Online Rounding and Pricing Schemes for k-Rental Problems

Hossein Nekouyan
University of Alberta
Edmonton, Canada
nekouyan@ualberta.ca

Bo Sun
University of Ottawa
Ottawa, Canada
Vector Institute
Toronto, Canada
bo.sun@uottawa.ca

Raouf Boutaba
University of Waterloo
Waterloo, Canada
rboutaba@uwaterloo.ca

Xiaoqi Tan
University of Alberta
Edmonton, Canada
xiaoqi.tan@ualberta.ca

Abstract

We study two online resource allocation problems with reusability in an adversarial setting, namely kRental-Fixed and kRental-Variable. In both problems, a decision-maker manages k identical reusable units and faces a sequence of rental requests over time. We develop theoretically grounded relax-and-round algorithms with provable competitive ratio guarantees for both settings. For kRental-Fixed, we present an optimal randomized algorithm that achieves the best possible competitive ratio. The algorithm first computes an optimal fractional allocation using a price-based approach, and then applies a novel lossless online rounding scheme to obtain an integral solution. For kRental-Variable, we first establish the impossibility of achieving lossless online rounding. We then introduce a limited-correlation rounding technique that treats each unit independently while introducing controlled dependencies across allocation decisions involving the same unit. Combined with a carefully-crafted price-based method for computing the fractional allocation, this approach yields an order-optimal competitive ratio for the variable-duration setting.

CCS Concepts

• **Theory of computation** → **Online algorithms**; **Computational pricing and auctions**.

Keywords

Online Algorithms, Competitive Analysis, Resource Allocation, Posted Pricing, Online Rounding

ACM Reference Format:
Hossein Nekouyan, Bo Sun, Raouf Boutaba, and Xiaoqi Tan. 2026. Online Rounding and Pricing Schemes for k-Rental Problems. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792368

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792368

1 Introduction

Online resource allocation is a central topic in computer science and operations research. It studies how to allocate limited resources to sequential online requests and has found wide applications in the digital platform economy, such as the allocation of computing and bandwidth resources in cloud computing [23, 24] and network routing [1–3], as well as dynamic pricing in hotel booking [20] and ride-sharing [6]. Traditionally, research has focused on sales-based models, in which resources are permanently allocated to requests and ownership is transferred from the seller to the buyer. However, in the platform economy, attention has shifted to rental-based models, where resources can be *reused*, as in cloud services, fashion rentals, or short-term accommodations such as hotels or Airbnbs. Reusability introduces an additional layer of complexity in online resource allocation due to overlapping usage periods and fluctuating demand. In this paper, we study online resource allocation problems involving *reusable* resources, which we refer to as *k-rental* problems.

In the k-rental setting, a seller (or decision-maker) manages an inventory of k identical units of a resource, such as k servers in a cloud data center or k hotel rooms. A sequence of requests arrives sequentially over a fixed horizon, each seeking to *rent* one unit for a specified duration and offering a corresponding value (or willingness to pay). Once allocated, a unit becomes unavailable until the rental duration ends, after which it returns to the inventory. The seller makes irrevocable accept-or-reject decisions for each request upon its arrival, with the objective of maximizing the total value of all accepted requests. In this paper, we study two variants of the k-rental problem, termed kRental-Fixed and kRental-Variable. In kRental-Fixed, each request occupies a unit for the same fixed duration, while its value can vary within a bounded range. This variant models fixed-duration rentals such as overnight hotel stays or time-slotted meeting room bookings, where the resource is released after each fixed block ends. kRental-Variable generalizes this setting by allowing rental durations to vary across requests, with each request's value assumed to be linear in its duration. This variable-duration model captures settings such as cloud instance allocation, virtual machine or container provisioning, and other usage-based services where resources are billed by the duration of use.

For the kRental-Fixed problem, [5] considers a more general online bipartite matching model in which each matched offline node becomes available again after a fixed d time units. However, their model does not capture the varying values of each request (equivalent to the varying edge weight of each matching), which are essential to online k-rental problem but cannot be easily handled by [5]. A more recent work by [7] examines an online job assignment problem that overlaps with both kRental-Fixed and kRental-Variable. It has developed *deterministic* algorithms that achieve order-optimal competitive ratios, but only in the *large-inventory* regime (i.e., $k \to \infty$). This leaves open the important question designing *randomized* algorithms with tight guarantees for both kRental-Fixed and kRental-Variable across all inventory sizes, including small and moderate regimes (i.e., k is finite).

1.1 Our Contributions

We address the above open question affirmatively by developing randomized algorithms for k-rental under general inventory settings. Our algorithms achieve the optimal competitive ratio for kRental-Fixed and a near-optimal performance guarantee for kRental-Variable. Specifically, our contributions are threefold:

γ-OCR and lossless online rounding schemes. We introduce an online rounding subroutine termed γ-Online Correlated k-Rental (γ-OCR) that captures the core challenge of rounding any feasible online fractional allocations in reusable settings with fixed rental durations. The goal is to round each fractional allocation to an integral one, allocating a unit with probability at least equal to its fractional value multiplied by $\gamma \in [0, 1]$. As a warm-up, we present an independent rounding scheme and show that it is lossless when the inventory is large but fails in small-inventory cases. To address this, we develop a new online rounding algorithm, 1-OCR, which introduces correlations across time steps to achieve lossless rounding. We believe this method has broader applicability and may be of independent interest.

Implications and insights of γ-OCR for pricing reusable resources. Our study highlights the role of *duration-oblivious dynamic pricing* in managing reusable resources through the online rounding framework γ-OCR. For kRental-Fixed, we develop an optimal randomized pricing scheme rooted in the 1-OCR framework. The algorithm first computes an optimal fractional allocation via a price-based mechanism, where each arriving request is priced according to the instantaneous utilization level of the resource unit, independent of future utilization fluctuations (hence, the term "duration-oblivious"). The fractional solution is then rounded to an integral allocation using the 1-OCR procedure. When rental durations are variable, as in kRental-Variable, we establish a fundamental impossibility result: no online rounding scheme can preserve the competitive ratio of an arbitrary fractional solution. To overcome this barrier, we design a randomized pricing-based algorithm that combines duration-oblivious dynamic pricing with a limited-correlation rounding scheme. This approach prices requests in the same utilization-dependent manner as in the fixed-duration case, but incorporates carefully structured dependencies across decisions on the same resource unit to mitigate the lossiness of rounding. Despite the inherent gap, we prove that this pricing-and-rounding strategy achieves order-optimal, best-known guarantees for kRental-Variable.

Techniques. Our overarching approach follows the *relax-and-round* framework, which first computes a fractional allocation for a relaxed version of the problem using a price-based method, and then rounds this solution to obtain an integral allocation. The second step, i.e., the rounding procedure, is more nuanced and central to our contribution. From a technical standpoint, to derive a pricing function that produces the optimal fractional solution for kRental-Fixed, we adopt an online primal-dual approach to guide the design of an appropriate pricing function. For kRental-Variable, we employ the LP-free certificate framework developed in [13] to construct the pricing function. In this approach, designing an α-competitive algorithm reduces to finding a feasible solution to a system of *delayed differential inequalities*, which the pricing function of DOP-ϕ-VARIABLE must satisfy in order to achieve the desired competitiveness. This system, parameterized by the achievable competitive ratio, characterizes the design of a pricing function that attains the smallest possible competitive ratio.

1.2 Related Work

The past few decades have witnessed a wide range of impactful applications of online resource allocation, including online advertising, ride-sharing platforms, and online auctions, making it a topic extensively studied across computer science, operations research, and economics. Below, we briefly review several key papers from the literature that are closely related to this work.

Online scheduling. The adversarial online interval scheduling problem, introduced by [18], introduced the problem of the task scheduling a sequence of intervals (jobs) on a single server as they arrive in order of their start times, assuming that the minimum and maximum job durations are unknown. Motivated by this work, several subsequent studies (e.g., [9, 12, 14]) have explored various extensions of the problem. For instance, [12] examined a variant in which the minimum and maximum job durations are known, referred to as the *online reservation problem*, and proposed a static pricing algorithm with a performance guarantee of $3 \cdot (1 + \ln(\Delta))$. In Section 4, we investigate whether a dynamic pricing scheme, one that continuously updates resource prices in response to changing market conditions, can achieve stronger performance guarantees.

Online matching with reusable resources. Several studies, including [5, 7, 11, 15, 22], have investigated the online matching problem and its variants in settings with resource reusability. In particular, as previously discussed, [5] examined an online bipartite matching problem in which resources are reusable. This setting extends the classical online matching model introduced by [17] by allowing each matched offline node to become available again after d time units. In Section 3, we focus on a special case of this model involving a single resource type and valuation uncertainty. We explore whether improved competitive ratios and stronger correlation schemes can be achieved compared to their results. [15] studied adversarial reusability in the context of online assortment planning, where the kRental-Variable problem can be reduced to their model. Their work also centers on deterministic algorithm design. Within the kRental-Variable framework, their algorithm achieves

a competitive ratio of 4 $\ln(\frac{d_{\max}}{d_{\min}})$ as the inventory size tends to infinity, where $d_{\max}$ and $d_{\min}$ denote the maximum and minimum rental durations, respectively. More recently, [7] considered a generalized version of the kRental-Variable problem within the online matching framework. Their model incorporates uncertainty in each request's per-unit-time valuation over the requested interval and across different item types. Their algorithm achieves a competitive ratio of $4 + \ln(d_{\max}/d_{\min})$ for kRental-Variable, which is the best-known guarantee in the large-inventory regime (as $k \to \infty$). Given the close connection between their formulation and ours, we provide a comparison in Subsection 4.4, showing that our algorithm can reproduce this result in the same large-inventory setting and outperforms their result in small inventory settings.

Online rounding. Recent work in computer science and operations research has demonstrated the effectiveness of online rounding frameworks [8, 19], which typically adopt a *relax-and-round* paradigm. For example, [8] introduced a subroutine called Online Correlated Selection, which imposes negative correlation across selected pairs, achieving improved competitive ratios over greedy algorithms limited to a $\frac{1}{2}$-competitive ratio. Building on this idea, [5] developed the Online Correlated Rental method for settings with reusable resources and fixed rental durations.

2 Online Correlated k-Rental

We first introduce Online Correlated k-Rental (OCR), which is an online rounding subroutine for k-rental problems. The OCR subroutine captures the key challenge of rounding a fractional solution in k-rental problems where resources are reusable, and serves as a building block for algorithm design in Section 3.

2.1 γ-OCR: Definitions and Objectives

Definition 1 (γ-OCR). *Consider a set of k identical balls, each uniquely labeled from the set $\{1, 2, \ldots, k\}$. Each ball can be rented to a player for a fixed duration of d time units, after which it becomes available for reuse. A sequence of N players arrives one by one, where each player n is characterized by a tuple $(\hat{x}_n, a_n)$. Here, $\hat{x}_n \in [0, 1]$ denotes the target probability with which the procedure should assign a ball to player n, and a_n is the arrival time of player n. For a fixed $\gamma \in [0, 1]$, a γ-OCR is an online rounding scheme that guarantees renting a ball to each player $n \in [N]$ with probability at least $\gamma\hat{x}_n$, for all input instances $\{(\hat{x}_n, a_n)\}_{n\in[N]}$ satisfying the following condition:*

$$\hat{x}_n \le \min\left\{1,\ k - \sum\nolimits_{j\in[n-1]} \hat{x}_j \cdot \mathbb{I}_{\{a_j+d>a_n\}}\right\}, \qquad \forall n \in [N]. \quad (1)$$

As a rounding scheme, γ-OCR is a randomized online algorithm that makes irrevocable decisions upon the arrival of each player, either assigning a ball to the player or rejecting it. γ-OCR focuses on the inputs that satisfy the regularity conditions in Eq. (1). In particular, the term $\sum_{j\in[n-1]} \hat{x}_j \cdot \mathbb{I}_{\{a_j+d>a_n\}}$ on the right-hand side quantifies the cumulative targeted probabilities of players who arrived prior to player n and whose rental intervals (each of length d) overlap with that of player n. Furthermore, the condition in Eq. (1) also guarantees that $\hat{x}_n \le 1$ for all $n \in [N]$, thereby ensuring that the assignment probability for any individual player does not exceed the availability of a single unit. Violation of this constraint would imply that the total targeted assignment at the arrival of player n exceeds the inventory limit of k units.

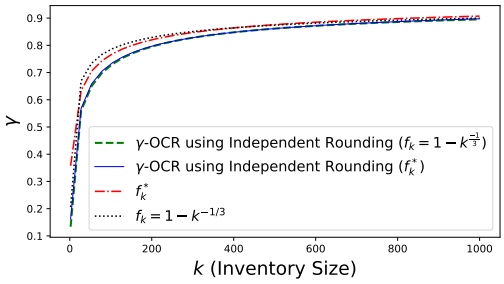

Figure 1: Comparison of the approximate parameter $f_k = 1 - \frac{1}{k^{1/3}}$ versus the optimal parameter f_k^*, and the resulting γ values for Algorithm 1 using f_k and f_k^*.

The goal is to design a γ-OCR that maximizes γ. A 1-OCR is referred to as a *lossless* rounding scheme, as it guarantees renting a ball to each player $n \in [N]$ with probability at least $\hat{x}_n$. Designing such a scheme, however, is non-trivial. As a warm-up, we first present a simple independent rounding algorithm that incurs rounding losses. This scheme highlights the limitations of uncorrelated rounding and underscores the need to correlate the allocation decision for each newly arriving player with the decisions made for previously served players who still hold rented balls.

2.2 Warm-up: Independent Rounding

Consider the rounding scheme in Algorithm 1, which is motivated by rounding schemes in the approximation algorithms literature. It samples a random seed S_n independently at the arrival of each player n from the uniform distribution $U(0, 1)$. A ball is then allocated to player n if $S_n \le f_k \cdot \hat{x}_n$ and at least one ball is available. The parameter $f_k \in [0, 1]$ serves as a downscaling factor that depends on the total number of balls, k, and targets to reduce the allocation probability for each player. This downshift increases the likelihood that a ball will be available for future arrivals, allowing more players to be served.

Proposition 1. *Given $f_k \in [0, 1]$, Algorithm 1 is γ_k-OCR, where* $\gamma_k = f_k \cdot \left(1 - \exp\left(-\frac{(k - f_k \cdot k)^2}{f_k \cdot k + k}\right)\right)$.

The proof of Proposition 1 follows standard techniques in the approximation algorithms literature (e.g., [4]) and applies the Chernoff bound to upper bound the probability of unfavorable events where $S_n \le f_k \cdot \hat{x}_n$ and $y_n \ge k$. This result also informs the design of the downscaling parameter f_k as a function of k to maximize γ_k. The optimal value f_k^* cannot be obtained in closed form, since it involves solving a transcendental equation of the form $A(x) = \ln(1 - D(x))$. However, it can be efficiently computed using numerical methods such as binary search, and is illustrated in Figure 1. Based on the numerical results, we also observe that the optimal parameter f_k^* closely follows a simple expression $f_k = 1 - \frac{1}{k^{1/3}}$. Figure 1 shows that the performances of Algorithm 1 with $f_k = 1 - \frac{1}{k^{1/3}}$ and f_k^* are also close. In addition, as seen in the figure, for large values of k the resulting γ approaches one; however, for small inventory sizes (small values of k), the algorithm is not an effective rounding scheme. This motivates our design for a lossless rounding scheme.

Algorithm 1: γ-OCR using Independent Rounding

Input: parameters k, d, and $f_k \in [0,1]$;
Output: Assignment $z_n \in \{0,1\}$ for each player $n \in [N]$;
for *each player n with the tuple* $(a_n, \hat{x}_n)$ **do**
  Compute the number of balls currently rented at time a_n: $y_n = \sum_{j=[n-1]} z_j \cdot \mathbb{I}_{\{a_j+d>a_n\}}$;
  Sample a random seed $S_n \sim U(0,1)$;
  if $S_n \le f_k \cdot \hat{x}_n$ ***and*** $y_n < k$ **then**
    $z_n \leftarrow 1$; ▷ Allocate the (y_n+1)-th available ball
  end
  else
    $z_n \leftarrow 0$. ▷ Reject request n
  end
end

2.3 A 1-OCR Through Dependent Rounding

In this subsection, we propose a 1-OCR by correlating the allocation decisions in Algorithm 2.

Key idea of creating correlations. At a high level, the key idea of Algorithm 2 is to correlate each current allocation decision with past decisions involving players whose allocated balls are expected to return sooner. Specifically, prior to the arrival of any player, the algorithm samples a single random seed $r \sim U(0,1)$, which remains fixed throughout the entire execution and serves as the only source of randomness. This shared seed induces correlation across allocation decisions for different players. For each player n, the algorithm uses this seed to determine whether to allocate a ball. It maintains two pointers: $m_n \in \{1,2,\dots,k\}$ and $p_n \in [0,1]$. The first pointer, m_n, indicates the ball under consideration for allocation to player n; this ball may be unavailable if it is currently assigned to a previous player. The second pointer, p_n, specifies a subinterval of $[0,1]$ used to guide the allocation decision. For each player n, the algorithm proceeds based on the relationship between p_n and $\hat{x}_n$, as described in the following two cases.

- **Case 1:** If $p_n + \hat{x}_n < 1$, then the algorithm allocates ball m_n to player n if $r \in [p_n,\ p_n+\hat{x}_n)$ and the ball is available in the system. Otherwise, the player is rejected and no ball is allocated.
- **Case 2:** If $p_n + \hat{x}_n \ge 1$, then the algorithm allocates ball m_n if $r \in [p_n,\ 1]$ (and the ball is available in the system), or allocates ball m_n+1 if $r \in [0,\ p_n+\hat{x}_n-1)$. Since $\hat{x}_n \le 1$, these intervals are non-overlapping, ensuring that no more than one ball is allocated.

Key invariants of Algorithm 2. It is worth noting that Algorithm 2 does not verify the availability of balls m_n and m_n+1 before assigning them to players. This omission is justified by some key invariants that we establish in Proposition 2 below concerning the availability of these balls.

Proposition 2 (Invariants of Algorithm 2). *Upon the arrival of the n-th player, the following holds:*

- *If $p_n + \hat{x}_n < 1$ and the random seed $r \in [p_n,\ p_n+\hat{x}_n)$, then the ball m_n is available.*
- *If $p_n + \hat{x}_n \ge 1$ and the random seed $r \in [p_n,\ 1]$, then the ball m_n is available.*

Algorithm 2: 1-OCR using Dependent Rounding

Input: Number of balls k, rental duration d.
Output: Assignment $z_n \in \{0,1\}$ for each player $n \in [N]$.
Initialize: Set $m_1 = 1$, $p_1 = 0$. sample a random seed $r \sim U(0,1)$.
for *each request n with the tuple* $(a_n, \hat{x}_n)$ **do**
  if $\sum_{j\in[n]} \hat{x}_j \cdot \mathbb{I}_{\{a_j+d>a_n\}} > k$ **then**
    $z_n \leftarrow 0$. ▷ Reject player n
  end
  if $p_n + \hat{x}_n < 1$ **then**
    if $r \in [p_n, p_n+\hat{x}_n)$ **then**
      $z_n \leftarrow 1$. ▷ Assign ball m_n to player n
    end
    else
      $z_n \leftarrow 0$. ▷ Reject player n
    end
    Update $p_{n+1} = p_n + \hat{x}_n$ and $m_{n+1} = m_n$.
  end
  else
    if $r \in [p_n, 1]$ **then**
      $z_n \leftarrow 1$. ▷ Allocate ball m_n to player n
    end
    else if $r \in [0, \hat{x}_n + p_n - 1)$ **then**
      $z_n \leftarrow 1$. ▷ Allocate ball m_n+1 to player n
    end
    else
      $z_n \leftarrow 0$. ▷ Reject player n
    end
    Update $p_{n+1} = \hat{x}_n + p_n - 1$ and $m_{n+1} = m_n + 1$ (if $m_{n+1} > k$, then set $m_{n+1} = 1$).
  end
end

- *If $p_n + \hat{x}_n \ge 1$ and the random seed $r \in [0,\ p_n+\hat{x}_n-1)$, then the ball m_n+1 is available.*

Proof. We prove the proposition by contradiction. Consider the first case where $p_n+\hat{x}_n < 1$, and suppose, for contradiction, that ball m_n is not available upon the arrival of player n. Then, there must exist some earlier player $n' < n$ to whom ball m_n was previously allocated, and whose rental interval overlaps with that of player n. This could have occurred in one of two ways: either $m_{n'} = m_n$ and the random seed $r \in [p_{n'},\ \min\{p_{n'}+\hat{x}_{n'},\ 1\})$, or $m_{n'}+1 = m_n$ and $r \in [0,\ p_{n'}+\hat{x}_{n'}-1)$. Now, based on how the pointer mechanism in Algorithm 2 updates the pointer m_n, to have $m_{n'}$ or $m_{n'}+1$ equal to m_n, and furthermore to have $p_n \ge r$, there must exist a sequence of players from n' to n whose cumulative target allocation exceeds the capacity k. That is, $\sum_{l=n'}^{n} x_l > k$. Furthermore, since a unit of ball is under player n''s allocation at time a_n, all players from n' to n must have overlapping rental intervals with a_n (due to fixed rental durations), i.e., $a_j + d > a_n \quad$ for all $j \in \{n',\dots,n\}$. Therefore, the total fractional target probabilities of players that their interval

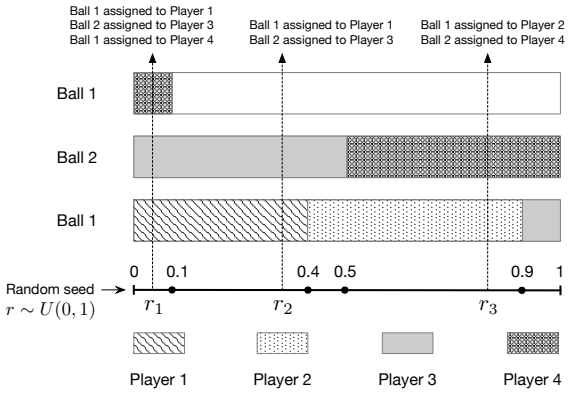

Figure 2: Illustration of the online rounding process in Algorithm 2 for an instance with two balls ($k = 2$), rental duration $d = 5$, and four players ($N = 4$). The players' arrival times and target probabilities are given by $\{(a_1 = 1, \hat{x}_1 = 0.4),\ (a_2 = 2, \hat{x}_2 = 0.5),\ (a_3 = 3, \hat{x}_3 = 0.6),\ (a_4 = 6, \hat{x}_4 = 0.6)\}$. As illustrated, when the random seed is sampled as $r = r_1 \in [0, 0.1)$, Ball 1 is first assigned to Player 1 and, upon return, reassigned to Player 4; meanwhile, Ball 2 is assigned to Player 3.

overlap with time a_n must satisfy:

$$\sum_{j\in[n-1]} \hat{x}_j \cdot \mathbb{I}\{a_j + d > a_n\} + \hat{x}_n > k,$$

which contradicts the feasibility constraint of the problem instance, as specified in Eq. (1). Hence, our initial assumption must be false, and the ball m_n must be available in the first case. A similar contradiction argument applies to the other two cases in the proposition, using the same logic of overlapping rental intervals, pointer behavior, and capacity violation. This concludes the proof. □

Leveraging the invariants established in Proposition 2, we can conclude that Algorithm 2 is lossless.

Theorem 1. *Algorithm 2 is a 1-OCR, namely, it is a lossless rounding scheme.*

The proof of Theorem 1 is given in Appendix A. See Figure 2 for an illustration of the lossless online rounding process using a concrete example.

3 k-Rental with Fixed Rental Durations

In this section, we define the kRental-Fixed problem and present an optimal randomized algorithm based on the rounding scheme, 1-OCR, developed in the previous section.

3.1 Problem Formulation and Assumptions

Let us formally define the online k-rental problem (kRental-Fixed) as follows. A decision maker has k units of an item and allocates them to online arriving requests. Each request $n \in [N]$ asks to rent one item for d time units. The n-th request arrives at time a_n, and has a valuation v_n for renting one item. In particular, if an item is allocated to request n, the item is rented starting from a_n, and becomes available again at time $a_n + d$. Let $x_n \in \{0, 1\}$ denote the decision to accept or reject the n-th request. The objective of the problem is to maximize the total valuation of requests that receive an item allocation, i.e., $\sum_{n\in[N]} x_n \cdot v_n$, while respecting the constraints on available items.

Let $I = \{v_n, a_n\}_{n\in[N]}$ denote an instance of kRental-Fixed. The maximum valuation from the optimal clairvoyant algorithm, $\texttt{OPT}(I)$, can be determined by solving the following optimization problem:

$$\max_{x_n} \quad \sum\nolimits_{n\in[N]} v_n \cdot x_n, \tag{2a}$$

$$\text{s.t.} \quad \sum\nolimits_{j\in[n]} x_j \cdot \mathbb{I}\{d + a_j > a_n\} \le k, \quad \forall n \in [N], \tag{2b}$$

$$x_n \in \{0, 1\}, \quad \forall n \in [N]. \tag{2c}$$

The constraint (2b) ensures that at any point in time throughout the horizon, no more than k items are allocated. In the online setting, the decision maker must make an irrevocable decision to accept or reject each request upon its arrival. The uncertainty regarding future requests' valuations and the overall demand for the items makes this decision challenging. To achieve a bounded performance, we follow the literature and assume the valuations of requests are bounded.

Assumption 1. *All valuations are within the range* $[v_{\min}, v_{\max}]$, *i.e.,* $v_n \in [v_{\min}, v_{\max}], \forall n \in [N]$.

Let $\mathcal{I}$ denote the set of all instances of the kRental-Fixed that satisfy Assumption 1. Our goal is to design online algorithms whose objective is competitive with that of $\texttt{OPT}(I)$ for every instance $I \in \mathcal{I}$. Specifically, an online algorithm ALG is said to be α-competitive if, for any input instance I, the following inequality holds $\alpha \ge \frac{\texttt{OPT}(I)}{\mathbb{E}[\texttt{ALG}(I)]}$, where the expectation $\mathbb{E}[\texttt{ALG}(I)]$ is taken over the randomness of the online algorithm. In the following section, we present an algorithm for the kRental-Fixed problem that achieves the minimum possible competitive ratio among all online algorithms. This result strictly improves upon the deterministic algorithm proposed by [7], which attains only an order-optimal competitive ratio in the asymptotic regime where the inventory size tends to infinity.

3.2 DOP-ϕ-FIXED: A Relax-and-Round Algorithm for kRental-Fixed

We introduce a randomized online algorithm, DOP-ϕ-FIXED, presented in Algorithm 3, which is based on a general *relax-and-round* framework. In the *relaxation step*, we design an online algorithm for a continuous version of problem (2) by relaxing the integrality constraint to $x_n \in [0, 1]$ for all $n \in [N]$. This algorithm produces an online fractional solution $\{\hat{x}_n\}_{n\in[N]}$. In the *rounding step*, the algorithm converts the fractional solution $\hat{x}_n$ into an integral decision $x_n \in \{0, 1\}$.

Core idea for obtaining fractional allocation: Pseudo-utility maximization in Eq. (3). Upon the arrival of request n, the algorithm first computes an expected utilization level $y_n = \sum_{j=1}^{n-1} x_j \cdot \mathbb{I}\{a_j + d > a_n\}$, which is the expected number of resource units that are currently rented by the previous requests whose rental durations overlap with that of request n. Using this expected utilization level, the algorithm determines the fractional allocation by solving a *pseudo-utility maximization* problem as described in Eq. (3). The first term $x \cdot v_n$ is the valuation from request n, and the second term

is a *pseudo-cost* of renting x unit of an item when the current utilization level is y_n. Specifically, the pseudo-cost, $k \int_{y_n/k}^{(y_n+x)/k} \phi(\eta)\, d\eta$, is estimated using a carefully-designed, normalized *marginal pricing* function $\phi : [0,1] \to [v_{\min}, v_{\max}]$. Thus, the integration of ϕ over the resource utilization interval $[\frac{y_n}{k}, \frac{y_n+x}{k}]$ gives the pseudo-cost. In this context, the fractional allocation $\hat{x}_n$ represents the optimal fraction of a resource unit to allocate to request n, given their valuation v_n and the pricing rule defined by ϕ.

Rounding subroutine: 1-OCR. In this step, the fractional solution $\hat{x}_n$ is passed to the lossless online rounding procedure 1-OCR. This procedure generates an integral decision x_n on whether to accept the request. This procedure operates as an online algorithm, with an instance of this algorithm initiated at the start of Algorithm 3. As each request n arrives at time a_n, the 1-OCR procedure receives a probability value $\hat{x}_n \in [0,1]$ (generated by the relaxation step) as input at time a_n. Then it makes an integral decision x_n on whether to accept or reject the request. All rounding decisions are based on one random seed, which is fixed when Algorithm 3 initiates the 1-OCR instance and remains the same for all requests.

Algorithm 3: Duration-Oblivious Price-based Algorithm for kRental-Fixed (DOP-ϕ-FIXED)

Input: pricing function $\phi : [0,1] \to [v_{\min}, v_{\max}]$;
Initiate an instance of 1-OCR procedure;
for *each arriving request n with* (v_n, a_n) **do**
  Compute the expected utilization at time a_n:

$$y_n = \sum_{j=1}^{n-1} \hat{x}_j \cdot \mathbb{I}\{a_j + d > a_n\};$$

  if $y_n < k$ **then**
    Compute the fractional allocation: ▷ Relax step

$$\hat{x}_n = \underset{x \in [0, \min\{1, k-y_n\}]}{\arg\max} \left(x \cdot v_n - k \cdot \int_{\eta=y_n/k}^{(y_n+x)/k} \phi(\eta) d\eta \right); \quad (3)$$

    Decide the integral allocation using Algorithm 2:
    $x_n = $ 1-OCR $(\hat{x}_n, a_n)$; ▷ Round step
  end
  else
    $x_n = 0$. ▷ Reject request n
  end
end

Before leaving this subsection, we remark that the pseudo-utility maximization that yields the fractional allocation in Eq. (3) differs fundamentally from the pseudo-cost frameworks of [7, 22]. Those works adopt a *forward-looking* approach that estimates the cost of accepting a request using the *entire* utilization trajectory over the request's rental interval. Thus, the pseudo-cost is a functional of the utilization curve throughout the interval, not merely the level observed at arrival. In contrast, our algorithm adopts a much simpler *myopic* approach that computes its pseudo-cost just based on the *current* utilization at the moment the request arrives (hence the term "*duration-oblivious*"). We next show that it is a simpler yet still competitive pricing rule.

3.3 Theoretical Guarantee of DOP-ϕ-FIXED

As explained in Section 3.2, DOP-ϕ-FIXED employs the 1-OCR procedure to round the fractional allocation $\hat{x}_n$ at each time step and to make an integral decision. For this procedure to function correctly, the fractional solution generated by the relax step must satisfy the feasibility conditions in Eq. (1) for 1-OCR.

LEMMA 1 (FEASIBILITY OF FRACTIONAL SOLUTIONS). *In DOP-ϕ-FIXED, the input of the round step (line 7 of Algorithm 3),* $\{(\hat{x}_n, a_n)\}_{n\in[N]}$, *satisfies the feasibility condition in Eq.* (1).

The above lemma naturally follows from the design of DOP-ϕ-FIXED. Based on this, we prove the performance guarantee for DOP-ϕ-FIXED.

THEOREM 2. *DOP-ϕ-FIXED is* $\left(1+\ln\left(\frac{v_{\max}}{v_{\min}}\right)\right)$*-competitive for kRental-Fixed when* ϕ *is given by*

$$\phi(y) = v_{\min} \cdot \exp\Big(\big(1 + \ln(\tfrac{v_{\max}}{v_{\min}})\big) \cdot y - 1 \Big), \quad y \in [0,1]. \quad (4)$$

The proof sketch of Theorem 2 is provided in Appendix C. Our proof follows the well-established online primal-dual (OPD) framework, with a notable deviation from the conventional approach: the updates to the dual variables are deferred until after the arrival of the final request similar to the work done in [15]. Finally, we can show that DOP-ϕ-FIXED achieves the optimal competitive ratio by showing a matching lower bound as follows.

PROPOSITION 3 (LOWER BOUND). *Under Assumption 1, no online algorithm, deterministic or randomized, can obtain a competitive ratio better than* $1 + \ln\left(\frac{v_{\max}}{v_{\min}}\right)$ *for the kRental-Fixed problem.*

The proof follows the approach of [16, 21] by constructing a *family of hard instances* to derive the desired lower bound.

4 k-Rental with Variable Rental Durations

This section focuses on the kRental-Variable problem, which generalizes the kRental-Fixed problem studied in Section 3 by allowing variable rental durations.

4.1 Problem Statement and Assumptions

We formally define the k-rental problem with variable rental durations (kRental-Variable) as follows. A decision maker has k identical units of a resource to allocate to N requests arriving online. Each request $n \in [N]$ arrives at time a_n and asks to rent one unit of the resource for d_n time units, where the rental duration d_n may vary across requests. If a unit is allocated to request n, it remains occupied until time $a_n + d_n$, after which it becomes available again for future requests. Each request has a valuation equal to its rental duration d_n. Let $x_n \in \{0,1\}$ indicate whether request n is accepted ($x_n = 1$) or rejected ($x_n = 0$). The objective is to maximize the total valuation of accepted requests, $\sum_{n\in[N]} x_n d_n$, subject to the resource constraint corresponding to the finite inventory capacity.

Let $I = \{a_n, d_n\}_{n\in[N]}$ denote an instance of the problem. The performance of the optimal clairvoyant algorithm on instance I, $\mathtt{OPT}(I)$, can be computed based on following integer linear program:

$$\max_{x_n} \quad \sum\nolimits_{n\in[N]} x_n \cdot d_n, \quad (5a)$$

$$s.t. \quad \sum_{j\in[n]} x_j \cdot \mathbb{I}_{\{d_j+a_j>a_n\}} \le k, \quad \forall n \in [N], \tag{5b}$$

$$x_n \in \{0,1\}, \quad \forall\, n \in [N]. \tag{5c}$$

To achieve a bounded competitive ratio, we still impose constraints on the adversary, ensuring that the requested rental durations of requests are bounded within a finite support.

Assumption 2. *All rental durations are within the range* $[d_{\min}, d_{\max}]$, *i.e.,* $d_n \in [d_{\min}, d_{\max}], \forall n \in [N]$.

In the following, we first prove that obtaining a lossless rounding scheme for a variant of the γ-OCR problem where players have variable rental durations is impossible. Given this, we then develop a new randomized algorithm that uses a new limited-correlation scheme to round the fractional solution, achieving a tight competitive ratio for kRental-Variable.

4.2 Impossibility Results

We show that the correlation scheme of the 1-OCR procedure in Algorithm 2 cannot be extended to settings with variable, request-dependent rental durations; in fact, when the duration of the requested rental intervals varies, it can be proven that no lossless rounding scheme exists.

Theorem 3. *There is no rounding scheme that can attain a lossless rounding, i.e.* $\gamma = 1$, *for the* γ-OCR *problem with variable rental durations.*

We prove the theorem by constructing a family of hard instances on which every online algorithm necessarily incurs a non-zero loss. Fundamentally, a lossless algorithm would need to randomize its decisions such that, across all sample paths, the number of allocated items remains balanced. Specifically, upon the arrival of request n, the algorithm must accept it on precisely those sample paths where the number of allocated items is lower than in any other path, or on paths where the number of allocated items is about to decrease due to the imminent expiration of rental durations for previously accepted requests. However, in the variable-duration setting, rental lengths fluctuate unpredictably with the arrival of new requests. A request allocated a unit later than others may return it earlier, introducing significant variability. This makes it infeasible to maintain the necessary correlation across sample paths. As a result, any online algorithm must incur a non-trivial competitive loss.

4.3 DOP-ϕ-VARIABLE: A Randomized Algorithm with Limited Correlation

We present DOP-ϕ-VARIABLE in Algorithm 4, which also follows a relax-and-round paradigm. Upon the arrival of request n, the algorithm computes the *probabilistic utilization level* $y_n^{(i)}$ for each unit $i \in [k]$. It then selects the unit i_n^* with the lowest utilization level (breaking ties arbitrarily). A fractional allocation $\hat{x}_n \in [0,1]$ is determined by solving the pseudo-utility maximization problem in Eq. (6), using a pricing function ϕ. Note that here ϕ does not refer to the function in Eq. (4), but instead denotes a new function specifically designed for DOP-ϕ-VARIABLE. Once $\hat{x}_n$ is computed, the algorithm proceeds to the rounding phase by correlating the current allocation decision for unit i_n^* with its prior allocation history. Specifically, if unit i_n^* is available, a new random seed is drawn, and the unit is allocated to request n with probability $\hat{x}_n/(1-y_n^{(i_n^*)})$. This ensures that the overall allocation probability for unit i_n^* is exactly $\hat{x}_n$. *This correlation process is carried out independently for each unit, introducing dependency only among allocation decisions involving the same unit.*

Algorithm 4: Duration-Oblivious Price-based Algorithm for kRental-Variable (DOP-ϕ-VARIABLE)

Input: pricing function $\phi : [0,1] \to [d_{\min}, d_{\max}]$;
for *each arriving request* n *with* (d_n, a_n) **do**
  Compute the probabilistic utilization level at a_n:
  $y_n^{(i)} = \sum_{j=1}^{n-1} \hat{x}_j \cdot \mathbb{I}\{a_j + d_j > a_n\} \cdot \mathbb{I}\{i_j^* = i\}, \forall i \in [k]$;
  Let $i_n^* = \arg\min_{i\in[k]} \{y_n^{(i)}\}$ and compute

$$\hat{x}_n = \arg\max_{x\in[0,1]} \; d_n \cdot x - \int_{y_n^{(i_n^*)}}^{y_n^{(i_n^*)}+x} \phi(\eta) d\eta; \tag{6}$$

  Sample a random seed $S_n \sim U(0,1)$;
  if $S_n \le \frac{\hat{x}_n}{1-y_n^{(i_n^*)}}$ *and unit* i_n^* *available in system* **then**
    $x_n = 1$; ▷ Allocate unit i_n^* to request n
  end
  else
    $x_n = 0$. ▷ Reject request n
  end
end

Overview of challenges and design principles of ϕ. Designing a competitive pricing function ϕ for DOP-ϕ-VARIABLE involves two primary challenges. First, unlike the relax-and-round approach adopted for kRental-Fixed, Algorithm 4 integrates the relaxation and rounding steps into a unified procedure, wherein the rounding phase directly influences the structure of the pricing function ϕ. Similar design principles have been employed in related works, such as [8, 10]. Second, as with Algorithm 3 and in contrast to the approaches in [7, 22], Algorithm 4 adopts a myopic, duration-oblivious decision-making strategy. It bases allocation decisions solely on the current utilization levels $\{y_n^{(i)}\}_{i\in[k]}$ at the arrival time a_n of request n, rather than considering the entire rental duration. This duration-oblivious pricing scheme may initially appear counterintuitive, as the pseudo-cost term in Eq. (6) does not depend on the rental duration d_n. Nevertheless, as elaborated in the next section, the pricing function ϕ is carefully designed to implicitly capture potential fluctuations in utilization throughout the entire duration of request n. In essence, our goal is to *maintain a simple and cognitively lightweight pricing scheme, while embedding the complexity into the internal structure of the pricing function ϕ itself.*

4.4 Theoretical Guarantees of DOP-ϕ-VARIABLE

We employ the LP-free certificate method developed by [13] to establish a performance guarantee for DOP-ϕ-VARIABLE. This involves constructing a system of linear constraints parameterized by α, such that the existence of a feasible solution certifies α-competitiveness. Satisfying this system effectively imposes a distinct constraint for

each request n. To address these constraints, we derive a system of differential inequalities that guide the design of the pricing function ϕ. For each possible utilization level at the arrival of request n, we consider the worst-case scenario for utilization fluctuations over its rental interval. Consequently, the design constraints for ϕ naturally arise from this worst-case analysis, embedding such fluctuations explicitly into the pricing function. The following theorem formalizes this result.

Theorem 4. *DOP-ϕ-VARIABLE is α-competitive for the kRental-Variable problem, provided that the pricing function ϕ is increasing and satisfies the following inequalities for all $d_n \in [d_{\min}, d_{\max}]$ and $n \in [N]$:*

$$\begin{cases} \displaystyle\int_{\eta=y_1}^{2y_1} \frac{2\alpha}{3}\phi(\eta)\,d\eta + \frac{\alpha}{3}d_n\big(\phi^*(d_n) - 2y_1\big) \;\geq\; d_n, \forall\, y_1 \in \left[0, \tfrac{\phi^*(d_n)}{2}\right], \\ \displaystyle\frac{\alpha}{3}d_n\phi^*(d_n) - \frac{\alpha}{3}y_2\big(d_n - \phi(y_2)\big) \;\geq\; d_n, \forall\, y_2 \in [0, \phi^*(d_n)], \end{cases} \tag{7}$$

where $\phi^*(d_n) = \sup\{x \in [0,1] \mid \phi(x) \leq d_n\}$.

The proof sketch of the above theorem is given in Appendix D. Our proof utilizes the LP-free certificate method of [13]. By Theorem 4, designing an algorithm with guaranteed performance reduces to constructing a pricing function ϕ that satisfies the constraints in Eqs. (7), while minimizing the competitive ratio α. However, solving ϕ is challenging in general because it involves solving a system of *delayed differential inequalities* with an inverse term.

Tightness of results from DOP-ϕ-VARIABLE. Based on Theorem 4, we can show that DOP-ϕ-VARIABLE is $3 \cdot (1 + \ln(\frac{d_{\max}}{d_{\min}}))$-competitive for the kRental-Variable problem when the pricing function ϕ is designed as: $\phi(y) = d_{\min} \cdot \exp([1+\ln(\frac{d_{\max}}{d_{\min}})] \cdot y - 1), \forall y \in [0,1]$. This also matches the best-known competitive ratio for kRental-Variable in prior work [12]. On the other hand, we can show that no online algorithm can achieve a competitive ratio better than $1 + \ln\left(\frac{d_{\max}}{d_{\min}}\right)$ for the kRental-Variable problem, using a construction of hard instances similar to the approach in [21] to establish the lower bound.

Comparison to prior work. To demonstrate the significance of our results, we further develop a numerical method in to solve Eqs. (7) and obtain a pricing function. In Figure 3, we show that the competitive ratio achieved by DOP-ϕ-VARIABLE (blue curve) surpasses the best known bound of $3\,(1 + \ln(d_{\max}/d_{\min}))$ from [12]. As a relax-and-round algorithm, we also investigate the performance of the fractional solution from the relaxation step. Recall that DOP-ϕ-VARIABLE adopts an integrated design of relaxation and rounding. Thus, deriving the fractional solution without the rounding step requires modifying the LP certificate conditions in Eq. (7). Using a similar approach as in the derivation of the differential inequalities in Theorem 4, we obtain a new design for the pricing function in the fractional setting and refer to the resulting algorithm, which uses this pricing function to generate fractional solutions, as DOP-ϕ-VARIABLE-FRACTIONAL. The competitive ratio of DOP-ϕ-VARIABLE-FRACTIONAL is illustrated in green curve in Figure 3. We observe that its performance is comparable to $4 + \ln(d_{\max}/d_{\min})$, which is the best-known competitive ratio of kRental-Variable in the large-inventory regime (as $k \to \infty$) [7]. Furthermore, in comparison to [7], both our proposed DOP-ϕ-VARIABLE and its fractional version DOP-ϕ-VARIABLE-FRACTIONAL employ a far simpler, *duration-oblivious* pseudo-utility maximization rule: the fractional decision for each arriving request depends only on the utilization level at its arrival, whereas the algorithm based on [7] must additionally track utilization fluctuations over the entire rental interval of each request.

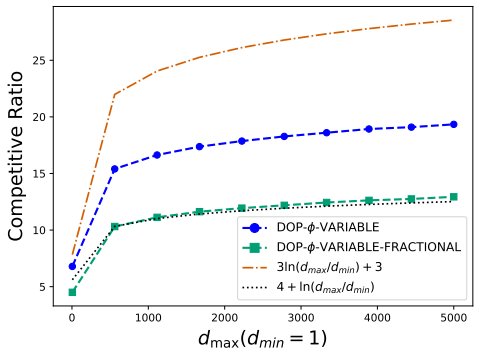

Figure 3: The blue and green curves illustrate the competitive ratios of Algorithm 4 in the integral and fractional settings (using the numerically derived pricing function ϕ). The brown and black curves serve as benchmarks, which correspond to the best known bound for general k [12] and the bound for large k [7].

5 Conclusion and Future Directions

We studied two adversarial online allocation problems involving reusable resources: the online k-rental problem with fixed rental durations (kRental-Fixed) and a more general variant with variable rental durations (kRental-Variable). To address both settings, we proposed a unified relax-and-round framework that leverages a price-based approach to compute fractional solutions and novel online rounding schemes to convert them into integral decisions. For kRental-Fixed, we integrated the price-based strategy with a new lossless online rounding scheme, 1-OCR, achieving the optimal competitive ratio. In the more general kRental-Variable setting, where lossless rounding is provably unattainable, we developed a limited-correlation rounding strategy that achieves an order-optimal guarantee. This work opens several promising directions for future research. In particular, two open problems stand out as especially compelling: (i) designing more powerful rounding schemes that further narrow the gap between fractional and integral solutions in the variable-duration setting, and (ii) establishing tight upper bounds on the best achievable γ-OCR under variable rental durations, despite the impossibility of attaining 1-OCR as shown by our result.

Acknowledgments

Xiaoqi Tan acknowledges support from Alberta Machine Intelligence Institute (Amii), Alberta Major Innovation Fund, and NSERC Discovery Grant RGPIN-2022-03646.

References

[1] Baruch Awerbuch, Yossi Azar, and Serge Plotkin. 1993. Throughput-competitive on-line routing. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. IEEE, 32–40.

[2] Niv Buchbinder and Joseph (Seffi) Naor. 2009. Online Primal-Dual Algorithms for Covering and Packing. *Mathematics of Operations Research* 34, 2 (2009), 270–286. http://www.jstor.org/stable/40538382

[3] Ying Cao, Bo Sun, and Danny HK Tsang. 2022. Online network utility maximization: Algorithm, competitive analysis, and applications. *IEEE Transactions on Control of Network Systems* 10, 1 (2022), 274–284.

[4] Ian DeHaan and Zachary Friggstad. 2023. Approximate Minimum Sum Colorings and Maximum k-Colorable Subgraphs of Chordal Graphs. In *Algorithms and Data Structures*, Pat Morin and Subhash Suri (Eds.). Springer Nature Switzerland, Cham, 326–339.

[5] Steven Delong, Alireza Farhadi, Rad Niazadeh, and Balasubramanian Sivan. 2022. Online Bipartite Matching with Reusable Resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*. Association for Computing Machinery.

[6] John P. Dickerson, Karthik A. Sankararaman, Aravind Srinivasan, and Pan Xu. 2021. Allocation Problems in Ride-sharing Platforms: Online Matching with Offline Reusable Resources. *ACM Trans. Econ. Comput.* 9, 3, Article 13 (June 2021), 17 pages. doi:10.1145/3456756

[7] Farbod Ekbatani, Yiding Feng, Ian Kash, and Rad Niazadeh. 2025. Online Job Assignment. In *International Conference on Internet and Network Economics (WINE '25)*.

[8] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. 2022. Edge-weighted online bipartite matching. *J. ACM* 69, 6 (2022), 1–35.

[9] Ulrich Faigle, Renate Garbe, and Walter Kern. 1996. Randomized Online Algorithms for Maximizing Busy Time Interval Scheduling. *Computing* 56, 2 (1996), 95–104. doi:10.1007/BF02309339

[10] Ruiquan Gao, Zhongtian He, Zhiyi Huang, Zipei Nie, Bijun Yuan, and Yan Zhong. 2021. Improved Online Correlated Selection. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. doi:10.1109/FOCS52979.2021.00123

[11] Minos Garofalakis, Yannis Ioannidis, Banu Özden, and Avi Silberschatz. 2002. Competitive On-line Scheduling of Continuous-Media Streams. *J. Comput. System Sci.* 64, 2 (2002), 219–248. doi:10.1006/jcss.2001.1799

[12] Shashank Goyal and Diwakar Gupta. 2020. The Online Reservation Problem. *Algorithms* 13, 10 (2020). doi:10.3390/a13100241

[13] Vineet Goyal, Garud Iyengar, and Rajan Udwani. 2024. Asymptotically Optimal Competitive Ratio for Online Allocation of Reusable Resources. *Operations Research* (2024). arXiv:https://doi.org/10.1287/opre.2021.0695 doi:10.1287/opre.2021.0695

[14] Diwakar Gupta and Fei Li. 2016. Reserve driver scheduling. *IIE Transactions* 48, 3 (2016), 193–204. arXiv:https://doi.org/10.1080/0740817X.2015.1078016 doi:10.1080/0740817X.2015.1078016

[15] Tianming Huo and Wang Chi Cheung. 2023. Online Reusable Resource Assortment Planning with Customer-Dependent Usage Durations. SSRN Working Paper. doi:10.2139/ssrn.4504713

[16] Hossein Nekouyan Jazi, Bo Sun, Raouf Boutaba, and Xiaoqi Tan. 2025. Posted Price Mechanisms for Online Allocation with Diseconomies of Scale. In *Proceedings of the ACM Web Conference 2025 (WWW '25)*.

[17] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. [n. d.]. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing (STOC '90)*.

[18] Richard J. Lipton and Andrew Tomkins. 1994. Online interval scheduling. In *ACM-SIAM Symposium on Discrete Algorithms*. https://api.semanticscholar.org/CorpusID:20204079

[19] Will Ma. 2024. Randomized Rounding Approaches to Online Allocation, Sequencing, and Matching. arXiv:2407.20419 [cs.DS]

[20] Paat Rusmevichientong, Mika Sumida, Huseyin Topaloglu, and Yicheng Bai. 2023. Revenue Management with Heterogeneous Resources: Unit Resource Capacities, Advance Bookings, and Itineraries over Time Intervals. *Oper. Res.* 71, 6 (Nov. 2023), 2196–2216. doi:10.1287/opre.2022.2427

[21] Bo Sun, Hossein Nekouyan Jazi, Xiaoqi Tan, and Raouf Boutaba. 2024. Static Pricing for Online Selection Problem and its Variants. In *The 20th Conference on Web and Internet Economics, WINE 2024, Edinburgh, United Kingdom Proceedings ,December 2-5, 2024*. Springer. https://arxiv.org/abs/2410.07378

[22] Bo Sun, Lin Yang, Mohammad Hajiesmaili, Adam Wierman, John CS Lui, Don Towsley, and Danny HK Tsang. 2022. The online knapsack problem with departures. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS '22)* 6, 3 (2022), 1–32.

[23] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. 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 (SIGMETRICS '15)* (Portland, Oregon, USA). 3–15.

[24] Zijun Zhang, Zongpeng Li, and Chuan Wu. 2017. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS '17)* 1, 1 (2017), 1–26.

Appendix

A Proof of Theorem 1

We consider two cases based on the relation between $1-p_n$ and $\hat{x}_n$.

Case I: $1-p_n \geq \hat{x}_n$. In this case, the algorithm assigns ball m_n to player n if ball m_n is available and $r \in [p_n, p_n+\hat{x}_n)$. Let E_{m_n} denote the event that ball m_n is available upon the arrival of the n-th player at time a_n, and let E denote the event $r \in [p_n, p_n+\hat{x}_n)$. Then the probability that ball m_n is allocated to player n is

$$\Pr[E_{m_n} \cap E].$$

Based on Proposition 2, if the random sample $r \in [p_n, p_n+\hat{x}_n)$, then ball m_n is guaranteed to be available at the arrival of player n. So if E occurs, ball m_n is available. Consequently,

$$\begin{aligned}\Pr[E_{m_n} \cap E] &= \Pr[E] \cdot \Pr[E_{m_n} \mid E] = \Pr[E] \\ &= \Pr[r \in [p_n, p_n+\hat{x}_n)] = \hat{x}_n,\end{aligned}$$

where in above the last inequality follows from the fact that random seed r is sampled uniformly from the range $[0,1]$. Hence, when $\hat{x}_n < 1-p_n$, player n gets ball m_n with probability $\hat{x}_n$.

Case II: $1-p_n < \hat{x}_n$. In this case, the algorithm first attempts to allocate ball m_n if it is available and if $r \in [p_n, 1]$. If this allocation does not occur, it then attempts to allocate ball m_n+1 provided that it is available and $r \in [0, \hat{x}_n-1+p_n)$. Let E_{m_n} denote the event that ball m_n is available at time a_n, and let E_1 denote the event $r \in [p_n, 1]$. Similarly, let E_{m_n+1} denote the event that ball m_n+1 is available, and let E_2 denote the event $r \in [0, \hat{x}_n-1+p_n)$. Then the total probability that a ball is allocated to player n is given by $\Pr[E_{m_n} \cap E_1] + \Pr\left[E_{m_n+1} \cap E_2 \cap (E_{m_n} \cap E_1)'\right]$. Based on Proposition 2, if E_1 occurs, ball m_n is available, and $\Pr[E_{m_n} \cap E_1] = \Pr[E_1] = 1-p_n$. Similarly, the event $E_{m_n+1} \cap E_2$ occurs with probability $\Pr[E_{m_n+1} \cap E_2] = \Pr[E_2] = \hat{x}_n-1+p_n$. Since the ranges $[p_n, 1]$ and $[0, \hat{x}_n-1+p_n)$ do not overlap (because $\hat{x}_n \leq 1$), we have

$$\begin{aligned}&\Pr\left[E_{m_n+1} \cap E_2 \cap (E_{m_n} \cap E_1)'\right] \\ &= \Pr\left[r \in \left([0, \hat{x}_n-1+p_n) \cap [0, p_n)\right)\right] \\ &= \Pr\left[r \in [0, \hat{x}_n-1+p_n)\right] \\ &= \hat{x}_n-1+p_n.\end{aligned}$$

Thus, the total probability that a ball is allocated to player n is

$$\begin{aligned}&\Pr[E_{m_n} \cap E_1] + \Pr\left[E_{m_n+1} \cap E_2 \cap (E_{m_n} \cap E_1)'\right] \\ &= (1-p_n) + (\hat{x}_n-1+p_n) \\ &= \hat{x}_n.\end{aligned}$$

Thus, in both cases, each player n gets a ball with the desired probability $\hat{x}_n$, which proves that Algorithm 2 is a lossless online scheme.

B Impossibility Result for Lossless Rounding in the Variable Duration Setting

Consider the γ-OCR problem introduced in Section 2.1, where each player rents a ball for a fixed duration d that is identical across all players and independent of their identity. In this setting, the rental period is uniform and known in advance.

In the following, we study a variant of this problem in which the rental duration is player-dependent and may vary across players.

Specifically, each player requests a rental duration that is revealed only upon their arrival. Thus, the algorithm receives information about a player's rental duration in an online fashion. A more formal definition of the OCR problem under this variable-duration setting is as follows.

Definition 2 (γ-OCR-V). *Consider a set of k identical balls, each uniquely labeled from the set $\{1, 2, \ldots, k\}$. Each ball can be assigned to a player for a variable duration, determined by the player's requested rental period, after which the ball becomes available for reassignment. A sequence of N players arrives one by one, with each player n characterized by a triple $(a_n, \hat{x}_n, d_n)$, where a_n is the arrival time, $\hat{x}_n \in [0, 1]$ is the target probability with which a ball should be assigned to player n, and d_n is the requested rental duration. For any $\gamma \in [0, 1]$, a γ-OCR-V algorithm, given the input sequence $\{(\hat{x}_n, a_n, d_n)\}_{n\in[N]}$, assigns a ball to each player $n \in [N]$ with probability at least $\gamma\hat{x}_n$. Similar to the definition of γ-OCR, for any given input instance of γ-OCR-V, $\{(\hat{x}_n, a_n, d_n)\}_{n\in[N]}$, we impose the following condition:*

$$\hat{x}_n \le \min\left(1, k - \sum_{j\in[n-1]} \hat{x}_j \cdot \mathbb{I}_{\{a_j+d_j>a_n\}}\right), \qquad \forall n \in [N]. \quad (8)$$

Proof of Theorem 3. We proceed by contradiction. Suppose, for the sake of contradiction, that a (possibly randomized) online algorithm ALG is *lossless* for the γ-OCR-V problem; that is, it achieves $\gamma = 1$ on every instance. Consider the following instance with inventory of $k = 2$ balls and:

$$\mathcal{I} = \Bigg\{ \underbrace{(a_1 = 1, x_1 = 0.5, d_1 = 5)}_{P_1}, \underbrace{(a_2 = 2, x_2 = 0.5, d_2 = 7)}_{P_2}, \underbrace{(a_3 = 5.5, x_3 = \tfrac{2}{3}, d_3 = 9)}_{P_3}, \underbrace{(a_4 = 6, x_4 = \tfrac{1}{3}, d_4 = 8)}_{P_4}, \underbrace{(a_5 = 8, x_5 = \tfrac{1}{2}, d_5 = 10)}_{P_5}, \underbrace{(a_6 = 14, x_6 = \tfrac{5}{6}, d_6 = 10)}_{P_6} \Bigg\}.$$

The above instance clearly satisfies the constraint in Eq. (8). Let $A_n^{(i)}$ ($i = 1, 2$) be the event that the i-th ball is allocated to player n by the online algorithm ALG upon processing the player's request. Likewise, let $E_t^{(i)}$ ($i = 1, 2$) denote the event that the i-th ball is still available at time t.

Next, we prove the following claim:

Lemma 2. *For the algorithm* ALG *to be lossless, we must have:*

$$\mathbb{P}[A_1^{(1)} \cap A_2^{(2)}] \; + \; \mathbb{P}[A_1^{(2)} \cap A_2^{(1)}] = 0$$

Proof. We prove the claim by contradiction. Suppose the lemma fails, i.e., $\mathbb{P}[A_1^{(1)} \cap A_2^{(2)}] + \mathbb{P}[A_1^{(2)} \cap A_2^{(1)}] > 0$.

Then the probability that *at least one* of the two balls is still available immediately after the decision for player 2 satisfies

$$\begin{aligned}\mathbb{P}[E_2^{(1)} \cup E_2^{(2)}] &= 1 - \mathbb{P}[(A_1^{(1)} \cup A_2^{(1)}) \cap (A_1^{(2)} \cup A_2^{(2)})] \\ &= 1 - \mathbb{P}[(A_1^{(1)} \cap A_1^{(2)}) \cup (A_2^{(1)} \cap A_2^{(2)})] \\ &= 1 - \mathbb{P}[A_1^{(1)} \cap A_2^{(2)}] - \mathbb{P}[A_1^{(2)} \cap A_2^{(1)}] < 1,\end{aligned}$$

where the strict inequality follows from the contradictory assumption.

Next, let us create a new instance $\mathcal{I}'$ that is identical to $\mathcal{I}$ for players P_1 and P_2, but replaces the remaining sequence by a single player P_3' with $(a_3 = 3, x_3 = 1, d_3 = 10)$. The feasibility condition (8) is easily verified for $\mathcal{I}'$. Because the first two players in $\mathcal{I}$ and $\mathcal{I}'$ are identical, algorithm ALG follows the same random path on both instances for the first two players. Thus, at $a_3 = 3$ in $\mathcal{I}'$ we still have $\mathbb{P}[E_2^{(1)} \cup E_2^{(2)}] < 1$, meaning that with probability less than one either of balls are available at arrival of request P_3'. Yet a lossless algorithm must allocate a ball to P_3' with probability $x_3 = 1$, an impossibility. Hence our initial assumption is false, and the lemma follows. □

Next, consider the arrival of the third player P_3 in instance $\mathcal{I}$, with parameters $a_3 = 5$, $x_3 = \frac{2}{3}$, and $d_3 = 9$. We prove the following statement.

Lemma 3. *For* ALG *to remain lossless on $\mathcal{I}$, it must hold that $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] = \frac{1}{6}$.*

Proof. First, let us prove that $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] \le \frac{1}{6}$. The proof is again by contradiction. Assume that the above inequality does not hold. Consider the instance $\mathcal{I}'$, in which the first three players are identical to those of instance $\mathcal{I}$, but the fourth player P_4' is characterised as $(a_4 = 7, d_4 = 10, x_4 = \frac{5}{6})$. It can be verified that the feasibility constraint in Eq. (1) is satisfied for every player. Because the first three arrivals in $\mathcal{I}$ and $\mathcal{I}'$ are identical, the online algorithm ALG behaves the same on both instances up to arrival of P_4'. Consequently, when P_4' arrives in $\mathcal{I}'$,

$$\begin{aligned}\mathbb{P}[E_7^{(1)} \cup E_7^{(2)}] &= 1 - \mathbb{P}[(A_2^{(1)} \cup A_3^{(1)}) \cap (A_2^{(2)} \cup A_3^{(2)})] \\ &= 1 - \mathbb{P}[(A_2^{(1)} \cap A_3^{(2)}) \cup (A_3^{(1)} \cap A_2^{(2)})] \\ &= 1 - \mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] - \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] < \frac{5}{6},\end{aligned}$$

where the last strict inequality is by the contradictory assumption to have $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] > \frac{1}{6}$. The allocation of P_1 does not influence the calculation above, because by $a_4 = 7$ any ball assigned to P_1 has already returned to the system. Hence, at the arrival of player P_4' in $\mathcal{I}'$ the probability that either ball is still available, $\mathbb{P}[E_7^{(1)} \cup E_7^{(2)}]$, is strictly less than $\frac{5}{6}$. Therefore, ALG cannot allocate a ball to P_4 with the required target probability $\frac{5}{6}$, contradicting the lossless property. Thus we must have $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] \le \frac{1}{6}$.

Next, let us prove that $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] \ge \frac{1}{6}$, after which we can conclude that the sum equals $\frac{1}{6}$. First, we compute the following probability:

$$\begin{aligned}&\mathbb{P}[(A_1^{(1)} \cup A_1^{(2)}) \cup (A_2^{(1)} \cup A_2^{(2)})] \\ &= \mathbb{P}[(A_1^{(1)} \cup A_1^{(2)})] + \mathbb{P}[(A_2^{(1)} \cup A_2^{(2)})] \\ &\quad - \mathbb{P}[(A_1^{(1)} \cup A_1^{(2)}) \cap (A_2^{(1)} \cup A_2^{(2)})] \\ &= x_1 + x_2 - \mathbb{P}[(A_1^{(1)} \cap A_2^{(2)}) \cup (A_1^{(2)} \cap A_2^{(1)})] \\ &= 1 - \mathbb{P}[(A_1^{(1)} \cap A_2^{(2)})] - \mathbb{P}[(A_1^{(2)} \cap A_2^{(1)})] = 1,\end{aligned}$$

where the last equality follows from Lemma 2. By the same reasoning we have $\mathbb{P}[(A_1^{(1)} \cup A_1^{(2)}) \cap (A_2^{(1)} \cup A_2^{(2)})] = 0$. Since $\mathbb{P}[A_3^{(1)} \cup A_3^{(2)}] = x_3 = \frac{2}{3}$, $\mathbb{P}[(A_1^{(1)} \cup A_1^{(2)})] = \frac{1}{2}$, and $\mathbb{P}[(A_1^{(1)} \cup A_1^{(2)}) \cup (A_2^{(1)} \cup A_2^{(2)})] = 1$, we must have $\mathbb{P}[(A_3^{(1)} \cup A_3^{(2)}) \cap (A_2^{(1)} \cup A_2^{(2)})] \ge \frac{1}{6}$.

Combining this with the earlier inequality establishes $\mathbb{P}[A_2^{(1)} \cap A_3^{(2)}] + \mathbb{P}[A_2^{(2)} \cap A_3^{(1)}] = \frac{1}{6}$. Thus, the proof is completed. □

Next, consider the arrival of player P_4 in instance $\mathcal{I}$, with parameters $a_4 = 6$, $x_4 = \frac{1}{3}$, and $d_4 = 7$.

Lemma 4. *For* ALG *to be lossless, after the arrival of player P_4 we must have* $\mathbb{P}[A_3^{(1)} \cap A_4^{(2)}] + \mathbb{P}[A_3^{(2)} \cap A_4^{(1)}] = 0$.

Proof. In case the above equality is not satisfied, a request P_5 could arrive at time $a_5 = 10$ with target probability $x_5 = 1$, and ALG would not be lossless since $\mathbb{P}[E_{10}^{(1)} \cup E_{10}^{(2)}] < 1$. □

Next, consider the arrival of player P_5 at time $a_5 = 8$, with target probability $x_5 = \frac{1}{2}$ and $d_5 = 10$. Let us refer to the time right before the arrival of fifth player as 8^-.

Lemma 5. *After the arrival of player P_5, $\mathbb{P}[A_3^{(1)} \cap A_5^{(2)}] + \mathbb{P}[A_3^{(2)} \cap A_5^{(1)}] \geq \frac{1}{2}$.*

Proof. From the previous two lemmas 3 and 4, we have

$$\mathbb{P}[(A_2^{(1)} \cup A_2^{(2)}) \cap (A_3^{(1)} \cup A_3^{(2)})] = \frac{1}{6},$$
$$\mathbb{P}[(A_3^{(1)} \cup A_3^{(2)}) \cap (A_4^{(2)} \cup A_4^{(1)})] = 0.$$

Let us define C_1, C_2, and C_3 as follows:

$$C_1 = \left(A_3^{(1)} \cup A_3^{(2)}\right) \cap \left(A_2^{(2)} \cup A_2^{(1)}\right)',$$
$$C_2 = \left(A_2^{(2)} \cup A_2^{(1)}\right) \cap \left(A_3^{(1)} \cup A_3^{(2)}\right)',$$
$$C_3 = \left(A_3^{(1)} \cup A_3^{(2)}\right) \cap \left(A_2^{(2)} \cup A_2^{(1)}\right).$$

These events are pairwise disjoint and $\mathbb{P}[C_1 \cup C_2 \cup C_3] = 1$.

Claim 1. $\mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_2] = 0$.

Proof. From Lemma 4 we have $\mathbb{P}[(A_3^{(1)} \cup A_3^{(2)}) \cap (A_4^{(2)} \cup A_4^{(1)})] = 0$, thus it follows that $\mathbb{P}[C_1 \cap (A_4^{(2)} \cup A_4^{(1)})] = 0$ and $\mathbb{P}[C_3 \cap (A_4^{(2)} \cup A_4^{(1)})] = 0$. Note that we have $\mathbb{P}[A_4^{(2)} \cup A_4^{(1)}] = \mathbb{P}[C_2] = \frac{1}{3}$, so $A_4^{(2)} \cup A_4^{(1)} = C_2$. Next,

$$\begin{aligned}
C_2 &= C_2 \cap C_2 \\
&= \left((A_4^{(2)} \cup A_4^{(1)}) \cap ((A_2^{(2)} \cup A_2^{(1)}) \cap (A_3^{(1)} \cup A_3^{(2)})')\right) \\
&= \left((A_4^{(2)} \cup A_4^{(1)}) \cap (A_2^{(2)} \cup A_2^{(1)})\right) \\
&\quad \cap \left((A_4^{(2)} \cup A_4^{(1)}) \cap (A_3^{(1)} \cup A_3^{(2)})'\right) \\
&= \left((A_4^{(2)} \cup A_4^{(1)}) \cap (A_2^{(2)} \cup A_2^{(1)})\right) \cap (A_4^{(2)} \cup A_4^{(1)}) \\
&= \left((A_4^{(2)} \cup A_4^{(1)}) \cap (A_2^{(2)} \cup A_2^{(1)})\right) \\
&= (A_2^{(2)} \cap A_4^{(1)}) \cup (A_2^{(1)} \cap A_4^{(2)}),
\end{aligned}$$

where the third equality uses $\mathbb{P}[(A_3^{(1)} \cup A_3^{(2)}) \cap (A_4^{(2)} \cup A_4^{(1)})] = 0$. Based on the above equations, under the event C_2, either $A_2^{(2)} \cap A_4^{(1)}$ or $A_2^{(1)} \cap A_4^{(2)}$ occurs, so neither ball 1 nor ball 2 is available. Therefore $\mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_2] = 0$. This completes the proof. □

Next, we can verify that $\mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_3] = 0$. In fact, $C_3 = (A_3^{(1)} \cup A_3^{(2)}) \cap (A_4^{(2)} \cup A_4^{(1)}) = (A_3^{(1)} \cap A_4^{(2)}) \cup (A_3^{(2)} \cap A_4^{(1)})$. Thus, under C_3, either $(A_3^{(1)} \cap A_4^{(2)})$ or $(A_3^{(2)} \cap A_4^{(1)})$ occurs, so neither ball 1 nor ball 2 is available, and $\mathbb{P}[E_4^{(1)} \cup E_4^{(2)} \mid C_3] = 0$.

Since at the arrival of P_5 we have $\sum_{i\in[4]} x_i \cdot \mathbb{I}_{\{a_i+d_i \geq a_5\}} = \frac{3}{2}$, with probability $\frac{1}{2}$ one of the two balls is available, and

$$\begin{aligned}
\frac{1}{2} &= \mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)}] \\
&= \mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_1] + \mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_2] + \mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_3] \\
&= \mathbb{P}[E_{8^-}^{(1)} \cup E_{8^-}^{(2)} \mid C_1] = \mathbb{P}[C_1].
\end{aligned}$$

Thus, we have $E_{8^-}^{(1)} \cup E_{8^-}^{(2)} = C_1$.

At the arrival of P_5, if either ball is available, the other ball is already allocated to P_3, since $C_1 = (A_3^{(1)} \cup A_3^{(2)}) \cap (A_2^{(2)} \cup A_2^{(1)})'$. Therefore, a lossless ALG must allocate one of the two balls to P_5 with probability $\frac{1}{2}$, and

$$\begin{aligned}
&\mathbb{P}[(A_5^{(1)} \cup A_5^{(2)}) \cap (A_3^{(1)} \cup A_3^{(2)})] \\
&= \mathbb{P}[(A_3^{(1)} \cap A_5^{(2)}) \cup (A_3^{(2)} \cup A_5^{(1)})] \\
&= \mathbb{P}[A_3^{(1)} \cap A_5^{(2)}] + \mathbb{P}[A_3^{(2)} \cup A_5^{(1)}] \\
&= \frac{1}{2}.
\end{aligned}$$

This completes the proof. □

Now consider the arrival of the sixth player with ($a_6 = 13.5, x_6 = \frac{5}{6}, d_6 = 10$). It can be verified that

$$\begin{aligned}
\mathbb{P}[E_6^{(1)} \cup E_6^{(2)}] &= 1 - \mathbb{P}[(A_3^{(1)} \cup A_5^{(1)}) \cap (A_3^{(2)} \cup A_5^{(2)})] \\
&= 1 - \mathbb{P}[(A_3^{(1)} \cap A_5^{(2)}) \cup (A_3^{(2)} \cap A_5^{(1)})] \\
&= \frac{1}{2},
\end{aligned}$$

where the last equality follows from Lemma 5. Thus, at the arrival of player P_6, with probability equal to $\frac{1}{2}$ either ball is available, and the algorithm cannot allocate a ball with target probability $\frac{5}{6}$ losslessly to player P_6. We thus complete the proof.

C Proof Sketch of Theorem 2

We use an online primal-dual approach to establish the competitive ratio of Algorithm 3. Consider the dual LP corresponding to the primal LP in Eq. (2):

$$\min_{\lambda,\mathbf{u}} \quad \sum_{n\in[N]} u_n + k \cdot \sum_{n\in[N]} \lambda_n, \tag{9}$$

$$\text{s.t.} \quad u_n + \sum_{j=n}^{N} \lambda_j \mathbf{1}_{\{a_n+d>a_j\}} \geq v_n, \quad \forall n \in [N], \tag{10}$$

$$\lambda_n \geq 0, \quad \forall n \in [N]. \tag{11}$$

Step-I: Design of dual solution, $\{u_n^{\mathsf{ALG}}, \lambda_n^{\mathsf{ALG}}\}_{n\in[N]}$, and feasibility of the dual solution. Let us initialize all dual variables to zero. For each rental request n, let $\hat{x}_n$ be the fractional allocation chosen by Algorithm 3. We then perform the following updates:

$$\lambda_{v_n^*}^{\mathsf{ALG}} = \lambda_{v_n^*}^{\mathsf{ALG}} + F \cdot \int_{\eta=\frac{y_n}{k}}^{\frac{y_n+\hat{x}_n}{k}} \phi(\eta) d\eta, \tag{12}$$

$$u_n^{\mathsf{ALG}} = F \cdot \hat{x}_n \cdot \left(v_n - \phi\left(\frac{y_n + \hat{x}_n}{k}\right)\right), \tag{13}$$

where $v_n^* = \max\{j \geq n | a_j < a_n + d\}$ and $F = 1 + \ln\left(\frac{v_{\max}}{v_{\min}}\right)$.

According to the dual variable update in Eq. (12)-(13), and utilizing the fact that requests rental durations are fixed, we can prove that the dual constraint in Eq. (10) corresponding to each request n is satisfied.

Step-II: Proof of $\sum_{n\in[N]} u_n^{\mathsf{ALG}} + k\cdot\sum_{n\in[N]} \lambda_n^{\mathsf{ALG}} \le F\cdot\mathsf{ALG}(I)$. Instead of directly proving this inequality, we show that for each request n:

$$\Delta_n^{D^{\mathsf{ALG}}} \le F\Delta_n^{\mathsf{ALG}},$$

where $\Delta_n^{D^{\mathsf{ALG}}}$ denotes the increase in the dual objective value after updating the dual variables for request n via Eqs. (12) and (13), and Δ_n^{ALG} denotes the increase in the objective value of the algorithm after allocating $\hat{x}_n$ fraction of an item to request n. We skip the proof details due to space constraints.

By the above two steps, the proof of $1+\ln\left(\frac{d_{\min}}{d_{\max}}\right)$-competitiveness of DOP-$\phi$-FIXED follows, and thus the proof of Theorem 2 is completed.

D Proof Sketch of Theorem 4

To establish the α-competitiveness of Algorithm 4, we apply the LP-free certificate approach developed [13]. We define a system of linear constraints that acts as a certificate of α-competitiveness when feasible. The variables in this system include $\{\{u_n, \lambda_n^{(i)}\}_{n\in[N]}, \theta_i\}_{i\in[k]}$. The constraints are given by:

$$\sum_{n\in[N]} u_n + \sum_{i\in[k]} \theta_i \le \alpha\cdot\mathsf{ALG}(I), \tag{14}$$

$$\theta_i + \sum_{n\in P_i} u_n \ge \sum_{n\in P_i} d_n = \mathsf{OPT}_i, \qquad \forall i\in[k], \tag{15}$$

where $\mathsf{ALG}(I)$ denotes the expected performance of the algorithm on instance I. For each i, the set P_i consists of the requests to whom the optimal clairvoyant algorithm allocates the i-th unit of the resource, and $\mathsf{OPT}_i = \sum_{n\in P_i} d_n$. Clearly, if the above constraints hold, then:

$$\mathsf{OPT} = \sum_{i=1}^{k}\mathsf{OPT}_i \le \sum_{i=1}^{k}\Big(\theta_i + \sum_{n\in P_i} u_n\Big) \le \sum_{i\in[k]}\theta_i + \sum_{n\in[N]} u_n \le \alpha\cdot\mathsf{ALG}(I),$$

where the above set of inequalities follow from the certificate constraints. In what follows, we first describe how to design a feasible solution for the system of constraints. Initially, all variables are set to zero. After the arrival of the final request in instance I, we update the variables as follows. For each rental request n, let $\hat{x}_n$ be the fractional allocation variable determined by Algorithm 4. We then perform the following updates:

$$u_n = \frac{\alpha}{3} d_n \hat{x}_n, \tag{16}$$

$$\lambda_j^{(i_n^*)} = \lambda_j^{(i_n^*)} + \frac{\alpha}{3}\big(a_{j+1} - a_j\big)\hat{x}_n, \quad \forall j, n\le j < v_n^*, a_j < a_n + d_n, \tag{17}$$

$$\lambda_{v_n^*}^{(i_n^*)} = \lambda_{v_n^*}^{(i_n^*)} + \frac{\alpha}{3}\Big(2d_n - \big(a_{v_n^*} - a_n\big)\Big)\hat{x}_n, \tag{18}$$

where $v_n^* = \max\{j \ge n | a_j < a_n + d_n\}$. After updating the variables above for all the N requests, let us set the value of variables $\theta_i = \sum_{n\in[N]} \lambda_n^{(i_n^*)}$ for all $i\in[k]$.

First constraint of the system. Based on the construction of the variables detailed above, we can verify that the constraint in Eq. (14) holds. Due to space constraints, we skip the proof details, but it follows by considering the increase in both the right- and left-hand sides of Eq. (14) under the updates performed for each request n, together with the increase in the expected objective value of the algorithm after processing each request n.

Second set of constraints (Eq. (15)). The value of θ_i can be lower bounded by:

$$\theta_i = \sum_{n\in[N]} \lambda_n^{(i)} \ge \sum_{n\in P_i}\sum_{j=n}^{N} \lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\}.$$

The inequality holds because, for each request $n\in P_i$, the set $\{j\in[N] \mid a_n + d_n > a_j\}$ cannot intersect with any other request's set. Otherwise, the optimal offline algorithm would have allocated unit i to two requests at the same time, which is infeasible. Hence, for the left-hand side of Eq. (15), we have

$$\theta_i + \sum_{n\in P_i} u_n \ge \sum_{n\in P_i}\sum_{j=n}^{N}\lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\} + \sum_{n\in P_i} u_n \ge \sum_{n\in P_i}\Big(\sum_{j=n}^{N}\lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\} + u_n\Big).$$

Thus, it suffices to show that for each request $n\in P_i$,

$$\sum_{j=n}^{N}\lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\} + u_n \ge d_n. \tag{19}$$

Lemma 6. *For any instance of kRental-Variable, the left-hand side of the above inequality can be lower bounded as follows:*

$$\sum_{j=n}^{N}\lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\} + u_n \;\ge\; \frac{2\alpha}{3}y_1\phi(y_1) - \frac{\alpha}{3}y_2\phi(y_1) + \int_{y_1}^{y_2}\frac{2\alpha}{3}\phi(\eta)\,d\eta + \frac{\alpha}{3}d_n\max\Big\{0, \phi^*(d_n) - y_2\Big\}.$$

where in above y_1 *and* y_2 *are some instance-dependent values such that* $y_2\in\big(0,\phi^*(d_n)\big]$ *and* $y_1\in\big(0,y_2\big]$.

The proof of the above lemma utilizes the dual updates in Eqs. (16)–(18) to derive a lower bound by reducing to the worst-case instance of kRental-Variable that minimizes the left-hand side of Eq. (19).

Following from the design of the pricing function ϕ given in Theorem 4, it can be proven that:

$$\sum_{j=n}^{N}\lambda_j^{(i)}\mathbb{I}\{a_n + d_n > a_j\} + u_n \ge \frac{2\alpha}{3}y_1\phi(y_1) - \frac{\alpha}{3}y_2\phi(y_1) + \int_{y_1}^{y_2}\frac{2\alpha}{3}\phi(\eta)\,d\eta + \frac{\alpha}{3}d_n\max\Big\{0,\phi^*(d_n) - y_2\Big\} \ge d_n.$$

The proof of the second inequality follows by analyzing the critical points of the right-hand side of the first inequality with respect to the values y_1 and y_2, where the RHS is minimized. We conclude that the second set of constraints in Eq. (15) is satisfied by the design of the system variables in Eqs. (16)–(18).

Therefore, by combining all the above results, if the increasing pricing function ϕ satisfies the system of constraints in Theorem 4, it establishes the α-competitiveness of Algorithm 4.