Data-driven Competitive Algorithms for Online Knapsack and Set Cover

Ali Zeynali,[1] **Bo Sun,** [2] **Mohammad Hajiesmaili,** [1] **Adam Wierman** [3]
[1] University of Massachusetts Amherst, USA
[2] The Hong Kong University of Science and Technology, Hong Kong
[3] California Institute of Technology, USA
azeynali@cs.umass.edu, bsunaa@connect.ust.hk, hajiesmaili@cs.umass.edu, adamw@caltech.edu

Abstract

The design of online algorithms has tended to focus on algorithms with worst-case guarantees, e.g., bounds on the competitive ratio. However, it is well-known that such algorithms are often overly pessimistic, performing sub-optimally on non-worst-case inputs. In this paper, we develop an approach for data-driven design of online algorithms that maintain near-optimal worst-case guarantees while also performing learning in order to perform well for typical inputs. Our approach is to identify policy classes that admit global worst-case guarantees, and then perform learning using historical data within the policy classes. We demonstrate the approach in the context of two classical problems, online knapsack and online set cover, proving competitive bounds for rich policy classes in each case. Additionally, we illustrate the practical implications via a case study on electric vehicle charging.

1 Introduction

As the adoption of machine learning (ML) in infrastructure and safety-critical domains grows, it becomes crucially important to ensure ML-driven algorithms can provide guarantees on their performance. Competitive analysis of online algorithms has been a remarkably successful framework for developing simple algorithms with worst-case guarantees. The ultimate goal in this framework is to devise algorithms with the best possible *competitive ratio*, which is defined as the worst-case ratio between the cost of an online algorithm and that of the offline optimal. However, because the focus is on worst-case guarantees, the algorithms developed are typically conservative and do not learn in a data-driven manner. The result is that worst-case optimized algorithms tend to under perform for typical scenarios where worst-case inputs are uncommon. This phenomenon is wide-spread, but the importance of provable guarantees for safety and robustness means relaxing worst-case assumptions is undesirable in many settings. An important open question is how to achieve the "best-of-both-worlds", both near-optimal performance in typical settings, which requires data-driven adaptation, and a near-optimal competitive ratio.

Toward this goal, there have been substantial efforts to improve the performance of competitive algorithms using predictions (Chen et al. 2016, 2015; Antoniadis et al. 2020a), ML advice (Lykouris and Vassilvtiskii 2018; Angelopoulos et al. 2020; Purohit, Svitkina, and Kumar 2018; Rohatgi 2020; Lee, Hajiesmaili, and Li 2019; Antoniadis et al. 2020b; Banerjee 2020; Wang, Li, and Wang 2020), and advice from multiple experts (Gollapudi and Panigrahi 2019). In these approaches the goal is to allow online algorithms to use (potentially noisy) predictions (or advice) about future inputs. Such predictions capture the fact that, often, something is known about the future that could be used to improve the performance in typical cases. These approaches have been successfully applied to design competitive algorithms that perform near-optimally in the typical case in settings such as the ski-rental problem (Purohit, Svitkina, and Kumar 2018; Kodialam 2019; Angelopoulos et al. 2020; Bamas, Maggiori, and Svensson 2020; Wei and Zhang 2020), online optimization with switching costs (Chen et al. 2016), online caching (Lykouris and Vassilvtiskii 2018; Rohatgi 2020), and metrical task systems (Antoniadis et al. 2020a), to name a few. However, in this literature, the power of ML models has been leveraged to *first*, predict the future input, and *then* modify the algorithms to use this additional input to further improve the performance. In this way, the learning and algorithmic parts are decoupled, and practical improvements could be obtained only when fine-grained predictions of individual inputs are (nearly) perfect.

The idea of this paper is inspired by the fact that practitioners typically prefer to learn from the coarse-grained patterns observed in previous problem instances and then optimize over a class of algorithms that achieves high performance given the coarse-grained patterns. This approach has been of interest of empirical studies for a long time (Semke, Mahdavi, and Mathis 1998; Leyton-Brown, Nudelman, and Shoham 2009; Kotthoff, Gent, and Miguel 2012; Winstein and Balakrishnan 2013; Akhtar et al. 2018; De Cicco, Cilli, and Mascolo 2019). However, developing a theoretical understanding of this approach has received attention only recently, after a seminal work of Gupta and Roughgarden (Gupta and Roughgarden 2017, 2020) and its follow-ups (Kleinberg et al. 2019; Balcan, Dick, and Vitercik 2018; Alabi et al. 2019; Cohen-Addad and Kanade 2017; Roughgarden 2021).

Inspired by the above high-level idea, in this paper, we focus on developing *data-driven competitive algorithms* with worst-case guarantees. The prior literature on online algo-

rithms with ML advice combines online algorithms with learning in order to learn the uncertain input. This work, in contrast, designs policy classes of algorithms that ensure competitive guarantees for all algorithms in the policy class and then learns the best algorithm based on historical data directly. The result is an adaptive algorithm, tuned based on historic data, that ensures worst-case robustness. Further, it allows the richness of the policy class to be balanced with the performance in the typical case – if the policy class is broadened then the competitive ratio grows but there is a potential to learn a better algorithm for the common case.

Realizing the potential of this approach requires two steps. First, developing a policy class of online algorithms whose worst-case competitive ratios are bounded. This is in contrast to the typical style of analysis in online algorithms that seeks an individual algorithm with an optimal competitive ratio. The form of the policy class is crucial, since it should be broad enough to allow adaptation in application settings, but still provide near-optimal competitive guarantees for all policies in the class; thus balancing worst-case guarantees with performance in the typical case. Second, the approach requires developing ML tools that can learn and adapt from historical data to select a policy from the policy class. This second task is standard, and can be approached with a variety of tools depending on the setting (as discussed in Section 5).

More formally, our goal in this paper is to derive policy classes of online algorithms such that all policies in the class have bounded degradation in terms of worst-case competitive ratio. To that end, we introduce the *degradation factor* $\texttt{DF}(A(\theta))$ of an algorithm as the worst-case performance degradation of an algorithm $A(\theta)$, parameterized by parameter θ, with respect to a baseline algorithm:

$$\texttt{CR}(A(\theta)) \leq \texttt{DF}(A(\theta))\texttt{CR}(A(\theta_0)), \tag{1}$$

where $A(\theta_0)$ is the baseline algorithm that could be the one that achieves the optimal competitive ratio, the algorithm in literature with the best known competitive ratio, or simply the current algorithm used in practice. Then, to characterize the degradation factor of a policy class, we define a class of *ϕ-degraded* algorithms as one where every algorithm in the class has degradation factor that is no larger than ϕ, i.e., $\mathcal{P}(\phi) = \{\theta | \texttt{DF}(A(\theta)) \leq \phi\}$. Given such a class, in runtime, ML tools can be used to learn algorithm $A(\hat{\theta})$, where $\hat{\theta}$ is the learned parameter, to optimize the performance while being ensured of a worst-case loss of at most ϕ.

As compared to (Gupta and Roughgarden 2017), where the data-driven algorithm design is proposed as a general framework of improving an algorithm, this work, to the best of our knowledge, is the first that brings the idea of learning the best online algorithm among a policy class of algorithms that all have provable a worst-case competitive ratio guarantees. In addition, the degradation factor introduced in the paper is a novel performance metrics that can explicitly characterize the worst-case performance loss of online algorithms due to the data-driven algorithms selection.

Summary of contributions. To introduce and develop the above framework, we first consider, as a warm-up example, the classic ski-rental problem (Section 2). We provide a simple, concrete construction of a policy class of algorithms for this problem. Then, we characterize the degradation factor as well as ϕ-degraded policies within the policy class. This result enables a trade-off between optimizing the typical case while ensuring near-optimal worst-case performance.

After this warm-up, our main technical results focus on illustrating the approach in two important online problems: online knapsack (OKP) (Section 3) and online set cover (OSC) (Section 4). For both OKP and OSC, we develop policy classes, explicitly characterize the degradation factor achieved by all policies in the class. For OSC, using a synthetic input, we provide intuition about how the proposed framework can leverage the coarse-grained structure of inputs to learn the best policy. This is a new design space that could not be captured by online algorithms with ML advice that only utilize the fine-grained prediction of future inputs.

Deriving the above results requires addressing two sets of technical challenges: (i) determining how to *choose the right policy class* of algorithms, which requires a delicate balance between the worst-case guarantees and the learning design space for practical improvement, and (ii) determining how to *bound the competitive ratio of a class of algorithms in a parametric manner*, which is not always a straightforward extension of the analysis of the classical analysis. For example, in our analysis of the policy class for OKP, we have to identify and analyze two worst-case instances that differ from the classical analysis of OKP.

Then, in Section 5, inspired by ideas in (Balcan, Dick, and Vitercik 2018), we discuss how to cast a regret minimization online learning problem for the selection of the best algorithm given a policy class of online algorithms. This section shows how to efficiently learn the best policy from the class, and highlights our work as a complementary view in the emerging space of data-driven algorithms.

Finally, to demonstrate the potential for achieving both data-driven adaptation and worst-case guarantees in practical settings, we apply our approach to the application of online admission control of electric vehicles in a charging station, which is an extended version of OKP (Alinia, Hajiesmaili, and Crespi 2019; Sun et al. 2020). Our experiments consider two months of electric vehicle data-traces from a parking lot at Caltech (Lee, Li, and Low 2019), and show that the approach can improve the observed performance by 13.7%, with only 3% of instances having worse performance than the worst-case optimized online algorithm. The proofs are given in (Zeynali et al. 2020).

2 Warm-up: The Ski-Rental Problem

To illustrate our approach using a simple example, we start with the classic ski-rental problem (Karlin et al. 1986; Borodin and El-Yaniv 2005), in which a skier goes skiing for an unknown number of days. On each day, the skier can either rent skis at a unit price or buy one at a higher integer price of $p > 1$, and ski for free from then on. The uncertainty is the number of days the skier will ski. Our focus is on deterministic algorithms, for simplicity, and the best known deterministic algorithm uses a *break-even point* and rents the first $p - 1$ days before buying on the pth day. It is

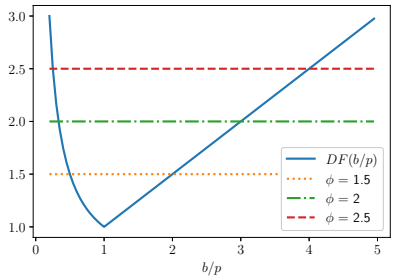

Figure 1: The lower bound of degradation factor as a function of normalized parameter b/p for the ski-rental problem

straightforward to see that this algorithm is 2-competitive, which is optimal. However, such a choice of the break-even point is overly conservative in typical situations.

Consider now a policy class of algorithms $A(b)$ with $b \in \mathcal{B} = \{1, 2, \dots\}$ defines the policy class. The parameter b is the number of renting days and can be optimized based on historical data in order to improve the typical performance, at the cost of an increased competitive ratio. The following theorem characterizes the degradation factor of $A(b)$ followed by a corollary characterizing the policy class of ϕ-degraded algorithms.

Theorem 1 *Let $A(b), b \in \mathcal{B} = \{1, 2, \dots\}$ be a policy class of algorithms. The degradation factor of algorithm $A(b)$, with respect to the baseline $A(p)$, the optimal 2-competitive algorithm, is* $\texttt{DF}(A(b)) = 1/2 + \max\{p/2b, b/2p\}$.

Note that with $b = p$, $\texttt{DF}(A(b)) = 1$, and $A(b)$ reduces to the algorithm that optimizes the competitive ratio. A proof follows quickly from standard results and is given in full version of the article in (Zeynali et al. 2020). Immediately from Theorem 1, the ϕ-degraded policy class is characterized as follows.

Corollary 2 *Let $\phi \geq 1$. The policy class of ϕ-degraded algorithms for the ski-rental problem is determined by $\mathcal{B}(\phi) = [p/(2\phi - 1), p(2\phi - 1)]$. That is, the degradation factor of any $A(b)$ with $b \in \mathcal{B}(\phi)$ is no larger than ϕ:* $\texttt{DF}(A(b)) \leq \phi, b \in \mathcal{B}(\phi)$.

To elaborate on typical design guidelines that follow the above analysis, in Figure 1, we plot the degradation factor as a function of the normalized parameter b/p. With $b/p > 1$, the competitive ratio degrades gracefully, while with $b/p < 1$ the competitive ratio degrades drastically. The figure also highlights a few ϕ-degraded policy classes. For example, the 2-degraded policy class, which leads to a degradation factor of at most 2, is $\mathcal{B}(2) = p \times [1/3, 3]$. This wide range shows that data-driven learning of a policy can be effective at optimizing the typical case. We discuss how to perform the data-driven learning in Section 5.

3 The Online Knapsack Problem

The goal of the online knapsack problem (OKP) is to pack items that are arriving online into a knapsack with unit capacity such that the aggregate value of admitted items is maximized. In each round, item $i \in [n] = \{1, \dots, n\}$, with value v_i and weight w_i, arrives, and an online algorithm must decide whether to admit or reject i with the objective of maximizing the total value of selected items while respecting the capacity. Given items' values and weights $\{v_i, w_i\}_{i\in[n]}$ in offline, OKP can be stated as

$$\begin{aligned} [\text{Offline OKP}] \quad \max \quad & \textstyle\sum_{i\in[n]} v_i x_i, \\ \text{s.t.}, \quad & \textstyle\sum_{i\in[n]} w_i x_i \leq 1, \\ \text{vars.}, \quad & x_i \in \{0, 1\}, \quad i \in [n], \end{aligned}$$

where the binary variable $x_i = 1$ denotes the admission of item i and $x_i = 0$ represents a decline. In an online setting, the admission decision x_i for item i must be made only based on causal information, i.e., the items' values and weights up to now $\{v_j, w_j\}_{j\in[i]}$ and previous decisions $\{x_j\}_{j\in[i-1]}$. Since there exists no online algorithm with a bounded competitive ratio for the general form of OKP (Zhou, Chakrabarty, and Lukose 2008), we focus on OKP under the following two standard assumptions, e.g., (Zhang, Li, and Wu 2017; Zhou, Chakrabarty, and Lukose 2008).

Assumption 1 *The weight of each individual item is much smaller than the unit capacity of the knapsack, i.e., $w_i \ll 1, \forall i \in [n]$.*

Assumption 2 *The value-to-weight ratio (or value density) of each item is lower and upper bounded between L and U, i.e., $L \leq v_i/w_i \leq U, \forall i \in [n]$.*

Assumption 1 naturally holds in large-scale systems, including the case study of this work in the context of electric vehicle charging. Assumption 2 is to eliminate the potential for rare items that have extremely high or low-value densities. This version of OKP has been used in numerous applications including online cloud resource allocation (Amarante et al. 2013; Zhang et al. 2017), budget constrained bidding in keyword auction (Zhou, Chakrabarty, and Lukose 2008), and online routing (Buchbinder, Naor et al. 2009). Also, as rigorously stated in appendix, OKP is closely related to the one-way trading problem (OTP) (El-Yaniv et al. 2001) in the sense that the optimal competitive ratios of OKP and OTP are the same and both can be achieved via threshold-based algorithms. Consequently, our results also hold for OTP.

Under Assumptions 1 and 2, and using $\gamma := U/L$ to represent the fluctuation of the value density, the optimal competitive ratio of OKP is $\ln(\gamma) + 1$, in the sense that no (deterministic or randomized) online algorithms can achieve a smaller competitive ratio (Zhou, Chakrabarty, and Lukose 2008). It has also been shown that a threshold-based algorithm can achieve this optimal competitive ratio (Zhou, Chakrabarty, and Lukose 2008; Zhang, Li, and Wu 2017). Let $\Psi(z) : [0, 1] \to [L, U] \cup \{+\infty\}$ denote a threshold function. The online algorithm for OKP admits an item i only if its value density is no less than the threshold value at current utilization level z_{i-1}, i.e., $v_i/w_i \geq \Psi(z_{i-1})$. With the threshold function designed in (Zhang, Li, and Wu 2017; Zhou, Chakrabarty, and Lukose 2008), this algorithm is proved to be $(\ln(\gamma) + 1)$-competitive.

3.1 A Competitive Policy Class for OKP

We consider a policy class of algorithms OKP-Alg(α), $\alpha \in \mathcal{A} := \{\alpha | \alpha > 0\}$. OKP-Alg($\alpha$) is a threshold-based

Algorithm 1 OKP−Alg(α): A parametric algorithm for OKP

Input: threshold function Ψ_α, initial utilization $z_0 = 0$;
while item i arrives **do**
  determine x_i^* by solving the problem (4) given the current utilization z_{i-1};
  update utilization $z_i = z_{i-1} + w_i x_i^*$
end while

algorithm whose threshold function Ψ_α is parameterized by α as follows

$$\Psi_\alpha(z) = \begin{cases} L & z \in [0, T), \\ \min\left\{U, Le^{\alpha(z/T-1)}\right\} & z \in [T, 1), \\ +\infty & z = 1, \end{cases} \quad (3)$$

where $T = 1/(\ln(\gamma) + 1)$ is a utilization threshold where all items will be admitted. Figure 2 depicts Ψ_α with respect to multiple parameter α. We can interpret $\Psi_\alpha(z)$ as the marginal cost of packing items into the knapsack when its utilization is z. This threshold function can then be used to estimate the cost of using a portion of the knapsack capacity and OKP−Alg(α) aims to balance the value from the item and cost of using the capacity. Particularly, upon the arrival of item i, OKP−Alg(α) makes the admission decision by solving a pseudo-utility maximization problem

$$x_i^* = \arg\max_{x_i \in \{0,1\}} v_i x_i - \Psi_\alpha(z_{i-1}) w_i x_i, \quad (4)$$

where $z_{i-1} = \sum_{j \in [i-1]} w_j x_j^*$, is the cumulative capacity utilization of the previous $i-1$ items. Given Assumption 1, $\Psi_\alpha(z_{i-1})w_i$ estimates the cost of packing item i. The online algorithm admits an item only if its value density is high enough such that a non-negative pseudo-utility can be obtained. The OKP−Alg(α) is summarized as Algorithm 1. Note that the parametric algorithm OKP−Alg(α) is a generalization of the classic threshold-based algorithm (Zhang, Li, and Wu 2017; Zhou, Chakrabarty, and Lukose 2008). When $\alpha = 1$, Ψ_α recovers the threshold function designed in (Zhang, Li, and Wu 2017) and OKP−Alg(1) can achieve the optimal competitive ratio.

To construct a competitive policy class for OKP, one could consider various ways of parametrizing the threshold Ψ_α. For instance, one could parameterize the minimum threshold T of the flat segment ($\alpha \in [0, T)$) and/or the increasing rate of the exponential segment ($\alpha \in [T, 1)$) to increase aggressiveness of OKP−Alg(α). Our results focus on the exponential rate, which provides a richness for tuning without incurring significant degradation in the competitive ratio. Our first result characterizes the degradation factor of OKP−Alg(α) with respect to the worst-case optimized algorithm OKP−Alg (1).

Theorem 3 *The degradation factor of* OKP−Alg(α), $\alpha \in \mathcal{A}$, *with respective to* OKP−Alg(1) *is*

$$\text{DF}(\text{OKP−Alg}(\alpha)) = \begin{cases} \frac{\alpha\gamma}{\alpha+\gamma-1} & \alpha \in [1, +\infty), \\ \frac{\alpha\gamma}{\alpha+\gamma^\alpha-1} & \alpha \in (0, 1). \end{cases} \quad (5)$$

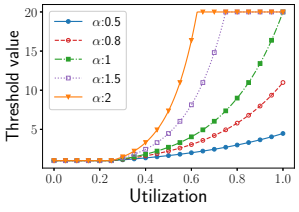

Figure 2: Threshold function Ψ_α in Eq. (3) with different values of α for OKP; $\gamma = 20$.

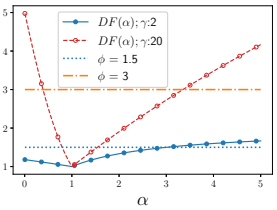

Figure 3: Degradation factor DF(OKP−Alg(α)) for different value of γ for OKP.

Figure 3 illustrates the degradation factor as the parameter α varies. In (Zeynali et al. 2020), we provide insights on the growth of the degradation factor and the rigorous proof of Theorem 3. The proof leverages the fact that, in the worst-case scenario, the arriving items can be divided into two batches. OKP−Alg(α) only admits the first batch of items while the offline optimal solution only admits the second batch. Depending on the parameter α, there exist two cases: when $\alpha \in [1, +\infty)$, the value densities of the first batch of items are exactly equal to the marginal cost of packing each item upon their arrivals and the total weight of the admitted items is z^u, at which the maximum value density is reached, i.e., $\Psi_\alpha(z^u) = U$. Then the second batch of items with value density $U - \epsilon$ ($\epsilon > 0$) arrives and their total weight is 1. The competitive ratio of this case can be derived as $(U-\epsilon)/\int_0^{z^u} \Psi_\alpha(s)ds$. When $\alpha \in (0, 1)$, since the maximum marginal cost of packing items is less than the maximum value density, i.e., $\Psi_\alpha(1) < U$, the first batch of items will occupy the whole capacity of the knapsack. Thus, in this case, the competitive ratio becomes $U/\int_0^1 \Psi_\alpha(s)ds$. Substituting Ψ_α gives the competitive ratios in the two cases. In order to adaptively tune α with worst-case performance guarantees, data-driven models can be proactively built to ensure a degradation factor no larger than ϕ by restricting α in a ϕ-degraded policy class. The following corollary characterizes the ϕ-degraded policy class of OKP−Alg(α).

Corollary 4 *Let $\phi \in [1, \gamma)$. The policy class of ϕ-degraded algorithms for* OKP *is*

$$\mathcal{A}(\phi) = \left[-\frac{\phi}{\gamma-\phi} - \frac{W_{\phi,\gamma}}{\ln(\gamma)}, \frac{\phi(\gamma-1)}{\gamma-\phi}\right], \quad (6)$$

where $W_{\phi,\gamma} := W\left(-\frac{\ln(\gamma)\phi}{\gamma-\phi} e^{-\ln(\gamma)\phi/(\gamma-\phi)}\right)$ and $W(\cdot)$ is the Lambert function.

The ϕ-degraded policy class models a tradeoff between the performance in typical settings and the robustness in worst-case scenarios. A larger ϕ corresponds to a weaker worst-case guarantee but provides a larger space for potentially learning an algorithm that can work better in common cases.

4 The Online Set Cover Problem

The classical (unweighted) version of online set-cover problem (OSC) is defined as follows. Let $\mathcal{I}$, with $n := |\mathcal{I}|$, be the ground set of elements, each indexed by i. Let $\mathcal{S}$, $m := |\mathcal{S}|$,

be a family of sets, such that each set $s \in \mathcal{S}$ includes a subset of elements in $\mathcal{I}$. In the online problem, a subset of elements $\mathcal{I}' \subseteq \mathcal{I}$ arrives element-by-element over time. Once element i arrives, the algorithm must cover i by selecting one (or more) sets containing i. The online algorithm knows $\mathcal{I}$ and $\mathcal{S}$ in advance, but not $\mathcal{I}'$. The ultimate goal is to pick the minimum possible sets from $\mathcal{S}$ in order to cover all elements in $\mathcal{I}'$. Given $\mathcal{I}'$, OSC can be formulated as follows.

$$\begin{aligned} \text{[Offline OSC]} \quad \min \quad & \textstyle\sum_{s\in\mathcal{S}} x_s, \\ \text{s.t.,} \quad & \textstyle\sum_{s|i\in s} x_s \geq 1,\ i \in \mathcal{I}', \\ \text{vars.,} \quad & x_s \in \{0,1\},\ s \in \mathcal{S}, \end{aligned}$$

where the binary variable $x_s = 1$ if set s is chosen, and $x_s = 0$, otherwise. The offline OSC problem is NP-hard and the online version is even more challenging due to the uncertainty of elements to be covered. The OSC problem was first introduced in (Alon, Awerbuch, and Azar 2003) and has been proved to be the core problem in numerous real-world applications such as online resource allocation (Pu et al. 2018; Wang et al. 2017), crowd-sourcing (Sheng, Tang, and Zhang 2012; Bagaria, Pananjady, and Vaze 2013), and scheduling (Pananjady, Bagaria, and Vaze 2015), etc. In the literature, there exists several online algorithms for OSC (Alon, Awerbuch, and Azar 2003; Buchbinder and Naor 2009). Our design is based on (Alon, Awerbuch, and Azar 2003) where the proposed algorithm is based on a specifically-designed potential function. Upon arrival of an element, the algorithm adds some subsets to cover the current element, and meanwhile ensures the potential function is non-increasing. This algorithm achieves a competitive ratio of $4\log n(2+\log m)$. We primarily focus on developing a class of algorithms for OSC by extending the algorithm in (Alon, Awerbuch, and Azar 2003), and characterize the degradation factor with respect to it. Also, we provide insights about how to learn an online algorithm from the class.

4.1 A Competitive Policy Class for OSC

In this section, we introduce OSC−Alg(θ), a parametric policy class for OSC that generalizes the existing algorithm, from (Alon, Awerbuch, and Azar 2003). The core of the policy class is a parameter θ that determines the rate at which the subsets should be covered. The algorithm works as follows. Let w_s represent the weight of set $s \in \mathcal{S}$. These weights evolve during the execution of OSC−Alg. Further, let $w_i = \sum_{s\in\mathcal{S}_i} w_s$ be the weight of element i, where $\mathcal{S}_i$ is the set of all subsets containing i, and define $\mathcal{I}^{\texttt{sel}}$ and $\mathcal{S}^{\texttt{sel}}$ as the running sets of covered elements and chosen subsets by the algorithm during its execution. The algorithm maintains a potential function Φ for the uncovered elements defined as $\Phi = \sum_{i\notin\mathcal{I}^{\texttt{sel}}} n^{2w_i}$.

The details on how the algorithm proceeds are summarized in Algorithm 2. Briefly, once a new element arrives, if the element is already covered, the algorithm does nothing. However, if the new element is uncovered, the algorithm first updates the weight of the set containing i according to Lines 6 and 7, and then selects at most $2\theta\log(n)$ subsets from $\mathcal{S}_i$ such that the potential function does not increase.

Algorithm 2 OSC−Alg(θ): A parametric algorithm for OSC

1: **initialization:** potential function Φ, $\theta > 1$, and $w_s = \frac{1}{\theta m}, s \in \mathcal{S}, \mathcal{I}^{\texttt{sel}} = \emptyset, \mathcal{S}^{\texttt{sel}} = \emptyset$
2: **while** element i arrives **do**
3: **if** $w_i \geq 1$, i.e., i is in $\mathcal{I}^{\texttt{sel}}$ **then**
4: do nothing
5: **else** do the weight-augmentation:
6: find the minimum k such that $\theta^k \times w_i > 1$
7: update $w_s \leftarrow \theta^k \times w_s, s \in \mathcal{S}_i$
8: select at most $2\theta\log(n)$ subsets from $\mathcal{S}_i$ such that value of potential function does not increase if we add these subsets to $\mathcal{S}^{\texttt{sel}}$. Add i to $\mathcal{I}^{\texttt{sel}}$
9: **end if**
10: **end while**

Note that, with $\theta = 2$, the algorithm degenerates to the existing algorithm in (Alon, Awerbuch, and Azar 2003), which covers at most $4\log n$ subsets in each round. For intuition behind potential functions and updating the weights, we refer to (Alon, Awerbuch, and Azar 2003). The following theorem characterizes the degradation factor of OSC−Alg(θ) using (Alon, Awerbuch, and Azar 2003) as the baseline.

Theorem 5 *The degradation factor of* OSC−Alg(θ) *with respect to* OSC−Alg(2) *is*

$$\texttt{DF}(\texttt{OSC−Alg}(\theta)) = \theta\left[\frac{2\log\theta+\log m}{2\log\theta(2+\log m)}\right]. \qquad (8)$$

As expected, this theorem recovers the result of (Alon, Awerbuch, and Azar 2003) for the case of $\theta = 2$, obtaining the competitive ratio of $\texttt{CR} = (4\log n)(\log m + 2)$ which is $O(\log n \log m)$. The proof (in (Zeynali et al. 2020)) is based on finding a feasible region for θ such that the number of iterations in the weight augmentation is upper bounded, and ensuring the feasibility of the operation in Line 7 of OSC−Alg. Also, note that with a slight change in the setting and assuming that the frequency of elements in subsets is bounded by d, OSC−Alg can be straightforwardly extended to the case with a competitive ratio of $(2\theta\log n)(2+\log d/\log\theta)$.

The next step is to characterize the ϕ-degraded policy class of algorithms, which involves calculations of the inverse of Eq. (8), and can be expressed by the r-Lambert function, i.e., the inverse of $f(x) = xe^x + rx$. However, for clarity, in Figure 4, we plot the degradation factor for two different values of m. Also, this figure shows 1.2 and 1.1-degraded policy classes. For example, it shows that with $m = 10^5$, by tuning $\theta = [1.67, 3.93]$, the competitive ratio will be degraded by at most 10%.

Last, to further highlight the practical difference of our proposed framework with the existing prediction-based online algorithms, we provide intuition on how the coarse-grained structure of input provides insights on finding the best policy. A key algorithmic nugget of OSC−Alg is on the number of subsets selected in Line 8. The higher the value of θ, the higher the probability of selecting more subsets in runtime. However, depending on the overlap of elements in subsets, a higher value θ might be good or bad. Specifically,

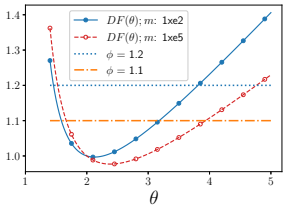

Figure 4: Degradation factor $\texttt{DF}(\texttt{OSC-Alg}(\theta))$ with different values of m for OSC

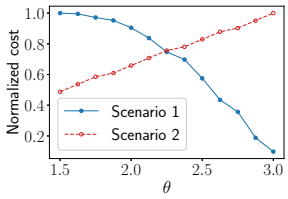

Figure 5: Normalized cost of $\texttt{OSC-Alg}(\theta)$ for two different problem scenarios

with higher θ, i.e., selecting more subsets in each round, is beneficial if the overlap of appearing elements in subsets is high since the algorithm covers some *useful* subsets in advance. We refer to this as "Scenario 1". As "Scenario 2", consider the case in which the overlap between elements in subsets is small. In this scenario, smaller θ might be better since in each round the algorithm will cover only subsets that are needed for the current element. To illustrate the impact of learning the right value of θ under Scenarios 1 and 2, in Figure 5, we report an experiment with 20 random instances of OSC with $n = 120$, $m = 3200$, and 80 elements appeared on the input. The result illustrates our intuition above, i.e., larger θ favors Scenario 1 and smaller θ favors Scenario 2. Our construction of policy class is able to capture these scenarios for learning the best policy. Instead, the existing prediction-based algorithms can only use fine-grained ML advice that predicts the upcoming elements in near future.

5 Data-driven Algorithm Selection as Online Learning

The previous sections have constructed competitive policy classes of online algorithms for ski-rental, OKP, and OSC. Given these classes, the next question is how to adaptively select the parametric algorithms using a data-driven approach. This task falls into the emerging framework of data-driven algorithm design, which has been introduced in (Gupta and Roughgarden 2017, 2020) and followed by (Balcan, Dick, and Vitercik 2018; Balcan, Dick, and Pegden 2020) by framing it as a principled online learning problem. In this section, we will discuss how the algorithm selection problem within our policy classes can be formulated as an online learning problem.

Consider the problem of adaptively selecting the parametric algorithm $A(\alpha)$ from a ϕ-degraded policy class $\mathcal{A}(\phi)$ in a total of M rounds, where each round t corresponds to an instance of the underlying online problem. At the beginning of round $t \in [M]$, we choose a parameter $\alpha_t \in \mathcal{A}(\phi)$ and run $A(\alpha_t)$ to execute the instance of this round $\mathcal{I}_t$. Let $R_t(\mathcal{I}_t, \alpha_t)$ denote the total reward in round t. Suppose the instances of all rounds are known from the start, the best fixed parameter α_c^{off} is given by $\alpha_c^{\text{off}} = \arg\max_{\alpha \in \mathcal{A}(\phi)} \sum_{t \in [M]} R_t(\mathcal{I}_t, \alpha)$. The problem of choosing the parameter is to design an online learning algorithm that can determine α_t in an online manner and minimize the regret of the adaptively selected parameter α_t with respective to the best fixed parameter α_c^{off}, i.e.,

$$\textsf{Regret}_M(\alpha_c^{\text{off}}) = \sum_{t \in [M]} R_t(\mathcal{I}_t, \alpha_c^{\text{off}}) - \sum_{t \in [M]} R_t(\mathcal{I}_t, \alpha_t).$$

To perform online learning for algorithm selection, we can apply results from prior literature. One approach is to convert the infinite set of parameters $\mathcal{A}(\phi)$ into a finite set $\tilde{\mathcal{A}}(\phi)$ using discretization, numerically evaluate the reward function, and then apply existing online learning algorithms to determine the parameter selection. Depending on the computational complexity of evaluating the reward function, we may choose Hedge algorithms (Freund and Schapire 1997) for full information feedback, i.e., known $R_t(\mathcal{I}_t, \alpha), \forall \alpha \in \tilde{\mathcal{A}}(\phi)$, or EXP3 algorithms (Auer et al. 2002) for bandit information feedback, i.e., when the reward $R_t(\mathcal{I}_t, \alpha_t)$ is known just for the selected α_t.

Another alternative is to obtain theoretical regret bounds on the online learning problems over the original infinite set $\mathcal{A}(\phi)$. The critical challenge in this setting for the regret analysis is to understand the properties of the per-round reward function. In particular, when the reward function is Lipschitz-continuous, sublinear regret algorithms can be shown in both the full information and bandit settings (Maillard and Munos 2010). For the one-way trading problem (OTP), which could be interpreted as a simplified version of the knapsack problem, one can show the reward function under our proposed parametric algorithms is Lipschitz-continuous (See (Zeynali et al. 2020) for more detail).

However, for OKP and OSC, the per-round reward functions are, in general, piecewise Lipschitz functions. It is known that online learning algorithms for general piecewise Lipschitz functions suffer linear regret bounds (Cohen-Addad and Kanade 2017), though recent work in (Balcan, Dick, and Vitercik 2018) shows that sublinear regrets can be achieved if the piecewise Lipschitz reward functions satisfy some additional *dispersion* conditions. The key step of verifying the dispersion condition is to characterize the discontinuity locations of the reward function. For several classic offline problems, (Balcan, Dick, and Vitercik 2018) provided such characterizations for the reward functions corresponding to their parametric offline algorithms. However, in the online learning problems of OKP and OSC, the reward functions are given by the optimal values of underlying online problems, and hence their discontinuities are challenging to characterize in general. However, proving sublinear regret bounds by verifying the dispersion condition in specific application settings is promising, since in the OTP setting, the reward function can even be Lipschitz-continuous.

6 Case Study

To illustrate the practical implications of being able to optimize within a policy class while still ensuring worst-case competitive bounds, we end the paper with a real-world case study on the admission control of electric vehicles (EVs) in a charging station, which is an extended version of OKP (Sun et al. 2020; Alinia et al. 2020). We consider a charging station with a charging demand more than its power capacity, which increasingly is the case. Upon the arrival of an EV, the

station has to either admit or reject the request based on the value and weight (amount of energy demand) of the charging request as well as the current utilization of the station. Clearly, this is similar to the OKP setting, with the difference being that the charging requests have flexibility within an available window and the stations may *schedule* the charging requests with this flexibility. The station during the time can be seen as the multiple knapsacks. Thus, the problem is a time-expanded version of single OKP. In (Zeynali et al. 2020), we show the the relation between EV charging problem and OKP more formally and characterize its degradation factor rigorously. Also, we explain how to achieve the optimal competitive ratio of OKP for this extended problem.

Experimental setup. To explore the performance of our framework for EV admission control, we use ACN-Data (Lee, Li, and Low 2019), an open EV dataset including over 50,000 fine-grained EV charging sessions. The data is collected from an EV parking lot at Caltech that includes more than 130 chargers. In this experiment, we use a two-month sequence of EV charging requests including arrival and departure times and charging demands. Since the data does not include the value of each request, we use a value estimation approach by modeling the distribution of historical arrivals and setting the values as a function of arrival, i.e., the higher the rate of arrival, the higher the value; details are provided in (Zeynali et al. 2020). Moreover, we used a *water-filling* scheduling policy to process the requests during the time. The water-filling policy splits the demand into the smaller parts then for each part, picks the slot with minimum utilization sequentially and updates the utilization by amount of smaller demand. In the experiments, each instance considers one day, and we randomly generated 100 instances for each day, each with different values, and report the results for $60 \times 100 = 6000$ instances.

We report the *empirical profit ratio*, profit of optimal offline algorithm over the profit of online algorithm in experiment, of different algorithms, which is the counterpart of the theoretical competitive ratio in the empirical setting. We compare the empirical profit ratio of three different algorithms: (1) $\texttt{OKP-Alg}(1)$, the worst-case optimized online algorithm that does not take into account the power of learning from historical data, but, guarantees the optimal competitive ratio; (2) $\texttt{OKP-Alg}(\alpha^{\texttt{off}})$, an algorithm that finds the best possible parameter in an offline manner. $\texttt{OKP-Alg}(\alpha^{\texttt{off}})$ is not practical since it is fed with the optimal parameter; however, it illustrates the largest possible improvement from learning; and (3) $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$, a simple, yet practical, algorithm that uses the optimal policy of the previous instance of the problem for the current instance.

Experimental results. Figure 6 plots the CDF of the empirical profit ratios of different algorithms. First, it shows that $\texttt{OKP-Alg}(\alpha^{\texttt{off}})$, e.g., the offline optimal policy, substantially improves the performance, i.e., the 80th percentile of $\texttt{OKP-Alg}(1)$ has the profit ratio of 1.87, while with $\texttt{OKP-Alg}(\alpha^{\texttt{off}})$, this is reduced to 1.46. Second, the performance of practical $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ is very close to that of $\texttt{OKP-Alg}(\alpha^{\texttt{off}})$. To scrutinize the microscopic behavior of $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$, in Figure 7, we plot the CDF of the improvement of the $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ as compared to

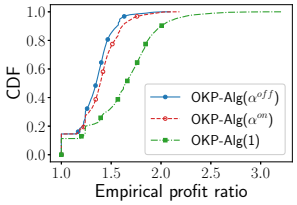

Figure 6: CDF of empirical profit ratios of different algorithms solving OKP

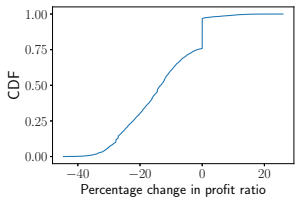

Figure 7: Improvement of $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ compared to $\texttt{OKP-Alg}(1)$

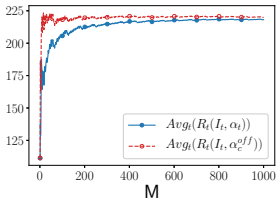

Figure 8: Average reward of the online learning algorithm and the static optimal

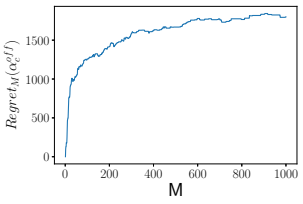

Figure 9: The regret of the online learning algorithm over different rounds

$\texttt{OKP-Alg}(1)$. Notable observations are as follows: (i) approximately in 76% of instances $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ outperforms $\texttt{OKP-Alg}(1)$, on average by 17.8%; (ii) in 21%, $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ has no benefits over $\texttt{OKP-Alg}(1)$; and finally, (iii) in 3% of cases, $\texttt{OKP-Alg}(\alpha^{\texttt{on}})$ does worse than $\texttt{OKP-Alg}(1)$, on average by 6.4%. A take-away from these experiments is that the average performance is substantially improved over the worst-case optimized algorithm, and that this improvement comes while only degrading the performance of a small fraction of instances by a small amount.

Algorithm selection via online learning. In this experiment, we use an online learning approach for selecting the best algorithm (as described in Section 5). Specifically, we use the adversarial Lipschitz learning algorithm in a full-information setting to implement the parameter selection (Maillard and Munos 2010). Figure 8 reports the average reward collected by the online learning approach and the best offline static algorithm over different rounds of running the problem. The average reward of an online algorithm converges to the optimal offline reward when as the learning process increases. The regret value's growth is presented in Figure 9. When the average reward of the online algorithm merges to the optimal offline value, the marginal rate of regret value decreases although the regret value increases.

7 Concluding Remarks

We developed an approach for characterizing policy classes of online algorithms with bounded competitive ratios, and introduce the *degradation factor*, a new performance metric that determines the worst-case performance loss of learning the best policy as compared to a worst-case optimized algorithm. We apply our approach to the ski-rental, knapsack, and set cover problems. These applications serve as illustrations of an integrated approach for *learning online algorithms*, while the majority of prior literature use ML for learning the uncertain input to online problems.

Acknowledgments

Ali Zeynali and Mohammad Hajiesmaili acknowledge the support from NSF grant CNS-1908298, and NSF CAREER 2045641. Adam Wierman's research is supported by NSF AitF-1637598, and CNS-1518941. Also, Bo Sun received the support from Hong Kong General Research Fund, GRF 16211220.

References

Akhtar, Z.; Nam, Y. S.; Govindan, R.; Rao, S.; Chen, J.; Katz-Bassett, E.; Ribeiro, B.; Zhan, J.; and Zhang, H. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proc. of the ACM SIGCOMM*, 44–58.

Alabi, D.; Kalai, A. T.; Liggett, K.; Musco, C.; Tzamos, C.; and Vitercik, E. 2019. Learning to Prune: Speeding up Repeated Computations. In *Conference on Learning Theory*, 30–33.

Alinia, B.; Hajiesmaili, M. H.; and Crespi, N. 2019. Online EV charging scheduling with on-arrival commitment. *IEEE Transactions on Intelligent Transportation Systems* 20(12): 4524–4537.

Alinia, B.; Hajiesmaili, M. H.; Lee, Z. J.; Crespi, N.; and Mallada, E. 2020. Online EV scheduling algorithms for adaptive charging networks with global peak constraints. *IEEE Transactions on Sustainable Computing* .

Alon, N.; Awerbuch, B.; and Azar, Y. 2003. The online set cover problem. In *Proc. of ACM STOC*, 100–105.

Amarante, S. R. M.; Roberto, F. M.; Cardoso, A. R.; and Celestino, J. 2013. Using the multiple knapsack problem to model the problem of virtual machine allocation in cloud computing. In *Proc. of IEEE CSE*.

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

Antoniadis, A.; Coester, C.; Elias, M.; Polak, A.; and Simon, B. 2020a. Online metric algorithms with untrusted predictions. *arXiv preprint arXiv:2003.02144* .

Antoniadis, A.; Gouleakis, T.; Kleer, P.; and Kolev, P. 2020b. Secretary and Online Matching Problems with Machine Learned Advice. *arXiv preprint arXiv:2006.01026* .

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32(1): 48–77.

Bagaria, V. K.; Pananjady, A.; and Vaze, R. 2013. Optimally approximating the lifetime of wireless sensor networks. *arXiv preprint arXiv:1307.5230* .

Balcan, M.-F.; Dick, T.; and Pegden, W. 2020. Semi-bandit Optimization in the Dispersed Setting. *The Conference on Uncertainty in Artificial Intelligence* .

Balcan, M.-F.; Dick, T.; and Vitercik, E. 2018. Dispersion for data-driven algorithm design, online learning, and private optimization. In *Proc. of IEEE FOCS*, 603–614.

Bamas, E.; Maggiori, A.; and Svensson, O. 2020. The Primal-Dual method for Learning Augmented Algorithms. *Advances in Neural Information Processing Systems* 33.

Banerjee, S. 2020. Improving Online Rent-or-Buy Algorithms with Sequential Decision Making and ML Predictions. *Advances in Neural Information Processing Systems* 33.

Borodin, A.; and El-Yaniv, R. 2005. *Online computation and competitive analysis*. Cambridge university press.

Buchbinder, N.; and Naor, J. 2009. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research* 34(2): 270–286.

Buchbinder, N.; Naor, J. S.; et al. 2009. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science* 3(2–3): 93–263.

Chen, N.; Agarwal, A.; Wierman, A.; Barman, S.; and Andrew, L. L. 2015. Online convex optimization using predictions. In *Proc. of ACM SIGMETRICS*, 191–204.

Chen, N.; Comden, J.; Liu, Z.; Gandhi, A.; and Wierman, A. 2016. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS* 44(1): 193–206.

Cohen-Addad, V.; and Kanade, V. 2017. Online optimization of smoothed piecewise constant functions. In *Artificial Intelligence and Statistics*, 412–420.

De Cicco, L.; Cilli, G.; and Mascolo, S. 2019. Erudite: a deep neural network for optimal tuning of adaptive video streaming controllers. In *Proc. of ACM MMSys*, 13–24.

El-Yaniv, R.; Fiat, A.; Karp, R. M.; and Turpin, G. 2001. Optimal search and one-way trading online algorithms. *Algorithmica* 30(1): 101–139.

Freund, Y.; and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55(1): 119–139.

Gollapudi, S.; and Panigrahi, D. 2019. Online algorithms for Rent-Or-Buy with expert advice. In *Proc. of ICML*, 2319–2327.

Gupta, R.; and Roughgarden, T. 2017. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing* 46(3): 992–1017.

Gupta, R.; and Roughgarden, T. 2020. Data-driven algorithm design. *Communications of the ACM* 63(6): 87–94.

Karlin, A. R.; Manasse, M. S.; Rudolph, L.; and Sleator, D. D. 1986. Competitive snoopy caching. In *Proc. of IEEE FOCS*, 244–254.

Kleinberg, R.; Leyton-Brown, K.; Lucier, B.; and Graham, D. 2019. Procrastinating with Confidence: Near-Optimal, Anytime, Adaptive Algorithm Configuration. In *Proc. of NeurIPS*, 8881–8891.

Kodialam, R. 2019. Optimal algorithms for Ski Rental with soft machine-learned predictions. *arXiv preprint arXiv:1903.00092* .

Kotthoff, L.; Gent, I. P.; and Miguel, I. 2012. An evaluation of machine learning in algorithm selection for search problems. *AI Communications* 25(3): 257–270.

Lee, R.; Hajiesmaili, M. H.; and Li, J. 2019. Learning-assisted competitive algorithms for peak-aware energy scheduling. *arXiv preprint arXiv:1911.07972* .

Lee, Z.; Li, T.; and Low, S. H. 2019. ACN-Data charging dataset: analysis and applications. In *Proc. of ACM eEnergy*.

Leyton-Brown, K.; Nudelman, E.; and Shoham, Y. 2009. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)* 56(4): 1–52.

Lykouris, T.; and Vassilvtiskii, S. 2018. Competitive Caching with Machine Learned Advice. In *Proc. of ICML*, 3302–3311.

Maillard, O.-A.; and Munos, R. 2010. Online learning in adversarial lipschitz environments. In *Joint european conference on machine learning and knowledge discovery in databases*, 305–320. Springer.

Pananjady, A.; Bagaria, V. K.; and Vaze, R. 2015. The online disjoint set cover problem and its applications. In *Proc. of IEEE INFOCOM*, 1221–1229.

Pu, L.; Jiao, L.; Chen, X.; Wang, L.; Xie, Q.; and Xu, J. 2018. Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks. *IEEE Journal on Selected Areas in Communications* 36(8): 1751–1767.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving online algorithms via ML predictions. In *Proc. of NeurIPS*, 9661–9670.

Rohatgi, D. 2020. Near-Optimal bounds for online caching with machine learned advice. In *Proc. of IEEE SOCA*, 1834–1845.

Roughgarden, T. 2021. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press. doi:10.1017/9781108637435.

Semke, J.; Mahdavi, J.; and Mathis, M. 1998. Automatic TCP buffer tuning. In *Proc. of ACM SIGCOMM*, 315–323.

Sheng, X.; Tang, J.; and Zhang, W. 2012. Energy-efficient collaborative sensing with mobile phones. In *Proc. of IEEE INFOCOM*, 1916–1924.

Sun, B.; Zeynali, A.; Li, T.; Hajiesmaili, M.; Wierman, A.; and Tsang, D. H. 2020. Competitive Algorithms for the Online Multiple Knapsack Problem with Application to Electric Vehicle Charging. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4(3): 1–32.

Wang, L.; Jiao, L.; Li, J.; and Mühlhäuser, M. 2017. Online resource allocation for arbitrary user mobility in distributed edge clouds. In *Proc. of IEEE ICDCS*, 1281–1290.

Wang, S.; Li, J.; and Wang, S. 2020. Online Algorithms for Multi-shop Ski Rental with Machine Learned Advice. *Advances in Neural Information Processing Systems* 33.

Wei, A.; and Zhang, F. 2020. Optimal Robustness-Consistency Trade-offs for Learning-Augmented Online Algorithms. *Advances in Neural Information Processing Systems* 33.

Winstein, K.; and Balakrishnan, H. 2013. TCP ex machina: Computer-generated congestion control. *ACM SIGCOMM Computer Communication Review* 43(4): 123–134.

Zeynali, A.; Sun, B.; ; Hajiesmaili, M.; and Wierman, A. 2020. Data-driven Competitive Algorithms for Online Knapsack and Set Cover. *arXiv preprint arXiv:2012.05361* .

Zhang, X.; Huang, Z.; Wu, C.; Li, Z.; and Lau, F. 2017. Online auctions in IaaS clouds: Welfare and pProfit maximization with server costs. *IEEE/ACM Trans. on Networking* 25(2): 1034–1047.

Zhang, Z.; Li, Z.; and Wu, C. 2017. Optimal posted prices for online cloud resource allocation. *Proc. of ACM SIGMETRICS* 1(1): 23:1–23:26.

Zhou, Y.; Chakrabarty, D.; and Lukose, R. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proc. of WINE*, 566–576.