the choice of either renting or buying skis but does not know how many days they will be skiing. This online optimization problem can be framed as satisfying a time-varying demand by either using a resource that charges by the average utilization (renting) or by the max utilization (buying). Although the demand is binary in the ski rental setting, the problem can be naturally generalized to rent/buy problems with continuous demand.

We consider a variant of the rent/buy problem where there are multiple renting options and multiple buying options, but resources now have capacity constraints. For brevity, we use the term AVG (resp., MAX) resource, instead of resource with a renting (resp. buying) pricing scheme. The rent/buy problem with multiple constrained resources has many practical applications, for instance running cloud applications (Khanafer et al., 2014), computation offloading in heterogeneous mobile clouds (Zhou et al., 2018), or minimizing bandwidth costs for large distributed systems (Adler et al., 2011) (see Section 5 for more details).

The rent/buy problem with multiple constrained resources can be stated as follows. For each time instant in a fixed time horizon $\mathcal{T} = \{1, \dots, T\}$, a demand $d(t)$ is revealed. This demand must be satisfied by using the AVG and MAX resources. Let $\mathcal{N}$ be the set of AVG resources (with $|\mathcal{N}| = n$) and $\mathcal{M}$ be the set of MAX resources (with $|\mathcal{M}| = m$). For an AVG resource $i \in \mathcal{N}$, the demand satisfied by that resource at time t is $x_i(t)$ and the corresponding total cost is proportional to $x_i = \frac{1}{T}\sum_{t\in\mathcal{T}} x_i(t)$. Similarly for a MAX resource $j \in \mathcal{M}$, the demand satisfied by that resource at time t is $y_j(t)$ and the corresponding total cost is proportional to $y_j = \max_{t\in\mathcal{T}} y_j(t)$. We use the convention that resources are indexed in ascending order of cost. Finally the resources have constant capacity over the time horizon, with $D_i^{\mathcal{N}}$ and $D_j^{\mathcal{M}}$ as the capacity of the i-th AVG and j-th MAX resource, respectively.

The resulting offline optimization problem is:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & C(\boldsymbol{x},\boldsymbol{y}) = \sum_{i\in\mathcal{N}} c_i^{\mathcal{N}} x_i + \sum_{j\in\mathcal{M}} c_j^{\mathcal{M}} y_j \qquad (1)\\ \text{s.t.} \quad & \sum_{i\in\mathcal{N}} x_i(t) + \sum_{j\in\mathcal{M}} y_j(t) = d(t), \quad t \in \mathcal{T},\\ & 0 \le x_i(t) \le D_i^{\mathcal{N}}, \quad i \in \mathcal{N}, t \in \mathcal{T},\\ & 0 \le y_j(t) \le D_j^{\mathcal{M}}, \quad j \in \mathcal{M}, t \in \mathcal{T}, \end{aligned}$$

where the first constraint ensures that all demand is satisfied.

3.1. Break-Even Structure of the Offline Optimal

Similarly to the ski rental problem, the offline optimal solution of the rent/buy problem with constrained resources has a break-even structure. Any instance of (1) has an optimal offline solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ with the following property: all arriving traffic up until a certain amount β^* (the optimal break-even point) is satisfied using the MAX resources, i.e. at each time instant $\sum_{j\in\mathcal{M}} y_j^*(t) = \min\{d(t), \beta^*, \sum_{j\in\mathcal{M}} D_j^{\mathcal{M}}\}$. All demand over that cutoff amount β^* is satisfied using the AVG resources, i.e. $\sum_{i\in\mathcal{N}} x_i^*(t) = d(t) - \sum_{j\in\mathcal{M}} y^*(t)$. Once the demand is divided among $\mathcal{N}$ and $\mathcal{M}$, the available resources are used greedily to capacity from the cheapest to the most expensive (see Algorithm 4). Thus the optimal solution of (1) is fully characterized by β^*.

3.2. Optimal Algorithm without Predictions

Previous work on the rent/buy problem with constrained resources introduced an optimal online deterministic algorithm (Adler et al., 2011). The optimal algorithm utilizes a *break-even point vector* $\boldsymbol{\beta} = \{\beta(1), \dots, \beta(T)\}$, so that the

break-even point can be refined as demand is revealed over the time horizon. Adler et al. demonstrate that this algorithm provably learns the correct value of β^* by the end of the time horizon and in doing so incurs a cost at most twice that of the offline optimal. We will refer to this algorithm as Dynamic Break-Even Optimization (DBO). Further details on DBO can be found in Appendix A.

4. Online Algorithms with Predictions

Although future-oblivious algorithms provide the best performance guarantee in the worst case setting, they are often pessimistic and overly conservative. In practice, algorithms that utilize predictions are preferred due to their better performance on typical inputs. The downside of using predictions is the lack of performance guarantees. We address this problem by first providing a theoretical analysis of data-driven methods for rent/buy problems with constrained resources. Specifically we consider the two extremes of predictors: (1) a static *parameter prediction* of the optimal break-even point available at the beginning of the time horizon and (2) a dynamic *input prediction* of the future demand that can be updated over time. Our analyses use the *competitive ratio* performance metric.

4.1. Parameter Predictions

Static predictions of the optimal break-even point are a natural starting point. We assume that there is an ML model that predicts $\hat{\beta}$ as the estimated value of the optimal offline break-even point β^*. The ML model can be trained using supervised methods on historical demand instances, with the labels generated by solving the offline optimization problem. We assume that this estimate is provided at the beginning of the time horizon. As predictions are rarely perfect, we denote the normalized error of the parameter predictor by $\epsilon_{\text{par}} = |\beta^* - \hat{\beta}| / \max_{t\in\mathcal{T}} d(t)$. Reliance on predictions with large ϵ_{par} may degrade the performance beyond the worst-case guarantees of pure online algorithms, while predictions with small ϵ_{par} almost match the offline optimal solution.

We now introduce Parameter Prediction Static Break-Even Optimization (parSBO), a simple data-driven algorithm that integrates the advice $\hat{\beta}$ into its decision-making. The algorithm mimics the offline optimal by dividing the demand among AVG and MAX resources and then uses the greedy subroutine. Details of parSBO are shown in Algorithm 1.

Theorem 4.1. *parSBO has a competitive ratio of* $1 + \epsilon_{par} \max\{\mu^-, \mu^+\}$*, where* $\mu^- = c_n^{\mathcal{N}}/c_1^{\mathcal{M}}$ *and* $\mu^+ = Tc_m^{\mathcal{M}}/c_1^{\mathcal{N}}$ *are problem-specific constants.*

See Appendix B.1 for the proof. The problem-dependent constants μ^- and μ^+ represent an upper bound on the worst-case ratio arising from respectively underestimating and overestimating β^*. Although parSBO achieves near optimal performance when $\hat{\beta}$ is of high quality, the algorithm is highly sensitive to the prediction error with performance that degrades linearly in ϵ_{par}.

Algorithm 1 parSBO (Static Break-even Optimization with Parameter Predictions)

Require: $\hat{\beta}$, $\mathbf{d}$, $\mathcal{N}$, $\mathcal{M}$
1: **for** $t = 1, \ldots, T$ **do**
2: Set $\beta_s(t) = \hat{\beta}$
3: Greedily satisfy the demand using Alg. 4 for $\beta_s(t)$
4: **end for**
5: **return** $(\boldsymbol{x}(\boldsymbol{\beta}_s), \boldsymbol{y}(\boldsymbol{\beta}_s)) = (\boldsymbol{x}, \boldsymbol{y})$

4.2. Input Predictions

Committing to a prediction $\hat{\beta}$ induces an inability to exploit new information that is revealed over the time horizon. As an alternative, we can instead use input predictions of the demand itself, that can be dynamically updated to improve their quality. In our model, at each time instant an algorithm will have access to predictions of the demand for the next w time instants, where $w \geq 2$. We represent the prediction of the demand at τ available at time t as $\hat{d}_t(\tau)$, the full window of predictions available at time t as $\hat{\boldsymbol{d}}_t = [\hat{d}_t(t), \ldots, \hat{d}_t(t+w-1)]$ and the full set of predictions available to the algorithm over the time horizon as $[\hat{\boldsymbol{d}}_t]_{t\in\mathcal{T}}$. Since the true current demand is available at each time instant, $\hat{d}_t(t) = d(t)$ for all $t \in \mathcal{T}$. However, it might not be the case that the remaining predictions are correct. We define the normalized total error ϵ_{in} of the input predictions $[\hat{\boldsymbol{d}}_t]_{t\in\mathcal{T}}$ as the sum of the error incurred in each prediction window:

$$\epsilon_{\text{in}} = \frac{1}{\max_{t\in\mathcal{T}} d(t)} \sum_{t=1}^{T} \sum_{\tau=t}^{\min\{t+w-1,T\}} |\hat{d}_t(\tau) - d(\tau)|.$$

Since input predictions forecast over a shorter time period and can be updated over the time horizon, usually $\epsilon_{\text{in}} \leq \epsilon_{\text{par}}$. Unlike the parameter predictions which only have two possible values given ϵ_{par} and β^*, there are infinitely many possible input predictions $[\hat{\boldsymbol{d}}_t]_{t\in\mathcal{T}}$ that have an associated error ϵ_{in}.

Although there are many classic algorithms in the class of Model Predictive Control that optimize using look-ahead windows, they are ill-suited to the time-coupling present in rent/buy problems. For example, the popular Receding Control Horizon algorithm (Bellingham et al., 2002) with access to perfect predictions achieves a competitive ratio of T/w on our problem. We instead introduce Input Prediction Dynamic Break-Even Optimization (inDBO), a modification of DBO that utilizes input predictions (shown as Algorithm 2).

Algorithm 2 inDBO (Dynamic Break-even Optimization with Input Predictions)

Require: $[\hat{\boldsymbol{d}}_t]_{t\in\mathcal{T}}$, $\mathbf{d}$, $\mathcal{N}$, $\mathcal{M}$
1: **for** $t = 1, \dots, T$ **do**
2: Set $d'(\tau) = d(\tau)$ for $\tau \le t$, $d'(\tau) = \hat{d}_t(\tau)$ for $t < \tau \le \min\{t + w - 1, T\}$, $d'(\tau) = 0$ otherwise
3: Solve (1) for $\mathbf{d}'$ and set $\beta_d(t) = \max\{\beta^*, \beta_d(t-1)\}$
4: Greedily satisfy the demand using Alg. 4 for $\beta_d(t)$
5: **end for**
6: **return** $(\boldsymbol{x}(\boldsymbol{\beta}_d), \boldsymbol{y}(\boldsymbol{\beta}_d)) = (\boldsymbol{x}, \boldsymbol{y})$

For the performance analysis of inDBO, the adversary has an *error budget* that can be allocated in any way to generate the worst-possible predictions (Comden et al., 2019). For example the error ϵ_{in} can be distributed evenly over the billing period or used entirely on the predictions for a specific time-slot.

Theorem 4.2. *inDBO is* $\big(2 - (w - \epsilon_{in}\mu^-)/T\big)$*-competitive.*

The proof can be found in Appendix B.2. Although the competitive ratios of both parSBO and inDBO have a linear dependence on the error of their respective predictions, the coefficient for parSBO is larger by a factor of T. Compared to parSBO, inDBO performs worse with perfect predictions, however it is much more robust to prediction error.

4.3. Comparison of Online Algorithms

We now discuss and compare the three online algorithms seen so far: (1) DBO: the best online algorithm that does not use any predictions (Adler et al., 2011); (2) parSBO: our first data-driven algorithm that uses a prediction of the parameter setting; and (3) inDBO: our second data-driven algorithm that uses multiple forecasts of future inputs.

4.3.1. Consistency and Robustness

The competitive ratio metric does not capture the full nuance of how different algorithms behave under variable prediction error levels. We now define two new performance metrics: *consistency* and *robustness*, introduced in Purohit et al. (2018) and Lykouris & Vassilvitskii (2018). Consistency is the competitive ratio of an algorithm when it has access to perfect predictions, while robustness is the competitive ratio of an algorithm when it has access to imperfect predictions. In our setting, the extent to which a prediction is incorrect can vary, so the robustness of an algorithm will be a function of the prediction error. Table 1 shows the consistency and robustness of the algorithms discussed previously in this section. Since DBO does not use predictions, its consistency and robustness are both 2.

Table 1 shows the trade-offs of each algorithm. Although DBO has bounded worst-case performance in any setting, it is unable to exploit available domain knowledge or historical data. On the other hand, parSBO matches the offline optimal when predictions are accurate, but its performance degrades rapidly with error. Finally, inDBO provides a middle ground between DBO and parSBO by offering a modest improvement in performance with accurate prediction in exchange for a lessened impact from inaccurate predictions. Figure 1 shows a practical take-away of Table 1 on how to choose between the three online algorithms. When parameter predictions are accurate, parSBO is the best choice. In the case that the parameter prediction is inaccurate, either DBO or inDBO could be the best choice, depending on the quality of the input predictions.

Algorithm	Consistency	Robustness
DBO	2	2
parSBO	1	$1 + \epsilon_{\text{par}} \max\{\mu^-, \mu^+\}$
inDBO	$2 - w/T$	$2 - w/T + \epsilon_{\text{in}}\mu^-/T$

Table 1. Consistency and robustness of online and prediction-based algorithms. Using only one of these 3 algorithms in all settings results in a hard trade-off between good performance in typical settings (low prediction error) and good performance in the worst-case (high prediction error).

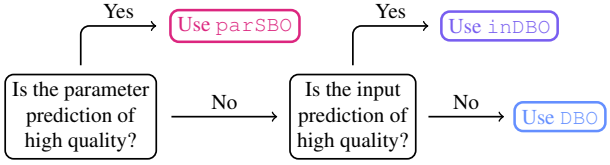

Figure 1. A practical flowchart that summarizes the theoretical results in Table 1. The flowchart suggests a classification approach, where the classifier can either be an ML model (see Section 6.1) or a human expert.

Theorem 4.3. *parSBO and inDBO achieve the optimal consistency for their respective predictors.*

The optimality of inDBO can be shown by a reduction to the ski-rental problem (Lu et al., 2012). Theorem 4.3 shows that the trade-off seen in Table 1 is not an artifact of the online algorithms parSBO and inDBO but rather a fundamental information theoretic property of the different types of predictors themselves. A parameter predictor that ambitiously predicts the optimal strategy is on the opposite end of the risk-reward spectrum compared to an input predictor that slowly learns the optimal break-even point by forecasting windows of future demand.

4.3.2. Computational Complexity

Another consideration when choosing which algorithm to implement is the cost of computation. The computational

cost has two components: (1) the cost of executing the online algorithms and (2) the cost of generating the predictions. For the former, parSBO has the lowest cost of $O(n \log n + m \log m)$ corresponding to sorting the resources for the greedy subroutine. However any of the $\star$DBO strategies must compute the offline optimization problem T times over the time horizon. At a cost of $O(L(\log m + T \log n))$ per iteration, both DBO and inDBO have a complexity of $O(TL(\log m + T \log n) + n \log n + m \log m)$, where L is the number of bits required to represent the max demand.

Regarding the cost of predictions, there has been a recent effort in designing algorithms that utilize fewer predictions following the work of Bhaskara et al. (2021) and Im et al. (2022). While DBO has no associated prediction cost and the cost of parSBO is negligible as the algorithm only requires one prediction per problem, inDBO requires at least T predictions of length w. For the best performance, it might even be necessary to retrain the input prediction model on new data. From this perspective DBO might be preferable over inDBO, even when input prediction error is low, simply because the improvement in algorithm performance does not justify the computational cost. Existing predictor-agnostic methods are unable to incorporate this nuance into their decision making unless explicitly instructed.

4.4. Robust Data-Driven Decision Making

The classification approach suggested by Figure 1 requires some degree of certainty in anticipated prediction quality. However it is also possible that practitioners are uncertain whether ϵ_{par} will be small and instead might want a way to hedge their bets. To allow for this, we introduce a *trust parameter* that allows the user to modify the performance of an algorithm by adjusting the reliance on predictions.

Definition 4.4. Let λ defined over the interval $(0, 1]$ be a trust parameter that indicates the level of trust in the parameter predictor. Employing customary notation, $\lambda \to 0$ represents full trust in the parameter predictor, whereas $\lambda \to 1$ indicates no trust in the parameter predictions at all. Any values other than 0 or 1 indicate partial trust in the parameter prediction.

The worst-case cost of an online algorithm equipped with a trust parameter is a function of the problem parameters, prediction errors and chosen value of λ.

We incorporate λ into decision-making by using the dynamic break-even strategies of DBO and inDBO but now incentivizing MAX resources used by parSBO with $\hat{\beta}$ and disincentivizing those that are not. Rather than solve the true problem (1), the algorithm instead solves another problem that only differs in the costs of the MAX resources, with incentivized resources having cost $\lambda c_j^{\mathcal{M}}$ and disincentivized resources having cost $c_j^{\mathcal{M}}/\lambda$. The algorithm is shown as Algorithm 3.

Algorithm 3 RoBO (Robustified Break-even Optimization)

Require: λ, $\hat{\beta}$, $[\hat{\boldsymbol{d}}_t]_{t\in\mathcal{T}}$, $\mathbf{b}$, $\mathcal{N}$, $\mathcal{M}$
for $t = 1, \ldots, T$ **do**
  if using both predictors **then**
    Set $d'(\tau) = d(\tau)$ for $\tau \le t$, $d'(\tau) = \hat{d}_t(\tau)$ for $t < \tau \le \min\{t + w - 1, T\}$, $d'(\tau) = 0$ o.w.
  else
    Set $d'(\tau) = d(\tau)$ for $\tau \le t$, $d'(\tau) = 0$ o.w.
  end if
  Solve (1) for $\mathbf{d}'$ with MAX resources used by $\hat{\beta}$ discounted by λ, all other MAX resources marked up by $1/\lambda$, and set $\beta_\lambda(t) = \max\{\beta^*, \beta_\lambda(t-1)\}$
  Greedily satisfy the demand using Alg. 4 for $\beta_{\boldsymbol{\lambda}}(t)$
end for
return $(\boldsymbol{x}(\boldsymbol{\beta}_{\boldsymbol{\lambda}}), \boldsymbol{y}(\boldsymbol{\beta}_{\boldsymbol{\lambda}})) = (\boldsymbol{x}, \boldsymbol{y})$

Our approach builds off work on other rent/buy problems that use a similar incentivization scheme to achieve good consistency-robustness tradeoffs (Purohit et al., 2018; Angelopoulos et al., 2020; Wang et al., 2020; Lee et al., 2021). Unlike previous work, we allow the user to either use DBO (for RoBO-1) or inDBO (for RoBO-2) to robustify the parameter predictor and provide protection in the worst-case. Table 2 shows the consistency and robustness results, with proofs in Appendix B.3. For the robustness analysis of RoBO-2, we assume a powerful adversary that couples ϵ_{par} and ϵ_{in} in the worst-possible way. Surprisingly, even with such a strong adversary, using both noisy predictors provides better worst-case performance than just one.

Algorithm	Consistency	Robustness
RoBO-1	$1 + \lambda$	$1 + \lambda + \epsilon_{\text{par}}\big(\min\{1/\lambda, \mu^-\} - \lambda\big)$
RoBO-2	$1+\lambda(1-w/T)$	$1+\lambda\big(1-(w-\epsilon_{\text{in}}\mu^-)/T\big)+ \epsilon_{\text{par}}\big(\min\{1/\lambda, \mu^+\} - \lambda\big)$

Table 2. Consistency and robustness of robustified algorithms that use either 1 (RoBO-1) or both (RoBO-2) predictors. Varying the trust parameter λ smoothly trades-off between performance in the low prediction error and high prediction error regimes.

Theorem 4.5. *RoBO-2 Pareto-dominates RoBO-1 when the following conditions are met: (i) symmetry of worst-case instances ($\mu^- \approx \mu^+$) and (ii) sufficiently large look-ahead window size ($w \ge \mu^+$).*

This counter-intuitive result holds for all values of λ and under mild assumptions. The first, *symmetry of worst-case instances*, requires that underestimating the break-even point is roughly as bad as overestimating it. This is necessary

because RoBO−1 and RoBO−2 have their worst-case instances from underestimating and overestimating the break-even point, respectively. The second requirement, *sufficiently large look-ahead window size*, requires that RoBO−2 have access to large enough windows of input predictions to improve over RoBO−1. Satisfying this condition can be as simple as having input prediction windows of size $w > 2$.

The vast majority of previous work on learning-augmented algorithms promotes defaulting to oblivious online decision-making when predictions are noisy. Theorem 4.5 challenges this approach by demonstrating that multiple noisy predictors can be combined and consistently provide improved performance, even in the worst-case.

5. Experimental Setup

We now experimentally evaluate the performance of our algorithms. Our goal is to provide empirical conclusions that will help future theory be better aligned with the desiderata of the real world. The experiments focus on the problem of *bandwidth cost minimization* for large distributed systems.

5.1. Bandwidth Cost Minimization

Modern internet-scale distributed services such as Amazon's web services (Jacquemart et al., 2019) or Akamai's CDN services rely on a large platform of servers deployed in thousands of data centers around the world (Maggs & Sitaraman, 2015). The operating expenditure of these systems is dominated by the bandwidth cost of the network traffic between the servers and clients (Hasan et al., 2014). For large CDNs, bandwidth costs are on the order of 100s of millions of dollars per year.

Bandwidth cost minimization is an example of a rent/buy problem with constrained resources (Adler et al., 2011). The resources correspond to different data centers, each with a bandwidth contract that dictates the maximum available capacity and either a usage or peak-based[1] billing scheme.

5.2. Real World Datasets

In our evaluation we use two different real world datasets to demonstrate challenges occurring in real large distributed computing systems. The first dataset showcases the variety in user traffic predictability across different geographic locations. The second dataset documents the shift in internet demand patterns during the COVID-19 lockdowns. Although we focus on bandwidth cost minimization, the challenges present in our traces arise from internet usage and services. The many other rent/buy optimization problems in the Internet ecosystem have similar challenges.

Dataset 1: Traffic Predictability Across Metro-Areas

The predictability of user internet demand can vary greatly across geographic locations. For example, larger metro-areas tend to have more predictable profiles due to the smoothing effects of mass aggregation, whereas the demands of smaller metro-areas may be more volatile and unpredictable. Other factors such as geopolitical or sporting events might affect user trends on a country-wide basis.

Our first dataset is a proprietary production dataset from Akamai, one of the largest content delivery networks (CDN) in the world. It consists of production traffic demands collected from 2000+ data centers in 75+ countries. In our experiments, we take the system-wide traffic and partition it into local instances of the bandwidth cost minimization problem for more than 180 different metro-areas. Each metro-area has a distinct profile, and relying on only one of DBO, parSBO or inDBO for all metro-areas can result in the hard trade-offs seen in Table 1.

Dataset 2: The Impact of COVID-19 Lockdowns

The global COVID-19 pandemic brought historically unprecedented changes in internet traffic demands. Lockdowns resulted in a migration of in-person interactions to virtually hosted alternatives. Bandwidth usage surged as video-conferencing services such as Zoom saw a 10x increase in usage. Over the month of March 2020, Akamai experienced a 30% increase in global internet usage, with a recorded peak traffic of 167 Tbps – more than twice the recorded peak traffic of 82 Tpbs in March 2019 (Branscombe, 2020).

Our second dataset captures this distributional shift. We digitized publicly available records of public peering traffic at Milan's internet exchange to recover demand data at 30 minute resolution for the duration of the COVID-19 lockdown in Italy (March 9 to 18 May 2020) as well as a baseline collected in January 2020[2]. Figure 2 shows the sudden shift in internet traffic demands as a result of lockdown measures.

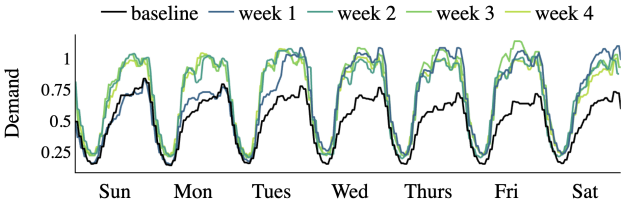

Figure 2. Dramatic shift of internet demand (Tb/s) in Milan due to COVID-related lockdowns. The first week of lockdowns saw average demand increase by 36% and peak demand increase by 31% compared to a pre-Covid baseline.

[1]Data centers typically charge using the 95$^{\text{th}}$ percentile for peak-based billing purposes, often referred to as *burstable billing*. However since traffic cannot be controlled precisely enough to take advantage of the 5% of "free" traffic per billing period, we can safely model the 95$^{\text{th}}$ percentile as a max function.

[2]Dataset 2 and related experiments are available at https://github.com/jmaghakian/covid_exp

6. Results and Discussion

We discuss 4 experiments. The first explores the efficacy of a simple meta-algorithm that uses a classification approach inspired by Figure 1. Our second experiment compares the performance of this simple classification meta-algorithm with an algorithm from the literature that uses multiple parameter predictions (Wang et al., 2020). Finally, the last two experiments investigate the performance of our algorithms in the presence of dramatic distributional shift.

6.1. Simple Classification Succeeds on Production Data

Inspired by Figure 1, we ask the following question: can system operators use a classification-based approach to determine which locations can benefit from relying on parameter predictions, which are more suited to input predictions and which are better using no predictions at all? The proposed meta-algorithm, `meta`, uses a classifier to a priori select which of `DBO`, `parSBO` or `inDBO` it will exactly mimic for the upcoming billing period.

We evaluate the performance of our heterogeneous predictor approach by implementing `DBO`, `parSBO` and `inDBO` across the 180+ metro-areas of Dataset 1. The data spans one month at five minute resolution and we use two weeks for training predictors and two weeks for testing. At each location and for each billing period, we instantiate 50 different instances of possible bandwidth prices to allow for a range of algorithm behaviors.

To generate the parameter predictions, we use linear regression with price setting, metro-area location and size as the dependent variables. For the input predictions, we predicted day-ahead windows using the AutoARIMA forecaster available through `sktime`. The classifier was trained using the same features as the parameter predictions, with the best choice of `DBO`, `parSBO` or `inDBO` on historical data as labels for the training dataset.

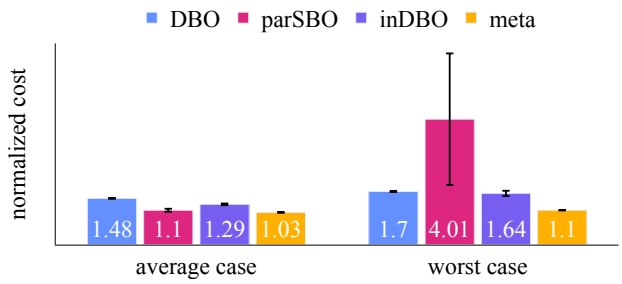

Figure 3. Our large-scale evaluations (18000+ trials) demonstrate that the empirical behavior of the three online algorithms matches the theoretical results summarized in Table 1. The success of our classification meta-algorithm shows that context and historical performance can be used to exploit predictors with no exploration.

For each location, we report the average and worst-case normalized cost for each algorithm. The results are aggregated over locations with the means and standard errors are shown in Figure 3.

Figure 3 demonstrates that in practice, although parameter predictions provide the strongest performance on average, `parSBO` is susceptible to very bad worst case performance. `DBO` provides the opposite trade-off, while `inDBO` is between the two extremes. Our classification-based meta algorithm can identify good strategies to use in each situation and impressively provides worst-case performance that matches the best average-case performance of the three online algorithms. Table 3 shows the frequency with which `meta` chooses each of the online algorithms.

DBO	parSBO	inDBO
1.96%	93.76%	4.28%

Table 3. Usage of each algorithm by our classification meta-algorithm. By selecting `parSBO` for the many scenarios where it performs very well and selecting either `DBO` or `inDBO` for the rare situations when ϵ_{par} is very large, our meta-algorithm can achieve the strong performance seen in Figure 3.

Since our meta-algorithm only incurs the overhead associated with its chosen online algorithm, Table 3 demonstrates that the approach has minimal computational cost. For example, `meta` uses at most 1 prediction for more than 95% of trials. In addition to a parsimonious use of predictions, our meta-algorithm exploits the static strategy and computes a dynamic approach for less than 7% of trials.

6.2. Simple Classification Even Outperforms SOTA

Now we compare the performance of `meta` against online algorithms in the literature that use multiple predictions. Of the previous work, only Wang et al. (2020) consider a setting similar enough that we could adapt their algorithm to bandwidth cost minimization. The adapted algorithm (which we refer to as `WLW`) requires multiple parameter predictions and utilizes a trust parameter λ to balance between data-driven and online decision making.

To generate multiple parameter predictions, we modified the setting of the previous experiment to create two different training sets for the different parameter predictors. We refer to the data-driven algorithms that use the first and second parameter predictions as `parSBO-1` and `parSBO-2`, respectively. Although `meta` was originally designed to use only one of each predictor type, the meta-algorithm can be easily adapted to use a classifier with 4 labels: `DBO`, `parSBO-1`, `parSBO-2` and `inDBO`.

We evaluate `parSBO-1`, `parSBO-2` and `meta`, as well as `WLW` with three settings of trust parameter: $\lambda = 0.25$, $\lambda = 0.5$ and $\lambda = 0.75$. As in the previous experiment,

the average and worst-case normalized cost are reported for each algorithm and aggregated over locations. The results of the experiment (mean and standard error for each algorithm and setting) are shown in Table 4.

Algorithm	Average Case	Worst Case
parSBO-1	1.014 ± 0.005	2.817 ± 1.625
parSBO-2	1.039 ± 0.011	6.107 ± 4.164
meta	$\mathbf{1.022 \pm 0.007}$	$\mathbf{1.236 \pm 0.055}$
$\text{WLW}_{\lambda=0.25}$	1.052 ± 0.004	1.630 ± 0.315
$\text{WLW}_{\lambda=0.5}$	1.102 ± 0.006	1.669 ± 0.312
$\text{WLW}_{\lambda=0.75}$	1.155 ± 0.008	1.730 ± 0.312

Table 4. Results over 9000+ trials show that our meta algorithm improved over the state-of-the-art by up to 11.5% in the average case and up to 28.6% in the worst-case, all while running an order of magnitude faster. Although increasing the value of λ (i.e. decreasing use of historical data) of WLW should decrease the worst-case cost, the opposite effect was observed in our experiments.

As observed in the previous experiment, parSBO-1 and parSBO-2 demonstrated strong average-case performance, at the cost of poor worst-case performance. Even with two different parameter predictors to choose from, meta successfully selected among the different online algorithms to achieve the best average-case and worst-case performance by a significant margin.

The state-of-the-art competitor, WLW, also achieved substantially improved worst-case performance compared to both parameter predictors alone. Surprisingly, increasing λ for WLW resulted in a higher worst-case competitive ratio. This observation supports our hypothesis that reverting to online decision making when predictions are noisy is often too conservative and pessimistic.

DBO	parSBO-1	parSBO-2	inDBO
5.04%	64.65%	26.40%	3.91%

Table 5. Usage of each algorithm by our classification meta-algorithm when there are two different parameter predictions. Our meta-algorithm leveraged both parameter predictions to achieve improved performance.

In addition to improved performance for both average-case and worst-case settings, our meta-algorithm also runs substantially faster than the competition. WLW is a dynamic strategy and thus must recompute an instance of the offline optimization problem at each time instance. In comparison, meta only utilizes a dynamic approach for less than 9% of the trials (see Table 5). Since running inDBO took 10^5 more time than running parSBO during our experiments, on average meta ran an order of magnitude faster than WLW.

6.3. On Real World Data, 2 Predictors Are Better than 1

We now discuss experiments conducted using our second dataset, which focuses on the distributional shift due to COVID-19 lockdowns. Given the uncertainty in how government regulations and local lockdown measures were unfolding in March 2020, it would be natural for system operators to want to hedge their bets by using an algorithm equipped with a trust parameter. This motivates the use of our algorithms RoBO-1 and RoBO-2. In theory, Theorem 4.5 states that under some mild assumptions, RoBO-2 is strictly better than RoBO-1. We now examine whether this is the case on real-world data, which may or may not satisfy the assumptions of Theorem 4.5.

We implemented RoBO-1 and RoBO-2 for a wide range of trust parameter settings during first month of lockdowns in Italy. Results are shown in Figure 4. The parameter prediction generated using historical data performed poorly, as seen in the performance of both algorithms for $\lambda = 0$. RoBO-1 exhibits performance very similar to the theoretical curves predicted by Table 2. In particular, it shows the classical graceful degradation for $\lambda \geq 0.3$.

Although RoBO-2 does not exhibit a smooth performance trade-off as a function of λ, it instead shows immediate improvement for very small values of λ. Notably, setting low values of λ for RoBO-2 results in better performance than parSBO and inDBO alone. One drawback of RoBO-2 is that the performance predicted in Table 2 is overly pessimistic due to the choice of error analysis, making it difficult to choose values of λ using theoretical results alone.

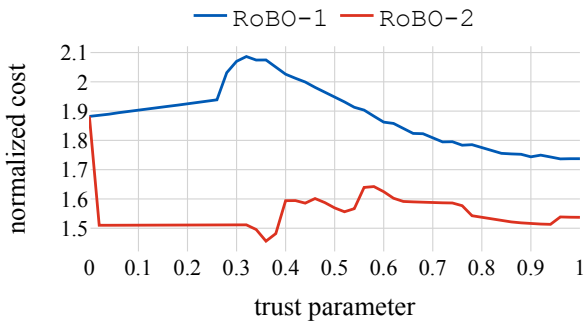

Figure 4. Performance of RoBO-1 and RoBO-2 with varying trust levels λ on the first month of lockdown data. Balancing the parameter prediction with online decision making actually makes RoBO-1's performance worse before eventually improving it. RoBO-2 immediately benefits from robustifying the parameter prediction with input predictions and achieves a significant improvement starting with λ values as low as 0.02.

6.4. Parameter Predictions Excel on the New Normal

For the months of April and May, our simulations showed that parSBO surprisingly outperformed the other online

algorithms by margins similar to pre-Covid baselines in the region. In other words, completely relying on parameter predictions generating using the small amount of lockdown bandwidth data performed significantly better than `parSBO`, `inDBO` or `RoBO-1` and `RoBO-2` with trust parameter settings $\lambda \neq 0$.

Figure 5 gives insight into this result. Although the change from pre-Covid demand to that of March 2020 was a significant distributional shift, internet demands soon stabilized into a "new normal" that could be predicted fairly accurately after a relatively short period of observation. The remaining uncertainty about the future from an optimization perspective was not about how bandwidth traffic demands would look under lockdown but rather how long the lockdowns would continue.

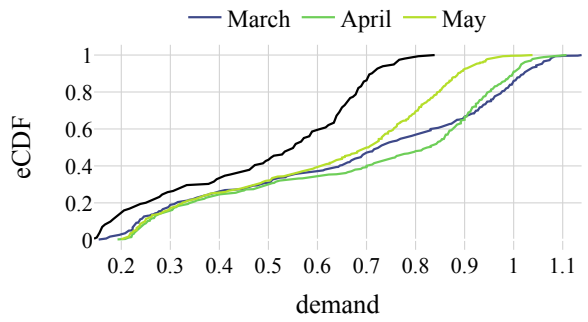

Figure 5. Distribution of monthly demand, with pre-Covid baseline shown in black. Although there is a large shift from the baseline to March demand, subsequent months are not too different in profile, exhibiting a "new normal" that allows `parSBO` to perform well.

Our experiments on the second dataset suggest that even the drastic motivation of distributional shift might not require algorithms with trust parameters post-shift. On real-world examples, more simple algorithms such as our classification-based approach in Section 6.1 can be surprisingly effective.

7. Related Work

Online algorithms were multiple predictions were first introduced for the ski rental problem (Gollapudi & Panigrahi, 2019), with subsequent work on multi-shop ski rental (Wang et al., 2020), facility location (Almanza et al., 2021) and online covering (Anand et al., 2022). The listed works are agnostic to how the predictions are generated, resulting in low explainability for why certain predictions are chosen.

The work of Dinitz et al. (2022) addresses the heterogeneity of the predictions arising from different hyperparameter choices for the ML models. However these differences are not known a priori and as a result Dinitz et al. focus on the challenge of learning which predictions in the portfolio are best in a computationally efficient manner. In comparison, our methods are more transparent and allow for easier integration of human expertise when selecting which predictors to use.

8. Conclusion

We initiated the study of online algorithms with heterogeneous predictors. By theoretically characterizing the performance of the best online algorithm for each predictor type, we provided improved explainability and clarity for the scenarios in which each should be used. We provided two ways to leverage the different predictors, (1) a classification-based meta-algorithm for when practitioners are confident in their ability to correctly predict error levels and (2) a trust-parameter equipped algorithm that allows the user to hedge their bets and robustify the parameter predictor as much as they desire. Our theoretical contributions are supplemented by large-scale empirical evaluations using real-world data.

Although we introduced our methodology for the rent/buy problem with constrained resources, our algorithms can be directly applied to other rent/buy problems. We anticipate that the performance guarantees will be similar. Our approach can also be applied to any other problem that has a clear choice of parameter prediction, such as facility location problems (Almanza et al., 2021; Jiang et al., 2022) or online load balancing (Lattanzi et al., 2020). Finally, the online algorithms with heterogeneous predictors framework need not be restricted to input and parameter predictors. Other quantities of interest could include distributional or aggregate statistics of problem inputs.

9. Acknowledgements

The authors thank the anonymous reviewers for their valuable comments and suggestions. Jessica Maghakian and Zhenhua Liu's research is supported in part by the NSF under grant numbers 1650114, 2214980, 2046444, 2106027 and 2146909. The work of Mohammad Hajiesmaili is supported by NSF CAREER-2045641, NGSDI-2105494, CPS-2136199, CNS-2106299, and CNS-2102963. Jian Li's research is supported in part by the NSF grants 2104880, 2148309 and the U.S. Army Research Office (ARO) grant W911NF-23-1-0072.

References

Adler, M., Sitaraman, R. K., and Venkataramani, H. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18):4007–4020, 2011.

Almanza, M., Chierichetti, F., Lattanzi, S., Panconesi, A., and Re, G. Online facility location with multiple advice. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*,

volume 34, pp. 4661–4673. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/250473494b245120a7eaf8b2e6b1f17c-Paper.pdf.

Anand, K., Ge, R., Kumar, A., and Panigrahi, D. Online algorithms with multiple predictions. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 582–598. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/anand22a.html.

Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., and Renault, M. Online computation with untrusted advice. In *Proc. of ITCS*, 2020.

Angelopoulos, S., Kamali, S., and Shadkami, K. Online bin packing with predictions. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 4574–4580. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/635. URL https://doi.org/10.24963/ijcai.2022/635. Main Track.

Bamas, E., Maggiori, A., Rohwedder, L., and Svensson, O. Learning augmented energy minimization via speed scaling. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15350–15359. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/af94ed0d6f5acc95f97170e3685f16c0-Paper.pdf.

Bampis, E., Dogeas, K., Kononov, A., Lucarelli, G., and Pascual, F. Scheduling with untrusted predictions. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 4581–4587. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/636. URL https://doi.org/10.24963/ijcai.2022/636. Main Track.

Bellingham, J., Richards, A., and How, J. P. Receding horizon control of autonomous aerial vehicles. In *Proc. of IEEE ACC*, 2002.

Bhaskara, A., Cutkosky, A., Kumar, R., and Purohit, M. Logarithmic regret from sublinear hints. *Advances in Neural Information Processing Systems*, 34:28222–28232, 2021.

Branscombe, M. *The Network Impact of the Global COVID-19 Pandemic*, 2020. URL https://thenewstack.io/the-network-impact-of-the-global-covid-19-pandemic/.

Comden, J., Yao, S., Chen, N., Xing, H., and Liu, Z. Online optimization in cloud resource provisioning: Predictions, regrets, and algorithms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(1): 16, 2019.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Algorithms with prediction portfolios. *arXiv preprint arXiv:2210.12438*, 2022.

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

Hasan, S., Gorinsky, S., Dovrolis, C., and Sitaraman, R. K. Trade-offs in optimizing the cache deployments of cdns. In *Proc. of IEEE INFOCOM*, 2014.

IBM. Ibm global ai adoption index 2022. *https://www.ibm.com/downloads/cas/GVAGA3JP*, 2022.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. *arXiv preprint arXiv:2202.04262*, 2022.

Jacquemart, Q., Pigout, C., and Urvoy-Keller, G. Inferring the deployment of top domains over public clouds using dns data. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 57–64. IEEE, 2019.

Jiang, S. H.-C., Liu, E., Lyu, Y., Tang, Z. G., and Zhang, Y. Online facility location with predictions. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=DSQHjibtgKR.

Khanafer, A., Kodialam, M., and Puttaswamy, K. P. To rent or to buy in the presence of statistical information: The constrained ski-rental problem. *IEEE/ACM Transactions on Networking*, 23(4):1067–1077, 2014.

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

Lee, R., Maghakian, J., Hajiesmaili, M., Li, J., Sitaraman, R., and Liu, Z. Online peak-aware energy scheduling with untrusted advice. *ACM SIGENERGY Energy Informatics Review*, 1(1):59–77, 2021.

Li, T., Yang, R., Qu, G., Shi, G., Yu, C., Wierman, A., and Low, S. Robustness and consistency in linear quadratic control with untrusted predictions. 50 (1), 2022. ISSN 0163-5999. doi: 10.1145/3547353.3522658. URL https://doi.org/10.1145/3547353.3522658.

Lu, T., Chen, M., and Andrew, L. L. Simple and effective dynamic provisioning for power-proportional data centers. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1161–1171, 2012.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. In *Proc of ICML*, 2018.

Maggs, B. M. and Sitaraman, R. K. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

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

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ml predictions. In *Proc. of NeurIPS*, 2018.

Sun, B., Lee, R., Hajiesmaili, M., Wierman, A., and Tsang, D. Pareto-optimal learning-augmented algorithms for online conversion problems. *Advances in Neural Information Processing Systems*, 34:10339–10350, 2021.

Wang, S., Li, J., and Wang, S. Online algorithms for multi-shop ski rental with machine learned advice. In *Proc. of NeurIPS*, 2020.

Wei, A. Better and simpler learning-augmented online caching. In *Proc. of APPROX/RANDOM*, 2020.

Wei, A. and Zhang, F. Optimal robustness-consistency tradeoffs for learning-augmented online algorithms. In *Proc. of NeurIPS*, 2020.

Zhou, B., Dastjerdi, A. V., Calheiros, R. N., and Buyya, R. An online algorithm for task offloading in heterogeneous mobile clouds. *ACM Transactions on Internet Technology (TOIT)*, 18(2):1–25, 2018.

A. Previous Work on Rent/Buy Problems with Constrained Resources (Adler et al., 2011)

All the algorithms in this paper use a greedy subroutine to optimally distribute the AVG and MAX demand among the available resources. For a specified break-even point β, Algorithm 4 returns the optimal allocations $\boldsymbol{x}(\beta)$ and $\boldsymbol{y}(\beta)$ among the AVG and MAX resources, respectively.

Algorithm 4 Greedy Subroutine

Require: $\beta, \mathbf{d}, \mathcal{N}, \mathcal{M}$
for $t = 1, \ldots, T$ **do**
  $i = 1, j = 1, d_m = 0, d_a = 0$
  while $d_m < \beta$ and $j \leq |\mathcal{M}|$ **do**
    $y_j(t) = \min\{d(t) - d_m, D_j^{\mathcal{M}}\}$
    $d_m = d_m + y_j(t), j = j + 1$
  end while
  while $d_a < d(t) - d_m$ and $i \leq |\mathcal{N}|$ **do**
    $x_i(t) = \min\{d(t) - d_m - d_a, D_i^{\mathcal{N}}\}$
    $d_a = d_a + x_i(t), i = i + 1$
  end while
end for
return $(\boldsymbol{x}(\beta), \boldsymbol{y}(\beta)) = (\boldsymbol{x}, \boldsymbol{y})$

Adler et al. also provided an optimal online deterministic algorithm for the rent/buy problem with constrained resources. Algorithm 5 shows this algorithm.

Algorithm 5 `DBO` (Dynamic Break-even Optimization)

Require: $\mathbf{d}, \mathcal{N}, \mathcal{M}$
for $t = 1, \ldots, T$ **do**
  Set $d'(\tau) = d(\tau)$ for $\tau \leq t$ and $d'(\tau) = 0$ otherwise
  Solve (1) for demand $\mathbf{d}'$ and set $\beta_o(t) = \beta^*$
  Greedily satisfy the demand with Algorithm 4 for $\beta_o(t)$
end for
return $(\boldsymbol{x}(\boldsymbol{\beta}_o), \boldsymbol{y}(\boldsymbol{\beta}_o)) = (\boldsymbol{x}, \boldsymbol{y})$

The following lemma provides properties about the break-even points $\boldsymbol{\beta}_o$ of the algorithm `DBO`, the optimal routing $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ through the AVG and MAX resources respectively, and the costs C_{AVG} and C_{MAX} incurred by using these routing. By combining properties (2) and (3), `DBO` can be proven to be 2-competitive.

Lemma A.1. *(Adler et al., 2011) For any input* $\mathbf{d}$ *to* `DBO`*, the following holds: (1)* $\beta_o(t) \leq \beta_o(t+1)$ *for all* $t \in \mathcal{T}$*, with* $\beta_o(T) = \beta^*$*; (2)* $C_{\text{MAX}}(\boldsymbol{y}(\boldsymbol{\beta}_o)) = C_{\text{MAX}}(\boldsymbol{y}^*)$*; (3)* $C_{\text{AVG}}(\boldsymbol{x}(\boldsymbol{\beta}_o)) \leq C(\boldsymbol{x}^*, \boldsymbol{y}^*)$

B. Proofs

B.1. Proof of Theorem 4.1

Theorem 4.1. *`parSBO` has a competitive ratio of* $1 + \epsilon_{par} \max\{\mu^-, \mu^+\}$*, where* μ^- *and* μ^+ *are problem-specific constants defined as* $\mu^- = c_n^{\mathcal{N}}/c_1^{\mathcal{M}}$ *and* $\mu^+ = T c_m^{\mathcal{M}}/c_1^{\mathcal{N}}$*.*

Proof. There are two cases: (i) underestimating β (i.e. $\hat{\beta} < \beta^*$) results in more demand satisfied with AVG resources compared to the offline optimal `OPT`, while (ii) overestimating β (i.e. $\hat{\beta} > \beta^*$) results in more demand satisfied with MAX resources compared to `OPT`. In both cases, `parSBO` must incur cost C^* equal to `OPT` as well as additional cost for decisions that differ from `OPT`. First consider case (i): in the worst case, `parSBO` uses the most expensive AVG resource to satisfy constant demand rather than the cheapest MAX resource. The total extra cost incurred by `parSBO` is at most

$c_n^{\mathcal{N}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)$, while the cost of OPT is at least $c_1^{\mathcal{M}} \max_{t\in\mathcal{T}} d(t)$. When $\hat{\beta} < \beta^*$, the competitive ratio is:

$$1 + \frac{c_n^{\mathcal{N}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)}{C^*} \leq 1 + \frac{c_n^{\mathcal{N}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)}{c_1^{\mathcal{M}} \max_{t\in\mathcal{T}} d(t)} = 1 + \epsilon_{\text{par}}\mu^-$$

For case (ii), in the worst case parSBO uses the most expensive MAX resource to satisfy a spike of demand rather than the cheapest AVG resource. The total extra cost of this decision is at most $c_m^{\mathcal{M}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)$, while OPT uses the AVG resources for a cost that is at least $(c_1^{\mathcal{N}}/T) \max_{t\in\mathcal{T}} d(t)$. As a result, when $\hat{\beta} < \beta^*$ the competitive ratio is:

$$1 + \frac{c_m^{\mathcal{M}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)}{C^*} \leq 1 + \frac{c_m^{\mathcal{M}}\big(\epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big)}{\frac{c_1^{\mathcal{N}}}{T} \max_{t\in\mathcal{T}} d(t)} = 1 + \epsilon_{\text{par}}\mu^+$$

Depending on whether underestimating or overestimating β^* is more costly, the adversary chooses $\max\{\mu^-, \mu^+\}$. □

B.2. Proof of Theorem 4.2

Theorem 4.2. *inDBO is $\big(2 - (w - \epsilon_{in}\mu^-)/T\big)$-competitive.*

Proof. Let $C(\boldsymbol{\beta}_d) = C_{\text{AVG}}(\boldsymbol{x}(\boldsymbol{\beta}_d)) + C_{\text{MAX}}(\boldsymbol{y}(\boldsymbol{\beta}_d))$ be the cost incurred by inDBO over the time horizon. Since $\beta_d(t) \geq \beta_o(t)$ for all $t \in \mathcal{T}$, inDBO spends no more on AVG resources than DBO (ie. $C_{\text{AVG}}(x(\boldsymbol{\beta}_d)) \leq C_{\text{AVG}}(x(\boldsymbol{\beta}_o))$). This upper bound suggests that the adversary will attempt to increase $C_{\text{MAX}}(\boldsymbol{y}(\boldsymbol{\beta}_d))$ rather than $C_{\text{AVG}}(\boldsymbol{x}(\boldsymbol{\beta}_d))$.

Denote the extra amount that inDBO spends on the MAX resources compared to DBO (and thus OPT) as $\Delta(\boldsymbol{y}) = C_{\text{MAX}}(\boldsymbol{y}(\boldsymbol{\beta}_d)) - C_{\text{MAX}}(\boldsymbol{y}^*)$. Specifically, $\Delta(\boldsymbol{y})$ corresponds to demand that is satisfied by the AVG resources in the offline optimal solution. Prior to mistakenly spending more on MAX resources, inDBO will have incurred a cost of $(2 - w/T)C^*$. To upper bound $\Delta(\boldsymbol{y})$, note that ϵ_{in} is used by the adversary to increase the amount of anticipated demand so that inDBO will switch to using MAX resources. For inDBO to switch, the cost of serving the true demand as well as fake demand using the AVG resources should be at least as much as serving the true and fake demand using the MAX resources. To make the overspending of inDBO as large as possible, the adversary will not use the error ϵ_{in} to increase the anticipated cost of using the MAX resources. Instead it will use ϵ_{in} to make the anticipated cost of using the AVG resources as large as possible. At most, it can increase the anticipated cost by $\epsilon_{\text{in}} \max_{t\in\mathcal{T}} d(t) \frac{c_n^{\mathcal{N}}}{T}$. The resulting competitive ratio is:

$$\frac{C_{\text{AVG}}(\boldsymbol{x}(\boldsymbol{\beta}_d)) + C_{\text{MAX}}(\boldsymbol{y}(\boldsymbol{\beta}_d))}{C^*} \leq 2 - \frac{w}{T} + \frac{\Delta(\boldsymbol{y})}{C^*} \leq 2 - \frac{w}{T} + \frac{\frac{c_n^{\mathcal{N}}}{T}\epsilon_{\text{in}} \max_{t\in\mathcal{T}} d(t)}{c_1^{\mathcal{M}} \max_{t\in\mathcal{T}} d(t)} =$$
$$= 2 - \frac{w - \epsilon_{\text{in}}\mu^-}{T},$$

where μ^- is the same constant as in Theorem 4.1 that provides a instance-dependent upper bound on the worst-case resulting from underestimating the break-even point β^*. □

B.3. Proof of Theorem 4.5

Theorem 4.5. *RoBO-2 Pareto-dominates RoBO-1 with respect to consistency and robustness when the following conditions are met: (i) symmetry of worst-case instances ($\mu^- \approx \mu^+$) and (ii) sufficiently large look-ahead window size ($w \geq \mu^+$).*

Before proving Theorem 4.5, we first derive the competitive ratios of RoBO-1 (Lemma B.1) and RoBO-2 (Lemma B.2).

Lemma B.1. *RoBO-1 is $(1 + \lambda)$-consistent and $\big(1 + \lambda + \epsilon_{par}(\min\{\frac{1}{\lambda}, \mu^-\} - \lambda)\big)$-robust.*

Proof. For the consistency result, when $\hat{\beta} = \beta^*$ the discrepancy between RoBO-1 and OPT arises from spending on the AVG resources. Depending on the choice of λ, RoBO-1 will spend at most $\lambda C_{\text{MAX}}(\boldsymbol{y}^*)$ extra compared to OPT. Combining this with lemma A.1 yields a competitive ratio of $1 + \lambda$.

For the robustness results, when $\hat{\beta} \neq \beta^*$ there are two cases: (i) $\hat{\beta} < \beta^*$ and (ii) $\hat{\beta} > \beta^*$. A combination of DBO and parSBO will achieve its worst performance when $\hat{\beta} < \beta^*$, as this exacerbates the failure modality of DBO. When $\hat{\beta} < \beta^*$, the worst-case for RoBO-1 is that $C^* = C_{\text{MAX}}(\boldsymbol{y}^*)$. RoBO-1 will spend at most $C_{\text{MAX}}(\boldsymbol{y}^*)$ on the MAX resources, as

well as a non-zero amount on the AVG resources. Let ΔC denote the amount that OPT spends on satisfying the demand $\hat{\beta} < d(t) < \beta^*$ with the MAX resources. For demand up to $\hat{\beta}$, RoBO-1 will be incentivized to use the MAX resources and will spend $\lambda(C_{\text{MAX}}(\boldsymbol{y}^*) - \Delta C)$ on AVG resources before switching to MAX resources. However for demand $d(t) > \hat{\beta}$, RoBO-1 will avoid using MAX resources and will pay up to $\frac{1}{\lambda}\Delta C$ on AVG resources before switching. In the worst-case, OPT satisfies all demand using the cheapest MAX resource for a total cost of $c_1^{\mathcal{M}} \max_{t\in\mathcal{T}} d(t)$, while RoBO-1 incurs $\Delta C = c_1^{\mathcal{M}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)$ at a marked up rate of $1/\lambda$. The extra cost of serving ΔC is capped at $c_n^{\mathcal{N}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)$, so the final competitive ratio of RoBO-1 is:

$$\frac{C_{\text{MAX}}(\boldsymbol{y}^*) + \lambda\big(C_{\text{MAX}}(\boldsymbol{y}^*) - \Delta C\big) + \min\left\{\frac{\Delta C}{\lambda}, c_n^{\mathcal{N}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\right\}}{C^*} \leq$$

$$1 + \lambda + \frac{\lambda\big(c_1^{\mathcal{M}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\big) + \min\left\{\frac{c_1^{\mathcal{M}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)}{\lambda}, c_n^{\mathcal{N}} \epsilon_{\text{par}} \max_{t\in\mathcal{T}} d(t)\right\}}{c_1^{\mathcal{M}} \max_{t\in\mathcal{T}} d(t)} =$$

$$1 + \lambda + \lambda\epsilon_{\text{par}} + \min\left\{\frac{\epsilon_{\text{par}}}{\lambda}, \frac{\epsilon_{\text{par}} c_n^{\mathcal{N}}}{c_1^{\mathcal{M}}}\right\} = 1 + \lambda + \epsilon_{\text{par}}\Big(\min\left\{\frac{1}{\lambda}, \mu^-\right\} - \lambda\Big)$$

□

Lemma B.2. *RoBO-2 is* $\big(1+\lambda(1-w/T)\big)$*-consistent and* $\Big(1+\lambda\big(1-(w-\epsilon_{in}\mu^-)/T\big)+\epsilon_{par}\big(\min\{1/\lambda,\mu^+\}-\lambda\big)\Big)$*-robust.*

Proof. Typical consistency and robustness results in the literature assume that one prediction is either correct or incorrect. For our analysis with multiple predictions of different quantities, we define our consistency results as the competitive ratio when $\epsilon_{\text{par}} = \epsilon_{\text{in}} = 0$. The robustness results correspond to the competitive ratio when both $\epsilon_{\text{par}} \neq 0$ and $\epsilon_{\text{in}} \neq 0$, and the adversary is able to couple ϵ_{par} and ϵ_{in} in the worst possible way.

First we provide the consistency analysis. RoBO-2 will spend the same amount on MAX resources as the offline optimal OPT. By using the input predictions, RoBO-2 will spend at most $C_{\text{MAX}}(\boldsymbol{y}^*)(1 - w/T)$ on AVG resources before switching to using MAX resources. Since the parameter prediction also incentivizes the use of MAX resources by a factor of λ, by using both predictors, RoBO-2 spends at most $\lambda C_{\text{MAX}}(\boldsymbol{y}^*)(1 - w/T)$ on AVG resources and $C_{\text{MAX}}(\boldsymbol{y}^*)$ on the MAX resources. The corresponding competitive ratio is $1 + \lambda(1 - w/T)$.

We now provide the robustness analysis. Although the performance of both predictors is not inherently correlated, we consider this extreme setting to illustrate the absolute worst-case scenario that can arise from using two different predictors. Since the worst-case of inDBO arises from overspending on MAX resources, we only need consider the settings where the parameter predictor $\hat{\beta}$ overestimates the break-even point. There are two possibilities: either (i) $\beta_d(T) < \hat{\beta}$ or (ii) $\beta_d(T) > \hat{\beta}$. In the first, inDBO will never attempt to use the MAX resources more than parSBO, while in the second inDBO will generate fake predictions of demand to attempt to use the MAX resources even more than parSBO. Since the latter is worse, we address the setting where $\beta_d(T) > \hat{\beta}$. For demand $d(t) < \beta^*$, RoBO-2 only benefits from using predictions, with a decreased expenditure on AVG resources as seen in the consistency analysis. For demand in the range $(\beta^*, \hat{\beta})$, the use of MAX resources is still incentivized by a discount factor of λ. Here, the overspending parallels the results in Lemma B.1, with an additive contribution to the competitive ratio of at most $\epsilon_{\text{par}}(\min\{\frac{1}{\lambda}, \mu^+\} - \lambda)$. Finally for demand $d(t) > \hat{\beta}$, due to the parameter predictions, RoBO-2 will increase its competitive ratio at most by $\lambda\epsilon_{\text{in}}\mu^-/T$. Combining the terms gives the desired robustness result. □

To prove Theorem 4.5, fix the consistency and use the parametric equations of Lemmas B.1 and B.2.