The Online Knapsack Problem with Departures

BO SUN, The Chinese University of Hong Kong, Hong Kong SAR
LIN YANG*, Nanjing University, China
MOHAMMAD HAJIESMAILI, University of Massachusetts Amherst, USA
ADAM WIERMAN, California Institute of Technology, USA
JOHN C.S. LUI, The Chinese University of Hong Kong, Hong Kong SAR
DON TOWSLEY, University of Massachusetts Amherst, USA
DANNY H.K. TSANG, The Hong Kong University of Science and Technology (Guangzhou), China and The Hong Kong University of Science and Technology, Hong Kong SAR

The online knapsack problem is a classic online resource allocation problem in networking and operations research. Its basic version studies how to pack online arriving items of different sizes and values into a capacity-limited knapsack. In this paper, we study a general version that includes *item departures*, while also considering *multiple knapsacks* and *multi-dimensional item sizes*. We design a threshold-based online algorithm and prove that the algorithm can achieve order-optimal competitive ratios. Beyond worst-case performance guarantees, we also aim to achieve near-optimal average performance under typical instances. Towards this goal, we propose a data-driven online algorithm that learns within a policy-class that guarantees a worst-case performance bound. In trace-driven experiments, we show that our data-driven algorithm outperforms other benchmark algorithms in an application of online knapsack to job scheduling for cloud computing.

CCS Concepts: • **Theory of computation** → **Online algorithms**; • **Applied computing** → **Decision analysis**; • **Networks** → *Network economics.*

Additional Key Words and Phrases: online knapsack problems; knapsack with departures; data-driven algorithms; competitive ratio; cloud job scheduling

ACM Reference Format:
Bo Sun, Lin Yang, Mohammad Hajiesmaili, Adam Wierman, John C.S. Lui, Don Towsley, and Danny H.K. Tsang. 2022. The Online Knapsack Problem with Departures. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 3, Article 57 (December 2022), 32 pages. https://doi.org/10.1145/3570618

1 INTRODUCTION

The online knapsack problem (OKP) is a classic online algorithms problem that studies how to pack arriving items of different sizes and values to capacity-limited knapsacks. It models an online decision-making process where one provider allocates resources (i.e., knapsack capacity) to sequentially arriving customers (i.e., items) to maximize the total return. OKP has been widely

*Corresponding author for this work; the first two authors contribute equally to this work.

Authors' addresses: Bo Sun, bsun@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong SAR; Lin Yang, linyang@nju.edu.cn, Nanjing University, China; Mohammad Hajiesmaili, hajiesmaili@cs.umass.edu, University of Massachusetts Amherst, USA; Adam Wierman, adamw@caltech.edu, California Institute of Technology, USA; John C.S. Lui, cslui@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong SAR; Don Towsley, towsley@cs.umass.edu, University of Massachusetts Amherst, USA; Danny H.K. Tsang, eetsang@ust.hk, The Hong Kong University of Science and Technology (Guangzhou), China and The Hong Kong University of Science and Technology, Hong Kong SAR.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2476-1249/2022/12-ART57 $15.00
https://doi.org/10.1145/3570618

used in networking and operations research applications, such as online job scheduling in cloud computing [32], online routing of virtual circuits [6], admission control in 5G mobile network slicing [25], online electric vehicle charging in smart grids [27], online hotel booking in revenue management [16], online bidding in repeated auctions [5], and beyond.

In the most basic version of OKP, there is only one knapsack, and each item is characterized by its value and one-dimensional (scalar) size. The problem is to irrevocably decide whether to admit each item upon its arrival with the goal of maximizing the total values of admitted items while respecting the capacity of the knapsack. The sequence of items can only be revealed one-by-one and may even be adversarial. From this basic version, a wide array of generalizations have been considered. Three important extensions are: (i) the *online multiple knapsack* (OMK) problem, where there exist multiple knapsacks and the decision becomes whether to admit each item, and which knapsack the item should be assigned to if admitted; (ii) the *online multi-dimensional knapsack* (OMdK) problem, where each item has a multi-dimensional (vector) size and the decision is whether to admit each item while simultaneously respecting multi-dimensional capacity limits; (iii) the *online knapsack with departures* (OKD) problem, where arriving items depart after finite time duration.

The most general form of OKP includes all three of these extensions: online multiple, multi-dimensional knapsacks with departures. It is this version that is most applicable to the applications listed above. For example, in the case of online cloud job scheduling, jobs have multi-dimensional requirements (e.g., computing, memory), there are multiple knapsacks, i.e., VM servers, and jobs depart after receiving the desired amount of service. Similarly, the application of online hotel booking also requires all three extensions. Different hotels correspond to multiple knapsacks, each with multiple types of rooms (e.g., single/double room). Then a new order requests to book a certain numbers of different room types (e.g., one single room and one double room for three people) and stay for a given duration. However, there currently do not exist algorithms with provable guarantees for this general setting. Providing the first such algorithm is the goal of this paper.

More specifically, OKP has been extensively studied under the framework of competitive analysis [27–29, 32, 33] with the goal of designing online algorithms that can achieve the minimal competitive ratio, which is the worst-case ratio of values obtained by the offline algorithm in hindsight and the online algorithm. It is well-known that even the most basic OKP has unbounded competitive ratios [23]. Thus, this line of research aims to achieve competitive ratios that depend on setup information, such as the numbers of knapsacks and dimensions, knapsack capacities, etc. Optimal online algorithms have been designed for the classic OKP [32, 33] and OMK [27, 33] settings, both achieving a competitive ratio of $\Theta(\ln\theta)$, where θ is the fluctuation of the item value density (i.e., the maximum value-to-size ratio). More recently, OMdK was shown to have a competitive ratio of $O(M\ln\theta)$ that increases linearly in the the number of knapsack dimension M [32]. This result has been then improved to $O(\ln(M\theta))$, which matches its lower bound $\Omega(\ln(M\theta))$ and thus is order-optimal [29].

While optimal algorithms (in terms of competitive ratios) exist for settings with multiple knapsacks and multi-dimensional items, handling extensions with departures has proven more difficult. A recent result from [32] extends OMdK to allow item departures. In particular, it treats each time slot in a T-slot horizon as one dimension of the knapsack and designs an online algorithm that achieves a competitive ratio of $O(T\ln\theta)$, linearly increasing in the time horizon T. Based on this time-expanded OMdK model, the competitiveness result can be further improved by the theoretical improvement of OMdK in [29] to $O(\ln(T\theta))$; however, the dependence on T remains and limits practical use. Further, both results focus on a single knapsack.

One may ask if it is possible to eliminate the dependence on T in the case of departures. Results from a related area suggest that this may be possible. Specifically, in another classic problem, online interval scheduling [16, 22], which can be considered as a special case of OKD with item durations

Table 1. State-of-the-art results for OKP and its extensions. All papers (except [29]) assume either the sizes of items are much smaller than the capacity of knapsacks or the item is fractionally admissible. Capacities of all dimensions are identical. We use $\Theta(\cdot)$ to denote a matching upper $O(\cdot)$ and lower bound $\Omega(\cdot)$, which indicates order optimality. Parameters θ and α are the fluctuation ratios of value density and duration, respectively.

paper	knapsack	dimension	duration	value density	competitive ratio
[27, 28, 32, 33]	1	1	$+\infty$	$[1, \theta]$	$\Theta(\ln \theta)$
[27, 33]	K	1	$+\infty$	$[1, \theta]$	$\Theta(\ln \theta)$
[29]	1	M	$+\infty$	$[1, \theta]$	$\Theta(\ln(M\theta))$
[32]	1	M	$[1, \alpha]$	$[1, \theta]$	$O(M\alpha \ln \theta)$
[16]	1	1	$[1, \alpha]$	1	$\Theta(\ln \alpha)$
Theorem 1	K	1	$[1, \alpha]$	$[1, \theta]$	$\Theta(\ln(\alpha\theta))$
Theorem 4	K	M	$[1, \alpha]$	$[1, \theta]$	$\Theta(\ln(M\alpha\theta))$

bounded within $[\underline{D}, \overline{D}]$ but with fixed value density and item size, the optimal competitive ratio has been shown to be $\Theta(\ln \alpha)$, where $\alpha = \overline{D}/\underline{D}$ is the duration fluctuation ratio. Thus, one may conjecture whether the optimal competitive ratio of OKD in its most general form is $O(\ln(M\alpha\theta))$. In this paper, the core open question we answer is:

Is there an algorithm for OKD *that can achieve a competitive ratio of* $O(\ln(M\alpha\theta))$*?*

We first provide two impossibility results that show the competitive ratios of direct extensions of prior algorithms [29, 31, 33] are lower bounded by $\Omega(\alpha)$ or $\Omega(\ln(\overline{D}))$, which is either linear in the duration ratio α or logarithmic in the maximum duration $\overline{D}$ (See Lemma 1). This motivates us to develop new algorithms and analysis tools to attain the target competitive ratio in this work.

1.1 Contributions

Our main result shows that the answer to the above question is "yes." As Table 1 shows, we provide the first algorithm with a competitive guarantee for the case of multiple knapsacks with multi-dimensional items and item departures. Further, we achieve an order-optimal competitive ratio. This result opens the door for a wide array of applications, like online job scheduling, which require the full generality of multiple knapsacks with multi-dimensional items and item departures.

Our algorithm extends a classic approach in the OKP literature to OKD settings, which decomposes the algorithm into subroutines that check the admissibility of each item into each single knapsack, and adopts a threshold-based algorithm to decide the admissibility. The algorithmic challenge lies in the design of threshold functions, and we formalize the challenge involved in this via an impossibility result that shows that two state-of-the-art designs fail to achieve the target ratio $O(\ln(M\alpha\theta))$ under one of two types of hard instances (see Lemma 1). To overcome this challenge, we design a threshold function that balances the worst-case ratios over the two types of instances that lead to difficulties for current state-of-the-art approaches. This results in an order-optimal competitive ratio. Further, our design provides a class of parameterized threshold functions and characterizes the regimes of the parameters such that all thresholds with properly selected parameters can achieve the target competitive ratio (see Theorems 1 and 4).

Underlying our competitive analysis is a novel instance partitioning procedure. Because analyzing the competitive ratio of OKD directly over a T-slot horizon has proven difficult, we take advantage of the weak dependence of items across the horizon, leading to a partitioning of one instance into sub-instances with a shorter interval. To be more precise, as each item in OKD stays in the knapsack for at most $\overline{D}$ consecutive slots, the items that start at time t are only correlated with items in slots $\{t - \overline{D} + 1, \ldots, t + \overline{D} - 1\}$. Leveraging this structure, we use a novel partitioning technique that reduces the competitive analysis over the original instance to sub-instances, each of which is of length $3\overline{D}$. This partitioning greatly simplifies the worst-case analysis under two different types of hard instances, making it possible to design a threshold function that balances the worst-case performances from the two cases. The partitioning procedure together with the newly-designed threshold function is essential to improve the dependence of competitive ratios on item duration, from $\Omega(\alpha)$ or $\Omega(\ln \overline{D})$ of prior designs to the optimal order $\Theta(\ln \alpha)$.

A common critique of competitive analysis is that it leads to algorithms optimized for the worst-case instances. As a result, such algorithms can under-perform for typical instances from real-world applications (see comparison in Figures 4 and 5 in §6). Motivated by this, we go beyond worst-case analysis and use our analysis to derive a data-driven online algorithm that learns to optimize its average performance based on past data while maintaining a worst-case competitive guarantee. Theorem 3 bounds the competitive ratio for the data-driven algorithm and then we empirically demonstrate its performance in §6 using an application to job scheduling in cloud computing. The results highlight that the data-driven online algorithm provides significant improvement in practice, while still maintaining bounded worst-case performance.

The remainder of the paper is structured as follows. We begin by studying the online multiple *one-dimensional* knapsack problem with departures in §2-4. We first introduce the model and its application scenarios in §2. Then, we show our algorithms and main results in §3. Proofs of the main results are deferred to §4. Next we show extensions to the multi-dimensional case in §5. In §6, we present numerical experiments using real traces from cloud job scheduling. Finally, we review the related literature in §7 and draw conclusions in §8.

2 ONLINE MULTIPLE KNAPSACKS WITH DEPARTURES

2.1 Problem statement

Consider K knapsacks in a slotted time horizon $[T] = \{1, \ldots, T\}$, where each knapsack $k \in [K]$ has capacity $C_k \in \mathbb{R}^+$. A total of N items arrive sequentially and each item n is characterized by its item information $\mathcal{I}_n = \{a_n, \{w_{nk}, v_{nk}, \mathcal{T}_{nk}\}_{k\in[K]}\}$, where a_n is the arrival time, and for each knapsack k, w_{nk} and v_{nk} are the size and value, and $\mathcal{T}_{nk} := \{s_{nk}, \ldots, s_{nk} + d_{nk} - 1\}$ is the set of time slots that item n requests to stay in knapsack k from its starting time s_{nk} to its departure time $s_{nk} + d_{nk} - 1$. The set $\mathcal{T}_{nk}$ contains d_{nk} consecutive time slots and we call d_{nk} the duration of the item.

Upon arrival of item n, a decision maker observes its item information $\mathcal{I}_n$ and determines (i) whether to admit this item, and (ii) which knapsack this item should be assigned to if it is admitted. Let $\boldsymbol{x}_n = \{x_{nk}\}_{k\in[K]}$ denote the decision variable, where $x_{nk} \in \{0, 1\}$ indicates whether to admit item n to knapsack k and $\sum_{k\in[K]} x_{nk} = 0$ represents declining the item. The goal is then to design an online algorithm to causally determine $\boldsymbol{x}_n$ based on the item information up to n, i.e., $\{\mathcal{I}_{n'}\}_{n'\le n}$, that maximizes the total value of all admitted items while ensuring the capacities of all knapsacks not to be violated over the time horizon.

Let $\mathcal{I} := \{\mathcal{I}_n\}_{n\in[N]}$ denote an instance of OKD. Given $\mathcal{I}$, the offline problem can be formulated as

$$\text{(offline OKD)} \quad \max_{x_{nk}} \quad \sum\nolimits_{n\in[N]} \sum\nolimits_{k\in[K]} v_{nk} x_{nk}, \tag{1a}$$

$$\text{s.t.} \quad \sum\nolimits_{n\in[N]: t\in\mathcal{T}_{nk}} w_{nk} x_{nk} \le C_k, \forall k \in [K], t \in [T], \tag{1b}$$

$$\sum\nolimits_{k\in[K]} x_{nk} \le 1, \forall n \in [N], \tag{1c}$$

$$x_{nk} \in \{0, 1\}, \forall n \in [N], k \in [K]. \tag{1d}$$

Let $\mathtt{OPT}(\mathcal{I})$ and $\mathtt{ALG}(\mathcal{I})$ denote the values obtained by the offline problem (1) and an online algorithm under the instance $\mathcal{I}$. The competitive ratio of the online algorithm is defined as the worst-case performance ratio of the offline and online algorithms, i.e., $\mathtt{CR} = \max_{\mathcal{I}} \mathtt{OPT}(\mathcal{I})/\mathtt{ALG}(\mathcal{I})$. Our goal is to design an online algorithm that can achieve the smallest competitive ratio.

2.2 Application scenarios

In the following, we highlight a few sample application scenarios that could be captured by OKD.

Online job scheduling in cloud computing. A cloud provider allocates cloud resources (e.g., computing, memory) from a pool of K VM servers to N cloud jobs that arrive sequentially over a time horizon T. Upon the arrival of a job $n \in [N]$, it submits its request information that includes its resource requirement w_{nk}, processing time $\mathcal{T}_{nk}$, and the corresponding value (willingness-to-pay) v_{nk} for each server k. Each job can be processed in any server $k \in [K]$ but it may have different requests and values across different servers that are located in different regions, configured in different resource bundles (e.g., a bundle consisting of 2vCPUs and 8GB memory), and run at different operating costs. Upon receiving each job, the cloud provider then decides whether to admit this job and if admitted, which server the job should be assigned to.

As pointed out by [28, 32], OKD also captures the model of *dynamic pricing for cloud resource allocation.* In this problem, each job n has its own *private* value v_{nk} for server k, which will not be submitted together with its request. The provider's decision is to post a set of prices for available servers. Then the job itself decides to take which price to join, or leave the platform. The online job scheduling and dynamic pricing converge to the same problem when we focus on threshold-based algorithms, where the threshold values (See equations (2) and (17)) are used to determine the scheduling or set as the posted prices in the two applications, respectively.

Online reservation problem. Motivated by the emerging online shopping and sharing-economy platforms such as Expedia (for hotel booking), Turo (for car rental), OpenTable (for restaurant reservation), etc., the platform service provider often faces an online reservation problem [16], which can be modeled by the OKD. Take the online hotel booking as an example, booking orders arrive sequentially, and for each potential hotel k (after being filtered based on prices and locations), each order n specifies how many rooms and how many people in each room (modeled by w_{nk}), the check-in and check-out dates (modeled by $\mathcal{T}_{nk}$) and the value v_{nk}. The provider then immediately decides to accept or decline each order, and which hotel the order should be allocated to if the order is accepted. Using a similar argument as in the previous example, the setting can be adapted to include dynamic pricing to cope with the private values of the orders.

Other applications. OKD has similarities with several other online problems in the literature. For example, the offline formulation of OKD can also be used to model the generalized assignment problem [20] where bins correspond to knapsacks. OKD is a special case of the online generalized assignment problem by putting additional assignment restrictions specified in Assumptions 1-3. Similarly, the problem of electric vehicle charging scheduling with fixed charging rate [15], the online traffic routing problem [19], the online appointment booking in healthcare [26] can all be considered as special cases of OKD.

2.3 Additional Notations and Assumptions

Even for the most basic version of OKP, it is impossible to design competitive algorithms without making additional assumptions [23]. Here, we present three standard technical assumptions that capture the key features of the above motivating applications and allow the derivation of competitive bounds for the proposed algorithms.

First, define the value density of item n in knapsack k as the item value per unit size per unit time, i.e., $v_{nk}/(w_{nk}d_{nk})$. To distinguish low-valued and high-valued items, we assume the value density varies within a bounded set. This assumption is consistent with those in the literature [29, 32, 33].

Assumption 1 (Value density fluctuation). *The value density of each item n in knapsack k is bounded, i.e., $v_{nk}/(w_{nk}d_{nk}) \in [1, \theta_k], \forall n \in [N]$ and value density (fluctuation) ratio is defined as θ_k.*

Next, since each item only stays in the knapsack for a finite duration that is much smaller than the time horizon, we assume that the duration d_{nk} of the item n in knapsack k is bounded.

Assumption 2 (Duration fluctuation). *The duration of each item n in knapsack k is bounded, i.e., $d_{nk} \in [\underline{D}_k, \overline{D}_k], \forall n \in [N]$ and duration (fluctuation) ratio is defined as $\alpha_k = \overline{D}_k/\underline{D}_k$.*

The duration ratio α_k is a dimensionless quantity that can model the variation of item duration. This assumption has been commonly used in the classic online interval scheduling problem [16, 22]. Under this assumption, OKD can be considered a generalized version of the online interval scheduling problem with varying value density and item sizes.

Last, we assume the size of each item is upper bounded and smaller than the capacities of the knapsacks. This eliminates the irrelevant items that are inadmissible to knapsacks.

Assumption 3 (Upper bound of item size). *The size of each item n is upper bounded, i.e., $w_{nk} \le \varepsilon_k \le C_k, \forall k \in [K], n \in [N]$.*

Finally, we want to emphasize that all three assumptions are consistent with the motivating applications we have discussed. For example, in the online job scheduling, the value of each job n is proportional to the required resources, e.g., computing, memory (modeled by w_{nk}), and its running duration (modeled by d_{nk}) in a server k. Each job only occupies the resources for a finite duration d_{nk} and then the required resources can be released for future jobs. In addition, the resources required by each job are smaller than the capacity provided by a server.

3 ALGORITHMS AND MAIN RESULTS

3.1 Algorithms

Our main results consist of two novel algorithms. First, in §3.1.1, we propose an online algorithm that achieves the order-optimal competitive ratio for OKD. Then, in §3.1.2, we extend this algorithm to design a data-driven algorithm that can learn to optimize the average-case performance while ensuring a competitive bound.

3.1.1 A worst-case optimized algorithm for OKD. We propose a simple yet effective online algorithm (OA-OKD) to solve OKD in Algorithm 1. The algorithm works as follows. Upon the arrival of item n, it first determines the admissibility of the item to each knapsack (lines 5-7) by calling an online threshold-based algorithm (OTA) subroutine in Algorithm 2, which takes a carefully-designed threshold function ϕ and real-time knapsack utilization over concerned time slots as inputs. If item n is admissible to at least one knapsack, OA-OKD then admits item n and assigns it to the knapsack k' that provides the maximum value among all admissible knapsacks. Otherwise, item n is rejected (lines 8-13). Finally, the knapsack utilization is updated (line 14), and is used to determine the admissibility for the next item. The key step is the admission control of items to each knapsack via OTA in Algorithm 2. To check admissibility, OTA defines a threshold value (line 3) as

$$\Phi = \sum\nolimits_{t \in \mathcal{T}} w\phi\left(z_t\right), \tag{2}$$

where $\phi(z_t)$ can be interpreted as the marginal cost of the unit item if it stays in the knapsack in slot t, and is a function of the real-time knapsack utilization z_t. Thus, Φ is the estimated total

Algorithm 1 Online Algorithms for Online Multiple Knapsacks with Departures (OA-OKD)

input: threshold function $\phi = \{\phi_k\}_{k\in[K]}$, knapsack capacities $\{C_k\}_{k\in[K]}$;
output: admission and assignment decision $\boldsymbol{x}_n = \{x_{nk}\}_{k\in[K]}$;
initialization: set initial knapsack utilization $z_{kt}^{(0)} = 0, \forall k \in [K], t \in [T]$;
for each item n with item information $\mathcal{I}_n = \{a_n, \{w_{nk}, v_{nk}, \mathcal{T}_{nk}\}_{k\in[K]}\}$ **do**
    for each knapsack $k \in [K]$ **do**
        call Algorithm 2 to check admissibility $\hat{x}_{nk} = \mathtt{OTA}(\mathcal{I}_n, \phi_k, \{z_{kt}^{(n-1)}\}_{t\in\mathcal{T}_{nk}}, C_k)$;
    end for
    if $\sum_{k\in[K]} \hat{x}_{nk} > 0$ **then**
        admit item n and assign it to knapsack $k' = \arg\max_{k\in[K]:\hat{x}_{nk}=1} v_{nk}$;
        set $x_{nk'} = 1$ and $x_{nk} = 0, \forall k \in [K] \setminus \{k'\}$;
    else
        decline item n and set $x_{nk} = 0, \forall k \in [K]$;
    end if
    update knapsack utilization $z_{kt}^{(n)} = z_{kt}^{(n-1)} + w_{nk}x_{nk}, \forall k \in [K], t \in \mathcal{T}_{nk}$.
end for

Algorithm 2 Online Threshold-based Algorithm for Admission Control (OTA)

input: item information $\{v, w, \mathcal{T}\}$, threshold function ϕ, utilization $\{z_t\}_{t\in\mathcal{T}}$, capacity C;
output: admission decision $\hat{x}$;
determine a threshold value $\Phi = \sum_{t\in\mathcal{T}} w\phi(z_t)$;
if $v \geq \Phi$ and $z_t + w \leq C, \forall t \in \mathcal{T}$ **then**
    item is admissible and set $\hat{x} = 1$;
else
    item is inadmissible and set $\hat{x} = 0$.
end if

cost of the item that has size w and stays in the knapsack over the time slots in $\mathcal{T}$. An item is admissible if it passes a *threshold check*, i.e., item value is greater than or equal to the threshold value ($v \geq \Phi$), and a *capacity check*, i.e., the knapsack capacity is not violated due to admission of the item ($z_t + w \leq C, \forall t \in \mathcal{T}$). The core of OA-OKD is the design of the threshold function ϕ such that the competitive ratio of OA-OKD can be guaranteed. For notational convenience, let OA(ϕ) denote OA-OKD with threshold function ϕ and CR(OA(ϕ)) denote the corresponding competitive ratio.

OA(ϕ) consists of two parts: decomposing the multiple knapsack problem into the admissibility check of each individual knapsack and admission control of each individual knapsack via OTA. The ideas of both parts can date back to the early work [33] for the classic OKP and we extend those ideas to settings that allow item departures in OA(ϕ). Although extending the algorithm from OKP to OKD is straightforward and some similar extensions exist in the literature (e.g., the time-expanded knapsack in [32]), designing a threshold function ϕ with the tightest competitiveness guarantee for OKD is still an open problem. The main algorithmic contribution of this paper is to design ϕ to achieve an order-optimal competitive ratio for OKD, and this is made possible by a careful redesign of the threshold function and a novel partitioning technique in the analysis of the competitive ratio. We discuss this more in §3.3.

3.1.2 Beyond the worst case: a data-driven algorithm for OKD. As is typical of optimal competitive algorithms, OA(ϕ) is conservative in its decisions in order to ensure an order-optimal competitive

Algorithm 3 Data-driven online algorithm (DOA)

1: **input:** bounds of item information $\{\theta_k, \alpha_k, \varepsilon_k\}_{k\in[K]}$, knapsack capacities $\{C_k\}_{k\in[K]}$;
2: **output:** reward of each round $R^\ell(\mathcal{I}^\ell, \phi^\ell)$;
3: **for** each round $\ell = 1, \dots, L$ **do**
4: select the threshold function ϕ^ℓ using a data-driven approach;
5: run $\mathtt{OA}(\phi^\ell)$ to execute the instance $\mathcal{I}^\ell$ and collect reward $R^\ell(\mathcal{I}^\ell, \phi^\ell)$.
6: **end for**

ratio. In this section, we propose an algorithm that moves beyond optimizing for the worst-case. The idea is to adaptively learn a policy that works well on practical instances by using a policy class that ensures any algorithm selected has a competitive ratio guarantee. In particular, consider running $\mathtt{OA}(\phi)$ repeatedly for L rounds. At the beginning of round $\ell \in [L]$, we select the threshold function ϕ^ℓ and then run $\mathtt{OA}(\phi^\ell)$ to execute the instance $\mathcal{I}^\ell$. Let $R^\ell(\mathcal{I}^\ell, \phi^\ell)$ denote the reward of round ℓ. The goal of the data-driven online algorithm (DOA) is to maximize the average reward over L rounds. Its pseudo-code is summarized in Algorithm 3. We say DOA achieves good average performance if its average reward is close to that obtained by a fixed threshold function selected in hindsight, i.e., $\phi^{\mathsf{off}} = \arg\max_\phi \sum_{\ell\in[L]} R^\ell(\mathcal{I}^\ell, \phi)/L$.

This approach has been successfully applied to the basic OKP and online set cover problem in [30]. However, existing results cannot be generalized to cope with the two technical challenges in our setting: (i) how to restrict the selection of ϕ to a feasible set, which not only contains ϕ that can achieve good average performance under typical instances but also has guaranteed worst-case performance; (ii) given the feasible set, how to design a data-driven algorithm to select ϕ to achieve a good average performance. In this paper, we provide theoretical results to tackle the first challenge (see Theorem 3) and show a viable empirical approach to solve the second challenge (see the numerical results in §6).

3.2 Main results

We now state our main results, which provide an upper bound on the competitive ratio of $\mathtt{OA}(\phi)$ (Theorem 1), a lower bound on the competitive ratio of any online algorithm (Theorem 2), and a competitive ratio bound for the data-driven algorithm (Theorem 3).

Theorem 1. *Under Assumptions 1-3, there exists $\gamma_k = O(\ln(\alpha_k\theta_k)), \forall k \in [K]$, if the item size is upper bounded by $\varepsilon_k \le C_k \ln 2/\gamma_k, \forall k \in [K]$, and the threshold function is $\phi^\gamma := \{\phi_k^{\gamma_k}\}_{k\in[K]}$, where*

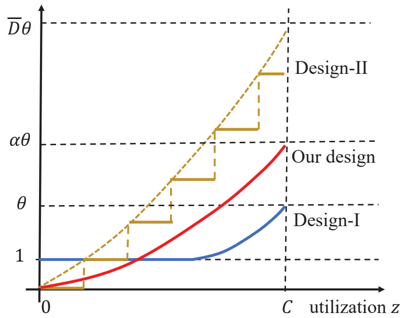
$$\phi_k^{\gamma_k}(z) = \exp\left(z\gamma_k/C_k\right) - 1, z \in [0, C_k], \forall k \in [K], \tag{3}$$

then the competitive ratio of $\mathtt{OA}(\phi^\gamma)$ is $O(\ln(\alpha\theta))$, where $\theta = \max_{k\in[K]} \theta_k$ and $\alpha = \max_{k\in[K]} \alpha_k$.

Theorem 2. *There is no online algorithm that can achieve a competitive ratio smaller than $\Omega(\ln(\alpha\theta))$ for the online multiple one-dimensional knapsacks with departures.*

Combining the upper bound result in Theorem 1 and the lower bound result in Theorem 2, we conclude that our proposed $\mathtt{OA}(\phi^\gamma)$ achieves an order-optimal competitive ratio for OKD. Before proceeding to the detailed proofs (§4 for Theorem 1 and §5 for Theorem 2), we first provide intuitions underlying the design and analysis of $\mathtt{OA}(\phi^\gamma)$. In fact, Theorem 1 provides a class of threshold functions in (3) parameterized by $\gamma := \{\gamma_k\}_{k\in[K]}$. When γ and the upper bound of item size (i.e., $\{\varepsilon_k\}_{k\in[K]}$) are both in the proper regimes, we have $\mathtt{CR}(\mathtt{OA}(\phi^\gamma)) = O(\ln(\alpha\theta))$.

Besides the optimal competitiveness guarantee, Theorem 1 also provides two additional new results compared to the competitive algorithms for OKP in the literature.



(a) Plots of different designs

Designs	Threshold functions
Design-I [32, 33]	$\phi^{I}(z) = \begin{cases} 1 & z \in \left[0, \frac{C}{1+\ln\theta}\right) \\ e^{(1+\ln\theta)z/C-1} & z \in \left[\frac{C}{1+\ln\theta}, C\right] \end{cases}$
Design-II [29]	$\phi^{II}(z) = 2^{\lfloor z\log(\overline{D}\theta)/C \rfloor} - 1, z \in [0, C]$
Our design	$\phi^{\texttt{wco}}(z) = e^{z\ln(\alpha\theta+1)} - 1, z \in [0, C]$

(b) Formulas of different designs

Fig. 1. Illustrating threshold functions of two benchmark designs and our proposed design. Design-I consists of a flat segment and an exponential segment. Design-II is a discretized version of an exponential function (i.e., the dashed line). Our proposed design is a continuous exponential function in Equation (3) with $\gamma^{\texttt{wco}} = \ln(\alpha\theta + 1)$.

First, to design online algorithms for OKP [28, 29, 31, 33], ε_k is commonly assumed to be infinitesimal compared to the knapsack capacity, i.e., $\varepsilon_k \ll C_k, \forall k \in [K]$, since the infinitesimal setting can eliminate challenges from the size variations of items and simplify the algorithms to focus on the key challenge from the varying value density. Our result removes the infinitesimal assumption and characterizes the regime of item size to achieve an order-optimal competitive result.

Second, Theorem 1 empowers us to design a data-driven online algorithm, which cannot only have worst-case performance guarantees but also learn from past data to optimize average-case performance of typical instances in real-world applications. In particular, for a given target competitive ratio β, we can characterize a parameter set $\Gamma(\beta)$ such that $\texttt{CR}(\texttt{OA}(\phi^{\gamma})) \le \beta, \forall \gamma \in \Gamma(\beta)$, which is formally presented as follows.

Theorem 3. *Given $\beta \ge \hat{\beta} := 10 + \frac{12}{\ln 2}\ln(\alpha\theta + 1)$, $\texttt{OA}(\phi^{\gamma})$ is β-competitive if the parameter $\gamma := \{\gamma_k\}_{k\in[K]}$ is chosen from $\Gamma(\beta)$, where the parameter set $\Gamma(\beta) \subseteq \mathbb{R}^K$ is given by*

$$\Gamma(\beta) = \left\{\gamma : (\beta-1)\zeta_k - 2W\left(\frac{(\beta-1)\zeta_k}{2\sqrt{2}}\exp\left(\frac{(\beta-1)\zeta_k}{2}\right)\right) \le \gamma_k \le \ln 2 \cdot \min\left\{\frac{(\beta-4)}{6}, \frac{C_k}{\varepsilon_k}\right\}\right\}, \quad (4)$$

where $\zeta_k := -\ln 2/(6\alpha_k\theta_k)$ and $W(\cdot)$ is the Lambert W function.

Theorem 3 essentially specifies a class of parameterized online algorithms for OKD that can provide the same competitiveness guarantee. Note that although those algorithms provide the same guarantee in the worst-case, their performances can be distinguished in practice since "typical instances" from real-world applications can be far from the worst-case instances. Theorem 3 further gives us the design space to choose the parameter of ϕ^{γ} with worst-case guarantees. Thus, in Algorithm 3, we can use a data-driven approach to adaptively select γ (equivalently ϕ^{γ}) from $\Gamma(\beta)$ to learn the best choice of γ in an online manner and, in the meantime, ensure the overall worst-case performance β. We evaluate the performance of data-driven algorithms in §6.1.

3.3 Discussion

We end this section with a discussion of the design decisions in our algorithm, and provide a contrast to the design of prior algorithms. In particular, Figure 1 compares our proposed design ϕ^{γ} with two important state-of-the-art designs ϕ^{I} and ϕ^{II}. ϕ^{I} is most widely used (e.g., [32, 33]) and has been proven optimal for the basic OKP and OMK. ϕ^{II} has recently been proposed for OMdK in [29], where it can achieve the order-optimal competitive ratio. Although these benchmark designs are the best possible designs for the classic OKP, OMK, and OMdK, applying them directly in $\texttt{OA}(\phi)$ cannot achieve the desired competitive results for OKD. We formalize this in the following lemma.

Lemma 1. *Under Assumptions 1-3,*

(1) if the threshold function is given by Design-I, then $\texttt{CR}(\texttt{OA}(\phi^I)) = \Omega(\alpha \ln \theta)$*;*
(2) if the threshold function is given by Design-II, then $\texttt{CR}(\texttt{OA}(\phi^{II})) = \Omega(\ln(\overline{D}\theta))$*.*

The above lemma gives impossibility results for applying the benchmark designs to achieve an order-optimal competitive ratio $\Theta(\ln(\alpha\theta))$. We provide a complete proof of Lemma 1 in Appendix A. In what follows we give insights on why both designs fail to achieve order-optimal performances. In fact, Design-I and Design-II fail due to two different types of worst-case instances. To be more precise, given a threshold function, the input instances that satisfy Assumptions 1-3 can be divided into two types: *capacity-free* instances where the final utilization values of all knapsacks lie below the capacity limits and hence the admission control in OTA is purely determined by the threshold check; and *capacity-limited* instances where the utilization of at least one knapsack in one slot approaches the capacity limit and thus admission control depends on both threshold and capacity checks. The lower bounds of Design-I and Design-II result from their poor performances compared to offline algorithms under capacity-limited and capacity-free instances, respectively.

In particular, ϕ^I neglects the item duration and thus any item with value density θ regardless of its duration can pass the threshold check in OTA. Then, the knapsack capacity can be quickly fully filled by short-duration items, while long-duration items that arrive later are declined due to capacity check. In contrast, the offline algorithm admits the long items declined by OTA and this leads to a unavoidable worst-case ratio that increases linearly in the duration ratio α.

To make order improvement in competitive ratios, ϕ^{II} proactively prohibits the occurrence of capacity-limited cases, by setting the terminal value $\phi^{II}(C)$ approximately equal to $\overline{D}\theta$. If an item has size w and its duration has overlap with any nearly full slots, then the item faces a threshold value of at least $\Phi \approx w\overline{D}\theta$, which is the largest possible value of a w-sized item. Thus, the item will be declined due to the threshold check, and in this way, ϕ^{II} essentially avoids the occurrence of capacity-limited cases. However, ϕ^{II} is over-conservative, reserving too much capacity for future high-valued items that may never come, and this results in a lower bound $\Omega(\ln(\overline{D}\theta))$ under capacity-free instances.

We note that the threshold functions of existing designs in the literature are generally of the form $\phi^\gamma(z) = O(\exp(z\gamma/C))$, where γ is a critical parameter. A larger γ leads to a steeper exponential function and accordingly a more conservative algorithm since more capacity will be reserved for high-valued items that may arrive later. ϕ^I and ϕ^{II} can be considered as two special cases by setting $\gamma = O(\ln \theta)$ and $\gamma = O(\ln(\overline{D}\theta))$, respectively. To overcome the limitations of ϕ^I and ϕ^{II}, we directly analyze the competitive performance of the algorithm with the general parameterized threshold function ϕ^γ. We can show that its competitive ratio is $O(\gamma)$ under capacity-free instances and $O(\alpha\theta\gamma/(\exp((\gamma - \ln 2)/2) - 1))$ under capacity-limited instances (see §4.2 for more details). Thus, we choose $\gamma = O(\ln(\alpha\theta))$ to tradeoff the performances under the two types of hard instances.

The key challenge in the competitive analysis is how to attain the optimal dependence on item duration ratio α. Since an instance for OKD is defined over a long time horizon $[T]$, the values of the online items may not be comparable with the offline values over the whole time horizon. Fortunately, as the maximum duration of each item is $\overline{D}$, the items in OKD are only correlated over a much shorter interval, i.e., the items that start in slot t are only correlated with items that start in slot $\{t - \overline{D} + 1, \dots, t + \overline{D} - 1\}$, and this motivates us to partition an instance into smaller sub-instances based on item's *starting time* and analyze the performance for each sub-instance.

Combining the new design of the threshold function with the corresponding new analysis, we show that our design achieves an optimal order competitive ratio.

Remark 1. *We focus on threshold-based algorithms in this paper. In the literature, a class of online primal-dual algorithms (*OPD*) [6, 7] has also been proposed to solve online knapsack problems. However, it is still challenging to extend* OPD *in [6] to the setting of* OKD *(See details in §7). In addition, to attain the optimal competitive ratio, we explicitly take advantage of the structural property of* OKD*. For example, the weak dependence of items across time horizon is utilized to partition the original instance into sub-instances with smaller intervals in our analysis. Such problem structure is crucial in our algorithm design and analysis. However, it is unclear how to take into account such structural information when designing* OPD *yet. Therefore, it is non-trivial to design* OPD *to achieve the optimal competitive ratio of* OKD*, but it is a promising future direction to explore.*

4 COMPETITIVE ANALYSIS

In this section, we analyze the competitive ratio of the online algorithm $\texttt{OA}(\phi^{\gamma})$ and formally prove Theorem 1 and Theorem 3. We first sketch the proofs of both theorems based on two technical lemmas and then prove the lemmas in §4.1 and §4.2, respectively.

To facilitate the competitive analysis of $\texttt{OA}(\phi^{\gamma})$, we first define K ancillary problems for OKD with K knapsacks. Let $\texttt{OKD}_k$ denote the k-th ancillary problem, which only allows items assigned to the k-th knapsack. The online decision of $\texttt{OKD}_k$ only depends on the item information related to knapsack k and is purely determined by the OTA with threshold function $\phi_k^{\gamma_k}$, which is called $\texttt{OTA}_k$. With the K ancillary problems, we decompose the analysis of $\texttt{OA}(\phi^{\gamma})$ into two lemmas.

Lemma 2. *Given* $\texttt{OTA}_k$ *is* $\texttt{CR}_k$*-competitive for* $k \in [K]$*, then* $\texttt{CR}(\texttt{OA}(\phi^{\gamma})) = 1 + \max_{k\in[K]} \texttt{CR}_k$*.*

Lemma 2 decomposes the analysis of OA-OKD into analysis of K ancillary algorithms $\{\texttt{OTA}_k\}_{k\in[K]}$, where each is the OTA for a single OKD. Based on Lemma 2, the competitiveness of OA-OKD is no worse than the worst $\texttt{CR}_k$ among $\{\texttt{OKD}_k\}_{k\in[K]}$ plus 1. Therefore, based on OA-OKD, multiple-knapsack OKD is not much harder than single OKD and we can focus on the analysis of $\texttt{OTA}_k, \forall k \in [K]$. Lemma 3 provides the competitive ratio of $\texttt{OTA}_k$.

Lemma 3. *Under Assumptions 1-3, if the threshold function* $\phi_k^{\gamma_k}$ *is given by Equation (3) with* $\gamma_k \in (\ln 2, +\infty)$ *and the item size is upper bounded by* $\varepsilon_k \le C_k \ln 2/\gamma_k$*, the competitive ratio of* $\texttt{OTA}_k$ *is*

$$\texttt{CR}_k(\gamma_k) = 3\max\left\{1 + \frac{2}{\ln 2}\gamma_k, \frac{2}{\ln 2}\cdot\frac{\alpha_k\theta_k\gamma_k}{\exp(\gamma_k - \ln 2)/2) - 1}\right\}. \tag{5}$$

The competitive ratio $\texttt{CR}_k(\gamma_k)$ consists of two terms, which capture the worst-case ratios under the capacity-free and capacity-limited instances, respectively. We choose to set $\gamma_k = O(\ln(\alpha_k\theta_k))$ that best balances these two worst-case ratios. Particularly, with $\hat{\gamma}_k = 2\ln(\alpha_k\theta_k + 1) + \ln 2$, we have

$$\texttt{CR}_k(\hat{\gamma}_k) := \max\left\{9 + \frac{12}{\ln 2}\ln(\alpha_k\theta_k + 1), 6 + \frac{12}{\ln 2}\ln(\alpha_k\theta_k + 1)\right\} = O(\ln(\alpha_k\theta_k)).$$

Therefore, there exist $\gamma_k = O(\ln(\alpha_k\theta_k))$ such that $\texttt{CR}_k = O(\ln(\alpha_k\theta_k))$. Combining with the decomposition in Lemma 2, there exist $\hat{\gamma} := \{\hat{\gamma}_k\}_{k\in[K]}$ such that

$$\texttt{CR}(\texttt{OA}(\phi^{\hat{\gamma}})) = 1 + \max_{k\in[K]} \texttt{CR}_k(\hat{\gamma}_k) = 10 + \frac{12}{\ln 2}\ln(\alpha\theta + 1) = O(\ln(\alpha\theta)),$$

which gives the main result in Theorem 1.

The proof of Theorem 3 also leverages the results in Lemma 2 and Lemma 3. Define $\hat{\beta} := \texttt{CR}(\texttt{OA}(\phi^{\hat{\gamma}}))$ as the reference competitive ratio. To ensure $\texttt{CR}(\texttt{OA}(\phi^{\gamma})) \le \beta, \forall \gamma \in \Gamma(\beta)$, we need to choose $\gamma := \{\gamma_k\}_{k\in[K]}$ such that $1 + \mathsf{CR}_k(\gamma_k) \le \beta, \forall k \in [K]$. In addition, the results of Lemma 3 requires $\gamma_k \le C_k \ln 2/\varepsilon_k$. Combining above results gives us the parameter set $\Gamma(\beta)$ in Theorem 3.

4.1 Proof of Lemma 2: Decomposition

Denote by $\mathcal{S}_k$ and $\mathcal{S}_k^*$ the sets of items assigned to knapsack k by OA-OKD and an offline algorithm that achieves an optimal solution, respectively. Then the total values of online and offline algorithms under instance $\mathcal{I}$ are $\mathsf{ALG}(\mathcal{I}) = \sum_{k\in[K]}\sum_{n\in\mathcal{S}_k} v_{nk}$ and $\mathsf{OPT}(\mathcal{I}) = \sum_{k\in[K]}\sum_{n\in\mathcal{S}_k^*} v_{nk}$.

For each knapsack k, let $\mathcal{S}_k^* \setminus \mathcal{S}_k$ denote the set of items that are admitted to knapsack k by the offline algorithm but not by OA-OKD. There can be two reasons why an item n in $\mathcal{S}_k^* \setminus \mathcal{S}_k$ is not admitted by the OA-OKD: (i) the item is inadmissible to knapsack k (i.e., $\hat{x}_{nk} = 0$ in Line 6 in Algorithm 1); or (ii) the item is admissible to k but it is finally assigned to knapsack k' since the item value in k' is larger than that in k, i.e., $v_{nk'} \ge v_{nk}$ (Line 9 in Algorithm 1). Let $\hat{\mathcal{S}}_k$ denote the set of items due to the second reason, which are the coupling items across knapsacks. Since items in $\hat{\mathcal{S}}_k$ are admitted by OA-OKD, we have $\{\hat{\mathcal{S}}_k\}_{k\in[K]} \subseteq \{\mathcal{S}_k\}_{k\in[K]}$, and thus

$$\sum\nolimits_{k\in[K]}\sum\nolimits_{n\in\hat{\mathcal{S}}_k} v_{nk} \le \sum\nolimits_{k\in[K]}\sum\nolimits_{n\in\mathcal{S}_k} v_{nk} = \mathsf{ALG}(\mathcal{I}). \tag{6}$$

We divide $\mathcal{S}_k^*$ to two subsets $\mathcal{S}_k^* \setminus \hat{\mathcal{S}}_k$ and $\hat{\mathcal{S}}_k$, and we have

$$\begin{aligned}\frac{\mathsf{OPT}(\mathcal{I})}{\mathsf{ALG}(\mathcal{I})} &= \frac{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k^*} v_{nk}}{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k} v_{nk}} = \frac{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k^*\setminus\hat{\mathcal{S}}_k} v_{nk} + \sum_{k\in[K]}\sum_{n\in\hat{\mathcal{S}}_k} v_{nk}}{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k} v_{nk}}\\ &\le 1 + \frac{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k^*\setminus\hat{\mathcal{S}}_k} v_{nk}}{\sum_{k\in[K]}\sum_{n\in\mathcal{S}_k} v_{nk}} \le 1 + \max_{k\in[K]} \frac{\sum_{n\in\mathcal{S}_k^*\setminus\hat{\mathcal{S}}_k} v_{nk}}{\sum_{n\in\mathcal{S}_k} v_{nk}},\end{aligned} \tag{7}$$

where the first inequality in (7) is due to Equation (6).

For each knapsack k, we construct an instance $\hat{\mathcal{I}}_k$ by extracting the items belonging to the set $\mathcal{S}_k \cup (\mathcal{S}_k^* \setminus \hat{\mathcal{S}}_k)$ from the original instance $\mathcal{I}$ and keeping the item sequence. Note that in the instance $\hat{\mathcal{I}}_k$, the set $\mathcal{S}_k$ includes all items that are admissible to knapsack k via OTA because other admissible items in $\hat{\mathcal{S}}_k$ have been excluded in the construction. If we present $\hat{\mathcal{I}}_k$ to the ancillary problem OKD_k, the online algorithm OTA_k also admits items $\mathcal{S}_k$ as does OA-OKD under instance $\mathcal{I}$, and achieves value $\sum_{n\in\mathcal{S}_k} v_{nk}$. In addition, the offline value of OKD_k under instance $\hat{\mathcal{I}}_k$ is no less than $\sum_{n\in\mathcal{S}_k^*\setminus\hat{\mathcal{S}}_k} v_{nk}$ since admitting items in $\mathcal{S}_k^* \setminus \hat{\mathcal{S}}_k$ is a feasible admission decision. Let CR_k denote the competitive ratio of OTA_k for OKD_k. By the definition of CR_k, we have

$$\frac{\sum_{n\in\mathcal{S}_k^*\setminus\hat{\mathcal{S}}_k} v_{nk}}{\sum_{n\in\mathcal{S}_k} v_{nk}} \le \mathsf{CR}_k, \forall k \in [K]. \tag{8}$$

Thus, the competitive ratio of OA-OKD is $\mathsf{CR} = 1 + \max_{k\in[K]} \mathsf{CR}_k$.

4.2 Proof of Lemma 3: Single Online Knapsack with Departures

We next analyze the competitive ratio of OTA_k for OKD_k. For convenience of notation, we omit the index k. Let $\mathcal{N}$ denote the instance for a single OKD and $N = |\mathcal{N}|$ denote the number of items in $\mathcal{N}$.

Partitioning of the time horizon into small segments. The first key step in our proof is to reduce the dependence of the competitiveness of the algorithm from the entire time horizon to smaller segments that are in the order of item duration. Let $\{z_t^{(N)}\}_{t\in[T]}$ denote the final utilization of all time slots after OTA executes all items in $\mathcal{N}$. Assume the time horizon T is an integer multiple of $\overline{D}$ and divide the time horizon into $H = T/\overline{D}$ segments. Let $\mathcal{T}^h := \{t \in [T] : (h-1)\overline{D} + 1 \le t \le h\overline{D}\}$ denote the set of time slots in the h-th segment and $\mathcal{N}^h$ denote a sub-instance of $\mathcal{N}$ that contains items whose *starting times* are in $\mathcal{T}^h$. We further denote a 3-segment sub-instance by $\tilde{\mathcal{N}}^h = \mathcal{N}^{h-1} \cup \mathcal{N}^h \cup \mathcal{N}^{h+1}, h \in [H]$, where $\mathcal{N}^0 = \mathcal{N}^{H+1} = \emptyset$. The order of the items in $\tilde{\mathcal{N}}^h$ is based on their arrival times.

Let $\texttt{OPT}(\mathcal{N})$ and $\texttt{ALG}(\mathcal{N})$ denote the value of items admitted from $\mathcal{N}$ by the offline algorithm and OTA, respectively. The competitive ratio of OTA is

$$\frac{\texttt{OPT}(\mathcal{N})}{\texttt{ALG}(\mathcal{N})} = \frac{\sum_{h\in[H]} \texttt{OPT}(\mathcal{N}^h)}{\sum_{h\in[H]} \texttt{ALG}(\mathcal{N}^h)} = \frac{3\sum_{h\in[H]} \texttt{OPT}(\mathcal{N}^h)}{\texttt{ALG}(\mathcal{N}^1) + \sum_{h\in[H]} \texttt{ALG}(\tilde{\mathcal{N}}^h) + \texttt{ALG}(\mathcal{N}^H)} \le 3 \max_{h\in[H]} \frac{\texttt{OPT}(\mathcal{N}^h)}{\texttt{ALG}(\tilde{\mathcal{N}}^h)}.$$

By partitioning the instance based on the items' starting time, the competitive analysis of OTA can be turned to the analysis of $\texttt{OPT}(\mathcal{N}^h)/\texttt{ALG}(\tilde{\mathcal{N}}^h)$, which is the ratio of the offline value from the sub-instance $\mathcal{N}^h$ and the online value from the 3-segment sub-instance $\tilde{\mathcal{N}}^h$. In the following, we focus on analyzing the upper bound of $\texttt{OPT}(\mathcal{N}^h)/\texttt{ALG}(\tilde{\mathcal{N}}^h)$ in two cases.

Case I: capacity-free case: The final utilizations of all time slots in $\hat{\mathcal{T}}^h := \mathcal{T}^h \cup \mathcal{T}^{h+1}$ are below the capacity, i.e., $z_t^{(N)} \le C - \varepsilon, \forall t \in \hat{\mathcal{T}}^h$.

In this case, the only reason why one item is rejected by OTA is that it fails to pass the *threshold check*, i.e., $v < \Phi$ in Line 4 in Algorithm 2. Then $\texttt{OPT}(\mathcal{N}^h)$ and $\texttt{ALG}(\tilde{\mathcal{N}}^h)$ can be connected via the final utilization $\{z_t^{(N)}\}_{t\in\hat{\mathcal{T}}^h}$. We first show that $\texttt{ALG}(\tilde{\mathcal{N}}^h)$ is lower bounded.

Proposition 1. *In Case I, the value of items in $\tilde{\mathcal{N}}^h$ admitted by* OTA *is lower bounded by*

$$\texttt{ALG}(\tilde{\mathcal{N}}^h) \ge \frac{\ln 2}{2\gamma} \sum_{t\in\hat{\mathcal{T}}^h} \phi(z_t^{(N)})C. \tag{9}$$

Proof. Since $\phi(0) = 0$, we can apply $\phi(z_t^{(N)}) = \sum_{n\in\mathcal{N}}[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})]$ and have

$$\begin{aligned}\sum_{t\in\hat{\mathcal{T}}^h} \phi(z_t^{(N)})C &= \sum_{t\in\hat{\mathcal{T}}^h} \sum_{n\in\mathcal{N}} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})] \\ &= \sum_{t\in\hat{\mathcal{T}}^h} \sum_{n\in\tilde{\mathcal{N}}^h: t\in\mathcal{T}_n} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})] \\ &\le \sum_{n\in\tilde{\mathcal{N}}^h} \sum_{t\in\mathcal{T}_n} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})],\end{aligned} \tag{10}$$

where $\mathcal{T}_n$ is the stay duration of item n. The second equality holds because the maximum duration of each item is $\overline{D}$ and hence the items that can stay in $\hat{\mathcal{T}}^h$ must be from $\tilde{\mathcal{N}}^h$. The last inequality holds since the items in $\tilde{\mathcal{N}}^h$ can stay up to segment $h+2$, which is outside $\hat{\mathcal{T}}^h$.

Let $\Delta\texttt{ALG}_n$ denote the increment of OTA due to processing item n. $\Delta\texttt{ALG}_n = 0$ if item n is declined and $\Delta\texttt{ALG}_n = v_n$ if it is admitted. We next show $\sum_{t\in\mathcal{T}_n} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})] \le (2\gamma/\ln 2)\Delta\texttt{ALG}_n, \forall n \in \tilde{\mathcal{N}}^h$ in the following two sub-cases.

Case I(a). When item n is declined by OTA, we have $z_t^{(n)} = z_t^{(n-1)}, \forall t \in \mathcal{T}_n$, and thus this gives $\sum_{t\in\mathcal{T}_n} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})] = 0 \le \frac{2\gamma}{\ln 2}\Delta\texttt{ALG}_n$.

Case I(b). When item n is admitted by OTA, we have $z_t^{(n)} = z_t^{(n-1)} + w_n, \forall t \in \mathcal{T}_n$, and then

$$\sum_{t\in\mathcal{T}_n} C[\phi(z_t^{(n)}) - \phi(z_t^{(n-1)})] = C\sum_{t\in\mathcal{T}_n} \exp(z_t^{(n-1)}\gamma/C)[\exp(w_n\gamma/C) - 1] \tag{11a}$$

$$\le C\sum_{t\in\mathcal{T}_n} \exp(z_t^{(n-1)}\gamma/C) \cdot \frac{w_n\gamma}{C\ln 2} \tag{11b}$$

$$= \frac{\gamma}{\ln 2}\sum_{t\in\mathcal{T}_n} w_n\phi(z_t^{(n-1)}) + \frac{\gamma}{\ln 2} w_n d_n \tag{11c}$$

$$\le \frac{\gamma}{\ln 2} v_n + \frac{\gamma}{\ln 2} v_n = \frac{2\gamma}{\ln 2}\Delta\texttt{ALG}_n. \tag{11d}$$

Equation (11a) is obtained by substituting threshold function (3). Inequality (11b) holds since $\exp(x \ln 2) - 1 \leq x$ if $0 \leq x \leq 1$, and $w_n\gamma/(C \ln 2) \leq \varepsilon\gamma/(C \ln 2) \leq 1$ based on the additional condition on the item size in Theorem 1. The last inequality (11d) holds since $\sum_{t \in \mathcal{T}_n} w_n \phi(z_t^{(n-1)}) \leq v_n$ when item n can pass the threshold check, and $w_n d_n \leq v_n$ based on Assumption 1.

Combining Equations (10)-(11) gives

$$\sum\nolimits_{t \in \hat{\mathcal{T}}^h} \phi(z_t^{(N)})C \leq \sum\nolimits_{n \in \tilde{\mathcal{N}}^h} \frac{2\gamma}{\ln 2} \Delta \mathtt{ALG}_n = \frac{2\gamma}{\ln 2} \mathtt{ALG}(\tilde{\mathcal{N}}^h),$$

which completes the proof. □

Next, we show the offline optimal value of items in $\mathcal{N}^h$ is upper bounded.

Proposition 2. *In Case I, the value of items in $\mathcal{N}^h$ admitted by the offline algorithm is*

$$\mathtt{OPT}(\mathcal{N}^h) \leq \mathtt{ALG}(\mathcal{N}^h) + \sum\nolimits_{t \in \hat{\mathcal{T}}^h} \phi(z_t^{(N)})C. \tag{12}$$

Proof. Let $\mathcal{S}^h \cap \mathcal{S}^{h*}$ denote the set of items in $\mathcal{N}^h$ that are admitted by both online algorithm OTA and the offline algorithm, and $\mathcal{S}^{h*} \setminus \mathcal{S}^h$ denote the set of items that are declined by OTA but admitted by the offline algorithm. We have

$$\sum\nolimits_{n \in \mathcal{S}^h \cap \mathcal{S}^{h*}} v_n \leq \mathtt{ALG}(\mathcal{N}^h), \tag{13}$$

$$\begin{aligned}\sum\nolimits_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h} v_n &\leq \sum\nolimits_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h} \sum\nolimits_{t \in \mathcal{T}_n} w_n \phi(z_t^{(n-1)}) \leq \sum\nolimits_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h} \sum\nolimits_{t \in \mathcal{T}_n} w_n \phi(z_t^{(N)}) \\ &= \sum\nolimits_{t \in \hat{\mathcal{T}}^h} \sum\nolimits_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h : t \in \mathcal{T}_n} w_n \phi(z_t^{(N)}) \leq \sum\nolimits_{t \in \hat{\mathcal{T}}^h} \phi(z_t^{(N)})C,\end{aligned} \tag{14}$$

where the first inequality in (14) holds since the item fails to pass the threshold check, and the last inequality holds since the items admitted by the offline algorithm cannot exceed the knapsack capacity, i.e., $\sum_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h : t \in \mathcal{T}_n} w_n \leq C$. Thus, we have

$$\mathtt{OPT}(\mathcal{N}^h) = \sum\nolimits_{n \in \mathcal{S}^h \cap \mathcal{S}^{h*}} v_n + \sum\nolimits_{n \in \mathcal{S}^{h*} \setminus \mathcal{S}^h} v_n \leq \mathtt{ALG}(\mathcal{N}^h) + \sum\nolimits_{t \in \hat{\mathcal{T}}^h} \phi(z_t^{(N)})C,$$

which completes the proof. □

Combining Proposition 1 and Proposition 2 gives $\frac{\mathtt{OPT}(\mathcal{N}^h)}{\mathtt{ALG}(\tilde{\mathcal{N}}^h)} \leq \frac{\mathtt{ALG}(\mathcal{N}^h) + \sum_{t \in \hat{\mathcal{T}}^h} \phi(z_t^{(N)})C}{\mathtt{ALG}(\tilde{\mathcal{N}}^h)} \leq 1 + \frac{2}{\ln 2}\gamma$.

Case II: capacity-limited case: There exists at least one time slot $t' \in \hat{\mathcal{T}}^h$ whose utilization approaches the knapsack capacity, i.e., $C - \varepsilon < z_{t'}^{(N)} \leq C$.

In this case, if one item is rejected, the reason can be the failure of passing either threshold check (the item value is smaller than the threshold) or capacity check (admitting this item violates the capacity constraint). A key observation is that if there exists one slot (say t') whose utilization approaches the capacity, then the final utilization of the knapsack around this fully utilized slot will be above a certain value due to the minimum duration assumption. In the following result, we leverage this observation to obtain a lower bound for $\mathtt{ALG}(\tilde{\mathcal{N}}^h)$.

Proposition 3. *In Case II, the value of items in $\tilde{\mathcal{N}}^h$ admitted by* OTA *is lower bounded by*

$$\mathtt{ALG}(\tilde{\mathcal{N}}^h) \geq \frac{\ln 2 C \underline{D}}{\gamma} \left[\exp((\gamma - \ln 2)/2) - 1\right]. \tag{15}$$

Proof. In Case II, there exists at least one time slot $t' \in \hat{\mathcal{T}}^h$, in which the final utilization by OTA approaches the capacity. Also, each item stays in the knapsack for at least $\underline{D}$ successive slots. In Figure 2, we show the final utilization of the capacity-limited case in the worst case, where there are two groups of items. Group-1 items start at the slot $t' - \underline{D} + 1$ and Group-2 items start at slot t'.

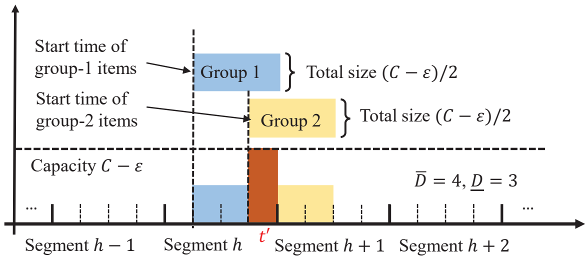


Fig. 2. Illustration of the worst-case final utilization in capacity-limited case.

All items stay in the knapsack for $\underline{D}$ slots, and the total sizes of both groups are $(C-\varepsilon)/2$. These two groups of items result in the nearly full capacity at slot t', i.e., $z_{t'}^{(N)} = C-\varepsilon$, and around half utilization in surrounding slots, i.e., $z_t^{(N)} = (C-\varepsilon)/2$ for $t = t'-\underline{D}+1, \dots, t'-1, t'+1, t'+\underline{D}-1$. Then based on Proposition 1, $\texttt{ALG}(\tilde{\mathcal{N}}^h)$ is lower bounded by

$$\texttt{ALG}(\tilde{\mathcal{N}}^h) \geq \frac{\ln 2}{2\gamma}[(2\underline{D}-2)\phi((C-\varepsilon)/2)+\phi(C-\varepsilon)]C \geq \frac{\ln 2C\underline{D}}{\gamma}\left[\exp((\gamma-\ln 2)/2)-1\right].$$

This completes the proof. □

In Case II, the offline value can only be trivially bounded by $\texttt{OPT}(\mathcal{N}^h) \leq 2C\theta\overline{D}$, which is the maximal possible value for items in $2\overline{D}$ slots. Thus, we have $\frac{\texttt{OPT}(\mathcal{N}^h)}{\texttt{ALG}(\tilde{\mathcal{N}}^h)} \leq \frac{2}{\ln 2}\cdot\frac{\alpha\theta\gamma}{\exp((\gamma-\ln 2)/2)-1}$.

Summarizing the results from the two cases, the competitive ratio of $\texttt{OTA}$ is

$$\texttt{CR}(\gamma) = 3\max\left\{1+\frac{2}{\ln 2}\gamma, \frac{2}{\ln 2}\cdot\frac{\alpha\theta\gamma}{\exp((\gamma-\ln 2)/2)-1}\right\}. \tag{16}$$

This completes the proof of Lemma 3.

5 MULTI-DIMENSIONAL ONLINE MULTIPLE KNAPSACKS WITH DEPARTURES

We now move to the general $\texttt{OKD}$, which additionally considers multi-dimensional items. In particular, the size of item n in knapsack k is modeled as an M_k-dimension vector $\boldsymbol{w}_{nk} = \{w_{nkm}\}_{m\in[M_k]}$. Accordingly, the capacity of each knapsack k is also a multi-dimensional vector $\boldsymbol{C}_k := \{C_{km}\}_{m\in[M_k]}$.

Before stating our main result in this setting, we first extend the assumptions on the value density and item size to the multi-dimensional setting.

Assumption 4 (Value density fluctuation in multi-dimensional setting). *The value density of each item n in knapsack k is bounded by $v_{nk}/(d_{nk}\sum_{m\in[M_k]} w_{nkm}) \in [1,\theta_k], \forall n \in [N]$.*

Compared to the one-dimensional setting, the size of item in the value density is replaced by the aggregate size over all dimensions in the multi-dimensional setting. Similar definitions have been used by previous works [29, 32] in studying online multi-dimensional knapsack problems.

Assumption 5 (Upper bound of item size in multi-dimensional setting). *The item size of each item n in dimension m in knapsack k is bounded by $w_{nkm} \leq \varepsilon_{km} \leq C_{km}, \forall n \in [N]$.*

To generalize our algorithms to this setting, we take into account the multi-dimensional item size by modifying the definition of threshold value in Line 3 in Algorithm 2 as

$$\tilde{\Phi} = \sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{m\in[M]} w_m\phi_m(z_{mt}), \tag{17}$$

where z_{mt} is the utilization of dimension m in slot t and ϕ_m is the threshold function for dimension m. By carefully designing the threshold function, we can further show the modified algorithm can achieve the order-optimal competitive ratio.

In the multi-dimensional setting, the worst-case performance depends on the capacities of the different dimensions. For knapsack k, define a new parameter $\eta_k = \sum_{m\in[M_k]} C_{km}/(\min_{m\in[M_k]} C_{km})$ as the ratio of the aggregate knapsack capacities over all dimensions and the minimum capacity of a single dimension. Let $\eta = \max_{k\in[K]} \eta_k$.

We can now state a generalization of Theorem 1 to the multi-dimensional setting as follows.

Theorem 4. *Under Assumptions 2, 4, and 5, there exists $\gamma_k = O(\ln(\eta_k\alpha_k\theta_k))$, if the item size is upper bounded by $\varepsilon_{km} \le C_{km}\ln 2/\gamma_k, \forall k \in [K], m \in [M_k]$, and the threshold function $\phi^\gamma := \{\phi_{km}^{\gamma_k}\}_{k\in[K],m\in[M_k]}$ is given by*

$$\phi_{km}^{\gamma_k}(z) = \exp\left(z\gamma_k/C_{mk}\right) - 1, \forall k \in [K], m \in [M_k], \tag{18}$$

then the competitive ratio of $\mathtt{OA}(\phi^\gamma)$ *is* $O(\ln(\eta\alpha\theta))$.

A special case that is often discussed is when all dimensions have identical capacities. Then, the capacity ratio $\eta_k = M_k, \forall k \in [K]$, and $\mathtt{OA}(\phi^\gamma)$ achieves a competitive ratio $O(\ln(M\alpha\theta))$ with $M = \max_{k\in[K]} M_k$. Our proof of Theorem 4 is involved, but uses standard techniques to build on Theorem 1. It is given in Appendix B. We can also obtain the following lower bound in this setting.

Theorem 5. *There exists no online algorithm that can achieve a competitive ratio smaller than $\Omega(\ln(\eta\alpha\theta))$ for online multiple multi-dimensional knapsacks with departures.*

Theorem 5 includes Theorem 2 as a special case and can be proved based on existing lower bound results in the literature. In particular, the lower bounds of competitive ratios for two special cases of the general OKD have been proven: $\Omega(\ln(\eta\theta))$ (Theorem 2 in [29]) for online multi-dimensional knapsack and $\Omega(\ln\alpha)$ (Theorem 2 in [16]) for online interval scheduling. Thus, the competitive ratio of OKD is lower bounded by

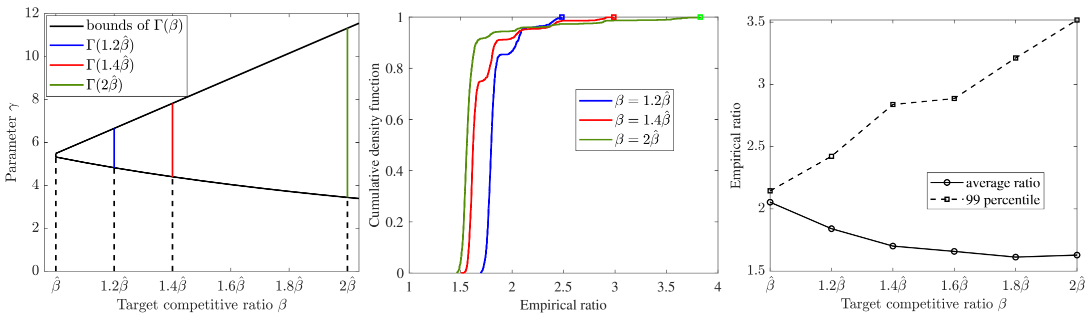
$$\mathtt{CR} \ge \max\{\Omega(\ln\alpha), \Omega(\ln(\eta\theta))\} \ge \frac{1}{2}\Omega(\ln\alpha) + \frac{1}{2}\Omega(\ln(\eta\theta)) = \Omega(\ln(\eta\alpha\theta)). \tag{19}$$

6 EXPERIMENTAL RESULTS

We focus our experimental results on evaluating the empirical performance of the data-driven online algorithm (DOA) in Algorithm 3. The worst-case optimized algorithm can be considered as a special case of DOA by choosing the same optimized threshold function over all instances.

We start our evaluation by demonstrating the essential trade-offs between average and worst-case performance in DOA. To do this, we first compare the algorithm with multiple baseline algorithms under a set of *hard instances* for OKD in §6.1. Then we show the performance of the algorithms under *typical instances* from real traces of the online cloud job scheduling in §6.2. Compared with benchmark algorithms in prior works and our worst-case optimized algorithm, DOA achieves significant improvement in the average performance at a moderate sacrifice of its worst-case guarantees under both hard and typical instances, and thus is of most practical use.

Experimental setup. We set a time horizon of $T = 3000$ slots. In each experiment, we test a total of $L = 1000$ instances, which are generated by 50 traces of item sequences (that include item arrival times, start times, and stay durations) and 20 random trials of item sizes and value densities for each trace. Each trace is either generated to capture hard instances for OKD in the worst case or sampled from real application traces. We refer to them as *hard instances* and *typical instances*, and detail how to generate them at the beginning of §6.1 and §6.2, respectively. We evaluate the performance of an online algorithm by its *empirical ratio*, which is defined as the ratio of rewards from the



(a) Parameter sets $\Gamma(\beta)$ (b) CDFs for different target ratios β (c) Average vs. worst performances

Fig. 3. Empirical performance of data-driven algorithms with different guaranteed competitive ratios.

offline algorithm to those of the online algorithm under a given instance. In each experiment, we sequentially compute the empirical ratios of L instances of different online algorithms, and report the average and 99 percentile of the empirical ratios to evaluate the average and worst-case performances of those algorithms.

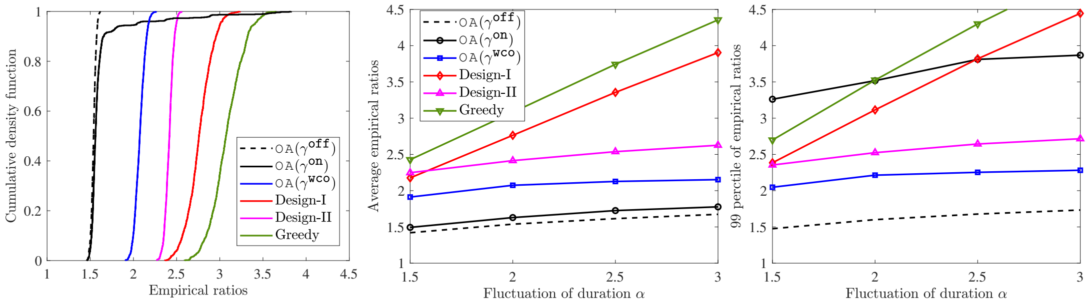
Implementation of DOA. Given a target competitive ratio guarantee β, we first determine the feasible parameter set $\Gamma(\beta)$ based on Theorem 3. The target ratio is set as a multiple of the reference value $\hat{\beta} = 10 + \frac{12}{\ln 2}\ln(\alpha\theta + 1)$. In the experiments, we implement a discretized version of DOA. In particular, we discretize $\Gamma(\beta)$ into $\tilde{\Gamma}(\beta)$ with a step size of 0.1 and let d denote the cardinality of $\tilde{\Gamma}(\beta)$. We refer to each $i \in [d]$ as an expert advice (which is equivalent to the selection of a threshold function). DOA can then be restated as a learning with expert advice problem. At the beginning of each round $\ell \in [L]$, we select an advice $i_\ell \in [d]$ and execute the instance using the selected threshold-based algorithm. After observing the whole instance, the rewards of all advices $\boldsymbol{r}_\ell := \{r_{\ell,i}\}_{i\in[d]}$ can be evaluated and used to determine the selection of the advice in next round.

In our experiments, the Hedge algorithm [14, 22] is used to determine the advice selection and it guarantees the regret $\mathbb{E}\left[\max_{i\in[d]}\sum_{\ell\in[L]} r_{\ell,i} - \sum_{\ell\in[L]} r_{\ell,i_\ell}\right] \le O(\sqrt{L\ln d})$, which is sublinear in the number of rounds. In Appendix D, we provide the details of DOA, including its implementation, regret guarantee, and limitations. In the rest of the paper, this DOA is denoted by $\mathsf{OA}(\gamma^{\mathsf{on}})$.

6.1 Performance of data-driven online algorithms

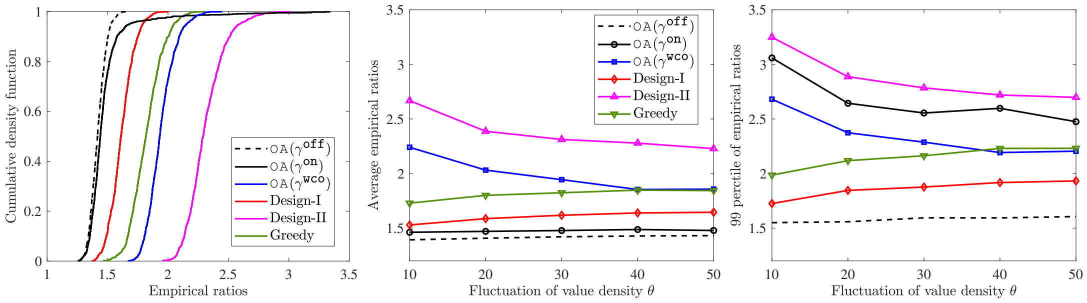
Hard instances for OKD. The hard instance has repeated patterns every $\underline{D}+\overline{D}$ slots. In each pattern, the hard instance consists of two batches of items. The first batch contains 50 items that have the shortest duration $\underline{D}$, fixed (normalized) item size 0.05 and value density randomly drawn from $[1,\theta]$. Then the second batch arrives and it has 50 items that have duration uniformly drawn from $[\underline{D},\overline{D}]$, fixed item size 0.05, fixed value density θ, and starting time that overlaps with the ending slot of the first batch of items. The value density ratio is set $\theta = 5$. The maximum duration is set $\overline{D} = 500$ and we test hard instances with varying α. Note that such instances are hard for OKD since any algorithm that aggressively admits items in the first batch can easily fill up the knapsack capacity in the first $\underline{D}$ slots of each pattern and declines the more valuable second batch of items. This hardness leads to the lower bound performance of Design-I as shown in Lemma 1. Our first set of empirical results are based on the *hard instances* that our algorithms are designed for.

6.1.1 Trade-offs between average and worst-case performances. We start by investigating the impact of the target ratio β on the performance of $\mathsf{OA}(\gamma^{\mathsf{on}})$. Note that β is an important hyper-parameter for $\mathsf{OA}(\gamma^{\mathsf{on}})$. Figure 3(a) illustrates the parameter sets when target ratios are set to $1.2\hat{\beta}$, $1.4\hat{\beta}$, and $2\hat{\beta}$



(a) CDF when $\theta = 5$ and $\alpha = 2$ (b) Average performance when $\theta = 5$ (c) Worst performance when $\theta = 5$

Fig. 4. Performance comparison of different algorithms under hard instances.



(a) CDF when $\theta = 30$ and $\alpha = 50$ (b) Average performance when $\alpha = 50$ (c) Worst performance when $\alpha = 50$

Fig. 5. Performance comparison of different algorithms under typical instances for cloud job scheduling.

(i.e., solid blue, red, and green lines). As we increase β, we tolerate a looser worst-case guarantee, in the meantime, we have a larger parameter set $\Gamma(\beta)$ for selecting γ, and thus a better chance to find the threshold function that can achieve a better average performance. Figure 3(b) shows the cumulative density functions (CDFs) of empirical ratios for different β that correspond to the parameter sets in Figure 3(a). We observe that a looser worst-case guarantee ($\beta = 2\hat{\beta}$) can give better empirical ratios for most of instances (more than 90%) but has a longer tail in the worst case as the cost. Figure 3(c) further compares the average and 99 percentile of the empirical ratios with varying β. The trade-off between average and worst-case performances can be clearly observed. In addition, we cannot always achieve a better average performance with a large β. As shown in Figure 3(c), the average ratio when $\beta = 1.8\hat{\beta}$ is smaller than that when $\beta = 2\hat{\beta}$. This is because the best parameter γ has already been included in $\Gamma(1.8\hat{\beta})$ and a further enlarged parameter set increases the difficulty of learning the best parameter, leading to a worse performance on average.

6.1.2 Performance comparisons with benchmark algorithms. Based on how γ is chosen, our proposed algorithm takes three forms: (i) **OA($\gamma^{\mathtt{on}}$)**, the DOA that selects γ based on the data-driven approach; (ii) **OA($\gamma^{\mathtt{wco}}$)**, the worst-case optimized algorithm that sets $\gamma^{\mathtt{wco}} = \ln(\alpha\theta+1)$ for all instances; (iii) **OA($\gamma^{\mathtt{off}}$)**, the average-case optimized algorithm that selects best possible static $\gamma^{\mathtt{off}}$ to minimize the average reward. Online determination of $\gamma^{\mathtt{off}}$ is impossible since it requires knowledge of all instances and thus this algorithm is just considered as a reference algorithm for OA($\gamma^{\mathtt{on}}$). We compare our proposed algorithms with three other benchmark algorithms. All of them correspond to OA(ϕ) with different threshold functions: (i) **Greedy** is a first-come-first-served algorithm that equivalently sets the threshold function to the smallest value density, i.e., $\phi(z) = 1, \forall z \in [0, C]$; (ii) **Design-I** adopts ϕ^I in Figure 1 and it is equivalent to the algorithm in [32] for the time-expanded `OKP`; (iii) **Design-II** adopts ϕ^{II} in Figure 1 that is used for `OMdK` in [29]. Generally speaking, these three algorithms are increasingly conservative due to their choice of threshold functions. The conservativeness of our worst-case optimized algorithm `OA`(γ^{wco}) lies between `Design-I` and `Design-II`.

Figure 4(a) compares the CDFs of empirical performances of different algorithms when $\theta = 5$ and $\alpha = 2$. Under the hard instances, a more conservative algorithm works better among the three benchmarks since such algorithms prohibit short-duration items from quickly occupying the knapsack capacities and blocking the followed long items. And the conservative design becomes increasingly more important as the duration ratio α increases, which is shown in Figures 4(b) and 4(c). Our worst-case optimized algorithm `OA`(γ^{wco}) outperforms all the benchmarks in both average and worst-case performances. Moreover, as α increases, the empirical ratio of `Design-I` grows linearly in α while that of `OA`(γ^{wco}) grows logarithmically, which is consistent with the lower bound results in Lemma 1 and Theorem 2. Compared with those algorithms with fixed threshold functions, our proposed data-driven algorithm `OA`(γ^{on}) achieves the best average performance, which is also close to that of the static offline benchmark `OA`(γ^{off}). This improvement in average performance is at the sacrifice of the worst-case performance as shown in Figure 4(c).

6.2 Trace-driven evaluation in online cloud job scheduling

To further validate the benefit of our proposed algorithms in real-world applications, we evaluate and compare the performances under typical instances for online cloud job scheduling.

Typical instances from cloud job traces. We extract traces of item sequences from the Google cluster traces [24]. One key feature from the cloud job traces is that there exist many short jobs and very few long jobs. To better show the comparisons, we set each time slot to be 10 seconds and restrict the duration of each job between 10 to 500 slots, i.e., any jobs that are shorter (longer) than 10 (500) are rounded to 10 (500). In this way, the duration ratio is fixed to $\alpha = 50$ for all typical instances. We consider a single server with one-dimension resource (e.g., CPU) and set its capacity to one. The resource requirement of each job is uniformly drawn from three possible values $w_n \in [0.01, 0.03, 0, 05]$ and the value of each job is set to $v_n = \xi_n d_n w_n$, where ξ_n is a uniform random variable within $[1, \theta]$. Then we evaluate the algorithms' average and worst-case performances when the value density ratio θ varies from 10 to 50.

Figure 5 illustrates the empirical performances of different algorithms under the typical instances. The behaviors of the algorithms with fixed threshold functions are very different from those under the hard instances. A more aggressive algorithm generally performs better except `Greedy`, while our worst-case optimized algorithm `OA`(γ^{wco}) is even worse than `Greedy`. This result is not unexpected since, even though the few elephant jobs are blocked due to capacity limitation, many mouse jobs can fill up the unused capacity, indicating an aggressive algorithm is already a good choice. Those typical instances are far from the hard instances and thus the algorithms optimized for the worst case cannot work well. In contrast, our data-driven algorithm `OA`(γ^{on}) adaptively adjusts the threshold function based on the instances and still outperforms all other benchmarks in the average performance at a moderate sacrifice of the worst-case performance.

7 RELATED WORK

The online knapsack problem. The fundamental difficulty of `OKP` is first shown by [23], which highlights that without additional assumptions on setup information, no competitive online algorithms can be designed. By assuming the value-to-size ratio of items is bounded within $[1, \theta]$

and the item size is infinitesimal, [33] first designs an online threshold-based algorithm for the (multiple) one-dimensional knapsack and shows it achieves an optimal competitive ratio $\Theta(\ln\theta)$. The follow-up work [32] extends the algorithm in [33] to the online multi-dimensional knapsack (OMdK) using the same threshold function and shows the competitive ratio is $O(M\ln\theta)$, where M is the number of dimensions. Then recent paper [29] redesigns the threshold function and obtains an order-optimal online algorithm for OMdK with a competitive ratio $\Theta(\ln(M\theta))$. The online knapsack with departures (OKD) problem is first approached by [32]. It treats each time slot in a T-slot horizon as one dimension of a knapsack and applies the result from OMdK that gives a competitive ratio $O(T\ln\theta)$. Based on the recent OMdK result in [29], this result can be further improved to $O(\ln(T\theta))$. However, all these results depend on the time horizon T, which can be a undesirable large value.

Online interval scheduling problem. The online interval scheduling (OIS) problem [22], (a.k.a., online reservation problem [16] in the operations research community) aims to schedule N jobs to S processors. Each job arrives at a random time and requires occupying one processor for a predetermined interval. Two jobs with overlapped intervals conflict with each other and must be placed on different processors. Upon arrival, each job informs the scheduler about its start time and stay duration, and the scheduler immediately assigns the job to one of the processors with no conflicts or declines it. The goal is to maximize the aggregate occupation of all processors over the time horizon. We note that OIS can be considered as a special case of OKD that has a single knapsack with capacity 1, fixed item size $1/S$, and unit value density. Based on the state-of-the-art result of OIS [16], if the duration of all jobs is bounded within $[\underline{D},\overline{D}]$ and the number of processors is large ($S\to\infty$), a randomized algorithm can achieve the optimal competitive ratio $\Theta(\ln\alpha)$, where $\alpha=\overline{D}/\underline{D}$ is the duration (fluctuation) ratio. This result basically provides a lower bound $\Omega(\ln\alpha)$ for OKD. However, it is unclear how to extend the algorithm in [16] to the general OKD problem. If we add the bounded duration assumption to the online threshold-based algorithms in [32] and [29], we can show their competitive ratios are lower bounded by $\Omega(\alpha\ln\theta)$ and $\Omega(\ln(\overline{D}\theta))$, respectively (see Lemma 1 for more details). Thus, there still exist no online algorithms in the literature that can achieve a competitive ratio $O(\ln(M\alpha\theta))$ for OKD.

Other variants to online knapsack problem. There exist many variants of the online knapsack problems for practical use. To capture the supply cost of using knapsack capacity (e.g., electricity cost), [18, 28] consider the online knapsack with supply cost, which is a convex function in the utilization of the knapsack. Some applications essentially have continuous decision variables (e.g., the online electric vehicle charging problem). The algorithms and results in OKP can be extended to continuous decisions under the infinitesimal item size assumption. Particularly, the classic one-way trading problem [12] can be considered as a continuous version of the basic OKP, and this connection has been extended to online fractional multiple knapsacks in [27].

Online primal-dual algorithms. Besides the online threshold-based algorithm (OTA), there exists an alternative class of online primal-dual algorithms (OPD) [7] that can potentially solve the online knapsack problems. The most relevant work is Buchbinder and Naor's paper [6] that designs OPD for a general online packing problem, which is equivalent to OMdK in the knapsack literature [29, 31]. However, there still exist algorithmic challenges in extending OPD to solve OKD. In particular, the key step of OPD is to design the dual variable update such that the increment ratio of dual and primal objectives is bounded, and both primal and (scaled) dual solutions are feasible. In addition, OPD produces a fractional solution and it needs a randomized rounding procedure to achieve the integral solution. It is unclear how to design the dual variable update and the randomized rounding in the setting of OKD with item departures and multiple knapsacks. Further, OPD in [6] replies on different assumptions and achieves different results compared to OTA (See the difference and connections in

Appendix C for more detail). This difference creates difficulties in extending OPD to solve OKD. Thus, it is non-trivial to extend OKD to the setting of OKD, but it is a promising future direction to explore.

Beyond the worst-case algorithms. Since the online algorithms that are designed for the worst case are usually too conservative for typical instances in practice, many works design beyond the worst-case algorithms for OKP and they usually make additional statistical assumptions on the input instance and present the problems as general resource allocation problems. For example, some works [1, 10] assume that items arrive in uniformly random order (i.e., random permutation model). Some others [5, 11, 21] assume that the item information is drawn i.i.d. from an unknown distribution (i.e., stochastic model). The algorithms from these works generally are designed to *first* learn the dual variables of the resource constraints by re-solving an optimization problem using past data [1, 21] or running online learning algorithms [5], and *then* determine the online allocation based on the learned dual variables. In addition, some recent works [13, 17] have also considered item departures in the general context of online allocation with reusable resources. However, the approaches and results in these works reply on the additional statistical assumptions, and thus are fundamentally different from our worst-case analysis in this paper.

8 CONCLUSIONS

This paper has designed an online algorithm that achieves order-optimal competitive ratios for the online multi-dimensional multiple knapsacks with departures problem. Our model and algorithms have generalized the state-of-the-art results in the online knapsack literature and opened the doors for real-world applications that require such full generality. From our trace-driven experiments, we have observed that the online algorithms that are optimized for the worst-case instances can be too conservative when faced with typical instances from real applications. To go beyond the worst case, we have further designed a data-driven online algorithm that can achieve the good performance under both worst-case and typical instances. Future works can further investigate how to improve the exact competitive ratios for online knapsack problems, instead of focusing only on the order-optimality. In addition, it is also interesting to explore how to provide a theoretical guarantee on the average-case performance of the original data-driven online algorithm.

ACKNOWLEDGMENTS

Bo Sun and Danny H.K. Tsang acknowledge the support received from the Hong Kong Research Grant Council (RGC) General Research Fund (Project 16202619 and Project 16211220). Adam Wierman acknowledges the support received from NSF grants (CNS-2146814, CPS-2136197, CNS-2106403, and NGSDI-210564) and the additional support from Amazon AWS. Mohammad Hajiesmaili's research is supported by NSF grants (CNS-2106299, CNS-2102963, CPS-2136199, NGSDI-2105494, and CAREER-2045641). The work of John C.S. Lui is supported in part by the RGC's SRFS2122-4S02.

REFERENCES

[1] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. 2014. A dynamic near-optimal algorithm for online linear programming. *Operations Research* 62, 4 (2014), 876–890.

[2] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32, 1 (2002), 48–77.

[3] Maria-Florina Balcan. 2020. Data-driven algorithm design. *arXiv preprint arXiv:2011.07177* (2020).

[4] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. 2018. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 603–614.

[5] Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. 2021. The Best of Many Worlds: Dual Mirror Descent for Online Allocation Problems. *Operations Research* (2021), forthcoming.

[6] Niv Buchbinder and Joseph Naor. 2009. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research* 34, 2 (2009), 270–286.

[7] Niv Buchbinder, Joseph Seffi Naor, et al. 2009. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science* 3, 2–3 (2009), 93–263.

[8] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.

[9] Vincent Cohen-Addad and Varun Kanade. 2017. Online optimization of smoothed piecewise constant functions. In *Artificial Intelligence and Statistics*. PMLR, 412–420.

[10] Nikhil R Devanur and Thomas P Hayes. 2009. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*. 71–78.

[11] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. 2019. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *Journal of the ACM (JACM)* 66, 1 (2019), 1–41.

[12] Ran El-Yaniv, Amos Fiat, Richard M Karp, and Gordon Turpin. 2001. Optimal search and one-way trading online algorithms. *Algorithmica* 30, 1 (2001), 101–139.

[13] Matthew Faw, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. 2022. Learning To Maximize Welfare with a Reusable Resource. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 2 (2022), 1–30.

[14] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[15] Lingwen Gan, Ufuk Topcu, and Steven H Low. 2012. Stochastic distributed protocol for electric vehicle charging with discrete charging rate. In *2012 IEEE Power and Energy Society General Meeting*. IEEE, 1–8.

[16] Shashank Goyal and Diwakar Gupta. 2020. The Online Reservation Problem. *Algorithms* 13, 10 (2020), 241.

[17] Vineet Goyal, Garud Iyengar, and Rajan Udwani. 2021. Asymptotically Optimal Competitive Ratio for Online Allocation of Reusable Resources. In *Web and Internet Economics - 17th International Conference, WINE 2021, Potsdam, Germany, December 14-17, 2021, Proceedings*, Vol. 13112. Springer, 543.

[18] Zhiyi Huang and Anthony Kim. 2019. Welfare maximization with production costs: A primal dual approach. *Games and Economic Behavior* 118 (2019), 648–667.

[19] Devansh Jalota, Dario Paccagnan, Maximilian Schiffer, and Marco Pavone. 2021. Online Traffic Routing: Deterministic Limits and Data-driven Enhancements. *arXiv preprint arXiv:2109.08706* (2021).

[20] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. Multiple knapsack problems. In *Knapsack Problems*. Springer, 285–316.

[21] Xiaocheng Li and Yinyu Ye. 2021. Online linear programming: Dual convergence, new algorithms, and regret bounds. *Operations Research* (2021).

[22] Richard J Lipton and Andrew Tomkins. 1994. Online Interval Scheduling. In *SODA*, Vol. 94. 302–311.

[23] Alberto Marchetti-Spaccamela and Carlo Vercellis. 1995. Stochastic on-line knapsack problems. *Mathematical Programming* 68, 1 (1995), 73–104.

[24] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. 2012. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the third ACM symposium on cloud computing*. 1–13.

[25] Vincenzo Sciancalepore, Konstantinos Samdanis, Xavier Costa-Perez, Dario Bega, Marco Gramaglia, and Albert Banchs. 2017. Mobile traffic forecasting for maximizing 5G network slicing resource utilization. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[26] Clifford Stein, Van-Anh Truong, and Xinshang Wang. 2020. Advance service reservations with heterogeneous customers. *Management Science* 66, 7 (2020), 2929–2950.

[27] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny H.K. Tsang. 2020. Competitive Algorithms for the Online Multiple Knapsack Problem with Application to Electric Vehicle Charging. *Proc. ACM Meas.*

Anal. Comput. Syst. 4, 3, Article 51 (Nov. 2020), 32 pages.

[28] Xiaoqi Tan, Bo Sun, Alberto Leon-Garcia, Yuan Wu, and Danny HK Tsang. 2020. Mechanism design for online resource allocation: A unified approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 2 (2020), 1–46.

[29] Lin Yang, Ali Zeynali, Mohammad H Hajiesmaili, Ramesh K Sitaraman, and Don Towsley. 2021. Competitive Algorithms for Online Multidimensional Knapsack Problems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–30.

[30] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. 2021. Data-driven competitive algorithms for online knapsack and set cover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10833–10841.

[31] Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis CM Lau. 2015. Online auctions in IaaS clouds: Welfare and profit maximization with server costs. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. 3–15.

[32] Zijun Zhang, Zongpeng Li, and Chuan Wu. 2017. Optimal Posted Prices for Online Cloud Resource Allocation. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1 (June 2017), 26 pages.

[33] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*. Springer, 566–576.

A PROOF OF LEMMA 1

To show the lower bounds of $\texttt{OA}(\phi^I)$ and $\texttt{OA}(\phi^{II})$, we construct two instances (one capacity-limited instance for Design-I and one capacity-free instance for Design-II), and show that $\texttt{CR}(\texttt{OA}(\phi^I))$ and $\texttt{CR}(\texttt{OA}(\phi^{II}))$ are at least $\Omega(\alpha \ln \theta)$ and $\Omega(\ln \overline{D}\theta)$ under the two instances, respectively.

Lower bound of $\texttt{OA}(\phi^I)$. Consider a capacity-limited instance that consists of two groups of items. Items in both groups have the same infinitesimal size w and starting time (e.g., slot 1). Each group has N items such that the capacity can be fully filled by either group, i.e., $Nw = C$. Group-1 items arrive first. Each item n requests to stay for $\underline{D}$ slots and its item value is equal to the threshold value of $\texttt{OA}(\phi^I)$ upon its arrival, i.e., $v_n = w \sum_{t\in[\underline{D}]} \phi^I(z_t^{(n-1)}) = w\underline{D}\phi^I(z_1^{(n-1)})$. Group-2 items arrive after Group-1, and each of its items requests to stay for $\overline{D}$ slots and has the same value $w\overline{D}\theta$.

Under above instance, $\texttt{OA}(\phi^I)$ admits all items in Group-1 that can fill up the capacity of the knapsack in the first $\underline{D}$ slots and declines all Group-2 items due to capacity limits. Then the online values obtained by $\texttt{OA}(\phi^I)$ is

$$\texttt{ALG}^I = \sum_{n=1}^{N} v_n = \underline{D} \sum_{n=1}^{N} w\phi^I(z_1^{(n-1)}) \approx \underline{D} \int_0^C \phi^I(u)du = \frac{\underline{D}C\theta}{1+\ln\theta}. \tag{20}$$

The offline algorithm will only admit Group-2 items and achieve the optimal value $\texttt{OPT}^I = \overline{D}C\theta$. Thus, the competitive ratio of $\texttt{OA}(\phi^I)$ is at least

$$\texttt{CR}(\texttt{OA}(\phi^I)) \geq \frac{\texttt{OPT}^I}{\texttt{ALG}^I} = \alpha(1+\ln\theta) = \Omega(\alpha\ln\theta). \tag{21}$$

Lower bound of $\texttt{OA}(\phi^{II})$. Consider a capacity-free instance that has N identical items. Each item has unit value density 1, fixed infinitesimal size w, starting time in slot 1, and stay duration $\underline{D}$. The total size of all items can fill up the capacity C, i.e., $Nw = C$.

Under above instance, $\texttt{OA}(\phi^{II})$ admits items up to utilization

$$z' = \operatorname*{arg\,max}_{z\in[0,C]:\phi^{II}(z)\leq 1} \quad z = \frac{2C}{\log(\overline{D}\theta)}, \tag{22}$$

and achieves a total value $\texttt{ALG}^{II} = \underline{D}z' = 2C\underline{D}/(\log(\overline{D}\theta))$. The offline algorithm will admit all items and achieve the optimal value $\texttt{OPT}^{II} = C\underline{D}$. Thus, the competitive ratio of $\texttt{OA}(\phi^{II})$ is at least

$$\texttt{CR}(\texttt{OA}(\phi^{II})) \geq \frac{\texttt{OPT}^{II}}{\texttt{ALG}^{II}} = \frac{\log(\overline{D}\theta)}{2} = \Omega(\ln\overline{D}\theta). \tag{23}$$

B PROOF OF THEOREM 4

Compared to the proof of Theorem 1, we note Lemma 2 still holds in the multi-dimensional setting. Therefore, to prove Theorem 4, we only need to prove the multi-dimensional counterpart of Lemma 3, which bounds the competitive ratio $\tilde{\texttt{CR}}_k$ of each ancillary problem $\texttt{OKD}_k$, which is a single multi-dimensional knapsack with departures. Thus, we next prove the following lemma.

Lemma 4. *Under Assumptions 2, 4, and 5, if the threshold function $\phi_{km}^{\gamma_k}$ is given by Equation (18) with $\gamma_k \in (\ln 2, +\infty)$ and the item size is upper bounded by $\varepsilon_{km} \leq C_{km} \ln 2/\gamma_k, \forall k \in [K], m \in [M_k]$, the competitive ratio of $\texttt{OTA}_k$ is*

$$\tilde{\texttt{CR}}_k(\gamma_k) = 3 \cdot \max\left\{1 + \frac{2}{\ln 2}\gamma_k, \frac{2}{\ln 2} \cdot \frac{\alpha_k\eta_k\theta_k\gamma_k}{\exp((\gamma_k - \ln 2)/2) - 1}\right\}. \tag{24}$$

We can apply the same partitioning procedure and then focus on analyzing the worst-case ratio of $\texttt{OPT}(\mathcal{N}^h)/\texttt{ALG}(\tilde{\mathcal{N}}^h), \forall h \in [H]$. Consider the following two cases.

Case I: capacity-free case. The final utilizations of all dimensions in $[M]$ across all time slots in $\hat{\mathcal{T}}^h := \mathcal{T}^h \cup \mathcal{T}^{h+1}$ are far from reaching the capacity, i.e., $z_{mt}^{(N)} \leq C_m - \varepsilon_m, \forall m \in [M], t \in \hat{\mathcal{T}}^h$.

In this case, the only reason why one item is rejected by $\texttt{OTA}$ is that it fails to pass the *threshold check*. Then $\texttt{OPT}(\mathcal{N}^h)$ and $\texttt{ALG}(\tilde{\mathcal{N}}^h)$ can be connected via the final utilization $\{z_{mt}^{(N)}\}_{m\in[M],t\in\hat{\mathcal{T}}^h}$. We first show that $\texttt{ALG}(\tilde{\mathcal{N}}^h)$ is lower bounded.

Proposition 4. *In Case I, the value of items in* $\tilde{N}^h$ *admitted by* $\texttt{OTA}$ *is lower bounded by*

$$\texttt{ALG}(\tilde{\mathcal{N}}^h) \geq \frac{\ln 2}{2\gamma} \sum\nolimits_{t\in\hat{\mathcal{T}}^h} \sum\nolimits_{m\in[M]} \phi_m(z_{mt}^{(N)})C_m. \tag{25}$$

Proof. Since $\phi_m(0) = 0$, we can have

$$\begin{aligned}\sum\nolimits_{t\in\hat{\mathcal{T}}^h} \sum\nolimits_{m\in[M]} \phi_m(z_{mt}^{(N)})C_m &= \sum\nolimits_{t\in\hat{\mathcal{T}}^h} \sum\nolimits_{m\in[M]} \sum\nolimits_{n\in\mathcal{N}} C_m[\phi_m(z_{mt}^{(n)}) - \phi(z_{mt}^{(n-1)})] \\ &= \sum\nolimits_{t\in\hat{\mathcal{T}}^h} \sum\nolimits_{m\in[M]} \sum\nolimits_{n\in\tilde{\mathcal{N}}^h} C_m[\phi_m(z_{mt}^{(n)}) - \phi(z_{mt}^{(n-1)})] \\ &\leq \sum\nolimits_{n\in\tilde{\mathcal{N}}^h} \sum\nolimits_{t\in\mathcal{T}_n} \sum\nolimits_{m\in[M]} C_m[\phi_m(z_{mt}^{(n)}) - \phi(z_{mt}^{(n-1)})]. \end{aligned} \tag{26}$$

Let $\Delta\texttt{ALG}_n$ denote the increment of $\texttt{OTA}$ due to processing item n. Thus, $\Delta\texttt{ALG}_n = 0$ if item n is declined and $\Delta\texttt{ALG}_n = v_n$ if it is admitted. We next show $\sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} C_m[\phi_m(z_{mt}^{(n)}) - \phi(z_{mt}^{(n-1)})] \leq (2\gamma/\ln 2)\Delta\texttt{ALG}_n, \forall n \in \tilde{\mathcal{N}}^h$ in the following two sub-cases.

Case I(a). When item n is declined by $\texttt{OTA}$, we have $z_{mt}^{(n)} = z_{mt}^{(n-1)}, \forall t \in \mathcal{T}_n, m \in [M]$ and thus

$$\sum\nolimits_{t\in\mathcal{T}_n} \sum\nolimits_{m\in[M]} C_m[\phi_m(z_{mt}^{(n)}) - \phi_m(z_{mt}^{(n-1)})] = 0 \leq \frac{2\gamma}{\ln 2}\Delta\texttt{ALG}_n. \tag{27}$$

Case I(b). When item n is admitted by $\texttt{OTA}$, we have $z_{mt}^{(n)} = z_{mt}^{(n-1)} + w_{nm}, \forall t \in \mathcal{T}_n, m \in [M]$, and

$$\sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} C_m[\phi_m(z_{mt}^{(n)}) - \phi_m(z_{mt}^{(n-1)})]$$

$$= \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} C_m \left[\exp\left(\frac{(z_{mt}^{(n-1)} + w_{nm})\gamma}{C_m}\right) - \exp\left(\frac{z_{mt}^{(n-1)}\gamma}{C_m}\right)\right] \tag{28a}$$

$$= \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} C_m \exp\left(\frac{z_{mt}^{(n-1)}\gamma}{C_m}\right)\left[\exp\left(\frac{w_{nm}\gamma}{C_m}\right) - 1\right] \tag{28b}$$

$$\leq \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} C_m \exp\left(\frac{z_{mt}^{(n-1)}\gamma}{C_m}\right) \cdot \frac{w_{nm}\gamma}{\ln 2 C_m} \tag{28c}$$

$$= \frac{\gamma}{\ln 2} \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} w_{nm} \exp\left[\left(\frac{z_{mt}^{(n-1)}\gamma}{C_m}\right) - 1\right] + \frac{\gamma}{\ln 2} \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} w_{nm} \tag{28d}$$

$$= \frac{\gamma}{\ln 2} \sum_{t\in\mathcal{T}_n} \sum_{m\in[M]} w_{nm}\phi_m(z_{mt}^{(n-1)}) + \frac{\gamma}{\ln 2} d_n \sum_{m\in[M]} w_{nm} \tag{28e}$$

$$\leq \frac{\gamma}{\ln 2} v_n + \frac{\gamma}{\ln 2} v_n = \frac{2\gamma}{\ln 2}\Delta\texttt{ALG}_n. \tag{28f}$$

Inequality (28c) holds since $\frac{w_{nm}\gamma}{C_m \ln 2} \leq \frac{\varepsilon_m \gamma}{C_m \ln 2} \leq 1$ based on the additional condition on the item size in Theorem 4. The last inequality (28f) is due to the decision rule in the multi-dimensional setting $\sum_{t\in\mathcal{T}_n}\sum_{m\in[M]} w_{nm}\phi_m(z_{mt}^{(n-1)}) \leq v_n$ when item n can pass the threshold check, and $d_n \sum_{m\in[M]} w_{nm} \leq v_n$ is based on Assumption 4.

Combining above equations gives

$$\sum\nolimits_{t\in\hat{\mathcal{T}}^h}\sum\nolimits_{m\in[M]}\phi_m(z_{mt}^{(N)})C_m \leq \sum\nolimits_{n\in\tilde{\mathcal{N}}^h}\frac{2\gamma}{\ln 2}\Delta\mathsf{ALG}_n = \frac{2\gamma}{\ln 2}\mathsf{ALG}(\tilde{\mathcal{N}}^h),$$

which completes the proof. □

Next, we show the offline optimal value of items in $\mathcal{N}^h$ is upper bounded.

Proposition 5. *In Case I, the value of items in $\mathcal{N}^h$ admitted by the offline algorithm is upper bounded by*

$$\mathsf{OPT}(\mathcal{N}^h) \leq \mathsf{ALG}(\mathcal{N}^h) + \sum\nolimits_{t\in\hat{\mathcal{T}}^h}\sum\nolimits_{m\in[M]}\phi_m(z_{mt}^{(N)})C_m. \tag{29}$$

Proof. Let $\mathcal{S}^h \cap \mathcal{S}^{h*}$ denote the set of items in $\mathcal{N}^h$ that are admitted by both online algorithm OTA and the offline algorithm and $\mathcal{S}^{h*} \setminus \mathcal{S}^h$ denote the set of items that are declined by OTA but admitted by the offline algorithm. We have

$$\sum\nolimits_{n\in\mathcal{S}^h\cap\mathcal{S}^{h*}} v_n \leq \mathsf{ALG}(\mathcal{N}^h), \tag{30}$$

$$\begin{aligned}\sum\nolimits_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h} v_n &\leq \sum\nolimits_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h}\sum\nolimits_{t\in\mathcal{T}_n}\sum\nolimits_{m\in[M]} w_{nm}\phi_m(z_{mt}^{(n-1)}) \\ &\leq \sum\nolimits_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h}\sum\nolimits_{t\in\mathcal{T}_n}\sum\nolimits_{m\in[M]} w_{nm}\phi_m(z_{mt}^{(N)}) \\ &= \sum\nolimits_{t\in\hat{\mathcal{T}}^h}\sum\nolimits_{m\in[M]}\phi_m(z_{mt}^{(N)})\sum\nolimits_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h:t\in\mathcal{T}_n} w_{nm} \\ &\leq \sum\nolimits_{t\in\hat{\mathcal{T}}^h}\sum\nolimits_{m\in[M]}\phi_m(z_{mt}^{(N)})C_m,\end{aligned} \tag{31}$$

where the first inequality in (31) holds since the item fails to pass the threshold check, and the last inequality holds since the items admitted by the offline algorithm cannot exceed the knapsack capacity in any dimension, i.e., $\sum_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h:t\in\mathcal{T}_n} w_{nm} \leq C_m, \forall m \in [M]$.

Thus, we have

$$\mathsf{OPT}(\mathcal{N}^h) = \sum\nolimits_{n\in\mathcal{S}^h\cap\mathcal{S}^{h*}} v_n + \sum\nolimits_{n\in\mathcal{S}^{h*}\setminus\mathcal{S}^h} v_n \leq \mathsf{ALG}(\mathcal{N}^h) + \sum\nolimits_{t\in\hat{\mathcal{T}}^h}\sum\nolimits_{m\in[M]}\phi_m(z_{mt}^{(N)})C_m, \tag{32}$$

which completes the proof. □

Combining Proposition 4 and Proposition 5 gives

$$\frac{\mathsf{OPT}(\mathcal{N}^h)}{\mathsf{ALG}(\tilde{\mathcal{N}}^h)} \leq \frac{\mathsf{ALG}(\mathcal{N}^h) + \sum_{t\in\hat{\mathcal{T}}^h}\sum_{m\in[M]}\phi_m(z_{mt}^{(N)})C_m}{\mathsf{ALG}(\tilde{\mathcal{N}}^h)} \leq 1 + \frac{2}{\ln 2}\gamma. \tag{33}$$

Case II: capacity-limited case. There exists at least one time slot $t' \in \hat{\mathcal{T}}^h$ whose utilization in one dimension $m' \in [M]$ approaches the knapsack capacity, i.e., $C_{m'} - \varepsilon_{m'} < z_{m't'}^{(N)} \leq C_{m'}$.

Proposition 6. *In Case II, the value of items in $\tilde{\mathcal{N}}^h$ admitted by* OTA *is lower bounded by*

$$\mathsf{ALG}(\tilde{\mathcal{N}}^h) \geq \frac{\ln 2 C_{m'} \underline{D}}{\gamma}\left[\exp\left(\frac{\gamma - \ln 2}{2}\right) - 1\right], \tag{34}$$

where $m' = \arg\min_{m\in[M]} C_m$.

PROOF. In this case, if there exists one slot (say t') whose utilization in one dimension (say m') approaches the capacity, then the final utilization of dimension m' in the worst-case is also in the pattern illustrated in Figure 2. Then based on Proposition 4, $\mathsf{ALG}(\tilde{\mathcal{N}}^h)$ is lower bounded by

$$\mathsf{ALG}(\tilde{\mathcal{N}}^h) \geq \frac{\ln 2}{2\gamma}\left[(2\underline{D}-2)\phi_{m'}\left(\frac{C_{m'}-\varepsilon_{m'}}{2}\right)+\phi_{m'}(C_{m'}-\varepsilon_{m'})\right]C_{m'} \tag{35a}$$

$$\geq \frac{\ln 2 C_{m'}\underline{D}}{\gamma}\phi_{m'}\left(\frac{C_{m'}-\varepsilon_{m'}}{2}\right) \tag{35b}$$

$$\geq \frac{\ln 2 C_{m'}\underline{D}}{\gamma}\left[\exp\left(\frac{\gamma-\ln 2}{2}\right)-1\right], \tag{35c}$$

which is minimized when $m' = \arg\min_{m\in[M]} C_m$. This completes the proof. □

In Case II, since the value density is upper bounded by θ. The total value of $2\overline{D}$ slots with total size of items $\sum_{m\in[M]} C_m$ is thus upper bounded by

$$\mathsf{OPT}(\mathcal{N}^h) \leq 2\theta\overline{D}\sum_{m\in[M]} C_m. \tag{36}$$

Thus, in Case II, we have

$$\frac{\mathsf{OPT}(\mathcal{N}^h)}{\mathsf{ALG}(\tilde{\mathcal{N}}^h)} \leq \frac{2}{\ln 2}\cdot\frac{\eta\alpha\theta\gamma}{\exp((\gamma-\ln 2)/2)-1}, \tag{37}$$

where $\eta = \sum_{m\in[M]} C_m/(\min_{m\in[M]} C_m)$.

Summarizing the results from the two cases, the competitive ratio of OTA is

$$\tilde{\mathsf{CR}}(\gamma) = 3\max\left\{1+\frac{2}{\ln 2}\gamma, \frac{2\eta\alpha\theta}{\ln 2}\cdot\frac{\gamma}{\exp((\gamma-\ln 2)/2)-1}\right\}. \tag{38}$$

This completes the proof of Lemma 4.

Based on Lemma 4, we can choose $\hat{\gamma}_k = 2\ln(\eta_k\alpha_k\theta_k+1)+\ln 2, \forall k\in[K]$ and this gives

$$\tilde{\mathsf{CR}}_k(\hat{\gamma}_k) = \max\left\{9+\frac{12}{\ln 2}\ln(\eta_k\alpha_k\theta_k+1), 6+\frac{12}{\ln 2}\ln(\eta_k\alpha_k\theta_k+1)\right\} = O(\ln(\alpha_k\theta_k)), \forall k\in[K]. \tag{39}$$

Then the competitive ratio of $\mathsf{OA}(\phi^\gamma)$ is $\tilde{\mathsf{CR}} = 1+\max_{k\in[K]}\tilde{\mathsf{CR}}_k(\hat{\gamma}_k) = O(\ln(\eta\alpha\theta))$.

C THRESHOLD-BASED ALGORITHMS VS. ONLINE PRIMAL-DUAL ALGORITHMS

The online primal-dual algorithm (OPD) and the online threshold-based algorithm (OTA) rely on different assumptions and achieve different competitive results. In the following, we show the differences and connections between the two algorithms.

The online packing problem, or the online multi-dimensional knapsack problem (OMdK), solved by OPD in [6] is a special case of OKD and can be formulated as a linear program

$$\max_{x_n} \quad \sum_{n\in[N]} v_n x_n, \tag{40a}$$

$$\text{s.t.} \quad \sum_{n\in[N]} w_{nm}x_n \leq C_m, \forall m\in[M], \tag{40b}$$

$$x_n \in \{0,1\}, \forall n\in[N], \tag{40c}$$

where $m\in[M]$ is the index for knapsack dimension and $n\in[N]$ is the index for items. In the online knapsack literature [29, 31], we commonly assume the value density is bounded $v_n/(\sum_{m\in[M]} w_{nm}) \in [1,\theta]$ since the value density is usually the most critical parameter in knapsack problems regardless of online or offline settings. θ can be considered as the ratio of the maximum to minimum value density. For clear comparison, we present the threshold-based algorithm for the OMdK in Algorithm 4.

Algorithm 4 Online Threshold-based Algorithm for Online Multi-dimensional Knapsack

input: threshold function $\phi := \{\phi_m(\cdot)\}_{m\in[M]}$, capacity $\{C_m\}_{m\in[M]}$;
output: admission decision x_n;
initialization: utilization $z_m^{(0)} = 0, \forall m \in [M]$;
for item $n = 1, \ldots, N$ **do**
  observe item value v_n and size $\{w_{nm}\}_{m\in[M]}$;
  determine a threshold value $\Phi = \sum_{m\in[M]} w_{nm}\phi_m(z_m^{(n-1)})$;
  if $v_n > \Phi$ and $z_m^{(n-1)} + w_{nm} \le C_m, \forall m \in [M]$ **then**
    admit the item and set $x_n = 1$;
  else
    decline the item and set $x_n = 0$.
  end if
  update utilization $z_m^{(n)} = z_m^{(n-1)} + w_{nm}x_n, \forall m \in [M]$.
end for

This algorithm can achieve the competitive ratio $O(\ln(\eta\theta))$, where $\eta = \sum_{m\in[M]} C_m / \min_{m\in[M]} C_m$ is the capacity variation. Its proof is similar to that of Theorem 4 by omitting the partitioning procedure. For completeness, the result can be summarized as the following lemma.

Lemma 5. *Under Assumptions 4 and 5, there exists $\gamma = O(\ln(\eta\theta))$, if the item size is upper bounded by $\varepsilon_m \le C_m \ln 2/\gamma, \forall m \in [M]$, and the threshold function $\phi^\gamma = \{\phi_m^\gamma\}_{m\in[M]}$ is given by*

$$\phi_m(z) = \exp(z\gamma/C_m) - 1, \forall m \in [M], \tag{41}$$

then the competitive ratio of Algorithm 4 is $O(\ln(\eta\theta))$.

In the online packing problem, the value of each item is usually set to be identical. For example, in the online virtual circuits routing problem (See §5.2 in [6]), the value of each virtual circuit is identical and equal to 1, i.e., $v_n = 1, \forall n \in [N]$. In this problem, the most crucial parameters are $w_m^{\max} = \max_{n\in[N]} w_{nm}$ and $w_m^{\min} = \min_{n\in[N]:w_{nm}\neq 0} w_{nm}$ that represent the maximum and minimal item size in each dimension. The OPD can achieve a competitive ratio of $O(\log M + \log \max_{m\in[M]} \frac{w_m^{\max}}{w_m^{\min}})$ (See Theorem 3.1 in [6]).

Although relying on different assumptions, OPD and OTA can be connected. We can in fact apply OTA to the setting of the OPD and achieve a slightly better competitive ratio guarantee. First, we normalize the item size of each dimension by its capacity and consider w_{nm}/C_m as the size of item n in dimension m. In this way, the capacities of all dimensions are normalized to 1. Then the capacity variation becomes $\eta = M$ and the value density becomes $v_n / (\sum_{m\in[M]} w_{nm}/C_m)$. In addition, the value density is bounded from below and above by

$$\frac{v_n}{\sum_{m\in[M]} w_{nm}/C_m} \le \frac{v_n}{\sum_{m\in[M]} \min_{n\in[N]:w_{nm}\neq 0} w_{nm}/C_m} = \frac{v_n}{\sum_{m\in[M]} w_m^{\min}/C_m}, \tag{42}$$

$$\frac{v_n}{\sum_{m\in[M]} w_{nm}/C_m} \ge \frac{v_n}{\sum_{m\in[M]} \max_{n\in[N]} w_{nm}/C_m} = \frac{v_n}{\sum_{m\in[M]} w_m^{\max}/C_m}. \tag{43}$$

Therefore, the variation of value density is upper bounded by

$$\theta = \frac{\frac{v_n}{\sum_{m\in[M]} w_m^{\min}/C_m}}{\frac{v_n}{\sum_{m\in[M]} w_m^{\max}/C_m}} = \frac{\sum_{m\in[M]} w_m^{\max}/C_m}{\sum_{m\in[M]} w_m^{\min}/C_m} \le \max_{m\in[M]} \frac{w_m^{\max}}{w_m^{\min}}. \tag{44}$$

Thus, OTA can achieve a competitive ratio of $O(\ln(M\theta))$ that is better than the ratio $O(\log M + \log \max_{m\in[M]} \frac{w_m^{\max}}{w_m^{\min}})$ achieved by the OPD.

In addition, OTA can achieve the same result as the OPD in a special case of the online packing problem with sparse demand requests. In the special case, the values of all items are identical, i.e., $v_n = v, \forall n \in [N]$ and all item sizes are binary, i.e., $w_{nm} \in \{0, 1\}$. Let $\Lambda = \max_{n\in[N]} \sum_{m\in[M]} w_{nm}$ denote the sparsity parameter, i.e., maximum number of dimensions used by each item. The OPD in [6] achieves a competitive ratio of $O(\ln \Lambda)$ when the capacity is large enough $C_m \geq O(\ln \Lambda), \forall m \in [M]$ based on the results of Theorem 3.2 and Lemma 5.4 in [6] for the integral version of the online packing. We can show that Algorithm 4 can also achieve a competitive ratio of $O(\ln \Lambda)$.

Lemma 6. *If all item sizes are binary $w_{nm} \in \{0, 1\}$ and item size is small compared to capacity with $C_m \geq \gamma/\ln 2$, the competitive ratio of Algorithm 4 is $O(\ln \Lambda)$ when the threshold function is*

$$\phi_m(z) = \frac{v}{\Lambda}[\exp(z\gamma/C_m) - 1], \forall m \in [M], \tag{45}$$

where $\gamma = \ln(\Lambda + 1)$.

Proof of Lemma 6. The proof is similar to that of the capacity-free case in Lemma 3. There is no capacity-limited case because the threshold check in Algorithm 4 can already guarantee no violations of the capacity constraints. To be precise, suppose one dimension m' reaches the capacity C_m after processing item n'. Then any item n that comes after n' and requests dimension m' (i.e., $w_{nm'} = 1$) will face a threshold value in Algorithm 4

$$\Phi = \sum\nolimits_{m\in[M]} w_{nm}\phi_m(z_m^{(n-1)}) \geq w_{nm'}\phi_{m'}(z_{m'}^{(n-1)}) = \phi_{m'}(C_{m'}) = v. \tag{46}$$

Therefore, the item n cannot pass the threshold check and the capacity violation is avoided. We next show that values obtained by Algorithm 4 and offline algorithm under the same instance $\mathcal{I}$ are lower bounded and upper bounded, respectively.

The value of admitted items by Algorithm 4 is lower bounded by

$$\mathsf{ALG}(\mathcal{I}) \geq \frac{\ln 2}{2\gamma} \sum\nolimits_{m\in[M]} \phi_m(z_m^{(N)})C_m, \tag{47}$$

where $z_m^{(N)}$ is the final utilization of dimension m after processing the N items in $\mathcal{I}$.

Since $\phi_m(0) = 0, \forall m \in [M]$, we have

$$\sum\nolimits_{m\in[M]} \phi_m(z_m^{(N)})C_m = \sum\nolimits_{n\in[N]} \sum\nolimits_{m\in[M]} [\phi_m(z_m^{(n)}) - \phi_m(z_m^{(n-1)})]C_m. \tag{48}$$

Next we show that $\sum_{m\in[M]} [\phi_m(z_m^{(n)}) - \phi_m(z_m^{(n-1)})]C_m \leq \frac{2\gamma}{\ln 2}\Delta\mathsf{ALG}_n$, where $\Delta\mathsf{ALG}_n$ is the increment of Algorithm 4 by processing item n. $\Delta\mathsf{ALG}_n = v$ if item n is admitted and $\Delta\mathsf{ALG}_n = 0$ otherwise.

Case I. When item n is declined, we have $z_m^{(n)} = z_m^{(n-1)}, \forall m \in [M]$, and thus

$$\sum\nolimits_{m\in[M]} [\phi_m(z_m^{(n)}) - \phi_m(z_m^{(n-1)})]C_m = 0 \leq \frac{2\gamma}{\ln 2}\Delta\mathsf{ALG}_n. \tag{49}$$

Case II. When item n is admitted, we have $z_m^{(n)} = z_m^{(n-1)} + w_{nm}, \forall m \in [M]$, and thus

$$\sum_{m\in[M]} [\phi_m(z_m^{(n)}) - \phi_m(z_m^{(n-1)})]C_m = \sum_{m\in[M]} \frac{vC_m}{\Lambda} \exp(z_m^{(n-1)}\gamma/C_m)[\exp(w_{nm}\gamma/C_m) - 1] \tag{50a}$$

$$\leq \sum_{m\in[M]} \frac{vC_m}{\Lambda} \exp(z_m^{(n-1)}\gamma/C_m)\frac{w_{nm}\gamma}{C_m \ln 2} \tag{50b}$$

$$= \frac{\gamma}{\ln 2} \sum_{m\in[M]} w_{nm}\phi_m(z_m^{(n-1)}) + \frac{v\gamma}{\Lambda \ln 2} \sum_{m\in[M]} w_{nm} \tag{50c}$$

$$\leq \frac{\gamma}{\ln 2} \cdot v + \frac{v\gamma}{\Lambda \ln 2} \cdot \Lambda = \frac{2\gamma}{\ln 2} \Delta \mathtt{ALG}_n, \tag{50d}$$

where Equation (50b) holds since $w_{nm}\gamma/(C_m \ln 2) \leq 1$ based on the assumption, and Equation (50d) holds because $\sum_{m\in[M]} w_{nm}\phi_m(z_m^{(n-1)}) \leq v$ if item n is admitted, and the maximum number of non-zero dimension is Λ, i.e., $\sum_{m\in[M]} w_{nm} \leq \Lambda$.

Thus, we can have $\mathtt{ALG}(\mathcal{I}) = \sum_{n\in[N]} \Delta\mathtt{ALG}_n \geq \frac{\ln 2}{2\gamma} \sum_{m\in[M]} \phi_m(z_m^{(N)})C_m$.

Based on the same arguments as Proposition 2, we can also show the value of offline algorithm is upper bounded by

$$\mathtt{OPT}(\mathcal{I}) \leq \mathtt{ALG}(\mathcal{I}) + \sum_{m\in[M]} \phi_m(z_m^{(N)})C_m. \tag{51}$$

Finally, by combining Equations (47) and (51), the competitive ratio of Algorithm 4 is

$$\frac{\mathtt{OPT}(\mathcal{I})}{\mathtt{ALG}(\mathcal{I})} \leq \frac{\mathtt{ALG}(\mathcal{I}) + \sum_{m\in[M]} \phi_m(z_m^{(N)})C_m}{\mathtt{ALG}(\mathcal{I})} \leq 1 + \frac{2\gamma}{\ln 2} = O(\ln \Lambda). \tag{52}$$

□

D IMPLEMENTATION OF DISCRETIZED DATA-DRIVEN ONLINE ALGORITHMS

Classical online algorithms are optimized for the worst-case instance; however, in practice, the worst-case rarely occurs and the average-case performance of online algorithms is often even worse than simple heuristics (e.g., greedy algorithms). Thus, this paper aims to propose a viable data-driven online algorithm (DOA) to improve the average-case performance while still providing worst-case guarantees. This approach generally consists of two steps: (i) constructing a class of parameterized online algorithms, each of which has a bounded competitive ratio; and (ii) adaptively selecting the online algorithm from the constructed algorithm class to optimize the average-case performance. In this paper, the first step has been completed in Theorem 3. The threshold-based algorithms have bounded competitive ratio β as long as the parameter of the threshold function is selected from the parameter set $\Gamma(\beta)$. In the following, we provide a viable approach for the second step by discretization and show the average-case performance. Finally, we discuss the limitations of this discretized DOA.

Algorithm implementation. In the experiments, we implement a discretized version of the DOA. In particular, we discretize $\Gamma(\beta)$ into $\tilde{\Gamma}(\beta)$ with step size of 0.1. Let d denote the cardinality of the discretized parameter set $\tilde{\Gamma}(\beta)$ and $[d] := \{1, \ldots, d\}$ denote the set of indices. In this discretized problem, we can alternatively use index $i \in [d]$ to represent the selected threshold function instead of parameter $\gamma \in \tilde{\Gamma}(\beta)$, and refer to i as an expert advice. Thus, we can restate the DOA as an exponential weights algorithm (or Hedge algorithm) [14] in Algorithm 5. This algorithm selects expert advice i_ℓ (that is equivalent to the selection of a threshold function) at the beginning of each

Algorithm 5 Data-driven online algorithm (DOA) via Hedge algorithm

1: **input:** parameter set $\tilde{\Gamma}(\beta)$ indexed by $[d]$, learning rate ν;
2: **output:** parameter selection $\{i_\ell\}_{\ell\in[L]}$;
3: **initialization:** initial selection probability $\boldsymbol{p}_1 := \{p_{1,i}\}_{i\in[d]} = [1/d,\dots,1/d]$;
4: **for** round $\ell = 1,\dots,L$ **do**
5: draw parameter i_ℓ from probability distribution $\boldsymbol{p}_\ell$;
6: observe reward $\boldsymbol{r}_\ell := \{r_{\ell,i}\}_{i\in[d]}$ and collect the reward r_{ℓ,i_ℓ};
7: update $\boldsymbol{p}_\ell$ by $p_{\ell+1,i} = \frac{p_{\ell,i}\exp(\nu r_{\ell,i})}{\sum_{j\in[d]} p_{\ell,j}\exp(\nu r_{\ell,j})}, \forall i \in [d]$.
8: **end for**

round $\ell \in [L]$. Then the rewards of all expert advices can be observed $\boldsymbol{r}_\ell := \{r_{\ell,i}\}_{i\in[d]}$ but only the reward of advice i_ℓ is collected by the algorithm.

Average-case performance. Following the classic results of the expert problem (e.g., Corollary 2.2 in [8]), the average-case performance of the DOA can be evaluated by the regret as follows.

Lemma 7. *Assume the per-round reward is bounded* $0 \le r_{\ell,i} \le \bar{r}, \forall \ell \in [L], i \in [d]$ *and the learning rate is set to* $\nu = \sqrt{2\ln d/(\bar{r}^2 L)}$*, the regret of Algorithm 5 is*

$$\mathbb{E}\left[\max_{i\in[d]} \sum_{\ell\in[L]} r_{\ell,i} - \sum_{\ell\in[L]} r_{\ell,i_\ell}\right] \le \bar{r}\sqrt{2L\ln d}. \tag{53}$$

Since the regret is sublinear in the number of rounds L, the average reward of Algorithm 5 approaches the reward obtained by the algorithm with a fixed threshold function selected in hindsight as $L \to \infty$.

Note that the regret is logarithmic in d and depends on the granularity of discretization and system parameters. In more detail, $d = \lceil \frac{\hat{d}_1}{\delta} \rceil \times \cdots \times \lceil \frac{\hat{d}_K}{\delta} \rceil$, where δ is the step size of discretization and $\hat{d}_k$ is the length of the k-th coordinate of the original parameter set $\Gamma(\beta)$. Based on Theorem 3, $\hat{d}_k$ can be upper bounded by

$$\begin{aligned}\hat{d}_k &= \ln 2 \cdot \min\left\{\frac{(\beta-4)}{6}, \frac{C_k}{\varepsilon_k}\right\} - (\beta-1)\zeta_k + 2W\left(\frac{(\beta-1)\zeta_k}{2\sqrt{2}}\exp\left(\frac{(\beta-1)\zeta_k}{2}\right)\right)\\ &\le \frac{\ln 2(\beta-1)}{6}\left[1+\frac{1}{\alpha_k\theta_k}\right],\end{aligned}$$

where $\zeta_k := -\ln 2/(6\alpha_k\theta_k)$ and $W(\cdot)$ is the Lambert W function. This upper bound increases linearly in the target competitive ratio β and decreases in the knapsack parameter $\alpha_k\theta_k$ since the knapsack with large $\alpha_k\theta_k$ dominates the worst-case ratio and thus gives less flexibility for parameter tuning. Let $\bar{d} = \max_{k\in[K]} \hat{d}_k$. Then the cardinality of the discretized set $\tilde{\Gamma}(\beta)$ is upper bounded by $d \le (\bar{d}/\delta)^K$. Therefore, the regret of the DOA can also be upper bounded by $\bar{r}\sqrt{2LK\ln(\bar{d}/\delta)}$, which is sublinear in the number of knapsacks K but increases when a finer grained step size δ is chosen.

Limitations of implementing the discretized problem. Algorithm 5 is based on the discretization of the original parameter set $\Gamma(\beta)$. The discretized problem can more accurately approximate the original problem by adopting a smaller step size δ while this will result in a larger parameter set d and thus a slower speed of learning the threshold function based on the regret bound (53). In this paper, we choose to set $\delta = 0.1$ to balance the approximation accuracy and learning speed in our numerical tests. In the problems with large d, it may become even computationally difficult to obtain the full information feedback (i.e., the reward of all possible expert advice $\boldsymbol{r}_\ell$) in each round. We can resort to the EXP3 algorithm [2] that only requires bandit information feedback (i.e., the reward of the selected parameter r_{ℓ,i_ℓ}) in such settings.

An alternative implementation of DOA is to design online learning algorithms to directly learn the parameter γ from the original continuous parameter set $\Gamma(\beta)$. However, it is challenging to provide theoretical regret bounds in this setting. Particularly, the regret analysis requires understanding the special properties (e.g., Lipschitz-continuous) of the per-round reward (as a function of the selected parameter γ). The per-round rewards of our online knapsack problem are in general piecewise Lipschitz functions, which suffer linear regret bounds without additional conditions [9]. Thus, it remains an open question to design sublinear regret online learning algorithm from the original continuous parameter set. A promising direction is to follow recent works on data-driven algorithm designs via online learning [3, 4] that identify additional properties (e.g., dispersion) of the reward function to improve the regret analysis.

Received August 2022; revised October 2022; accepted November 2022