Essays on

The Online Multiple Knapsack Problem &

The Online Reservation Problem

A THESIS

SUBMITTED TO THE FACULTY OF THE

UNIVERSITY OF MINNESOTA

BY

Shashank Goyal

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Advisor: Diwakar Gupta

June, 2018

Acknowledgements

I am very grateful to my advisor, Professor Diwakar Gupta, for giving me full independence in choosing my field of research, for motivating and guiding me whenever I was stuck, for not letting me be insincere with my work, and for being extremely patient, tolerant, and loving. I would also like to thank my committee members - Professors Andrew Odlyzko, Qie He, and Ankur Mani - for their their valuable inputs.

I would like to thank Professors Russell King, Reha Uzsoy, Vipul Jain, Qie He, Sasha Voronov, Michael Kay and Yunan Liu for being awesome teachers. My research will not have been possible at all without what I learned from them.

I would like to thank my labmates - Paola Martin, Yibin Chen, Xiaoxu Tang, Daniel Escobar Naranjo, Biswaranjan Mohanty, Yuhao Feng, and Brandon McConnell - for making my time at the graduate school so much fun! Although it is difficult to describe here how each one of my friends and family members helped me though my time as a grad student, I would like to thank all of them profusely, for it is they who made the good times great and the bad ones tolerable.

Most importantly, I would like to thank God for always helping me.

Dedication

This work is dedicated to all my teachers.

Abstract

We study two problems in this thesis, viz. the *online multiple knapsack problem* (OMKP) and the *online reservation problem* (ORP).

The OMKP has applications in revenue management and scheduling. There are multiple identical knapsacks. Items arrive one at a time, each having a value-density and a size. Upon arrival, an item must either be placed into a knapsack or turned away irrevocably, without any information about future items, except the largest possible values of item size and value-density. An accepted item yields a reward equal to its value. The decision maker seeks decision rules that have the best *competitive-ratio* (CR), which is the worst-case ratio of the optimal reward in the full-information case to the reward earned by the algorithm. We derive lower bounds on CR of any algorithm, analyze CRs of some known algorithms, and propose three new algorithms that have CRs significantly better than the known algorithms for specific ranges of parameter values. We also study the *online revenue management problem* where all items have sizes equal to the knapsack capacities.

In the ORP, we model many problem features faced by resource owner in sharing-economy platforms, many of which operate as follows. Owners list availability of resources (such as apartments, cars or tutoring services), prices, and contract-length limits. Customers propose contract start times and lengths. Owners decide immediately (or within a short time window) whether to accept or decline each proposal, even if the contract is for a future date. Accepted proposals generate a revenue for the owner. Declined proposals are lost. At any decision epoch, the owner has no information about future requests. The owner seeks easy-to-implement algorithms that have the best CR. We derive lower bounds on CR of any algorithm, analyze CRs of intuitive algorithms that are fair in the sense that they always accept a proposal whenever a resource is available, and propose two new algorithms

that have CRs significantly better than any fair algorithm for specific ranges of parameter values. We also study a special case of the ORP where all proposals arrive exactly at their start times.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we study two online problems — the online multiple knapsack problem and the online reservation problem. The adjective "online" has been used in a variety of contexts, and broadly speaking, it signifies lack of information about the future in a sequential decision making problem. To the contrary, the adjective "offline" indicates full information. Sequential decision making under uncertainty is a broad subject and not every problem that falls within this class is qualified by the adjective online, e.g. Markov decision processes (Howard 1960), stochastic programming (Birge and Louveaux 2011), and multi-armed bandit problems (Bubeck et al. 2012). Therefore, to put things in perspective, we begin by giving a very brief overview of the two problems that we study.

The Online Multiple Knapsack Problem (OMKP):

Suppose there are multiple identical knapsacks. Items arrive one at a time, each having a value density and a size. Value of an item is its value density multiplied by its size. The decision maker knows the largest possible values of item size and value density. A decision epoch arises when an item arrives. At that moment, an item must be either placed into a knapsack or turned away irrevocably, without any additional

information about future items. An accepted item yields a reward equal to its value. Declined items are lost. The decision maker seeks decision rules that have the best worst-case *relative performance*, which is also called the *competitive ratio* (CR). For a given problem instance, relative performance is the ratio of the optimal reward in the full-information case to the reward earned by the algorithm. We derive lower bounds on CR of any algorithm, analyze CRs of some known algorithms, and propose three new algorithms that have CRs significantly better than the known algorithms for specific ranges of parameter values. We also study the *online revenue management problem* (ORMP) in which all items have sizes equal to the knapsack capacities. Our approach results in smallest CR for the ORMP. The OMKP has many applications, including those in revenue management and scheduling, which have received attention in the Operations Management literature.

The Online Reservation Problem (ORP):

Resource owners participating in sharing economy platforms list availability of resources (such as apartments, cars or tutoring services), prices, and contract-length limits. Customers propose contract start times and lengths. Owners decide immediately (or within a short time window) whether to accept or decline each proposal, even if the contract is for a future date. Accepted proposals generate a revenue for the owner. Declined proposals are lost. At any decision epoch, the owner has no information about future requests. The owner seeks easy-to-implement algorithms that have the best CR. This is the online reservation problem that we study. We derive lower bounds on CR of any algorithm, analyze CRs of intuitive algorithms that are fair in the sense that they always accept a proposal whenever a resource is available, and propose two new algorithms that have CRs significantly better than

any fair algorithm for specific ranges of parameter values. We also study a special case of the ORP where all proposals arrive exactly at their start times.

The two problems described above have many similarities. In both cases, the problem instance is revealed sequentially and the decisions have to be made without any information about the future except some global constraints on problem parameters (e.g. the minimum and maximum item sizes). Furthermore, the goodness of an algorithm is measured in terms of its worst-case relative performance or CR. Because of these features, our research contributes to the growing literature on the *competitive analysis of online algorithms*.

Fiat and Woeginger (1998, Chapter 1, Section 3) give a historical background of this field. Some highlights from their description are as follows. Graham (1966) was perhaps the first paper to perform competitive analysis of an algorithm. Knuth (1968) gave a definition of what an online algorithm means, which is similar to today's conception of online algorithms — i.e. an algorithm that does not need the knowledge of the 'future' to produce an output. Johnson (1973), a seminal work on the bin packing problem, was the first work to use the adjective online in the context of approximation algorithms. The popularity of the field of competitive analysis was triggered by two papers — Sleator and Tarjan (1985a) and Sleator and Tarjan (1985b). The term *competitive analysis* itself was coined by Karlin et al. (1988). There are numerous surveys available of papers dealing with competitive analysis – see, e.g. Fiat and Woeginger (1998), Borodin and El-Yaniv (1998), Albers (2003), and Buchbinder et al. (2009).

One result, which is not specific to competitive analysis of online algorithms, but has been used extensively in numerous papers dealing with competitive analysis is the Yao's principle (Yao 1977). According to Krumke and Thielen (2015, Section 2.4), this concept has emerged as *the* standard technique for proving a lower bound on CR of any algorithm

for a given setting. We use it to do the same for the ORP.

With the advent of e-commerce and because of fluctuating costs and prices that depend on numerous socio-economic factors that are themselves challenging to ascertain and model, buyers and sellers may find themselves making on the spot decisions based solely upon the present state of the system. An auctioneer may have to accept or reject bids that come one at a time without having a clue as to what precise bids may come in the future. A customer may have to book flight tickets immediately, without knowing how prices will change in the future. A supply-chain manager may have to place order for raw materials knowing just the bounds within which the demand for the final product will lie. Competitive analysis is relevant in all such Operations Management problem settings.

According to Elmachtoub and Levi (2016), the framework of competitive analysis has only recently been applied in the field of Operations Management. Below we list some of the application domains where this technique has been applied, along with a few examples for each domain. Although these papers share a common theme that uncertainty characterization is unavailable at the decision making epoch and that CR is used for ranking algorithms, the specific models, analyses, and results in these papers are quite different from our work and from each other.

Lan et al. (2008), Ball and Queyranne (2009), and Lan et al. (2011) study various online revenue management problems. Wagner (2010), den Heuvel and Wagelmans (2010), and Wagner (2011) study online versions of inventory management and lot sizing problems. Ma and Simchi-Levi (2017) analyze an online resource allocation problem. Elmachtoub and Levi (2015) and Elmachtoub and Levi (2016) study online customer selection problems. An online assortment optimization problem is analyzed by Golrezaei et al. (2014) and an online traveling salesman problem by Jaillet and Wagner (2008).

We next proceed to Chapters 2 and 3 that deal with the OMKP and the ORMP, respectively. These are written as separate papers, each with its own introduction and literature survey. Also, the notation is specific to the chapter in which it is used and is defined at the beginning of the chapter.

Chapter 2

The Online Multiple Knapsack Problem

2.1 Introduction

Consider the following problem. There are a finite number of knapsacks (or bins) of known capacity. Jobs (or items) arrive one at a time. Each job is characterized by two attributes: value and size (or duration). Value density of a job is its value divided by its size. Upon arrival, a job must either be immediately placed into a knapsack or turned away irrevocably, without any information about the future jobs, except the largest possible values of size and value density. A job can be placed only into those knapsacks that have enough free space to accommodate the job. Those jobs that do not fit into any knapsack must be declined. An accepted job yields a reward equal to its value, whereas a declined job yields no reward. The decision maker seeks decision rules that have the best worst-case ratio of the optimal reward in the full information case to the reward earned by the decision rule. Some applications of the problem we just described are as follows:

Application 1: Integrated steel mills produce custom grades of steel slabs for their important customers. The chemical process of manufacturing steel requires a minimum batch size, which may be more than the order size (Denton et al. 2003). In that case, some slabs are leftover and subsequently listed for sale in the product catalog. Steel mills may receive orders for these leftover slabs made of special grades of steel one at a time. Some orders may be for non-integer number of slabs, in which case slabs are cut to meet customer requirements, if the order is accepted. Different customers may offer to pay different amounts for different-sized orders. There is insufficient historical data to develop an estimate for future orders for the leftover slabs. Then, the mathematical optimization problem that the steel mill faces is essentially the problem introduced above.

Application 2: The online version of the single-leg revenue management problem (Ball and Queyranne 2009) is also the problem described above when all jobs have sizes equal to the knapsack capacities. Each seat on a particular flight can be thought of as a knapsack, incoming booking requests are jobs, and the fare associated with each job is the value of that job.

Application 3: Hospitals assign blocks of operating-room (OR) time to surgeons according to a repeating schedule (Guerriero and Guido 2011). A particular surgeon may have several blocks in a planning period, e.g. a two-week period. The need for surgery is identified one patient at a time, and surgeons typically book cases in the earliest available planning period. Different surgical cases have different planned case lengths and values. The latter may be thought of as the negative of the cost of postponing the case to the next planning period. This cost includes the cost of prolonged discomfort to the patient, delay-induced worse medical outcomes, and delayed revenue for the surgeon. The surgeon would like to maximize the worst-case performance without knowing the full sequence of cases that may

be scheduled in the planning period. Upon treating OR blocks as knapsacks, and surgical cases as jobs, we obtain the problem described earlier.

The problem we study is the *online* version of the well known *multiple-knapsack problem* (MKP), and we refer to this online version as the OMKP. The *offline* version, MKP, is NP-hard (Kellerer et al. 2004) and makes accept/decline decisions based on full knowledge of all jobs in each problem instance. Although the knapsacks may have different capacities in some applications, we focus on the identical-knapsacks setting, which occurs frequently enough as evidenced by the above examples. In the OMKP, accepting a job results in an immediate reward, but it may also prevent the decision maker (DM) from accepting future high-value jobs. Furthermore, the DM has insufficient information about job-arrival patterns to develop a probabilistic estimate of the attributes of jobs that may arrive in the future. Therefore, he or she considers a max-min type objective. That is, the DM seeks an algorithm that results in the best worst-case *relative performance*, where relative performance for a given instance is the ratio of the offline optimal reward to the reward earned by the algorithm. The worst-case relative performance is called the *competitive ratio* (CR), and smaller the CR, the better the algorithm.

2.1.1 The Model

The model primitives, also shown in Figure 2.1, are as follows: n = number of knapsacks, each of unit capacity (or having unit space), $d_j \in (0, \alpha]$ = the duration (or size) of job j, and $\rho_j \in [1, \Delta]$ = the value density of job j. Note that $\alpha \in (0, 1]$ and $\Delta \in [1, \infty)$. An instance of the OMKP, $L \in \mathcal{L}(\alpha, \Delta)$, is a finite sequence of jobs with at least one job. $\mathcal{L}(\alpha, \Delta)$ is the set of all instances with parameters α and Δ. Knapsacks are arbitrarily indexed $1, 2, \ldots, n$ at the start of each problem instance. We use $v_j \doteq d_j \rho_j$ to denote the value of an arbitrary

job j. In Equation 2.1 below, we define quantities m and M, the integer floor and ceiling of $1/\alpha$, both of which will appear frequently in various results and their proofs.

$$m \doteq \left\lfloor \frac{1}{\alpha} \right\rfloor \text{ and } M \doteq \left\lceil \frac{1}{\alpha} \right\rceil. \tag{2.1}$$

Note that m is an upper bound on the number of maximum-duration jobs that can be fitted into a single knapsack.

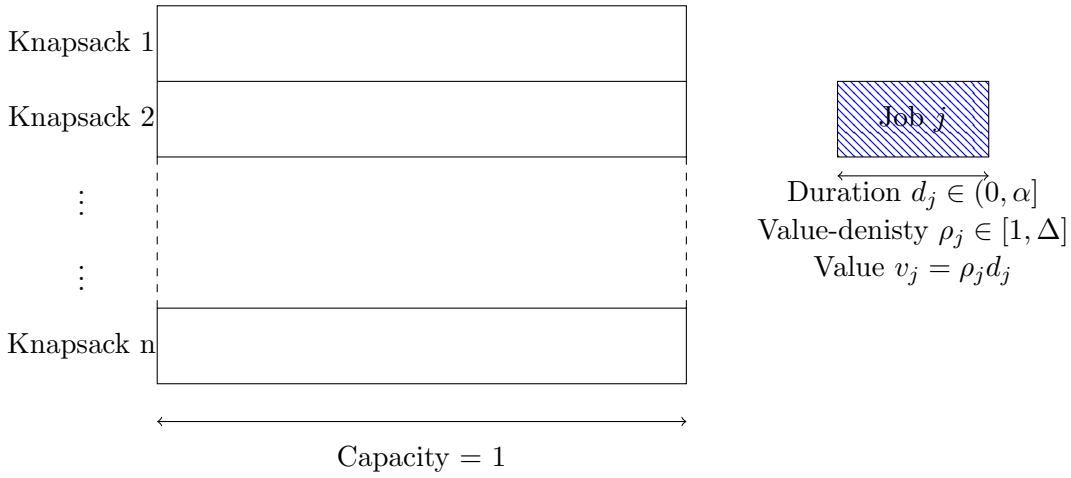

Figure 2.1: n empty knapsacks and an arbitrary job j.

The DM knows the problem parameters $n,\ \alpha$ and Δ, and all knapsacks are empty at the start of a problem instance. Jobs in the instance L arrive one at a time. The duration and the value of a job are revealed only upon its arrival and the remainder of the problem instance remains unknown. The DM must then immediately decide whether to assign the job to a knapsack with enough empty (or free) space for that job, if at least one such knapsack exists, or decline it. Therefore, any algorithm that we consider makes decisions based on the knowledge of $n,\ \alpha,\ \Delta$, attributes of jobs that have already arrived, and its past decisions.

Algorithms for the OMKP can be either deterministic or randomized. A deterministic algorithm takes a unique decision at each job-arrival epoch, whereas a randomized algorithm

takes a decision that depends on the value of a random draw from a probability distribution specified by the algorithm. Let A denote an arbitrary algorithm, $v_A(L)$ the expected total value of jobs accepted by A in problem instance L, and $OPT(L)$ the optimal total value of accepted jobs that could be obtained if the job sequence L was known upfront. Then, the relative performance of Algorithm A for instance L is defined as $OPT(L)/v_A(L)$, and the CR of Algorithm A is defined as:

$$r_A(n, \alpha, \Delta) = \sup_{L \in \mathcal{L}(\alpha, \Delta)} \left(\frac{OPT(L)}{v_A(L)} \right). \tag{2.2}$$

We seek an algorithm that achieves the smallest CR. Often, we are able to find only an upper bound on the CR of an algorithm, and not the exact value. In such a case, we use this upper bound as a proxy for CR. The upper bound is also used to determine the competitiveness of an algorithm, where for given values of n, α, and Δ, we say that Algorithm A is *c-competitive* if $c \geq r_A(n, \alpha, \Delta)$.

Note that when using a deterministic algorithm A, the ratio of $OPT(L)$ and the reward earned by A is greater than or equal to r_A every time A is applied to L. However, this is not true for randomized algorithms. Therefore, even when there exists a randomized algorithm that produces a higher expected value $v_A(L)$, some DMs may prefer to use a deterministic algorithm.

We also consider the *online revenue management problem* (ORMP), which was described as Application 2 earlier. In the ORMP, all jobs in every instance are of unit duration. The CR of an arbitrary algorithm A in the context of the ORMP is defined as:

$$r_A^{RM}(n, \Delta) = \sup_{L \in \mathcal{L}^1(\Delta)} \left(\frac{OPT(L)}{v_A(L)} \right), \tag{2.3}$$

where $\mathcal{L}^1(\Delta) \subset \mathcal{L}(1, \Delta)$ is the set of all instances that consist only of unit-sized jobs.

2.1.2 Our Approach

To the best of our knowledge, a systematic way needed to search for an algorithm that has the smallest CR for the OMKP does not exist. Instead, papers in this line of research establish a lower bound on the CR of all algorithms (either randomized or deterministic, depending on the context), and establish the goodness of the proposed algorithm by comparing its CR to the lower bound. We follow a similar approach.

We first derive separate lower bounds on CRs of all deterministic and all randomized algorithms. The lower bounds imply that no deterministic algorithm can be better than $\mathcal{O}(\ln \Delta)$-competitive in the limiting cases $n \to \infty$ and $\alpha \to 0$. Also, no randomized algorithm can be better than $\mathcal{O}(\ln \Delta)$-competitive for any n and α. We then tweak some known algorithms for the online bin-packing problem (OBP) for application to the OMKP and analyze their CRs. We show that the First-Fit Algorithm (Algorithm F) is $\mathcal{O}(\Delta)$-competitive and has the smallest CR amongst all the OBP algorithms considered. Algorithm F assigns an incoming job to the smallest-indexed knapsack with enough free space. Algorithm F is $\mathcal{O}(\Delta)$-competitive because it does not consider the value-densities of jobs when making accept/decline decisions, and hence may end up filling the knapsacks with jobs with value density equal to one and declining many jobs with value density equal to Δ. The gap between the $\mathcal{O}(\Delta)$ CR of Algorithm F and the best possible $\mathcal{O}(\ln \Delta)$ CR motivates us to develop new algorithms, we propose three new algorithms in the sequel.

We first propose the Segregating Algorithm (Algorithm S) that filters out jobs of certain sizes and segregates the remaining based on their sizes while placing them in the knapsacks in an attempt to minimize avoidable empty space in the knapsacks. Algorithm S also considers only the job durations, and not the value densities, for making accept/decline decisions. Therefore, just like Algorithm F, it is also $\mathcal{O}(\Delta)$-competitive. But it is worth

considering because it achieves a CR that is strictly smaller than that of Algorithm F if n is sufficiently large.

Next, we propose two algorithms that take into account both job durations and value densities to make assign/decline decisions. The first of these is the Deterministic-Threshold Algorithm (Algorithm D). Algorithm D is a deterministic algorithm that creates a threshold for each incoming job, and assigns any job with value greater than or equal to the threshold in the same way that Algorithm F would. We find an upper bound on the CR of Algorithm D for arbitrary values of n, α, and Δ, and also show that it is $\mathcal{O}(\ln \Delta)$-competitive in the limiting cases $n \to \infty$ and $\alpha \to 0$. The second algorithm is the Randomized-Threshold Algorithm (Algorithm R), which samples a unique threshold from a distribution that we prescribe. Algorithm R then assigns any job with value greater than or equal to this threshold the same way that an algorithm A would, where Algorithm A is any OMKP algorithm, such as Algorithm F or S, that considers only job durations when making assign/decline decisions. When using Algorithm R in conjunction with Algorithm A, we say that we are using Algorithm $R : A$. We derive an upper bound on the CR of Algorithm $R : A$ for arbitrary A and arbitrary values of n, α, and Δ. We also show that Algorithms $R : F$ and $R : S$ are both $\mathcal{O}(\ln \Delta)$-competitive for any n and α. Since no algorithm can be better than $\mathcal{O}(\ln \Delta)$-competitive, Algorithms $R : F$ and $R : S$ are both optimal in this sense.

We analyze the ORMP along the same lines as the OMKP, and show that Algorithm D achieves the smallest CR possible for any deterministic algorithm for the ORMP. We also prove that Algorithm $R : F$ achieves the smallest CR possible for any randomized algorithm for the ORMP.

2.1.3 Contribution

We present a unifying model that generalizes many existing online knapsack problem formulations. Specifically, many previous papers study special cases in which the values of problem primitives are either fixed or restricted in some fashion, e.g. unit-sized jobs, single knapsack, etc. We place no such restrictions, and study the general version of the problem. Our algorithms possess CRs that are significantly better than those of selected known algorithms for certain range of parameter values. Our approach results in the smallest possible CR for the ORMP, thus improving upon an earlier result by Ball and Queyranne (2009).

2.2 Related Literature

There have been numerous studies on the MKP, the offline problem. There are also papers on online versions of the MKP where, unlike the OMKP or the ORMP, some stochastic information is available about the future jobs. The analyses and results obtained in papers dealing with these two settings – i.e. offline MKP and stochastic MKP – cannot be be applied to the online setting because the OMKP does not assume availability of such information. Therefore, we focus only on those papers that assume no information is available about future jobs. We recommend readers to refer to Kellerer et al. (2004) for an extensive survey of literature on MKP, and to Marchetti-Spaccamela and Vercellis (1995), Kleywegt and Papastavrou (1998), Kleywegt and Papastavrou (2001), Slyke and Young (2000), Lin et al. (2008), Stein et al. (2016), and Wang et al. (2015) for some analyses of variants of MKP with stochastic information.

We divide the relevant literature into three groups, namely online knapsack problems (Section 2.2.1), secretary problems (Section 2.2.2), and online revenue and operations man-

agement problems (Section 2.2.3).

2.2.1 Online Knapsack Problems

Numerous online knapsack problems have been studied in the literature. Many papers study special cases of the OMKP in which problem parameters (e.g. n, α, and Δ) are either set equal to specific values or restricted in some sense. We do not place such restrictions and study the general version.

Cygan et al. (2016) is the paper most closely related to our efforts. The authors study various online versions of the MKP. We discuss their non-removable sum-objective proportional-value version, which is identical to the OMKP with $\alpha = \Delta = 1$. The authors prove that Algorithm F is 2-competitive when $n > 1$ and that no algorithm can have CR less than $(1 + \ln 2)$ for any $n \in \mathbb{N}$. We prove both these results for arbitrary values of n, α, and Δ, and furthermore show that both the results in Cygan et al. (2016) can be obtained by setting $\alpha = \Delta = 1$ and appropriate values of n in the expressions we derive. We further extend their work by proposing three new algorithms that achieve better CRs for certain parameter values.

Azar et al. (2000) study multiple variants of the OMKP. The one of interest to us is the one in which instances L consist of jobs with value equal to 1 and durations in the set $\{1/k, 2/k, 3/k, \ldots, (k-1)/k, 1\}$, where k is an arbitrary integer. This set of allowed jobs is a subset of allowed instances for the OMKP when $\alpha = 1$ and $\Delta = k$. The authors show that Algorithm F has a CR equal to k. They also propose an algorithm that is $\mathcal{O}(\ln k)$-competitive. The authors further prove that no algorithm can have a CR better than $\mathcal{O}(\ln k)$. When $\alpha = 1$ and $\Delta = k$ in the OMKP, although Algorithm F has CR equal to ∞, we do show that Algorithms $R : F$ and $R : S$ are $\mathcal{O}(\ln k)$-competitive. The difference

in results for Algorithm F is because we do not restrict the set of allowed job durations and values in the same way as Azar et al. (2000) do.

Zhou et al. (2008a) study the single knapsack case ($n = 1$) of the OMKP. The authors present a deterministic algorithm whose CR converges to $(1 + \ln \Delta)$ in the limit $\alpha \to 0$, and they also prove that this is the best possible CR in this limiting case. In another version of the paper, Zhou et al. (2008b) present a randomized algorithm with CR converging to $(1 + \ln \Delta)$ in the limit $\alpha \to 0$. The CR of both these algorithms are not known for arbitrary values of α. We also prove that $(1 + \ln \Delta)$ is best possible CR in the limit $\alpha \to 0$, but our lower bound proofs are different from both Zhou et al. (2008a) and Zhou et al. (2008b). CRs of Algorithms D, $R : F$ and $R : S$ also converge to $(1 + \ln \Delta)$ in the limit $\alpha \to 0$. Furthermore, we present upper bounds on the CRs of our algorithms for arbitrary values of n, α, and Δ.

Buchbinder and Naor (2009) give a $\mathcal{O}(\ln \Delta)$-competitive algorithm for a variant of the single knapsack case ($n = 1$) of the OMKP where the decision maker can accept parts of jobs rather than complete jobs only. Both Böckenhauer et al. (2014) and Han et al. (2015) give a 2-competitive algorithm for the OMKP when $n = \alpha = \Delta = 1$ and prove that it is the best possible CR. The CR results of both these papers are better than ours if we plug $n = \alpha = \Delta = 1$ into our results, but their algorithm and proof techniques cannot be directly applied to cases in which n, α and Δ are not all equal to 1. In contrast, our algorithms and results do not require $n = \alpha = \Delta = 1$.

2.2.2 Secretary Problems

In the standard secretary problem (Ferguson 1989, Freeman 1983), a known number of jobs arrive one at a time and the value of each job is revealed upon its arrival. Upon seeing a

job, the DM must either accept it or decline it irrevocably, and the goal is to accept the largest-valued job with the highest probability. Therefore, the problem is similar to the ORMP with $n = 1$. However, unlike the ORMP, the number of jobs is known upfront and the value-densities of jobs are not bounded above, that is they belong to the interval $[1, \infty)$.

Multiple-choice secretary problem (Babaioff et al. 2007, Kleinberg 2005) is an extension of the standard secretary problem where multiple jobs can be accepted. Babaioff et al. (2007) study the knapsack secretary problem which is similar to the OMKP with $n = 1$, but the value-densities belong to the interval $[1, \infty)$. For both these problems, the CR is used to rank different algorithms.

Despite the apparent similarities, the secretary problems and the OMKP/ORMP differ quite significantly as explained next. In the secretary problems, an instance is a set of jobs, as opposed to a sequence. It is assumed that every permutation of a set of jobs is equally likely to arrive. Therefore, when CR is used as the objective function, the expected total reward of an algorithm for an instance is defined as the mean of expected total reward from each permutation. This difference in objective functions precludes us from utilizing the analyses and results from the secretary problems for the OMKP/ORMP. More specifically, we show that no algorithm can have a finite CR for the OMKP/ORMP if the job value-densities are not bounded above. But, an algorithm with a finite CR for the knapsack secretary problem is possible and presented in Babaioff et al. (2007). Similarly, finite-CR algorithms for the multiple-choice secretary problem are presented in Kleinberg (2005) and Babaioff et al. (2007).

2.2.3 Online Revenue and Operations Management Problems

Various online versions of the single-leg revenue management problem are analyzed in Ball and Queyranne (2009). Amongst these, one version is called the discrete bid-price control problem, which is identical to the ORMP. They propose a deterministic algorithm for the problem and analyze its CR. Their result on CR can be found Section 2.3 of their paper. The expression itself is a bit involved, and therefore, we do not present it here. We improve on this result by showing that two of the algorithms that we propose for the OMKP, Algorithms D and $R : F$, can be modified for application to the ORMP, and that Algorithm D achieves the smallest CR possible for any deterministic algorithm, and Algorithm $R : F$ achieves the smallest CR possible for any algorithm for the ORMP.

Although the online optimization has been used extensively in theoretical computer science (see Albers 2003), Elmachtoub and Levi (2016) state that its application to problems in operations management has begun only recently. Apart from the work on revenue management described above, some other streams where online problems have been studied are inventory management (den Heuvel and Wagelmans 2010, Wagner 2010, 2011), resource allocation (Ma and Simchi-Levi 2017), traveling salesman problem (Jaillet and Wagner 2008), assortment optimization (Golrezaei et al. 2014), and customer selection (Elmachtoub and Levi 2015, 2016). Despite sharing a common high-level theme of online optimization, the specific ideas used in these papers are very different from ours.

2.3 Notation

Key notation needed to specify the OMKP and the ORMP were presented in Section 2.1.1. In this section, we present some more notation that we will use in the technical results and

in the descriptions of the algorithms that we study. Note that although some variables or functions that we either defined in Section 2.1.1 or will define in this section may depend on one or more of the parameters n, α, and Δ, we do not always make this dependence explicit to keep the notation readable. For example, although M defined in Equation 2.1 depends on α, we will use the notation M instead of $M(\alpha)$. We summarize all the notation presented in Section 2.1.1 and this section in Table 2.1.

We begin by presenting some set theoretic notation. Then, we present a function in Definition 2.1 that will be used in derivation of the lower bound on CR of any deterministic algorithm. It will also be used for constructing Algorithms S and D and proving results on their CRs. Then, we present a couple of lemmas on this function that will be required in subsequent proofs. Then, in Definitions 2.2 and 2.3, we present notation relevant to Algorithms S and D, respectively. Finally, the CDF of the distribution from which Algorithm R samples the threshold is presented in Definition 2.4.

Table 2.1: Summary of notation from Sections 2.1.1 and 2.3.

n	The number of knapsacks.
α	The maximum job duration.
m	$\lfloor 1/\alpha \rfloor$. See Equation 2.1.
M	$\lceil 1/\alpha \rceil$. See Equation 2.1.
Δ	The maximum job value density.
$OPT(L)$	Offline optimal total value for instance L.
$v_A(L)$	Expected value of jobs accepted by Algorithm A for instance L.
$\mathcal{L}(\alpha, \Delta)$	The set of all non-empty finite sequences of jobs with duration $\in (0, \alpha]$ and value density $\in [1, \Delta]$.

r_A	CR of Algorithm A for the OMKP. See Equation 2.2.
$\mathcal{L}^1(\Delta)$	The set of all non-empty finite sequences of jobs with unit duration and value density $\in [1, \Delta]$.
r_A^{RM}	CR of Algorithm A for the ORMP. See Equation 2.3.
$\mathbb{R}$	The set of real numbers.
$\mathbb{N}$	The set$\{1, 2, \ldots\}$.
$[k]$	The set$\{1, 2, \ldots, k\}$.
$A \setminus B$	$\{x \mid x \in A \text{ and } x \notin B\}$.
f	A function relevant to the lower bound derivation in Section 2.4 and to Algorithms S and D. See Definition 2.1.
w	A function relevant to Algorithm S. See Definition 2.2.
I_S	$\lceil mn/w(n, \alpha) \rceil$. See Definition 2.2.
θ_i	A function relevant to Algorithm S. See Definition 2.2.
Θ_j	A function relevant to Algorithm S. See Definition 2.2.
u	A function relevant to Algorithm S. See Definition 2.2.
γ	A function relevant to Algorithm S. See Definition 2.2.
t	A function relevant to Algorithm D. See Definition 2.3.
I_D	$\lceil mn/t(n, \alpha, \Delta) \rceil$. See Definition 2.3.
ϕ	A function relevant to Algorithm D. See Definition 2.3.
Φ	A function relevant to Algorithm D. See Definition 2.3.
τ	A function relevant to Algorithm D. See Definition 2.3.
ψ	A function relevant to Algorithm D. See Definition 2.3.
$\mathcal{I}$	A set relevant to Algorithm D. See Definition 2.3.

G	CDF of a distribution relevant to Algorithm R. See Definition 2.4.

We begin by defining the set theoretic notation that we will use in this chapter. $\mathbb{R}$ denotes the set of real numbers, $\mathbb{N}$ the set $\{1, 2, \ldots\}$, and $[k]$ the set $\{1, 2, \ldots, k\}$ for any $k \in \mathbb{N}$. Given sets A and B, $A \setminus B \doteq \{x | x \in A \text{ and } x \notin B\}$. Next, we define a a function that will be used in subsequent results and discussions.

Definition 2.1. *Let function $f : [1, \infty) \times \mathbb{N} \mapsto \mathbb{R}$ be defined as*

$$f(x, a) \doteq x \frac{\left\lceil \frac{a}{x} \right\rceil}{a} \left(1 + \frac{x}{a}\right)^{a - \left\lceil \frac{a}{x} \right\rceil}.$$

Below in Lemma 2.1, we prove that the function f is strictly increasing in its first argument whenever its second argument belongs to the set $\mathbb{N}$.

Lemma 2.1. *Given $a \in \mathbb{N}$, the function $f(x, a)$ is strictly increasing in x when $x \in [1, \infty)$.*

Proof. Let $k \doteq a$. Partition $[1, \infty)$ into k intervals such that

$$I_j \doteq \left[\frac{a}{a - (j-1)}, \frac{a}{a - j}\right), \qquad j \in [k-1], \text{ and}$$

$$I_k \doteq [a, \infty).$$

It is easy to verify that if $x \in I_j$, then $\left\lceil \frac{a}{x} \right\rceil = a - (j-1)$, and that if $x \in I_j$ and $x' \in I_{j+1}$, then $x' > x$. Given this setup, the proof is based on two algebraic arguments. First, we prove that $f(x, a)$ is strictly increasing in x whenever $x \in I_j$. Next, we show that if $x \in I_j$ and $x' \in I_{j+1}$, then this implies $f(x, a) < f(x', a)$ for every such pair, completing the proof.

Suppose $x \in I_j$ and x is changed only such that it continues to belong to a particular I_j where $i \doteq \left\lceil \frac{a}{x} \right\rceil$ is a constant. Then, it is easy to verify that the function f is the product of two strictly increasing functions of x, specifically $\frac{xi}{a}$ and $(1 + \frac{x}{a})^{a-i}$ and therefore increasing in x.

Next, we consider an arbitrary $x \in I_j$ and an arbitrary $x' \in I_{j+1}$. Denoting $i \doteq \left\lceil \frac{a}{x} \right\rceil$ and $i' \doteq \left\lceil \frac{a}{x'} \right\rceil$, we must have $i = i' + 1$ and $x' > x$, and the following sequence of inequalities must hold.

$$
\begin{aligned}
& i' \geq \frac{a}{x'} \implies i' + \frac{i'x'}{a} \geq i' + 1 \implies i' \left(1 + \frac{x'}{a}\right) \geq i \\
& \implies i' \left(1 + \frac{x'}{a}\right)^{a-i+1} > i \left(1 + \frac{x}{a}\right)^{a-i} \implies i' \left(1 + \frac{x'}{a}\right)^{a-i'} > i \left(1 + \frac{x}{a}\right)^{a-i} \\
& \implies \frac{x'i'}{a} \left(1 + \frac{x'}{a}\right)^{a-i'} > \frac{xi}{a} \left(1 + \frac{x}{a}\right)^{a-i} \implies f(x', a) > f(x, a).
\end{aligned}
$$

Hence, proved. □

In the next lemma, we prove a result that will be used in some of the subsequent proofs.

Lemma 2.2. $\inf\{x \geq 1 | f(x, a) \geq \mu\}$ *is real-valued for any* $a \in \mathbb{N}$ *and* $\mu \in [1, \infty)$.

Proof. Given $a \in \mathbb{N}$ and $\mu \in [1, \infty)$, if we prove that $\{x \geq 1 | f(x, a) \geq \mu\}$ is non-empty, then that would imply that $\inf\{x \geq 1 | f(x, a) \geq \mu\}$ lies between 1 and one of the elements of $\{x \geq 1 | f(x, a) \geq \mu\}$, proving the result. Consider the fact that

$$
f(\mu a, a) = \frac{\mu a}{a} \left\lceil \frac{a}{\mu a} \right\rceil \left(1 + \frac{\mu a}{a}\right)^{a - \left\lceil \frac{a}{\mu a} \right\rceil} = \mu (1 + \mu)^{a-1} \geq \mu.
$$

Further, $\mu a \geq 1$. Therefore, $\mu a \in \{x \geq 1 | f(x, a) \geq \mu\}$. Hence, proved. □

Next, we define notation that will be used for constructing and in results for Algorithm S.

Definition 2.2. *Given the maximum job duration* α*, the maximum job density* Δ*, and* n *knapsacks, we define the following notation:*

- *Function* $w : \mathbb{N} \times (0, 1] \mapsto \mathbb{R}$ *such that* $w(n, \alpha) \doteq \inf\{x \geq 1 | f(x, mn) \geq (m + 1)/m\}$. *Function* w *is well-defined by Lemma 2.2.*

- $I_S \doteq \left\lceil \frac{mn}{w(n,\alpha)} \right\rceil$
- $\theta_i \doteq \begin{cases} \frac{1}{m+1} & \text{for } i \in [I_S] \\ \frac{w(n,\alpha)I_S}{mn(m+1)} \left(1 + \frac{w(n,\alpha)}{mn}\right)^{i-I_S-1} & \text{for } i =\in [mn+1] \setminus [I_S] \end{cases}$
- $\Theta_j \doteq \sum_{i=1}^{mj} \theta_i$ *for* $j \in [n]$
- $u(j) \doteq \begin{cases} 1 & \text{if } j > 1 \\ 0 & \text{otherwise} \end{cases}$
- $\gamma(j) \doteq u(j) \min\left((j-1)\frac{m+1}{m+2}, (j-2)\left(1 - \frac{m}{(m+1)^2}\right) + \frac{m}{m+1}\right) + 1 - \alpha + \Theta_{n-j}$

In Definition 2.3 below, we define notation that will be used for constructing and in results for Algorithm D.

Definition 2.3. *Given the maximum job duration α, the maximum job density Δ, and n knapsacks, we define the following notation:*

- $t : \mathbb{N} \times (0,1] \times [1,\infty) \mapsto \mathbb{R}$ *such that* $t(n,\alpha,\Delta) \doteq \inf\{x \geq 1 | f(x,mn) \geq \Delta\}$. *$t$ is well-defined by Lemma 2.2.*
- $I_D \doteq \left\lceil \frac{mn}{t(n,\alpha,\Delta)} \right\rceil$
- $\mathcal{I} \doteq \{(i,j) | i \in [n], j \in [m], (i-1)m + j \leq I_D\}$
- $\phi(i,j) \doteq \begin{cases} 1 & \text{if } (i,j) \in \mathcal{I} \\ \frac{t(n,\alpha,\Delta)I_D}{mn}\left(1 + \frac{t(n,\alpha,\Delta)}{mn}\right)^{(i-1)m+j-I_D-1} & \text{if } (i,j) \notin \mathcal{I}, i \in [n], j \in [m] \end{cases}$

- $\Phi(i,j) \doteq \begin{cases} \phi(i,j+1) & \text{if } i \in [n] \text{ and } j \in [m-1] \\ \phi(i+1,1) & \text{if } i \in [n-1] \text{ and } j = m \\ f(t,mn) & \text{if } i = n \text{ and } j = m \end{cases}$

- $\tau(i,j) \doteq \dfrac{n\Phi(i,j)}{\sum\limits_{k=1}^{i-1}\left[\sum\limits_{\ell=1}^{m-1}\dfrac{\phi(k,\ell)}{m}+\dfrac{\phi(k,m)}{m(m+1)}\right]+\sum\limits_{\ell=1}^{j-1}\dfrac{\phi(i,\ell)}{m}+\dfrac{\phi(i,j)}{m(m+1)}}$ *for* $i \in [n]$ *and* $j \in [m]$

- $\psi(i,j) \doteq \dfrac{n\Phi(i,j)}{\sum\limits_{k=1}^{i-1}\left[\sum\limits_{\ell=1}^{m-1}\dfrac{\phi(k,\ell)}{m}+\phi(k,m)\dfrac{\alpha}{j}\right]+\sum\limits_{\ell=1}^{j-1}\dfrac{\phi(i,\ell)}{m}+\phi(i,j)\left(\dfrac{1}{m}-\alpha\right)}$ *for* $i \in [n]$ *and* $j \in [m]$

Finally, in Definition 2.4, we present the CDF of the distribution from which Algorithm R samples the threshold.

Definition 2.4. $G : [1,\Delta] \mapsto [0,1]$ *such that* $G(x) \doteq (1+\ln x)/(1+\ln \Delta)$.

2.4 Lower Bounds on CRs of OMKP Algorithms

We begin our presentation of technical results by proving a lower bound on the CR of any deterministic algorithm (Theorem 2.1) and then a lower bound on the CR of any randomized algorithm (Theorem 2.2) for the OMKP. A separate bound for deterministic algorithms is derived because, as mentioned in Section 2.1.1, risk-averse decision makers may choose to consider only deterministic algorithms. The lower bounds serve as benchmarks for the algorithms proposed in Section 2.5. Complete proofs of analytical results are provided in the Appendices. We include a sketch of these proofs and intuitive arguments in the paper.

Theorem 2.1. *The CR of any deterministic algorithm A is bounded below by a lower bound*

$\underline{r}_{det}$, *where*

$$\underline{r}_{det}(n, \alpha, \Delta) \doteq \sup\{x | f(x, Mn) \leq \Delta(M+1)/M\}. \tag{2.4}$$

Sketch of the proof. We first consider only those α that are reciprocals of integers. Later we explain why the lower bound $\underline{r}_{det}$ must also hold for other values of α. The proof is by contradiction. We assume that there exists a deterministic algorithm A for the OMKP such that $r_A(n, \alpha, \Delta) < \underline{r}_{det}(n, \alpha, \Delta)$, which implies that $r_A(n, \alpha, \Delta) < \underline{r}_{det}(n, \alpha, \Delta) - \delta$ must be true for some $\delta > 0$. We then present an instance L for which $OPT(L)/v_A(L) \leq r_A(n, \alpha, \Delta)$ must hold from the definition of CR, but Lemma 2.1 implies that $r_A(n, \alpha, \Delta) \geq \underline{r}_{det}(n, \alpha, \Delta) - \delta$, which is a contradiction. A complete proof is presented in Appendix A. □

The lower bound derived above is not an explicit function. To help with an intuitive understanding of the function, we give its numerical values for selected cases in Table 2.2. Further, we are able to obtain explicit lower bounds for some limiting cases in Corollary 2.1 below.

Corollary 2.1. *For an arbitrary deterministic algorithm A,*

1. $\lim_{n\to\infty} r_A(n, \alpha, \Delta) \geq 1 + \ln \dfrac{M+1}{M} + \ln \Delta.$
2. $\lim_{\alpha\to 0} r_A(n, \alpha, \Delta) \geq 1 + \ln \Delta.$

We get the above result by taking appropriate limits in Equation 2.4. We next prove a lower bound on CRs of randomized algorithms for the OMKP.

Theorem 2.2. *The CR of any randomized algorithm A is bounded below by* $\underline{r}_{ran}$, *where*

$$\underline{r}_{ran}(n, \alpha, \Delta) \doteq 1 + \ln \frac{M+1}{M} + \ln \Delta. \tag{2.5}$$

Sketch of the proof. Our argument generalizes the proof of a similar result in Cygan et al. (2016) who show that $\underline{r}_{ran}(n, 1, 1) = 1 + \ln 2$, which is consistent with Equation 2.5 in the

Table 2.2: Numerical values of $\underline{r}_{det}$.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.23	1.43	1.80	1.96	2.21	2.72	2.71	3.04	3.74
100	1.22	1.41	1.70	1.92	2.10	2.40	2.61	2.80	3.11

statement of Theorem 2.2. We first consider only those α that are reciprocals of integers. The lower bound $\underline{r}_{ran}$ also holds for other values of α by the same logic as in Theorem 2.1. We construct $(k+1)$ specific instances, where k is an arbitrary non-negative integer. Note that given an algorithm A and an instance L, $OPT(L)/v_A(L)$ is a lower bound on $r_A(n, \alpha, \Delta)$ from the definition of CR. Therefore, for an arbitrary algorithm A, we choose the best lower bound on $r_A(n, \alpha, \Delta)$ that we can obtain from the $(k+1)$ instances that we consider. The minimum of these best lower bounds over the space of all possible randomized algorithms must be a lower bound on the CR of any randomized algorithm. We show that this search can be performed by solving a linear program (LP), and observe that the lower bound is increasing in k. Therefore, we let $k \to \infty$ and get the lower bound presented in Equation 2.5. A complete proof is presented in Appendix B. □

Table 2.3 gives values of lower bounds on CR of any randomized algorithm for some parameter values. It can be shown that $\underline{r}_{ran}$ is increasing in α and Δ. Also, $\underline{r}_{det}$ converges to $\underline{r}_{ran}$ in the limiting cases as either $n \to \infty$ or $\alpha \to 0$, and it is always greater than $\underline{r}_{ran}$ for any n, α and Δ. This is because the set of deterministic algorithms is a proper subset of the set of randomized algorithms for the OMKP.

In general, lower bounds derived above may or may not be tight. For example, it

Table 2.3: Numerical values of $\underline{r}_{ran}$.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.22	1.41	1.69	1.92	2.10	2.39	2.61	2.79	3.08
100	1.22	1.41	1.69	1.92	2.10	2.39	2.61	2.79	3.08

is known that in the case $n = \alpha = \Delta = 1$, no deterministic algorithm has a finite CR (Marchetti-Spaccamela and Vercellis 1995) and the best randomized algorithm is 2-competitive (Böckenhauer et al. 2014, Han et al. 2015). Therefore, both $\underline{r}_{det}$ $(= 2)$ and $\underline{r}_{ran}$ $(= 1 + \ln 2)$ are loose for this case. But, as we will see in Theorems 2.5 and 2.6, both the lower bounds are tight in the limit as $\alpha \to 0$.

2.5 Algorithms for the OMKP

First, in Section 2.5.1, we study some existing algorithms that can be tweaked and used for the OMKP. Then, in the subsequent sections (Sections 2.5.2, 2.5.3, and 2.5.4), we propose three new algorithms and analyze their CRs. The general sequence of presentation is as follows: we first present the key idea behind the algorithms being discussed, then present a rigorous description of the algorithms, then obtain their CRs or upper bounds on CRs, and finally, we discuss intuitive reasons behind the particular dependence of the CRs on the parameters n, α, and Δ. Note that the CR of any algorithm must be increasing in α and Δ because CR is a supremum over $\mathcal{L}(\alpha, \Delta)$ and $\mathcal{L}(\alpha, \Delta) \subseteq \mathcal{L}(\alpha', \Delta')$ if $\alpha \leq \alpha'$ and $\Delta \leq \Delta'$.

In the expressions for CRs or upper bounds that we present, if plugging in the parameters n, α, and Δ makes the denominator 0, then this implies that the particular CR or the upper

bound is ∞. The same holds true for any expression appearing in the proofs. Complete proofs of analytical results are provided in the Appendices. We include a sketch of these proofs and intuitive arguments in the paper.

2.5.1 Online Bin-Packing Algorithms

Many online bin-packing (OBP) algorithms can be used for the OMKP with some modifications. In OBP, unlike the OMKP, there are infinitely many identical knapsacks (or bins) and the objective is to fit all the jobs in the minimum number of knapsacks. Therefore, when used for the OMKP, the algorithms for OBP focus only on fitting the jobs, disregarding the job-values. We present OMKP adaptations of the following algorithms from the seminal work of Johnson (1973) and then analyze their CRs:

- **First-Fit (Algorithm F) :** Algorithm F assigns an incoming job to the smallest-indexed knapsack with enough free space and declines the job if no knapsack has enough free space.

- **Next-Fit (Algorithm N) :** N assigns an incoming job to the largest-indexed knapsack with at least one job, if this knapsack has enough free space. Otherwise, N assigns the job to the smallest-indexed empty knapsack, if there is one. If there is no empty knapsack, it declines current and all subsequent jobs.

- **Any-Fit algorithms (AF) :** It is a set of algorithms such that any algorithm in AF assigns an incoming job to one of the knapsacks with at least one job and with enough free space for the job. If no such knapsack is available, then the algorithm assigns the job to the smallest-indexed empty knapsack if at least one empty knapsack is available and declines the job otherwise.

- **Almost-Any-Fit algorithms (AAF) :** Define P to be the set of knapsacks with at least one job and with enough free space for the incoming job, and let knapsack $k \in P$ be the largest-indexed knapsack amongst those in P that have the largest amount of free space. Then AAF is the set of algorithms such that an algorithm in AAF assigns the incoming job to any knapsack in $P \setminus \{k\}$, unless $P = \{k\}$ or $P = \emptyset$. If $P = \{k\}$, the algorithm must assign the job to knapsack k. Else if $P = \emptyset$, the algorithm must assign the job to the smallest-indexed empty knapsack if at least one is available and decline it otherwise.

Below in Theorem 2.3, we present results on CRs of algorithms described above. Refer to Table 2.1 for meaning of any notation not explained here.

Theorem 2.3. *Given n knapsacks, the maximum job duration α, and the maximum value density Δ, the following statements are true.*

1. $r_F(n, \alpha, \Delta) = \max\left(\dfrac{m+1}{m}\Delta, \dfrac{nm\alpha(\Delta-1)+n}{(n-1)(1-\frac{1-\alpha}{m})+1-\alpha}\right)$.
2. *Suppose A is an Almost-Any-Fit algorithm, then $r_A(n, \alpha, \Delta) = r_F(n, \alpha, \Delta)$.*
3. *Suppose A is an Any-Fit algorithm, then $r_F(n, \alpha, \Delta) \leq r_A(n, \alpha, \Delta) \leq r_N(n, \alpha, \Delta)$.*

Sketch of the proof.

1. We consider only those instances for which Algorithm F declines at least one job because $OPT(L)/v_F(L) = 1$ if Algorithm F does not decline any job in L and 1 is a trivial lower bound on $r_F(n, \alpha, \Delta)$. We begin by showing that if Algorithm F declines at least one job for an arbitrary instance L, then at most one knapsack can have free capacity $\geq 1/(m+1)$ after all the jobs have been either assigned or declined. We then divide the problem into different cases depending upon the number and location of

such knapsacks. For each case, we establish an upper bound on $OPT(L)$ and a lower bound on $v_F(L)$. The ratio of the two gives an upper bound on $OPT(L)/v_F(L)$ for each case. We combine these bounds to get an upper bound on $r_F(n, \alpha, \Delta)$. Finally, we present some specific instances to prove that this bound is tight. A complete proof is presented in Appendix C.

2. We omit the proof because it is similar to that for Statement 1.

3. We pick an arbitrary Any-Fit algorithm A. Given n, α and Δ, we first prove $r_F(n, \alpha, \Delta) \leq r_A(n, \alpha, \Delta)$. We apply Algorithm F on an arbitrary instance L_1 and construct a new instance L_2 by observing the decision made by Algorithm F. The construction is such that $OPT(L_1) = OPT(L_2)$ and $v_A(L_2) = v_F(L_1)$. Thus, $OPT(L_1)/v_F(L_1) = OPT(L_2)/v_A(L_2) \leq r_A$. Since L_1 was arbitrary, we have $r_F \leq r_A$.

   In order to prove that $r_A(n, \alpha, \Delta) \leq r_N(n, \alpha, \Delta)$, we apply A on an arbitrary instance L_3 and construct a new instance L_4 by observing the decision made by A. The construction is such that $OPT(L_3) = OPT(L_4)$ and $v_N(L_4) \leq v_A(L_3)$. Thus, $OPT(L_3)/v_A(L_3) \leq OPT(L_4)/v_N(L_4) \leq r_N$. Since L_3 was arbitrary, we have $r_A \leq r_N$. A complete proof is presented in Appendix D.

□

To help with our discussions later, we obtain CR of Algorithm F for some limiting cases in Corollary 2.2 below.

Corollary 2.2.

1. $\lim_{n\to\infty} r_F(\alpha, n, \Delta) = \frac{m+1}{m}\Delta$.

Table 2.4: Numerical values of r_F.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.25	1.50	2.00	2.50	3.00	4.00	5.00	6.00	8.00
100	1.25	1.50	2.00	2.50	3.00	4.00	5.00	6.00	8.00

2. $\lim_{\alpha \to 0} r_F(\alpha, n, \Delta) = \Delta$.

We get the above result by using the Statement 1 of Theorem 2.3 and taking appropriate limits.

Theorem 2.3 states that Algorithm F has the best CR amongst the OBP algorithms we considered. Most other OBP algorithms, such as the Harmonic $_{\mathrm{k}}$ algorithms (Lee and Lee 1985), when used for the OMKP, cannot have CRs much better than Algorithm F because their CR is bounded below by $((m+1)/m)\,\Delta$. To prove this, consider an instance where the first mn jobs are of duration $1/(m+1) + \epsilon$ each ($0 < \epsilon$, small enough), and then there are $(m+1)n$ jobs of duration $1/(m+1)$ each. The value density of first mn jobs is 1 and the last $(m+1)n$ jobs is Δ. Any of the Harmonic $_{\mathrm{k}}$ algorithms will accept only the first mn jobs, but it is optimal to select the last $(m+1)n$ jobs. Therefore, $\lim_{\epsilon \to 0} (OPT(L)/v(L)) = ((m+1)/m)\,\Delta$, proving our claim.

Table 2.4 gives values of r_F for some parameter values. It can be shown that r_F is decreasing in n and increasing in α and Δ. The CR of Algorithm F is linear in Δ. This is because Algorithm F does not consider the value-densities of jobs when making accept/decline decisions, and hence may end up filling the knapsacks with jobs with value density equal to one and declining many jobs with value density equal to Δ. In Section 2.5.2, we present the

Segregating Algorithm (S) that also does not consider the value-densities of jobs for making decisions and hence has a CR that is linear in Δ, but it tries to fit jobs in a different way than Algorithm F. As a result, its CR is smaller than that of Algorithm F for some range of parameter values. Then in Sections 2.5.3 and 2.5.4, we present algorithms that use ideas from Algorithms F and S, and also consider the value-densities of jobs for decision-making to achieve CRs that are closer to the lower bounds presented in Section 2.4.

2.5.2 The Segregating Algorithm (Algorithm S)

Like OBP algorithms considered previously, Algorithm S tries to fit jobs well into the knapsacks, without concerning itself with the job values. Loosely speaking, Algorithm S tries to pack smaller jobs together in lower-indexed knapsacks and larger jobs in the higher-indexed knapsacks because this causes the space in these knapsacks to be filled more efficiently. The intermediate jobs are declined. The idea of segregating jobs based on their durations is well known and has been used in OBP algorithms such as the Harmonic $_{\mathrm{k}}$ algorithms (Lee and Lee 1985). Intermediate jobs are declined to avoid bad relative performance for instances similar to the one presented in Section 2.5.1 to prove that CR of any Harmonic $_{\mathrm{k}}$ algorithm is at least $((m+1)/m)\,\Delta$.

Description of Algorithm S: Initialize $counter = 1$. For each incoming job, make a decision and update the value of *counter* using the following rules.

if no knapsack has enough free space for the job **then**

    decline the job

else if duration of the job $\in (1/(m+1), \theta_{counter})$ **then**

    decline the job

else if duration of the job $\leq 1/(m+1)$ **then**

assign the job to smallest-indexed knapsack with enough free space

else

assign the job to largest-indexed knapsack with enough free space and $counter \leftarrow counter + 1$

end if

The thresholds, θs, are chosen as in Definition 2.2 because they result in a small r_S. Next, we establish a bound on r_S. Refer to Table 2.1 for meaning of any notation not explained here.

Theorem 2.4.

$$r_S(n, \alpha, \Delta) \leq \max\left(w(n, \alpha), \frac{n}{\min\limits_{j \in [n]} \gamma(j)}\right)\Delta. \tag{2.6}$$

Sketch of the proof. Using arguments similar to those in the proof of Statement 1 in Theorem 2.3, we consider only those instances for which Algorithm S declines at least one job. Algorithm S may decline a job in one of two ways. First, Algorithm S may decline a job because no knapsack has enough free space for the job. Second, although there is at least one knapsack with enough free capacity for the incoming job, the duration of the job belongs to the interval $(1/(m+1), \theta_{counter})$.

In case no job in the instance is declined because of lack of free space, all declines jobs have duration $> 1/(m+1)$ and we use algebraic arguments to prove that $OPT(L)/v_S(L) \leq w(n, \alpha)\Delta$. Otherwise, if at least one job is declined because of lack of free space, we utilize arguments similar to those in the proof of Statement 1 in Theorem 2.3 to prove that $OPT(L)/v_S(L) \leq n\Delta/\min\limits_{j \in [n]}\{\gamma(i)\}$. Combining these two bounds and noting that the instance L that we considered was arbitrary gives us the desired result. The details are presented in Appendix E. □

Table 2.5: Numerical values of upper bound on r_S from Theorem 2.4.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.27	1.58	2.50	2.53	3.16	5.00	5.06	6.32	10.00
100	1.24	1.46	1.83	2.49	2.92	3.66	4.97	5.84	7.32

We are able to obtain explicit upper bounds on the CR of Algorithm S for some limiting cases in Corollary 2.3 below.

Corollary 2.3.

1. $\lim_{n\to\infty} r_S(n,\alpha,\Delta) \leq \frac{m+2}{m+1}\left(\frac{1+\ln\frac{m+1}{m}}{1+\ln\frac{m+2}{m+1}}\right)\Delta.$
2. $\lim_{\alpha\to 0} r_S(n,\alpha,\Delta) \leq \Delta.$

We get the above result by taking appropriate limits in Equation 2.6 in the statement of Theorem 2.4.

Table 2.5 gives values of the upper bound on the CR of Algorithm S, as derived in Theorem 2.4, for some parameter values. Algorithm S is worth considering because given α and Δ, CR of Algorithm S is strictly smaller than that of Algorithm F for sufficiently large n. Also, as we mentioned in Section 2.5.1, just like the CR of Algorithm F, the upper bound on the CR of Algorithm F is also linearly increasing in Δ because Algorithm S does not consider value densities when deciding which jobs to accept. Therefore, in the sequel we propose algorithms that take into account job value densities for making accept or decline decisions.

2.5.3 The Deterministic-Threshold Algorithm (Algorithm D)

Algorithm D creates a deterministic state-dependent threshold for each incoming job, and assigns it in the same way as Algorithm F if the value of the job is greater than or equal to the threshold. State refers to the attributes of the current and past jobs, along with the past decisions. This way, Algorithm D inherits desirable properties of Algorithm F, while favoring higher value density jobs that Algorithm F may have declined.

Knapsack 1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
Knapsack 2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
Knapsack 3	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
Knapsack 4	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)
Knapsack 5	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)

Figure 2.2: Knapsacks with $m = 5$ segments for Algorithm D when $n = 5$ and $\alpha = \frac{1}{5}$.

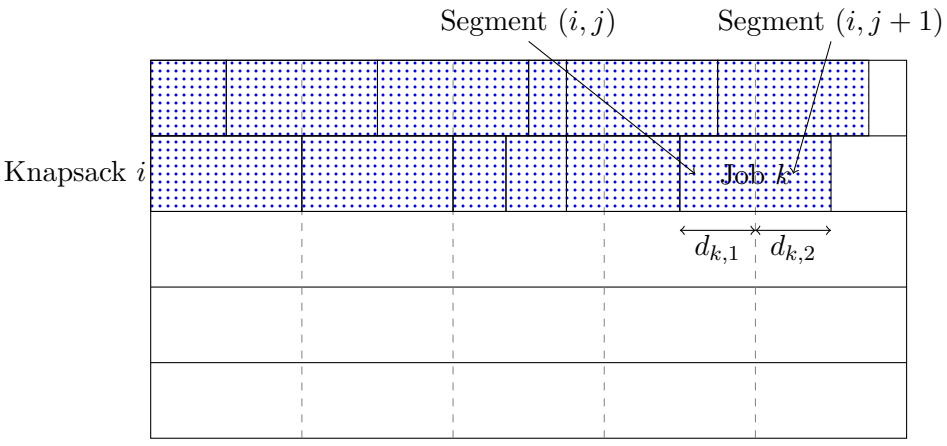

Figure 2.3: Placement of jobs in the knapsacks by Algorithm D for an arbitrary instance.

Description of Algorithm D: We divide each knapsack into m segments of length $1/m$ each (Figure 2.2). The segments of knapsack i are labelled by $(i, 1), (i, 2), \ldots, (i, m)$ from left to right. Each segment (i, j) has a value $\phi(i, j)$ associated with it. Although the

OMKP does not require jobs to be assigned to particular positions within the knapsacks, Algorithm D places the jobs at specific positions. The jobs may be moved around within each knapsack after all the placement decisions have been made. Algorithm D places the first job in a knapsack against the knapsack's left end, the next job is stacked against first job's right end, and so on (Figure 2.3). Algorithm D declines a job if no knapsack has enough free space for it. Else, let knapsack i be the smallest-indexed knapsack with enough free space for the incoming job k, and let (i, j) be the leftmost segment with non-zero free space. Let $d_{k,1}(> 0)$ be the duration of job in segment (i, j) and $d_{k,2}(\geq 0)$ be the duration of job in segment $(i, j+1)$ (Figure 2.3). Algorithm D assigns the job to the knapsack if the job's value $\geq d_{k,1}\phi(i, j) + d_{k,2}\phi(i, j+1)$ and declines it otherwise.

The ϕ values that are used for constructing thresholds are chosen as in Definition 2.3 because they result in a small CR. Next, we derive an upper bound on the CR of Algorithm D. Refer to Table 2.1 for meaning of any notation not explained here.

Theorem 2.5.

$$r_D(n, \alpha, \Delta) \leq \max\left(\max_{(i,j)\in\mathcal{I}} \tau(i, j), \max_{(i,j)\in\mathcal{I}} \psi(i, j)\right). \tag{2.7}$$

Sketch of the proof. Using arguments similar to those in the proof of Statement 1 in Theorem 2.3, we consider only those instances for which Algorithm D declines at least one job. Let L be an arbitrary instance from which Algorithm D declines at least one job. We begin by identifying the rightmost partially filled segment, (i^*, j^*), in the largest-indexed knapsack with at least one job assigned to it after all the jobs in an arbitrary instance L have been either placed in knapsacks or declined by Algorithm D. Because of the way Algorithm D works and because at least one job from L is declined, $(i^*, j^*) \in \mathcal{I}$. We use the same idea as that in the proof of Statement 1 in Theorem 2.3 to find lower bounds on the duration

Table 2.6: Numerical values of upper bound on r_D from Theorem 2.5.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.25	1.50	2.00	2.31	2.86	3.60	3.46	4.40	5.44
100	1.25	1.50	2.00	2.13	2.56	3.40	3.01	3.62	4.81

of accepted jobs. We use these lower bounds and the ϕ values to compute a lower bound on $v_D(L)$. We also compute an upper bound on $OPT(L)$. The ratio of these bounds gives us an upper bound on $OPT(L)/v_D(L)$. This upper bound is of the form $\max(\tau, \psi)$, which is similar in structure to Statement 1 in Theorem 2.3 because we use similar logic in these proofs. We combine these bounds for all possible values that i^* and j^* can take and note that L was arbitrary to get the desired upper bound on $r_D(n, \alpha, \Delta)$. Details are presented in Appendix F. □

Just as we did for Algorithms F and S, we obtain explicit upper bounds on the CR of Algorithm D for some limiting cases in Corollary 2.4 below.

Corollary 2.4.

1. $\lim_{n\to\infty} r_D(n, \alpha, \Delta) \leq \frac{m+1}{m}(1 + \ln \Delta)$.

2. $\lim_{\alpha\to 0} r_D(n, \alpha, \Delta) \leq 1 + \ln \Delta$.

We get the above result by taking appropriate limits in Equation 2.7 in the statement of Theorem 2.5.

Table 2.6 gives values of the upper bound on the CR of Algorithm D, as derived in Theorem 2.5, for some parameter values. Corollary 2.4 shows that in the limiting cases

$n \to \infty$ and $\alpha \to 0$, considering value-densities of jobs while making accept or decline decisions and using the same fitting technique as Algorithm F enables Algorithm D to achieve a CR that is $\mathcal{O}(\ln \Delta)$ instead of $\mathcal{O}(\Delta)$. Further, this $\mathcal{O}(\ln \Delta)$ CR in the limiting cases is in line with the lower bounds presented in Corollary 2.1.

In the next section, we present an approach that also uses the idea of using thresholds, but unlike Algorithm D, the threshold is sampled from a distribution. This leads to a randomized algorithm.

2.5.4 The Randomized-Threshold Algorithm (Algorithm *R*)

The main idea behind Algorithm R is to construct a randomized threshold for incoming jobs. The jobs that satisfy the threshold criterion are then accepted or declined using another OMKP algorithm, such as Algorithm F or S, that only considers job-durations and not the values or value-densities of jobs for making decisions. The randomized threshold prioritizes higher value density jobs, resulting in a CR that is $\mathcal{O}(\ln \Delta)$.

Description of Algorithm *R:A*: Sample x from the distribution with CDF G. If the value of the job is $< x$, decline it. Else, accept or decline the job using Algorithm A, where A is an OMKP algorithm that considers only the job durations for making decisions.

Theorem 2.6.

$$r_{R:A}(n, \alpha, \Delta) \leq r_A(n, \alpha, 1)(1 + \ln \Delta).$$

Sketch of the proof. For the proof, we consider an arbitrary instance L and use the definition of r_A to get a lower bound on $g(L_x)$, the total duration of jobs that are accepted by Algorithm $R : A$ when the sampled threshold is equal to x. Because $v_R(L) = E\left[xg(L_x)\right]$, the lower bound on $g(x)$ gives us a lower bound on $v_R(L)$. Then, we use this lower bound on $v_R(L)$ and some algebraic arguments to show that $\dfrac{OPT(L)}{v_R(L)} \leq r_A(n, \alpha, 1)(1 + \ln \Delta)$.

Table 2.7: Numerical values of upper bound on $r_{R:F}$ from Theorems 2.3 and 2.6.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.25	1.50	2.00	2.12	2.54	3.39	2.98	3.58	4.77
100	1.25	1.50	2.00	2.12	2.54	3.39	2.98	3.58	4.77

Table 2.8: Numerical values of upper bound on $r_{R:S}$ from Theorems 2.4 and 2.6.

Δ	1			2			4		
n \ α	0.25	0.50	1.00	0.25	0.50	1.00	0.25	0.50	1.00
5	1.27	1.58	2.50	2.14	2.67	4.23	3.02	3.77	5.97
100	1.24	1.46	1.83	2.10	2.47	3.10	2.97	3.49	4.37

Since L was arbitrary, this gives us the desired upper bound on $r_{R:A}(n, \alpha, \Delta)$. Details are presented in Appendix G. □

Tables 2.7 and 2.8 give values of upper bounds on the CRs of Algorithms $R : F$ and $R : S$ respectively for some parameter values, as derived in Theorems 2.3, 2.4, and 2.6.

Note that the randomized threshold is effective only when $\Delta > 1$. When Algorithm F or S are used in conjunction with the randomized threshold, the resulting CR is $\mathcal{O}(\ln \Delta)$ instead of the original $\mathcal{O}(\Delta)$ for any value of α and n. To contrast, Algorithm D achieves an $\mathcal{O}(\ln \Delta)$ CR only in the limiting cases when $n \to \infty$ or $\alpha \to 0$.

2.6 The ORMP

In this section, we present results for the ORMP. In Theorem 2.7 below, we prove that Algorithm $R : F$ has the smallest possible CR for the ORMP, and that Algorithm D has the smallest possible CR amongst all deterministic algorithms for the ORMP. To use Algorithm D for the ORMP, set $\alpha = 1$ throughout the notation and description given in Section 2.5.3. We also prove that the CR of Algorithm D converges to the CR of Algorithm $R : F$ in the limit $n \to \infty$. Thus, our algorithms achieve better CRs than the algorithm proposed by Ball and Queyranne (2009) for the ORMP, which is referred to as the discrete bid-price control problem in their work. Ball and Queyranne (2009)'s algorithm is very similar to Algorithm D with α set to 1, but the specific threshold values they propose are different from ours.

Theorem 2.7. *The following statements are true in the context of the ORMP.*

1. *The CR of any randomized algorithm A is bounded below by $\underline{r}_{ran}^{RM}$, where*

$$\underline{r}_{ran}^{RM}(n, \Delta) \doteq 1 + \ln \Delta.$$

2. *Algorithm $R : F$ achieves the smallest possible CR, that is,*

$$r_{R:F}^{RM}(n, \Delta) = \underline{r}_{ran}^{RM}(n, \Delta) = 1 + \ln \Delta.$$

3. *The CR of any deterministic algorithm A is bounded below by $\underline{r}_{det}^{RM}$, where*

$$\underline{r}_{det}^{RM}(n, \Delta) \doteq \sup\{x | f(x, n) \leq \Delta\}. \tag{2.8}$$

4. *Algorithm D achieves the smallest CR amongst all deterministic algorithms, that is,*

$$r_{D}^{RM}(n, \Delta) = \inf\{x \geq 1 | f(x, n) \geq \Delta\} = \underline{r}_{det}^{RM}(n, \Delta).$$

5. Algorithm D achieves the smallest possible CR in the limit $n \to \infty$, that is,

$$\lim_{n\to\infty} r_D^{RM}(n, \Delta) = 1 + \ln \Delta.$$

Sketch of the proof. The proofs of Statements 1, 3, and 4 are similar to those for corresponding results for the OMKP and are presented in Appendices H, I, and J, respectively. Moreover, the proof of Statement 2 is identical to that for Theorem 2.6 once we note that $r_F^{RM}(n, \Delta) = \Delta$. Statement 5 can be proved by taking appropriate limits in Equation 2.8 and then using Statement 1. □

Chapter 3

The Online Reservation Problem

3.1 Introduction

Popular web platforms such as Airbnb and Zilok provide owners the ability to list resources available for sharing, prices, and contract-length limits. The list of resources that may be shared is long and includes items such as apartments, tools, clothes, and intangibles such as the owners' time (e.g. tutoring or handyman services). The sharing/renting process is known by many names such as peer-to-peer renting (Fraiberger and Sundararajan 2017), collaborative consumption (Hamari et al. 2016), and sharing economy (Martin 2016). Sharing economy is growing — in early 2018, there were 1,229 sharing-economy start-ups listed on AngelList, a US website for start-ups, with an average valuation of $3.4M (AngelList 2018). An individual interested in a listed resource can see its availability and place a request with the owner, indicating the desired contract start time and length. The owner must then decide immediately (or within a short period of time) whether to accept or decline the request, even if the contract is for a future date. Accepted requests earn a reward equal to the product of the posted price and the contract duration. Declined requests are lost and

earn nothing. A key trade-off that the owner confronts is the reward he or she earns from the current request and the potential loss of reward from higher-value future requests that may clash with the current one.

We assume that the owner has no information about future requests that may arise, other than the contract-length limits, some of which may depend on the type of resource. The minimum contract duration is predetermined either by the platform, e.g. one night on Airbnb, or by the owner, e.g. the number of hours on the car-sharing platform Getaround. The maximum contract duration is usually set by the owners, e.g. two days for a wedding gown rental (Zilok 2018). In case of services such as tutoring and handyman work, such a restriction may be a result of limited human capacity. At times, an owner may also place an upper limit on contract length because of local laws. For example, renting an apartment for more than 30 days automatically confers tenant rights to the customer in California (DCA 2012), and this may not be acceptable to the apartment owner. Finally, market forces may result in a de facto upper bound on the contract length. This may happen on platforms such as Getaround whose selling point is availability of short duration contracts that are not offered by conventional renting agencies. However, using such platforms becomes more expensive for the longer duration contracts. This will naturally limit the length of the contracts.

There are many situations in which our assumption that the owner does not have information about future requests, other than the contract-length limits, is reasonable. This may happen when demand depends on a multitude of frequently changing external factors that are difficult to anticipate. Additionally, the owner may not have access to the web platform data, and the data he or she observes may be very limited. Because of the lack of information about future requests, the owner's problem is an *online* version of a problem that

Kolen et al. (2007) call the *interval scheduling problem with given machines*. In the *offline* version, the entire sequence of requests and their attributes are known upfront. Because our motivating application is the reservation of resources on sharing economy platforms, we call this problem the *online reservation problem* (ORP).

In some instances, owners may have limited information about future requests. Even in those cases, it may be desirable to use an online algorithm because they are robust against forecast errors (Ma and Simchi-Levi 2017) and distributional misspecifications (Buchbinder et al. 2013). Moreover, the decision maker will not need to customize the algorithm to exploit the type, quantity and quality of information available. Specifically, for the application we have in mind, the owner is unlikely to have the expertise and the resources to develop customized approaches. Our approach is designed to be easily implementable in practice.

The *relative performance* of an algorithm for a given problem instance is the ratio of the offline optimal reward to the reward earned by the algorithm. *Competitive ratio* (CR) of an algorithm is defined as its worst-case relative performance. Given that we have no information about future requests, we use CR of an algorithm as a measure of its goodness. CR has been used widely for evaluating different decision making strategies for a whole host of online problems (see Albers 2003).

3.1.1 The Problem Statement

An instance L of the ORP is a non-empty finite sequence of requests or jobs. Each job i in L has three attributes: arrival time $a_i \geq 0$, start time s_i ($\geq a_i$), and length (or duration) d_i. Let $D_{min} > 0$ and $D_{max} < \infty$ be the contract-length limits. We constrain L to belong to a set $\mathcal{L}(\mathcal{D})$, where $\mathcal{D} \doteq [D_{min}, D_{max}]$. What this means is that while L can consist of an arbitrary number of jobs with arbitrary arrival and start times, the length of each job must

belong to the interval $[D_{min}, D_{max}]$. We also define $\Delta \doteq D_{max}/D_{min}$.

Let n denote the maximum number of concurrently available identical resources or servers, indexed $1, 2, \ldots, n$. Each server can process only one job at a time. Jobs in L arrive one at a time and the attributes of each job are revealed only upon its arrival. As soon as a job arrives, the owner must immediately assign the job to one of the servers or decline it, knowing only that the lengths of any remaining jobs in L will be in $\mathcal{D}$. Once assigned, jobs cannot be preempted. Similarly, overlapping or clashing jobs may not be assigned to the same server, i.e. jobs i and j with $s_i \leq s_j$ and $s_i + d_i > s_j$ cannot be assigned to the same server. Given job i, we say that server j is with no conflicts if no job already assigned to server j clashes with job i. Without loss of generality, we set the unit rent equal to 1. Therefore, the owner receives a reward equal to the duration of the job if it is assigned to one of the n servers, and zero otherwise. Declined jobs are lost.

Because jobs are assumed to arrive one at a time, no two jobs can have the same precise arrival time. However, different jobs may have identical start times and/or durations. We say that job-$i \equiv$ job-j, i.e. i and j represent the same job, if $a_i = a_j$, $s_i = s_j$, and $d_i = d_j$.

Given L, let $OPT(L)$ denote the total value of jobs in an *offline optimal set*, which is an optimal set of jobs that would be accepted if L was known entirely in advance. Let A denote an arbitrary algorithm in the set $\mathcal{A}$, where $\mathcal{A}$ is the set of algorithms that upon arrival of a job, either immediately assign it to one of the servers with no conflicts, or decline it, based only on the characteristics of the previously-arrived jobs, the past assign/decline decisions, the number of servers n, and the set $\mathcal{D}$. Given A, let $v_A(L)$ denote the expected total value of jobs accepted by A for the problem instance L. Algorithm A may be either deterministic or randomized. A randomized algorithm makes the accept or decline decisions based on the outcome of a random draw from a distribution specified by the algorithm. In case of a

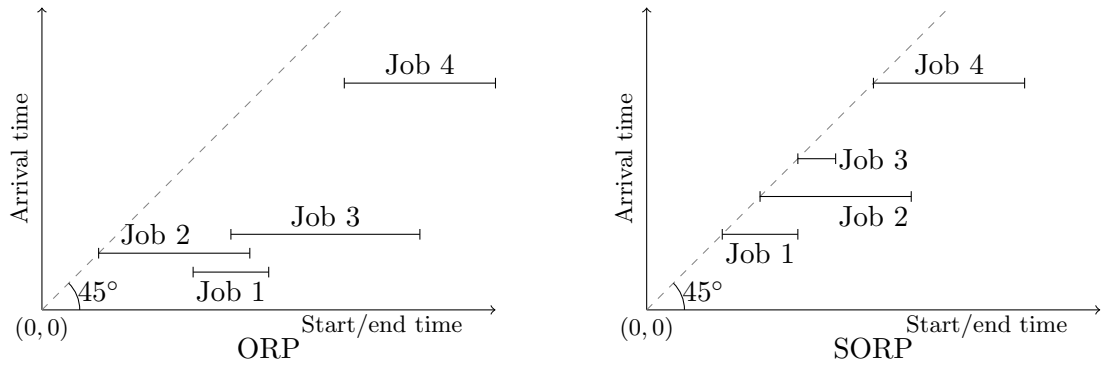

Figure 3.1: Examples of problem instances with four jobs each under the ORP and the SORP settings.

randomized algorithm, $v_A(L)$ denotes the expected value of jobs accepted by A in repeated implementations of A over the same L. Given this notation, the relative performance of A for instance L is defined as $OPT(L)/v_A(L)$, and the competitive ratio (CR) of A is defined as follows:

$$r_A(n, \Delta) := \sup_{L \in \mathcal{L}(\mathcal{D})} \frac{OPT(L)}{v_A(L)}. \tag{3.1}$$

Given n and Δ, our objective is to find an A^* such that $r_{A^*} \leq r_A$ for every $A \in \mathcal{A}$.

We also study a special case, abbreviated as SORP. The difference between the ORP and the SORP is that in the latter $s_i = a_i$ for each i (see Figure 3.1). Such scenarios fit operations at platforms such as Uber and Lyft, and in problems dealing with the assignment of open jobs to reserve drivers in transit operations (Gupta and Li 2016). We use $\mathcal{L}^*(\mathcal{D})$ to denote the set of all possible non-empty finite sequences in which jobs arrive at their start times. Then, the CR of an arbitrary $A \in \mathcal{A}$ for the SORP is defined as $r^*_A(n, \Delta) := \sup_{L \in \mathcal{L}^*(\mathcal{D})} \left(OPT(L)/v_A(L)\right)$.

In the remainder of this paper, we use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{N}$ to denote the set $\{1, 2, \ldots\}$, and $[k]$ to denote the set $\{1, 2, \ldots, k\}$ for any $k \in \mathbb{N}$. Given sets A and B, $x \in A \setminus B \iff x \in A$ and $x \notin B$. Later, in Definitions 3.2, 3.3, and 3.4, we will present some notation that will be used in the technical results and the description of the

algorithms that we propose. In Table 3.1, we summarize the notation presented in this section and in the Definitions 3.2, 3.3, and 3.4.

Table 3.1: Summary of notation from Section 3.1.1 and Definitions 3.2, 3.3, and 3.4.

a_i	Arrival time of job i.
s_i	Start time of job i.
d_i	Duration of job i.
D_{min}	Minimum value of job duration.
D_{max}	Maximum value of job duration.
$\mathcal{D}$	The interval $[D_{min}, D_{max}]$.
Δ	D_{max}/D_{min}
n	The number of servers.
$OPT(L)$	Offline optimal total value for instance L.
$v_A(L)$	Expected value of jobs accepted by Algorithm A for instance L.
$\mathcal{L}(\mathcal{D})$	The set of all non-empty finite sequences of jobs with duration in $\mathcal{D}$.
r_A	CR of Algorithm A for the ORP. See Equation 3.1.
$\mathcal{L}^*(\Delta)$	The set of all non-empty finite sequences of jobs with duration in $\mathcal{D}$ that arrive precisely at their respective start times.
r_A^*	CR of Algorithm A for the SORP.
$\mathbb{R}$	The set of real numbers.
$\mathbb{N}$	The set $\{1, 2, \ldots\}$.
$[k]$	The set $\{1, 2, \ldots, k\}$.
$A \setminus B$	$\{x \mid x \in A \text{ and } x \notin B\}$.
t	A function relevant to Algorithm D for the ORP. See Definition 3.2.

I	$\lceil 3n/t(n,\Delta)\rceil$. See Definition 3.2.
ϕ	Admission thresholds relevant to Algorithm D for the ORP. See Definition 3.2.
μ	CDF of a distribution relevant to Algorithm R. See Section 3.3.
t^*	A function relevant to Algorithm D for the SORP. See Definition 3.4.
I^*	$\lceil 2n/t^*(n,\Delta)\rceil$. See Definition 3.4.
ϕ^*	Admission thresholds relevant to Algorithm D for the SORP. See Definition 3.4.

3.1.2 Our Approach

It is often difficult to determine the exact value of CR of an algorithm. In such cases, we instead find an upper bound on the CR and use it as a proxy for the CR. This difficulty arises because of a multitude of reasons. First, a closed-form expression for $OPT(L)$ is not available for every L. In fact, the offline version of the ORP is NP-Hard (Gupta and Li 2016). Second, given an algorithm A, evaluating $v_A(L)$ for all instances may be difficult, especially when A is a randomized algorithm. Note that there are an infinite number of instances in the set $\mathcal{L}(\mathcal{D})$. Given this difficulty, we use the notion of c-competitiveness with a suitably defined c (Fiat and Woeginger 1998). For given n and Δ, we say that Algorithm A is c-competitive if $c \geq r_A$. We also find a lower bound on the CR of any algorithm that could be brought to bear on the ORP and use that as a benchmark for the goodness of any algorithm in $\mathcal{A}$. Therefore, there are two research challenges addressed in this paper: (i) how to establish a lower bound on CRs of all algorithms in $\mathcal{A}$, and (ii) how to construct algorithms whose CRs or upper bound on CRs can be computed and whose CRs or upper bounds on CRs are close to the lower bound on CRs of all algorithms in $\mathcal{A}$.

We first find a lower bound on the CR of any algorithm in $\mathcal{A}$ in Section 3.2, proving

that no algorithm can have a CR better than $\mathcal{O}(\ln \Delta)$. Then, in Section 3.3.1 we analyze the CRs of all fair algorithms, which are those that do not decline a request whenever a server is available. The term 'fair' comes from Boyar and Larsen (1999). We study fair algorithms because they are intuitive and have been studied in the context of other problems such as the bin-packing problem (Azar et al. 2000). We prove that any fair algorithm is $\mathcal{O}(\Delta)$-competitive. The gap between the CR of the fair algorithms and the best possible CR motivates the need for better algorithms. The linear dependence of the CRs of fair algorithms on Δ arises because in some instances they end up accepting jobs with duration D_{min} and declining those with duration D_{max}. Therefore, we design two new algorithms using the intuition that reserving capacity for higher-value future jobs may help improve the CR and that this may be operationalized using thresholds. We first propose a deterministic algorithm, Algorithm D, in Section 3.3.2 that uses server specific admission thresholds. An incoming job is assigned to the smallest-indexed server with no conflicts and with admission threshold smaller than or equal to the job's duration. We find the upper bound on the CR of Algorithm D for any values of n and Δ, and prove that the algorithm is $\mathcal{O}(\ln \Delta)$-competitive in the limit $n \to \infty$. We also propose a randomized algorithm, Algorithm R, in Section 3.3.3. At the beginning of the problem instance, Algorithm R samples a threshold from a distribution that we prescribe, and any job that has duration strictly smaller than this threshold is declined. Any qualifying job is assigned to the smallest-indexed server with no conflicts. We prove that Algorithm R is $\mathcal{O}(\ln \Delta)$-competitive for any n and Δ, and is therefore optimal in the sense that no algorithm can be better than $\mathcal{O}(\ln \Delta)$-competitive. The expressions for the CRs or upper bounds on CRs for all algorithms that we study imply that for each algorithm, there exists a specific parameter-value range in which it is better than all the other algorithms that we study.

For the SORP, we follow the same approach as we do for the ORP. Before we outline our contributions, we discuss the literature next.

3.1.3 Related Literature

A large number of studies that analyze online scheduling problems assume that preemption of already accepted jobs is allowed. Papers in this stream propose algorithms for problems with a variety of different features and analyze the CRs of proposed algorithms; see Miyazawa and Erlebach (2004); Seiden (1998); Faigle and Nawijn (1995); Fung et al. (2008); Koren and Shasha (1995); and Baruah et al. (1992). Preemption is not allowed in our setting, and therefore, we will focus in the sequel on papers that do not allow preemption.

Boyar and Larsen (1999) analyze online seat reservations for a train that stops at k stations. They study two scenarios, first where all tickets yield the same reward, and second where the reward from a ticket is proportional to the number of stops in the journey. If we think of tickets as jobs and seats as servers, it is easy to see the connection between the latter scenario and the ORP. Boyar and Larsen (1999) consider only fair algorithms. They prove that CR of any fair deterministic algorithm is $\Theta(k)$, which means that there exist real numbers $c_1, c_2 > 0$ and integer $k_0 > 0$ such that $c_1 k \leq CR \leq c_2 k$ for all $k \geq k_0$. By putting in $D_{min} = 1$ and $D_{max} = k - 1$ in the expressions for CRs of fair algorithms for the ORP, we find that they are indeed $\Theta(k)$-competitive. But since we do not limit ourselves to fair algorithms, we are able to achieve smaller CRs for some ranges of parameter values. Moreover, Boyar and Larsen (1999)'s formulation allows jobs to have durations in the integer set $\{1, 2, \ldots, k-1\}$, whereas the ORP allows durations to belong to the interval $[1, k-1]$. Boyar and Larsen (1999) also prove some results on *accommodating ratio*, which is same as CR except that the supremum is taken only over special job sequences, which have

the property that all the jobs can be accepted in the offline version. We do not analyze accommodating ratio of the algorithms because in our motivating application, there is no reason to assume that the owner will face only special job sequences.

Some other papers, e.g. Gupta and Li (2016), Lipton and Tomkins (1994), and Garofalakis et al. (2002), are similar to the SORP in the sense that they assume that jobs arrive exactly at their start times. We cannot use the algorithms or other results from these papers for the ORP because the set of allowed instances for the SORP is a proper subset of those for the ORP. However, our approach for the ORP can be utilized to solve the SORP. Moreover, as shown in Section 3.4, we can tweak our ORP algorithms to achieve even smaller CRs for the SORP.

Gupta and Li (2016) study the multi-processor online fixed job scheduling problem for reserve driver scheduling. Their problem is identical to the SORP, except that the servers in their version are only available for specific time intervals, which model driver shifts. They propose a randomized algorithm for their problem that is $\mathcal{O}(\ln \Delta)$-competitive. One of our algorithms is also $\mathcal{O}(\ln \Delta)$-competitive. Furthermore, the numerical values of the upper bounds on CRs of our algorithms are much smaller than that of Gupta and Li (2016)'s algorithm.

Lipton and Tomkins (1994) study a problem that is identical to the SORP with a single server, except that D_{max} and D_{min} are not known. They show that no algorithm can have a CR that is $\mathcal{O}(\ln \Delta)$ or better. They also propose an algorithm that is $\mathcal{O}(\ln \Delta)^{1+\epsilon}$-competitive, where ϵ is any positive real number. This randomized algorithm relies on a specific converging sequence of coin-flip probabilities. Faigle et al. (1996) extend these results in two ways. First, they show that multiple different sequences $z(1), z(2), \ldots$ can be used to construct randomized algorithms for the problem if $\sum_{i=1}^{\infty} 1/z(i)$ converges, and that

the resulting algorithm will be $\mathcal{O}(\ln(z(\Delta)))$-competitive. Second, the authors generalize the results to include the multiple server case. Unlike the problems studied by Lipton and Tomkins (1994) and Faigle et al. (1996), D_{min} and D_{max} are known in case of sharing economy platforms. Therefore, we exploit this information to obtain better CRs.

Garofalakis et al. (2002) consider the online scheduling of continuous media streams. This is identical to the single server case of the SORP, except that jobs now demand only a portion of the bandwidth of the server. They prove that no algorithm can be better than $\mathcal{O}(\ln \Delta/(1-\gamma))$-competitive, where γ is the maximum fractional bandwidth that any job can demand. They also propose an algorithm that has CR within a constant factor of $\ln \Delta$ if $\gamma < 1/\lceil \ln \Delta \rceil$. In case of rental platforms, the customer must demand at least one unit of resource, and therefore, the fractional bandwidth assumption of Garofalakis et al. (2002) does not hold.

3.1.4 Contribution

Our key contributions are as follows. We model problem features that many owners face when sharing resources via web platforms. Upon doing so, we obtain an online version of the interval scheduling problem that has not been studied before. Although similar problems have been studied, they do not capture all the problem features that we do. We prove a lower bound on CR of any algorithm for the ORP. Although fair algorithms have been studied in various applications, we establish their CRs or upper bounds on their CRs in the context of the ORP for the first time. We propose Algorithms D and R that are easy to implement and use intuitively-appealing threshold-criteria for making accept/decline decisions. We show that the CRs of both these algorithms are significantly better than that of any fair algorithm for specific ranges of parameter values. We modify our algorithms and analysis

to apply to the SORP setting, and show that the CRs of our algorithms are smaller than those of the algorithm proposed in Gupta and Li (2016).

3.2 A Lower Bound on CRs

To derive a lower bound on the CR of any algorithm for the ORP, we use a particular version of Yao's principle (Yao 1977). A statement and a proof of this version can be found in Krumke and Thielen (2015, Section 2.4.2). We start by stating this version of the Yao's principle, as applicable to the ORP, in Theorem 3.1 below.

Theorem 3.1. ***Yao's Principle for the ORP****: Let $\mathcal{A}$ be the set of all algorithms for the ORP as defined in Section 3.1.1, and let $\mathcal{A}_D$ be the subset of deterministic algorithms in $\mathcal{A}$. If $\mathcal{K} \subseteq \mathcal{L}(\mathcal{D})$ is a set of instances and if $\mathcal{Q}$ is a probability distribution with $\mathcal{K}$ as its support, then $\inf_{A \in \mathcal{A}_D} (E(OPT(L))/E(v_A(L)))$ is a lower bound on the CR of any algorithm in $\mathcal{A}$. The expectations are over the probability distribution $\mathcal{Q}$.*

Below in Theorem 3.2, we derive a lower bound on the CR of any algorithm for the ORP by using Theorem 3.1.

Theorem 3.2. *The CR of any algorithm for the ORP is bounded below by $\underline{r}$, where $\underline{r}(n, \Delta) \doteq \ln \Delta + 2$.*

Sketch of the proof. In the detailed proof presented in Appendix K, we present a set of instances $\mathcal{K}$, a probability distribution $\mathcal{Q}$ over $\mathcal{K}$, find a lower bound on $\inf_{A \in \mathcal{A}_D} (E(OPT(L))/E(v_A(L)))$, and apply Theorem 3.1. The key challenge in this exercise is to come up with $\mathcal{K}$ and $\mathcal{Q}$ such that a closed-form expression for the lower bound on $\inf_{A \in \mathcal{A}_D} (E(OPT(L))/E(v_A(L)))$ can be derived. We also want to construct $\mathcal{Q}$ and $\mathcal{K}$ such that the lower bound is as large as possible. □

The lower bound is increasing in Δ. Intuitively, this happens because the trade-offs become more pronounced for larger values of Δ, that is, there is possibility of loosing a larger reward from future jobs upon accepting a smaller job at each decision epoch. Therefore, the CR of every algorithm increases with Δ, and so does the LB on the CR of any algorithm. This lower bound is tight for some parameter values, as we will show later. It is also tight in the sense that we present an algorithm later (Algorithm R) that is $\mathcal{O}(\ln \Delta)$-competitive.

3.3 Algorithms

We begin this section by presenting Lemma 3.1 that will be the basis for deriving an upper bound on CRs of algorithms in subsequent proofs. Then in Section 3.3.1, we analyze CRs of all fair algorithms for the ORP. In Sections 3.3.2 and 3.3.3, we analyze CRs of Algorithms D and R respectively. Lemma 3.1 is based on the notions of *allocation scheme* and *allocation*, which we define next.

Definition 3.1. *Given two sets of jobs, B and C, an allocation scheme from B to C is a function w that maps the Cartesian product $B \times C$ to the set of non-negative real numbers such that for any job $b \in B$, $\sum_{c \in C} w(b,c)$ is less than or equal to d_b, the duration of job b. Furthermore, $w(b,c)$ is called the allocation from job b to job c.*

We are now ready to present Lemma 3.1.

Lemma 3.1. *Given Algorithm $A \in \mathcal{A}_D$ and an instance L, let B_L be the set of jobs accepted by A and C_L be an offline optimal set. If we are given a positive real number a and if for every instance $L \in \mathcal{L}(\Delta)$, we are given an allocation scheme w_L from B_L to C_L such that for any job $c \in C_L$, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq a$ holds, then a is an upper bound on the CR of Algorithm A.*

Sketch of the proof. The proof involves algebraic arguments, which are presented in Appendix L. The intuition behind the lemma is that if for a given instance L, we can allocate the durations of jobs accepted by Algorithm A to jobs in the offline optimal set such that each job in the offline optimal set is allocated at least $1/a$ of its duration, then $OPT(L)/v_A(L) \leq a$. Furthermore, if we can do the same for every instance $L \in \mathcal{L}(\Delta)$, then $r_A \leq a$. □

3.3.1 Fair Algorithms

A fair algorithm is one that assigns a job to a server with no conflicts, whenever at least one such server exists. Otherwise it declines the job. Next, in Theorem 3.3, we present results on the CRs of fair algorithms.

Theorem 3.3. *The following statements regarding the CR of an arbitrary fair algorithm A are true.*

1. $r_A(1, \Delta) = 2$ *if* $\Delta = 1$.
2. $r_A(1, \Delta) = 2\Delta + 1$ *if* $\Delta > 1$.
3. $r_A(n, \Delta) \leq 3$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.
4. $2\Delta + 1 \leq r_A(n, \Delta) \leq 2\Delta + 2$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

Sketch of the proof. The key challenge in establishing r_A is to come up with allocation schemes from the sets of jobs accepted by Algorithm A to offline optimal sets such that Lemma 3.1 can be applied. The upper bound in the first statement is tight because of a matching lower bound from Theorem 3.2. The lower bounds in the second and fourth statement are proved by identifying specific instances and calculating the ratio $OPT(L)/v_A(L)$

for them. The lower bound so proved is tight for the second statement. The detailed proof is given in Appendix M. □

It follows from Theorems 3.2 and 3.3 that the CR of any fair algorithm is optimal when $n = \Delta = 1$. Also, just like the lower bound on CR of any online algorithm (Theorem 3.2), the CR is increasing in Δ. However, the rate of increase is linear instead of logarithmic. This gap motivates the need for algorithms with better CRs. Intuitively, the rate of increase is linear because in some instances, a fair algorithm only accepts jobs with duration close to D_{min}. These decisions prevent it from accepting overlapping jobs with duration D_{max} that arrive in the future and that would have been accepted instead if we knew the entire instance upfront.

3.3.2 The Deterministic-Selection Algorithm (Algorithm D)

Job j is said to be admissible to server i if it does not conflict with any previously-accepted job of server i and if the job j's duration, d_j, is greater than or equal to the admission threshold of server i, $\phi(i)$ (the latter is defined in Definition 3.2 below). Algorithm D assigns an incoming job to the smallest-indexed server to which it is admissible, if at least one such server exists. Otherwise, it declines the job.

Admission thresholds serve to increase the likelihood that some longer duration jobs will be accepted by turning away small jobs and thereby overcoming a shortcoming of the fair algorithms. The thresholds, $\phi(i)$'s, and other notation required for subsequent results and discussions, are specified in Definition 3.2. Then we present a result on the CR of Algorithm D in Theorem 3.4.

Definition 3.2.

- $t : \mathbb{N} \times [1, \infty) \mapsto \mathbb{R}$ *such that* $t(n, y) \doteq \inf \left\{ x \geq 1 \;\middle|\; \frac{x}{3n} \left\lceil \frac{3n}{x} \right\rceil \left(1 + \frac{x}{3n}\right)^{n - \left\lceil \frac{3n}{x} \right\rceil} \geq y \right\}$
- $I \doteq \left\lceil \dfrac{3n}{t\,(n, \Delta)} \right\rceil$
- $\phi(i) \doteq \begin{cases} D_{min} & \text{if } i \leq I, i \in [n+1] \\ \dfrac{t\,(n, \Delta)\, I D_{min}}{3n} \left(1 + \dfrac{t\,(n, \Delta)}{3n}\right)^{i - I - 1} & \text{if } i > I, i \in [n+1] \end{cases}$

We are now ready to present Theorem 3.4.

Theorem 3.4. *The following statements regarding the CR of Algorithm D are true.*

1. $r_D(1, \Delta) = 2$ *if* $\Delta = 1$.
2. $r_D(1, \Delta) = 2\Delta + 1$ *if* $\Delta > 1$.
3. $r_D(n, \Delta) \leq 3$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.
4. $r_D(n, \Delta) \leq t\,(n, \Delta) + 1$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

Sketch of the proof. The first three statements follow directly from Theorem 3.3 because $\phi(i) = D_{min}$ for all i, and therefore, Algorithm D is a fair algorithm for these cases. So, we present the proof for the fourth statement only, the details of which are given in Appendix N. Just like Theorem 3.3, the crux of this proof lies in identifying allocation schemes from the sets of jobs accepted by the algorithm to the offline optimal sets such that Lemma 3.1 can be applied to prove the proposed upper bound. □

When $\Delta > 1$, we are able to obtain an explicit upper bound on the CR of Algorithm D for the limiting case $n \to \infty$ in the corollary below.

Corollary 3.1. $\lim\limits_{n \to \infty} r_D(n, \Delta) \leq 3 \ln \Delta + 4$.

Table 3.2: Numerical values of upper bounds on r_D from Theorem 3.4.

n \ Δ	2	4	8	16	32	64	128
5	6.40	9.16	12.23	16.00	19.74	24.09	29.12
100	6.10	8.21	10.33	12.48	14.63	16.80	18.99

From the definition of function t and upon taking limit, we have $\lim_{n\to\infty} t(n, \Delta) = 3\ln\Delta + 3$. Then, using Theorem 3.4, we get the desired result.

Therefore, by cleverly setting individual thresholds for the servers, Algorithm D achieves a CR that is $\mathcal{O}(\ln\Delta)$ in the limit as $n \to \infty$. In terms of CR, Algorithm D outperforms any fair algorithm when n and Δ are sufficiently large. Table 3.2 gives values of either the CR or the upper bound on the CR of Algorithm D, as derived in Theorem 3.4, for some parameter values.

3.3.3 The Randomized-Selection Algorithm (Algorithm R)

Algorithm R also uses a threshold, but it is sampled from a specified distribution. In the beginning, Algorithm R samples the threshold x from the distribution with CDF μ (defined in Definition 3.3 below). The same threshold x is then used for all jobs in the instance. Algorithm R declines all incoming jobs with duration strictly smaller than the threshold x. Any qualifying job is assigned to the smallest-indexed server with no conflicts, if at least one such server exists. Otherwise, the job is declined.

The idea behind the threshold in Algorithm R is same as that in Algorithm D: to increase the likelihood that some longer duration jobs will be accepted by turning away small jobs and thereby overcoming a shortcoming of the fair algorithms.

Below, in Definition 3.3, we define CDF μ of the distribution from which Algorithm R samples the threshold.

Definition 3.3. $\mu : [D_{min}, D_{max}] \mapsto [0, 1]$ *such that* $\mu(x) \doteq (1 + \ln(x/D_{min}))/(1 + \ln \Delta)$.

Next, we present results on the CR of Algorithm R in Theorem 3.5.

Theorem 3.5. *The following statements regarding the CR of Algorithm R are true.*

1. $r_R(1, \Delta) = 2$ *if* $\Delta = 1$.
2. $r_R(1, \Delta) \leq 3 \ln \Delta + 3$ *if* $\Delta > 1$.
3. $r_R(n, \Delta) \leq 3$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.
4. $r_R(n, \Delta) \leq 4 \ln \Delta + 4$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

Sketch of the proof. Statements 1 and 3 follow directly from Theorem 3.3 because the sampled threshold is equal to D_{min} with probability 1, and therefore, Algorithm R is a fair algorithm for these cases. For the proof of other two statements, we consider an arbitrary instance, L, and present an allocation scheme. Using this allocation scheme, we derive a lower bound on the total duration of jobs that are accepted by Algorithm R when the sampled threshold is equal to x. We then use this lower bound to show that $OPT(L)/v_R(L) \leq 3 \ln \Delta + 3$ when $n = 1$, and $OPT(L)/v_R(L) \leq 4 \ln \Delta + 4$ when $n \in \mathbb{N}$. The details are given in Appendix O. □

CR of Algorithm R is $\mathcal{O}(\ln \Delta)$, just as the lower bound on CRs derived in Theorem 3.2, and is therefore optimal in this sense. In terms of CR, R outperforms any fair algorithm when Δ is large enough, and it outperforms Algorithm D when n is small enough and Δ is large enough. Table 3.3 gives numerical values of lower bounds and upper bounds on CRs derived in Sections 3.2 and 3.3.

Table 3.3: Numerical values of lower bound on CRs from Theorems 3.2 and either CRs or upper bounds on CRs from Theorems 3.3, 3.4, and 3.5. Algorithm A represents any fair algorithm.

n	1			10			100		
Δ	1	5	25	1	5	25	1	5	25
$\underline{r}$	2.00	3.61	5.22	2.00	3.61	5.22	2.00	3.61	5.22
Algorithm A	2.00	11.00	51.00	3.00	12.00	52.00	3.00	12.00	52.00
Algorithm D	2.00	11.00	51.00	3.00	9.45	15.89	3.00	8.89	13.86
Algorithm R	2.00	7.83	12.66	3.00	10.44	16.88	3.00	10.44	16.88

3.4 Jobs Arrive at Their Start Times (SORP)

In this section, we study the special case of the ORP, abbreviated as SORP, where all the jobs arrive exactly at their start times, and this fact is known upfront to the decision maker. Although we apply Algorithm R to the SORP exactly the way we did for the ORP, we modify the server-specific admission thresholds of Algorithm D. This is done to exploit the extra information known to the decision maker that each job will arrive at its start time.

Similar to Sections 3.2 and 3.3, we derive the lower bound on the CR of any algorithm for the SORP (Theorem 3.6). Then, we analyze CRs of fair algorithms and Algorithms D and R for the SORP (Theorems 3.7, 3.8, and 3.9). The proofs are very similar to the corresponding derivations for the ORP, and are presented in Appendices P, Q, R, and S, respectively. The lower bound on CR of any algorithm and the CRs or upper bounds on the CRs of the studied algorithms are smaller than those for the ORP because the set of allowed instances for the SORP is a proper subset of those for the ORP and the CR is the supremum of relative performance over the set of all instances. The take-aways from the

results in this section are similar to those from the results on the ORP.

We begin by deriving a lower bound on CR of any algorithm for the SORP in Theorem 3.6.

Theorem 3.6. *The CR of any algorithm for the SORP is bounded below by* $\underline{r}^*$*, where* $\underline{r}^*(n, \Delta) \doteq \ln \Delta + 1$.

We next present results on CRs of fair algorithms in Theorem 3.7.

Theorem 3.7. *The following statements regarding the CR of an arbitrary fair algorithm* A *are true.*

1. $r_A^*(1, \Delta) = 1$ *if* $\Delta = 1$.
2. $r_A^*(1, \Delta) = \Delta + 1$ *if* $\Delta > 1$.
3. $r_A^*(n, \Delta) \leq 2$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.
4. $\Delta + 1 \leq r_A^*(n, \Delta) \leq \Delta + 2$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

Gupta and Li (2016) also prove Statement 2 in Theorem 3.7, but they do so using arguments different from ours. We prove all four statements using the common framework of allocation schemes and Lemma 3.1.

Algorithm D works the same way for the SORP as it does for the ORP, the only difference is that the server thresholds are given by ϕ^*s (defined in Definition 3.4) instead of ϕs. Next, we present some notation in Definition 3.4.

Definition 3.4.

- $t^* : \mathbb{N} \times [1, \infty) \mapsto \mathbb{R}$ *such that* $t^*(n, y) \doteq \inf \left\{ x \geq 1 \;\middle|\; \frac{x}{2n} \left\lceil \frac{2n}{x} \right\rceil \left(1 + \frac{x}{2n}\right)^{n - \left\lceil \frac{2n}{x} \right\rceil} \geq y \right\}$
- $I^* \doteq \left\lceil \frac{2n}{t^*(n, \Delta)} \right\rceil$

- $\phi^*(i) \doteq \begin{cases} D_{min} & \text{if } i \leq I^*, i \in [n+1] \\ \dfrac{t^*(n,\Delta)\, I^* D_{min}}{2n} \left(1 + \dfrac{t^*(n,\Delta)}{2n}\right)^{i-I^*-1} & \text{if } i > I^*, i \in [n+1] \end{cases}$

The notation defined above differs from that in Definition 3.2 in that each occurrence of 3 in Definition 3.2 is replaced by 2 here. Next, we present the result on the CR of Algorithm D in Theorem 3.8 below.

Theorem 3.8. *The following statements regarding the CR of Algorithm D are true.*

1. $r_D^*(1,\Delta) = 1$ *if* $\Delta = 1$.

2. $r_D^*(1,\Delta) = \Delta + 1$ *if* $\Delta > 1$.

3. $r_D^*(n,\Delta) \leq 2$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.

4. $r_D^*(n,\Delta) \leq t^*(n,\Delta) + 1$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

When $\Delta > 1$, we are able to obtain an explicit upper bound on the CR of Algorithm D for the limiting case $n \to \infty$ in the corollary below.

Corollary 3.2. $\lim_{n\to\infty} r_D^*(n,\Delta) \leq 2\ln\Delta + 3$.

Finally, we present results on the CR of Algorithm R in Theorem 3.9.

Theorem 3.9. *The following statements regarding the CR of Algorithm R are true.*

1. $r_R^*(1,\Delta) = 1$ *if* $\Delta = 1$.

2. $r_R^*(1,\Delta) \leq 2\ln\Delta + 2$ *if* $\Delta > 1$.

3. $r_R^*(n,\Delta) \leq 2$ *if* $\Delta = 1$ *and* $n \in \mathbb{N}$.

4. $r_R^*(n,\Delta) \leq 3\ln\Delta + 3$ *if* $\Delta > 1$ *and* $n \in \mathbb{N}$.

Table 3.4: Numerical values of lower bound on CRs from Theorem 3.6 and either CRs or upper bounds on CRs from Theorems 3.7, 3.8, and 3.9, and Gupta and Li (2016). Algorithm A represents any fair algorithm.

n	1			10			100		
Δ	1	5	25	1	5	25	1	5	25
$\underline{r}^*$	1.00	2.61	4.22	1.00	2.61	4.22	1.00	2.61	4.22
Algorithm A	1.00	6.00	26.00	2.00	7.00	27.00	2.00	7.00	27.00
Algorithm D	1.00	6.00	26.00	2.00	6.64	10.93	2.00	6.26	9.57
Algorithm R	1.00	5.22	8.44	2.00	7.83	12.66	2.00	7.83	12.66
Gupta and Li (2016)	–	54.96	56.37	–	109.93	112.74	–	109.93	112.74

Table 3.4 gives numerical values of lower bounds and CRs or upper bounds on CRs as derived in this section. We also compare our results with Gupta and Li (2016) who study a variant of the SORP in which the servers are available only for specific time intervals. By making these time intervals arbitrarily long, we get back the SORP. Gupta and Li (2016) only consider cases where $\Delta > 1$ and propose an $\mathcal{O}(\ln \Delta)$-competitive algorithm for it. The algorithm requires as input two additional parameters (in addition to n and Δ) that can be tuned as desired by the owner. The authors also present an upper bound on the CR of their algorithm for any value of input parameters. Since the expression for the upper bound are quite involved, we do not present it here. However, given n and Δ, we tune the two additional parameters to achieve the smallest value of this upper bound, which is presented in the last row of Table 3.4. Note that the CRs or upper bounds on CRs of our algorithms are significantly smaller than those of the algorithm in Gupta and Li (2016).

Chapter 4

Concluding Remarks

In Chapter 2, we studied the OMKP, which has applications in scheduling and revenue management. We derived lower bounds on the CR of any algorithm (deterministic or randomized, depending upon the context), analyzed CRs of some known algorithms from the online bin-packing literature after adapting them to the OMKP setting, and proposed three new algorithms that have CRs much smaller than the known algorithms for specific ranges of parameter values. We also studied the ORMP, a variant of OMKP in which all items have sizes equal to the knapsack capacities. We modified the algorithms proposed for the OMKP for application to the ORMP. We showed that one of our deterministic algorithms achieves the smallest CR amongst any deterministic algorithm for the ORMP, and another one of our algorithms achieves the smallest CR amongst any randomized algorithm.

In Chapter 3, we studied the ORP, which models the trade-offs faced by resource owners on popular sharing economy web platforms. We derived lower bounds on CR of any algorithm, analyzed CRs of fair algorithms, and proposed two new algorithms that have CRs significantly better than any fair algorithm for specific ranges of parameter values. We also modified the algorithms proposed for the ORP for application to the SORP and analyzed

their CRs.

Both the OMKP and the ORP have many common features, one of which is that at each decision epoch, the decision maker faces a trade-off between the reward (or value) he or she earns from the current job and the potential loss of value from future jobs that may need to be turned away because of this decision. This trade-off arises because of capacity constraints. Further, in both the problems, the decision maker must make the accept or decline decision without any information about the jobs that are yet to arrive. Decision makers may intuitively realize that using thresholds to reserve capacity for jobs with higher rewards that may arrive in the future will help improve the CR relative to the fair algorithms. The new algorithms we proposed for both the OMKP and the ORP use precisely this intuition. However, the choice of thresholds is important. In fact, the CR may deteriorate depending upon the choice of thresholds. For example, if all the thresholds are strictly greater than 1 for the OMKP or strictly greater than D_{min} for the ORP, then the CR for that algorithm will be ∞.

For both the OMKP and the ORP, we not only propose new algorithms and characterize their CRs, but also analyze CRs of known algorithms or intuitive algorithms that a decision maker may come up with. Therefore, we present a rigorous basis to help the decision maker choose an appropriate algorithm. Even in cases where he or she chooses a known or an intuitive algorithm, he or she can be more confident in his or her decision because the trade-offs have been made explicit by our analysis.

For both the problems, we studied special cases, namely, the ORMP for the OMKP and the SORP for the ORP. In both special cases, the set of allowed instances is a proper subset of the set of instances allowed in the original formulation. Although the algorithms for original formulations can be applied to the special cases and the results on the CRs of these

algorithms will still be valid, we showed that by tweaking algorithms to use the additional information that comes from having to deal with the smaller set of allowed instances and by analyzing CRs for the special cases, we get better worst-case relative performance guarantees when compared to the respective original formulations. The same approach can be used for other applications that are special cases of either the OMKP or the ORP, but which have not been studied in this thesis.

Although we designed all our algorithms with the objective of achieving the smallest CR in mind, they are also candidates for applications in which the future uncertainty is completely characterized by known probability distributions. In such a situation, the expected reward of various algorithms available can be estimated by sampling job sequences from the known probability distribution. If one of our algorithms yields higher average reward as compared to other available algorithms, then it can be used. Our algorithms are easy to implement and do not require detailed computations. Another advantage of our algorithms is that they have known worst-case relative performances that will hold even if job arrivals deviate from the distributional assumptions, thereby giving greater confidence to the decision makers.

There are many problems that remain open. For both the OMKP and the ORP, algorithms that achieve the smallest possible CRs are not known as yet. This also holds true for the SORP. Future research could focus on establishing the smallest possible CRs. In addition, in some applications of OMKP, the knapsacks have different capacities. Such generalization of OMKP could form the basis of future efforts. Finally, another possible future direction is to consider a variant of the ORP where the rewards are increasing and concave in job durations, reflecting the discounts for longer-duration contracts.

Bibliography

Albers, Susanne. 2003. Online algorithms: a survey. *Mathematical Programming* **97**(1) 3–26.

AngelList. 2018. Sharing economy startups. URL `https://angel.co/sharing-economy-4`.

Azar, Yossi, Joan Boyar, Lene M. Favrholdt, Kim S. Larsen, Morten N. Nielsen. 2000. Fair versus unrestricted bin packing. *Algorithm Theory - SWAT 2000: 7th Scandinavian Workshop on Algorithm Theory Bergen, Norway, July 5–7, 2000 Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 200–213.

Babaioff, Moshe, Nicole Immorlica, David Kempe, Robert Kleinberg. 2007. A knapsack secretary problem with applications. Moses Charikar, Klaus Jansen, Omer Reingold, José D. P. Rolim, eds., *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007. Proceedings*. Springer Berlin Heidelberg, 16–28.

Ball, Michael O., Maurice Queyranne. 2009. Toward robust revenue management: Competitive analysis of online booking. *Operations Research* **57**(4) 950–963.

Baruah, S., G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, F. Wang. 1992. On the competitiveness of on-line real-time task scheduling. *Real-Time Systems* **4**(2) 125–144.

Birge, John R, Francois Louveaux. 2011. *Introduction to stochastic programming*. Springer Science & Business Media.

Böckenhauer, Hans-Joachim, Dennis Komm, Richard Královič, Peter Rossmanith. 2014. The online

knapsack problem: Advice and randomization. *Theoretical Computer Science* **527** 61–72.

Borodin, Allan, Ran El-Yaniv. 1998. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA.

Boyar, J., K. S. Larsen. 1999. The seat reservation problem. *Algorithmica* **25**(4) 403–417.

Bubeck, Sébastien, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* **5**(1) 1–122.

Buchbinder, Niv, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, Maxim Sviridenko. 2013. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Operations Research* **61**(4) 1014–1029.

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

Buchbinder, Niv, 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) 93–263.

Cygan, Marek, Łukasz Jeż, Jiří Sgall. 2016. Online knapsack revisited. *Theory of Computing Systems* **58**(1) 153–190.

DCA. 2012. *California Tenants: A Guide to Residential Tenants' and Landlords' RIghts and Responsibilities*. California Department of Consumer Affairs. URL `http://www.dca.ca.gov/publications/landlordbook/catenant.pdf`.

den Heuvel, Wilco Van, Albert P. M. Wagelmans. 2010. Worst-case analysis for a general class of online lot-sizing heuristics. *Operations Research* **58**(1) 59–67.

Denton, Brian, Diwakar Gupta, Keith Jawahir. 2003. Managing increasing product variety at integrated steel mills. *Interfaces* **33**(2) 41–53.

Elmachtoub, Adam N., Retsef Levi. 2015. From cost sharing mechanisms to online selection problems. *Mathematics of Operations Research* **40**(3) 542–557.

Elmachtoub, Adam N., Retsef Levi. 2016. Supply chain management with online customer selection. *Operations Research* **64**(2) 458–473.

Faigle, U., R. Garbe, W. Kern. 1996. Randomized online algorithms for maximizing busy time interval scheduling. *Computing* **56**(2) 95–104.

Faigle, Ulrich, Willem M. Nawijn. 1995. Note on scheduling intervals on-line. *Discrete Applied Mathematics* **58**(1) 13 – 17.

Ferguson, Thomas S. 1989. Who solved the secretary problem? *Statistical Science* **4**(3) 282–289.

Fiat, Amos, Gerhard J. Woeginger, eds. 1998. *Online Algorithms: The State of the Art*, *Lecture Notes in Computer Science*, vol. 1442. Springer.

Fraiberger, Samuel P., Arun Sundararajan. 2017. Peer-to-peer rental markets in the sharing economy. NYU Stern School of Business Research Paper. Available at SSRN: https://ssrn.com/abstract=2574337.

Freeman, P. R. 1983. The secretary problem and its extensions: A review. *International Statistical Review / Revue Internationale de Statistique* **51**(2) 189–206.

Fung, Stanley P. Y., Chung Keung Poon, Feifeng Zheng. 2008. Online interval scheduling: randomized and multiprocessor cases. *Journal of Combinatorial Optimization* **16**(3) 248–262.

Garofalakis, Minos, Yannis Ioannidis, Banu Özden, Avi Silberschatz. 2002. Competitive on-line scheduling of continuous-media streams. *Journal of Computer and System Sciences* **64**(2) 219 – 248.

Golrezaei, Negin, Hamid Nazerzadeh, Paat Rusmevichientong. 2014. Real-time optimization of personalized assortments. *Management Science* **60**(6) 1532–1551.

Graham, R. L. 1966. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal* **45**(9) 1563–1581.

Guerriero, Francesca, Rosita Guido. 2011. Operational research in the management of the operating theatre: a survey. *Health care management science* **14**(1) 89–114.

Gupta, Diwakar, Fei Li. 2016. Reserve driver scheduling. *IIE Transactions* **48**(3) 193–204.

Hamari, Juho, Mimmi Sjöklint, Antti Ukkonen. 2016. The sharing economy: Why people participate in collaborative consumption. *Journal of the Association for Information Science and Technology* **67**(9) 2047–2059.

Han, Xin, Yasushi Kawase, Kazuhisa Makino. 2015. Randomized algorithms for online knapsack problems. *Theoretical Computer Science* **562** 395 – 405.

Howard, Ronald A. 1960. *Dynamic Programming Dynamic Programming and Markov Processes*. The M.I.T. Press.

Jaillet, Patrick, Michael R. Wagner. 2008. Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research* **56**(3) 745–757.

Johnson, David S. 1973. Near-optimal allocation algorithms. Ph.D. thesis, Massachusetts Institute of Technology. URL `http://dspace.mit.edu/handle/1721.1/57819`.

Karlin, Anna R., Mark S. Manasse, Larry Rudolph, Daniel D. Sleator. 1988. Competitive snoopy caching. *Algorithmica* **3**(1) 79–119.

Kellerer, Hans, Ulrich Pferschy, David Pisinger. 2004. *Knapsack Problems*. Springer.

Kleinberg, Robert. 2005. A multiple-choice secretary algorithm with applications to online auctions. *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '05, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 630–631.

Kleywegt, Anton J., Jason D. Papastavrou. 1998. The dynamic and stochastic knapsack problem. *Operations Research* **46**(1) 17–35.

Kleywegt, Anton J., Jason D. Papastavrou. 2001. The dynamic and stochastic knapsack problem with random sized items. *Operations Research* **49**(1) 26–41.

Knuth, D.E. 1968. *The Art of Computer Programming; Volume 2: Seminumerical Algorithms*. 1st ed. Addison-Wesley.

Kolen, Antoon W.J., Jan Karel Lenstra, Christos H. Papadimitriou, Frits C.R. Spieksma. 2007. Interval scheduling: A survey. *Naval Research Logistics (NRL)* **54**(5) 530–543.

Koren, Gilad, Dennis Shasha. 1995. Dover: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM J. Comput.* **24**(2) 318–339.

Krumke, Sven O., Clemens Thielen. 2015. Introduction to online optimization. URL `http://www.mathematik.uni-kl.de/fileadmin/AGs/opt/Lehre/SS14/Online_SS14/online-optimization.pdf`.

Lan, Yingjie, Michael O. Ball, Itir Z. Karaesmen. 2011. Regret in overbooking and fare-class allocation for single leg. *Manufacturing & Service Operations Management* **13**(2) 194–208.

Lan, Yingjie, Huina Gao, Michael O. Ball, Itir Karaesmen. 2008. Revenue management with limited demand information. *Management Science* **54**(9) 1594–1609.

Lee, C. C., D. T. Lee. 1985. A simple on-line bin-packing algorithm. *J. ACM* **32**(3) 562–572.

Lin, Grace Y., Yingdong Lu, David D. Yao. 2008. The stochastic knapsack revisited: Switch-over policies and dynamic pricing. *Operations Research* **56**(4) 945–957.

Lipton, Richard J., Andrew Tomkins. 1994. Online interval scheduling. *In Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. 302–311.

Ma, Will, David Simchi-Levi. 2017. Online resource allocation under arbitrary arrivals: Optimal algorithms and tight competitive ratios. Available at SSRN: `https://ssrn.com/abstract=2989332`.

Marchetti-Spaccamela, A., C. Vercellis. 1995. Stochastic on-line knapsack problems. *Mathematical Programming* **68**(1) 73–104.

Martin, Chris J. 2016. The sharing economy: A pathway to sustainability or a nightmarish form of neoliberal capitalism? *Ecological Economics* **121**(Supplement C) 149 – 159.

Miyazawa, Hiroyuki, Thomas Erlebach. 2004. An improved randomized on-line algorithm for a weighted interval selection problem. *Journal of Scheduling* **7**(4) 293–311.

Seiden, Steven S. 1998. Randomized online interval scheduling. *Operations Research Letters* **22**(4) 171 – 177.

Sleator, Daniel D, Robert E Tarjan. 1985a. Amortized efficiency of list update and paging rules. *Communications of the ACM* **28**(2) 202–208.

Sleator, Daniel Dominic, Robert Endre Tarjan. 1985b. Self-adjusting binary search trees. *Journal of the ACM (JACM)* **32**(3) 652–686.

Slyke, Richard Van, Yi Young. 2000. Finite horizon stochastic knapsacks with applications to yield management. *Operations Research* **48**(1) 155–172.

Stein, C., V.A. Truong, X. Wang. 2016. Advance reservations with heterogeneous customers. Available at `http://www.columbia.edu/~vt2196/OnlineCapacityAllocation9.pdf`.

Wagner, Michael R. 2010. Fully distribution-free profit maximization: The inventory management case. *Mathematics of Operations Research* **35**(4) 728–741.

Wagner, Michael R. 2011. Online lot-sizing problems with ordering, holding and shortage costs. *Operations Research Letters* **39**(2) 144 – 149.

Wang, Xinshang, Van-Anh Truong, David Bank. 2015. Online advance admission scheduling for services with customer preferences. Available at `http://www.columbia.edu/~vt2196/OnlineStochasticScheduling4.pdf`.

Yao, A. C. C. 1977. Probabilistic computations: Toward a unified measure of complexity. *Foundations of Computer Science, 1977., 18th Annual Symposium on*. 222–227.

Zhou, Yunhong, Deeparnab Chakrabarty, Rajan Lukose. 2008a. Budget constrained bidding in keyword auctions and online knapsack problems. Christos Papadimitriou, Shuzhong Zhang, eds., *Internet and Network Economics: 4th International Workshop, WINE 2008, Shanghai, China, December 17-20, 2008. Proceedings*. Springer Berlin Heidelberg, 566–576.

Zhou, Yunhong, Deeparnab Chakrabarty, Rajan M. Lukose. 2008b. Budget constrained bidding in keyword auctions and online knapsack problems. *Techinal Report* URL `http://www.hpl.hp.com/techreports/2008/HPL-2008-9.pdf`.

Zilok. 2018. Rental wedding gown. URL `http://us.zilok.com/rental/119802-wedding-gown.html?l=81615#ItemDescription`.

Appendices

A Proof of Theorem 2.1

Proof. We first prove the result only for those α that are reciprocals of integers. Note that $M = 1/\alpha$ for such α from Equation 2.1. The proof is by contradiction. Let A be a deterministic algorithm such that $r_A(n, \alpha, \Delta) < \underline{r}_{det}(n, \alpha, \Delta)$. Given n, α and Δ, denote $r_A(n, \alpha, \Delta)$ by r and $\underline{r}_{det}(n, \alpha, \Delta)$ by $\underline{r}_{det}$. Consider an instance L such that the duration of any job i in L is given by $d_i = \min\{\alpha, v_i\}$, where v_i is the value of job i. Because the duration and value density of such jobs are uniquely determined by their values, it suffices to characterize these jobs only by their values. The sequence of values of jobs in L is one that is obtained by appending the jobs in sequences $s_0, s_1, \ldots, s_{Mn-I+1}$, presented in Table A.1, one after the other. Note that $I \doteq \lceil Mn/r \rceil$, $\epsilon > 0$ is arbitrarily small, $O_1(\epsilon) \doteq (1+\epsilon)rI\epsilon/(Mn)$, and $O_k(\epsilon) \doteq (1+\epsilon)r/(Mn)\left(I\epsilon + \sum_{\ell=1}^{k-1} O_\ell(\epsilon)\right)$ for $k = 2, 3, \ldots, Mn - I$.

Integers $j_0, j_1, \ldots, j_{Mn-I}$ are such that when A is applied to L, I jobs are accepted from subsequence s_0, one job each is accepted from subsequences $s_1, s_2, \ldots, s_{Mn-I}$, and no job is accepted from subsequence s_{Mn-I+1}. In particular, the sequence of values of accepted jobs

Table A.1: Sequences $s_1, s_2, \ldots, s_{Mn-I+2}$. Each job is denoted by its value.

Notation	Sequence of Jobs
s_0	$\underbrace{\frac{1}{M+1}+\epsilon, \ldots, \frac{1}{M+1}+\epsilon}_{j_0}$
s_1	$\underbrace{\frac{rI}{Mn(M+1)}+O_1(\epsilon), \ldots, \frac{rI}{Mn(M+1)}+O_1(\epsilon)}_{j_1}$
s_2	$\underbrace{\frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)+O_2(\epsilon), \ldots, \frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)+O_2(\epsilon)}_{j_2}$
$\vdots$	$\vdots$
s_{Mn-I}	$\underbrace{\frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)^{Mn-I-1}+O_{Mn-I}(\epsilon), \ldots, \frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)^{Mn-I-1}+O_{Mn-I}(\epsilon)}_{j_{Mn-I}}$
s_{Mn-I+1}	$\underbrace{\alpha\Delta, \ldots, \alpha\Delta}_{Mn}$

can be written as:

$$\underbrace{\frac{1}{M+1}+\epsilon, \frac{1}{M+1}+\epsilon, \ldots, \frac{1}{M+1}+\epsilon}_{I},$$
$$\underbrace{\frac{rI}{Mn(M+1)}+O_1(\epsilon), \ldots, \frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)^{Mn-I-1}+O_{Mn-I}(\epsilon)}_{Mn-I}.$$

We next prove the following two claims to show that such an instance L indeed exists:

1. The maximum duration of any job in L is less than or equal to α and the value is less than or equal to $\alpha\Delta$.

2. There exist integers $j_0, j_1, \ldots, j_{Mn-I}$ such that when Algorithm A is applied to L, the sequence of values of accepted jobs is as claimed.

Proof of Claim 1: The duration of every job i in L is less than or equal to α because $d_i = \min\{\alpha, v_i\}$. The value of each job in L is less than or equal to $\alpha\Delta$ because the following statements are true.

- Value of each job in the subsequence s_{Mn-I+1} is no more than $\alpha\Delta$.

- The value of any job in any one of the subsequence $s_0, s_1, \ldots, s_{Mn-I}$ is less than $(rI/(Mn(M+1)))\,(1+r/(Mn))^{Mn-I}$, which by Definition 2.1 is equal to $f(r,Mn)/(M+1) = \alpha M/(M+1) f(r,Mn)$. Because $r < \underline{r}_{det}$, Lemma 2.1 and Equation 2.4 imply that $f(r,Mn) \leq \Delta(M+1)/M$. Therefore, the value of any job in any of subsequence $s_0, s_1, \ldots, s_{Mn-I}$ is less than or equal to $\alpha\Delta$. □

Proof of Claim 2: The proof is by construction. Note that the n knapsacks can accommodate any Mn jobs from L and this is also the maximum number possible because the size of each job in L is strictly greater than $1/(M+1)$ and the capacity of each knapsack is exactly 1.

Consider an instance L_0 that has Mn jobs of the same kind as in sequence s_1. If we apply Algorithm A to this sequence and it accepts less than I jobs, then $\frac{OPT(L_1)}{v_A(L_1)} \geq \frac{Mn\,(1/(M+1)+\epsilon)}{(I-1)\,(1/(M+1)+\epsilon)} = \frac{Mn}{r}\frac{r}{\lceil Mn/r\rceil - 1} > r$, where the last inequality holds because $\frac{Mn}{r} > \left\lceil \frac{Mn}{r} \right\rceil - 1$. But this is not possible because A is r-competitive. Therefore, Algorithm A must accept at least I jobs from the sequence. Let job i_0 be the Ith job that A accepted from the sequence. Then, j_0 is the position of job i_0 in L_0.

Consider an instance L_1 with first j_0 jobs of the same kind as in sequence s_1, and then Mn jobs of the same kind as in sequence s_2. The offline optimal solution for L_1 is to accept the last Mn jobs. If we apply Algorithm A to L_1, it must accept I jobs from the first j_0 jobs. If it accepts no other job from the next Mn jobs, then $\frac{OPT(L_1)}{v_A(L_1)} = \frac{Mn\,(rI/(Mn(M+1)) + O_1(\epsilon))}{I\,(1/(M+1)+\epsilon)} = \frac{rI + (M+1)MnO_1(\epsilon)}{I + (M+1)I\epsilon} = \frac{rI + (1+\epsilon)r(M+1)I\epsilon}{I + (M+1)I\epsilon} > \frac{rI + r(M+1)I\epsilon}{I + (M+1)I\epsilon} = r$. But, this is not possible because A is r-competitive. Therefore, Algorithm A must accept at least one job from the last Mn jobs in L_1. Let job i_1 be the (j_0+1)th job that Algorithm A accepted from L_1. Then, j_1 = position of job i_1 in L_1

minus j_0.

Similarly, we can prove the existence of $j_2, j_3, \ldots, j_{Mn-I}$ using the arguments described above for j_0 and j_1. □

By definition, $v_A(L)$ is the total value of jobs that Algorithm A will accept when we apply it to L. Therefore,

$$\implies v_A(L) = I\left(\frac{1}{M+1} + \epsilon\right) + \sum_{k=1}^{Mn-I}\left(\frac{rI}{Mn(M+1)}\left(1+\frac{r}{Mn}\right)^{k-1} + O_k(\epsilon)\right)$$

$$\implies v_A(L) = I\left(\frac{1}{M+1} + \epsilon\right) + \frac{rI}{(M+1)Mn}\frac{\left(1+\frac{r}{Mn}\right)^{Mn-I} - 1}{(1+\frac{r}{Mn}) - 1} + \sum_{k=1}^{Mn-I} O_k(\epsilon)$$

$$\implies \lim_{\epsilon\to 0}\frac{n}{v_A(L)} = \frac{1}{\frac{I}{(M+1)n}\left(1+\frac{r}{Mn}\right)^{Mn-I}}. \tag{1}$$

If we knew L upfront, it was optimal to accept only the last Mn jobs of value $\alpha\Delta$ each. Therefore, $OPT(L) = n\Delta$. Further, there exists a $\delta > 0$ such that $r < \underline{r}_{det} - \delta$ because $r < \underline{r}_{det}$. Since A is r-competitive, we have

$$\frac{OPT(L)}{v_A(L)} \leq r \text{ for any } \epsilon > 0$$

$$\implies \lim_{\epsilon\to 0}\frac{n\Delta}{v_A(L)} \leq r$$

$$\implies r\frac{I}{(M+1)n}\left(1+\frac{r}{Mn}\right)^{Mn-I} \geq \Delta \qquad \text{(substituting } \lim_{\epsilon\to 0}\frac{n}{v_A(L)} \text{ from Equation 1)}$$

$$\implies r\frac{I}{Mn}\left(1+\frac{r}{Mn}\right)^{Mn-I} \geq \Delta(M+1)/M$$

$$\implies r\frac{I}{Mn}\left(1+\frac{r}{Mn}\right)^{Mn-I} \geq (\underline{r}_{det} - \delta)\frac{I}{Mn}\left(1+\frac{(\underline{r}_{det}-\delta)}{Mn}\right)^{Mn-I}$$

(from Equation 2.4 and Lemma 2.1)

$$\implies f(r, Mn) \geq f(\underline{r}_{det} - \delta, Mn) \qquad \text{(by Definition 2.1)}$$

$$\implies r \geq \underline{r}_{det} - \delta \text{ (by Lemma 2.1),}$$

which is a contradiction. Hence, proved.

The lower bound $\underline{r}_{det}(n, \alpha, \Delta)$ must also hold for those values of α that are not reciprocals of integers because of two reasons. First reason is that the CR of any algorithm is increasing in α by the definition of CR. Second, for any given n and Δ, $\underline{r}_{det}(n, \alpha, \Delta)$ is constant-valued for $\alpha \in [\frac{1}{\ell}, \frac{1}{\ell-1})$ for any $\ell = 2, 3, \cdots$. Figure A.1 illustrates the intuition behind this line of reasoning.

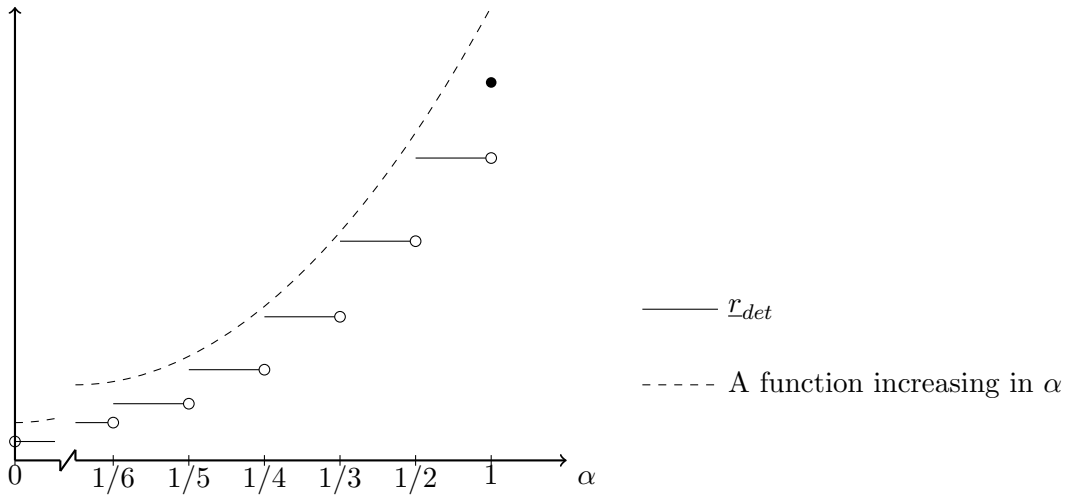

Figure A.1: Variation of $\underline{r}_{det}$ (for some n and Δ) and an arbitrary function that is increasing in α.

□

B Proof of Theorem 2.2

Proof. Consider only those values of α that are reciprocals of integers because the lower bound $\underline{r}_{ran}(n, \alpha, \Delta)$ remains valid for other values of α following the same reasoning that is described at the end of the proof of Theorem 2.1 (Appendix A). Note that $M = \dfrac{1}{\alpha}$ for such α from equation (2.1).

We construct a sequence of jobs, L, such that for job i in the sequence $d_i = \min\{\alpha, v_i\}$. Because the duration and value density of such jobs are uniquely determined by their values, it suffices to characterize these jobs only by their values. Let k be an arbitrary non-negative

integer and let $\epsilon > 0$ be arbitrarily small. Then L consists of $k+1$ subsequences, where the j-th subsequence consists of Mn jobs of value a_j and where $a_0 \doteq 1/(M+1) + \epsilon < a_1 < a_2 < \ldots < a_k \doteq \Delta/M$, and $a_j \doteq a_0 \left(\frac{a_k}{a_0}\right)^{j/k}$. In particular, we can write the sequence L as follows:

$$\underbrace{a_0, a_0, \cdots, a_0}_{Mn \text{ jobs}}, \underbrace{a_1, a_1, \cdots, a_1}_{Mn \text{ jobs}}, \cdots, \underbrace{a_k, a_k, \cdots, a_k}_{Mn \text{ jobs}}. \tag{2}$$

Let A be an arbitrary randomized algorithm for the OMKP. Given A, we can calculate x_j, the expected number of jobs with value a_j accepted by A from sequence L. Next, we apply A to an instance that consists of only the first j subsequences. The n knapsacks can accommodate any Mn jobs from this instance and this is also the maximum number possible because the size of each job in the instance is strictly greater than $1/(M+1)$ and the capacity of each knapsack is exactly 1. Therefore, the offline optimal solution is to accept the Mn jobs with values equal to a_j from the instance. But, the expected value of jobs accepted by A is $\sum_{i=0}^{j} x_i a_i$. By the definition of CR, $r_A(n, \alpha, \Delta) \geq \zeta_j^A \doteq \frac{Mna_j}{\sum_{i=0}^{j} x_i a_i}$, and also $r_A(n, \alpha, \Delta) \geq \max_{j=0,1,\cdots,k} \zeta_j^A$. Therefore, $\min_{A \in \mathcal{A}} \max_{j=0,1,\cdots,k} \zeta_j^A$, where $\mathcal{A}$ denotes the set of all randomized algorithms for the OMKP, is a valid lower bound on the CR of any randomized algorithm for the OMKP. We can translate the search for the minimum value of $\max_{j=0,1,\cdots,k} \zeta_j^A$ over $\mathcal{A}$ to a search over all the possible values of x_j's that randomized algorithms induce. Equivalently, we determine $\max_{x_j's} \min_{j=0,1,\cdots,k} \frac{1}{\zeta_j^A}$ using LP formulation (LP1).

$$\begin{aligned} &\max\ z && \text{(LP1)} \\ &\text{subject to: } z \leq \frac{\sum_{i=0}^{j} x_i a_i}{Mna_j} \quad \text{for every } j = 0, 1, \cdots, k \\ &\sum_{i=0}^{k} x_i \leq Mn \qquad \text{and} \\ &x_i \geq 0,\ i = 0, 1, \cdots, k \end{aligned}$$

The constraint $\sum_{i=0}^{k} x_i \leq Mn$ follows from the fact that the n knapsacks can accommodate a maximum of Mn jobs from sequence 2. It can be shown that given $a_0, \ldots, a_k$, there exists an optimal solution $(x_0^*, x_1^*, \ldots, x_k^*, z^*)$ to the above LP such that

$$z^* = \frac{\sum_{i=0}^{0} x_i^* a_i}{Mna_0} = \frac{\sum_{i=0}^{1} x_i^* a_i}{Mna_1} = \ldots = \frac{\sum_{i=0}^{k} x_i^* a_i}{Mna_k} \text{ and} \tag{3}$$

$$\sum_{i=0}^{k} x_i^* = Mn. \tag{4}$$

$$\text{Define } \zeta_j^* \doteq \frac{Mna_j}{\sum_{i=0}^{j} x_i^* a_i},\ j = 0, 1, \ldots, k. \tag{5}$$

Because ζ_j^*'s are identical by Equation 3 for all j, we drop subscript j, i.e. $\zeta^* \doteq \zeta_j^*$. We know that for any a, b, c and d such that $b, d \neq 0$ and $b \neq d$, $a/b = c/d \implies a/b = c/d = (a-c)/(b-d)$. Therefore, using Equation 5, we obtain

$$\zeta^* = \frac{Mna_j - Mna_{j-1}}{\sum_{i=0}^{j} x_i^* a_i - \sum_{i=0}^{j-1} x_i^* a_i} = Mn\frac{a_j - a_{j-1}}{a_j x_j^*}. \tag{6}$$

From Equation 5, $x_0^* = \dfrac{Mn}{\zeta^*}$, and from Equation 6, $x_j^* = Mn\dfrac{a_j - a_{j-1}}{a_j \zeta^*}$ for all $j = 1, \ldots, k$.

Plugging these values in Equation 4, $\sum_{i=0}^{k} x_i^* = Mn$, gives us

$$\begin{aligned} \zeta^* &= 1 + \sum_{j=1}^{k} \left(1 - \frac{a_{j-1}}{a_j}\right) = k + 1 - \sum_{j=1}^{k} \frac{a_{j-1}}{a_j} \\ &= k + 1 - k\left(\frac{a_0}{a_k}\right)^{1/k} = 1 + k\left(1 - \left(\frac{\frac{1}{M+1} + \epsilon}{\frac{\Delta}{M}}\right)^{1/k}\right). \end{aligned} \tag{7}$$

As discussed previously, ζ^*, being the reciprocal of the optimal objective value to LP1, gives us a valid lower bound on the CR of any randomized algorithm for given $n,\ \alpha$, and Δ. We note that this lower bound is increasing in k and decreasing in ϵ and that we may choose k

and ϵ arbitrarily. Therefore, we let $k \to \infty$ and $\epsilon \to 0$ and still get a lower bound that does not depend on k and ϵ. From Equation 7,

$$\lim_{\epsilon \to 0} \zeta^* = 1 + k \left(1 - \left(\frac{M}{(M+1)\Delta} \right)^{1/k} \right)$$

$$\implies \lim_{k \to \infty} \lim_{\epsilon \to 0} \zeta^* = 1 + \ln \frac{M+1}{M} + \ln \Delta. \text{ (Doing some algebra and using L'Hospital's Rule)}$$

□

C Proof of Statement 1 in Theorem 2.3

Proof. We use E_i to denote empty space and F_i to denote filled space in knapsack i after all the jobs in an instance have been either placed in knapsacks or declined (Figure C.1). Clearly, $E_i + F_i = 1$ for any $i \in [n]$.

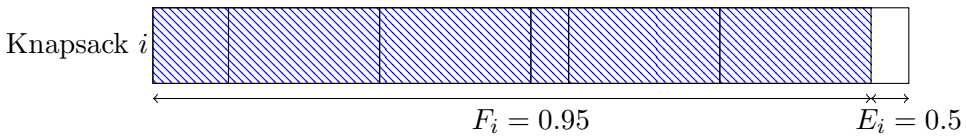

Figure C.1: Empty and filled spaces in knapsack i after all the jobs in an arbitrary instance have been either placed in knapsacks or declined. Shaded rectangles represent accepted jobs.

Given n, α and Δ, we denote $r_F(n, \alpha, \Delta)$ by r and v_F by v. Because of the reasons discussed in the sketch of the proof, we consider only those instances for which Algorithm F declines at least one job. Note that $1/(m+1) < \alpha \leq 1/m$ from Equation 2.1. We begin by proving an intermediate claim in Proposition C.1, which will help in streamlining this proof.

Proposition C.1. *If Algorithm F declines at least one job from an instance, then at most one knapsack has free capacity $\geq 1/(m+1)$ after all the jobs in the instance have been either placed in knapsacks or declined.*

Proof of Proposition C.1. Consider an instance where Algorithm F declines at least one job. Assume that free capacity in knapsack i, E_i, is greater than or equal to $1/(m+1)$. Because at least one job was declined, $E_i < \alpha$. Therefore,

$$\frac{1}{m+1} \leq E_i < \alpha \leq \frac{1}{m} \implies \frac{m-1}{m} < F_i \leq \frac{m}{m+1} \implies (m-1)\alpha < F_i \leq \frac{m}{m+1}.$$

Because α is the largest duration any job can have, knapsack i contains at least m jobs. Further, $F_i \leq m/(m+1)$ implies that at least one of these jobs has duration $\leq 1/(m+1)$. If there are two knapsacks with free capacity $\geq 1/(m+1)$, then as shown above, the knapsack with the larger index contains a job of duration $\leq 1/(m+1)$, which could have been fitted into the knapsack with the smaller index. But, this is impossible with Algorithm F. Hence, proved. □

The above proposition implies that only one of the following three cases can happen:

Case 1: There does not exist a knapsack $\widehat{i} \in [n]$ with free space $E_{\widehat{i}} \geq 1/(m+1)$.

$n\Delta$ is a trivial upper bound on $OPT(L)$ and since all knapsacks have free space strictly less than $1/(m+1)$,

$$v(L) > n(1 - 1/(m+1)) = nm/(m+1). \tag{8}$$

Therefore,

$$\frac{OPT(L)}{v(L)} < \frac{n\Delta}{\frac{nm}{m+1}} = \frac{m+1}{m}\Delta. \tag{9}$$

Case 2: There exists a unique knapsack $\widehat{i} \in [n]$ with free space $E_{\widehat{i}} \geq 1/(m+1)$ and such that $\widehat{i} \neq n$.

This case cannot happen when $n = 1$. So, we consider $n \geq 2$. We choose a knapsack $j \in \{\widehat{i}+1, \widehat{i}+2, \ldots, n\}$. Now,

$$F_j = 1 - E_j > 1 - \frac{1}{m+1} = \frac{m}{m+1} > \frac{m-1}{m} \geq (m-1)\alpha.$$

Because α is the largest duration any job can have, knapsack j has at least m jobs. Because we used Algorithm F, duration of each job in knapsack j is strictly greater than $E_{\widehat{i}}$, the free space in knapsack $\widehat{i}$. Thus,

$$F_j > mE_{\widehat{i}} \implies F_{\widehat{i}} + F_j > F_{\widehat{i}} + mE_{\widehat{i}} = 1 + (m-1)E_{\widehat{i}} \geq 1 + \frac{m-1}{m+1} = \frac{2m}{m+1}.$$

Therefore,

$$v(L) \geq \sum_{i \in [n]} F_i = \left(\sum_{i \in [n] \setminus \{\widehat{i}, j\}} F_i \right) + F_{\widehat{i}} + F_j > \frac{nm}{m+1}. \tag{10}$$

$n\Delta$ is a trivial upper bound on $OPT(L)$ and hence,

$$\frac{OPT(L)}{v(L)} < \frac{n\Delta}{\frac{nm}{m+1}} = \frac{m+1}{m}\Delta. \tag{11}$$

Case 3: There exists a unique knapsack $\widehat{i} \in [n]$ with free space $E_{\widehat{i}} \geq 1/(m+1)$ and such that $\widehat{i} = n$.

Because at least one job was declined,

$$E_n < \alpha \implies E_n < \frac{1}{m} \implies \frac{m-1}{m} < F_n \implies (m-1)\alpha < F_n,$$

which implies that knapsack n has at least m jobs because α is the largest duration any job can have. Again, $E_n < \alpha \implies F_n > 1-\alpha \implies$ the smallest job in knapsack n has duration $= (1-\alpha)/m + \epsilon$, for some ϵ such that $-(1-\alpha)/m < \epsilon \leq 1/(m+1) - (1-\alpha)/m$. Because we used Algorithm F and this particular job did not go into any of the previous knapsacks $\implies E_j \leq (1-\alpha)/m + \epsilon$ for any knapsack $j \in [n-1]$. But if $\epsilon < 0$, then $E_j \leq (1-\alpha)/m$ still holds. Therefore, $E_j \leq (1-\alpha)/m + \epsilon^+$ for some ϵ^+ such that $0 \leq \epsilon^+ \leq 1/(m+1) - (1-\alpha)/m$. Therefore, $F_j \geq 1 - (1-\alpha)/m - \epsilon^+$ for any knapsack $j \in [n-1]$. Further, $F_n \geq 1 - \alpha + m\epsilon$ because knapsack n has at least m jobs, the smallest of which has duration $(1-\alpha)/m + \epsilon$. Because at least one job was declined, $F_n > 1 - \alpha$. Thus, $F_n \geq 1 - \alpha + m\epsilon^+$, which implies

that

$$\sum_{i\in[n]} F_i = \left(\sum_{i\in[n-1]} F_i\right) + F_n \geq (n-1)\left(1-(1-\alpha)/m\right) + 1 - \alpha + (m+1-n)\epsilon^+. \quad (12)$$

Given L, if we solve the MKP, the offline multiple knapsack problem, then there is at least one and possibly multiple subsets of jobs from L that are optimal solutions. Denote the set of all these optimal subsets by Q. Given $q \in Q$, let x_q denote the total duration of jobs in q that were declined by Algorithm F from L and y_q be the cumulative duration of those jobs in a which were accepted by Algorithm F from L, and let z_q be the cumulative duration of the jobs that were accepted by Algorithm F from L but do not belong to q. Let $\bar{v}(x_q),\ \bar{v}(y_q)$, and $\bar{v}(z_q)$ denote the average value density of jobs in $x_q,\ y_q$, and z_q respectively. Then, $\dfrac{OPT(L)}{v(L)} = \dfrac{x_q\bar{v}(x_q) + y_q\bar{v}(y_q)}{z_q\bar{v}(z_q) + y_q\bar{v}(y_q)}$. $\dfrac{OPT(L)}{v(L)} \geq 1$ implies that $x_q\bar{v}(x_q) \geq z_q\bar{v}(z_q)$, and therefore, $\dfrac{x_q\bar{v}(x_q) + y_q\bar{v}(y_q)}{z_q\bar{v}(z_q) + y_q\bar{v}(y_q)}$ is decreasing in $\bar{v}(y_q)$. Further, since $1 \leq \bar{v}(y_q) \leq \Delta$, $\dfrac{OPT(L)}{v(L)} \leq \dfrac{x_q\bar{v}(x_q) + y_q}{z_q\bar{v}(z_q) + y_q}$. $\bar{v}(x_q), \bar{v}(z_q) \in [1,\Delta]$ implies that $\dfrac{OPT(L)}{v(L)} \leq \dfrac{x_q\Delta + y_q}{z_q + y_q}$. Because $1/(m+1) \leq E_n$, every job declined by Algorithm F had duration $> 1/(m+1)$. Therefore, x_q cannot contain more than mn jobs, which implies that $x_q \leq nm\alpha$ because α is the largest duration any job can have. Further, we already know that $x_q + y_q \leq n,\ y_q + z_q = \sum_{i\in[n]} F_i \geq (n-1)\left(1-(1-\alpha)/m\right) + 1 - \alpha + (m+1-n)\epsilon^+$. Therefore,

$$\frac{OPT(L)}{v(L)} \leq \frac{nm\alpha\Delta + (n - nm\alpha)}{(n-1)(1-\frac{1-\alpha}{m}) + 1 - \alpha + (m+1-n)\epsilon^+}.$$

Using the facts that the denominator of the above bound is linear in ϵ^+ and that $0 \leq \epsilon^+ \leq 1/(m+1) - (1-\alpha)/m$, we can show by simple algebraic manipulation that

$$\frac{OPT(L)}{v(L)} \leq \max\left(\frac{m+1}{m}\Delta, \frac{nm\alpha\Delta + (n - nm\alpha)}{(n-1)\left(1-\frac{1-\alpha}{m}\right) + 1 - \alpha}\right). \quad (13)$$

Combining Equations 9, 11, and 13 from the above three cases, the following statement

holds for any arbitrary instance L from which Algorithm F declines at least one job.

$$\frac{OPT(L)}{v(L)} \leq \max\left(\frac{m+1}{m}\Delta, \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}\right)$$
$$\implies r \leq \max\left(\frac{m+1}{m}\Delta, \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}\right). \tag{14}$$

To complete the proof, we need to establish that this upper bound is tight. We will do that next by deriving a matching lower bound on r. We do this by first proving that $r \geq \frac{m+1}{m}\Delta$ and then that $r \geq \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}$.

Consider an instance L_1 such that the first mn jobs in it are of duration $(1/(m+1)+\epsilon)$ each ($\epsilon > 0$ and arbitrarily small) and then there are $(m+1)n$ jobs of duration $1/(m+1)$ each. The value density of first mn jobs is 1 and the last $(m+1)n$ jobs is Δ. Algorithm F accepts only the first mn jobs but it is optimal to select the last $(m+1)n$ jobs. Thus,

$$\frac{OPT(L_1)}{v(L_1)} = \frac{n}{nm\left(\frac{1}{m+1}+\epsilon\right)}\Delta \implies r \geq \frac{n}{nm\left(\frac{1}{m+1}+\epsilon\right)}\Delta.$$

Because $\epsilon > 0$ was arbitrary, Equation 15 implies that

$$r \geq \frac{m+1}{m}\Delta. \tag{15}$$

Next, we show that $r \geq \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}$ by considering three cases, namely $n = 1$, $2 \leq n \leq m$, and $n > m$. This completes the proof.

Case A: $n = 1$.

Consider an instance L_2 such that the first job in it is of duration $1 - m\alpha$ (ignore if $1 - m\alpha = 0$), next m jobs of duration $\left(\frac{(m-1)\alpha}{m} + \epsilon\right)$ each ($\epsilon > 0$ and arbitrarily small), and finally there are m jobs of duration α each. The value density of first $(m+1)$ jobs is 1 and of last m jobs is Δ. F only accepts the first $(m+1)$ jobs. But, it is optimal to select

the first job and the last m jobs. Thus,

$$\frac{OPT(L_2)}{v(L_2)} = \frac{1 - m\alpha + m\alpha\Delta}{1 - m\alpha + m\left(\frac{(m-1)\alpha}{m} + \epsilon\right)} = \frac{nm\alpha(\Delta - 1) + n}{(n-1)\left(1 - \frac{1-\alpha}{m}\right) + 1 - \alpha + m\epsilon}. \tag{16}$$

Because $\epsilon > 0$ was arbitrary, Equation 16 implies that

$$r \geq \frac{nm\alpha(\Delta - 1) + n}{(n-1)\left(1 - \frac{1-\alpha}{m}\right) + 1 - \alpha}. \tag{17}$$

Case B: $2 \leq n \leq m$.

Consider an instance L_3 such that the first n jobs in it are of duration $(1 - m\alpha)$ (ignore these if $1 - m\alpha = 0$), next m jobs of duration $\dfrac{1 - \frac{1-\alpha}{m} - n(1 - m\alpha)}{m}$ each, next $m(n-2)$ jobs are of duration $\dfrac{1 - \frac{1-\alpha}{m}}{m}$ each, next m jobs are of duration $((1-\alpha)/m + \epsilon)$ each ($\epsilon > 0$ and arbitrarily small), and finally there are mn jobs of duration α each. The value density of first $n(m+1)$ jobs is 1 and of last mn jobs is Δ. Algorithm F assigns the first $n + m$ jobs to the first knapsack, the next $m(n-2)$ jobs m each to the next $(n-2)$ knapsacks, the next m jobs to the last knapsack, and declines the remaining jobs. But, it is optimal to select first n and last mn jobs. Thus,

$$\begin{aligned}\frac{OPT(L_3)}{v(L_3)} &= \frac{n(1 - m\alpha) + mn\alpha\Delta}{n(1 - m\alpha) + m\left(\frac{1 - \frac{1-\alpha}{m} - n(1-m\alpha)}{m}\right) + m(n-2)\left(\frac{1 - \frac{1-\alpha}{m}}{m}\right) + m\left(\frac{1-\alpha}{m} + \epsilon\right)} \\ &= \frac{nm\alpha(\Delta - 1) + n}{(n-1)\left(1 - \frac{1-\alpha}{m}\right) + 1 - \alpha + m\epsilon}. \end{aligned} \tag{18}$$

Because $\epsilon > 0$ was arbitrary, Equation 18 implies that

$$r \geq \frac{nm\alpha(\Delta - 1) + n}{(n-1)\left(1 - \frac{1-\alpha}{m}\right) + 1 - \alpha}. \tag{19}$$

Case C: $n > m$.

It can be shown that $n > m$ implies that $\frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha} \leq \frac{m+1}{m}\Delta$. We have already shown in Equation 15 that $r \geq \frac{m+1}{m}\Delta$. Therefore,

$$r \geq \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}. \tag{20}$$

Equations 14, 15, 17, 19, and 20 imply that

$$r = \max\left(\frac{m+1}{m}\Delta, \frac{nm\alpha(\Delta-1)+n}{(n-1)\left(1-\frac{1-\alpha}{m}\right)+1-\alpha}\right).$$

Hence, proved. □

D Proof of Statement 3 in Theorem 2.3

Proof. Let A be an arbitrary Any-Fit algorithm. Given n, α and Δ, let us denote $r_F(n,\alpha,\Delta)$ by r_F, $r_A(n,\alpha,\Delta)$ by r_A and $r_N(n,\alpha,\Delta)$ by r_N. Let L_1 be an arbitrary instance. We use Algorithm F to make assign/decline decisions for jobs in L_1 and construct a new instance L_2 observing the job assignment done by Algorithm F for L_1. In L_2, we first put those jobs that were assigned to knapsack 1 by Algorithm F, in any arbitrary order. Then those that are assigned to knapsack 2, in any arbitrary order. We do this for all the remaining knapsacks. Finally, we put the declined jobs, again in any arbitrary order. Note that $OPT(L_1) = OPT(L_2)$. If we apply A to L_2, A will assign and decline the jobs the same way as Algorithm F would for L_2. Also, the jobs accepted by Algorithm F for L_2 are same as those accepted by Algorithm F for L_1. Thus, $v_A(L_2) = v_F(L_2) = v_F(L_1)$. Then $\frac{OPT(L_1)}{v_F(L_1)} = \frac{OPT(L_2)}{v_A(L_2)} \leq r_A$. Therefore, $r_F = \sup_{L_1 \in \mathcal{L}(\alpha,\Delta)} \frac{OPT(L_1)}{v_F(L_1)} \leq r_A$.

Let L_3 be an arbitrary instance. We use Algorithm A to make assign/decline decisions for jobs in L_3 and as we do it, we construct a new instance L_4 by observing the decisions A takes. If A assigns a job to the largest-indexed knapsack among the ones which have at least

one job or assigns it to an empty knapsack, we add the job to L_4. When A declines a job for the first time, we add it to L_4, and then, in any arbitrary order, add all the remaining jobs, including those which were not added previously, to L_4. The sequence of jobs in L_4 is same as the order in which we add them to L_4. Note that $OPT(L_3) = OPT(L_4)$. But if we apply N to L_4, it will accept only those job which we added to L_4 before the first job in L_3 was declined by A. Therefore, $v_N(L_4) \leq v_A(L_3)$. Thus, $\frac{OPT(L_3)}{v_A(L_3)} \leq \frac{OPT(L_4)}{v_N(L_4)} \leq r_N$. Therefore, $r_A = \sup_{L_3 \in \mathcal{L}(\alpha,\Delta)} \frac{OPT(L_3)}{v_A(L_3)} \leq r_N$.

Hence, proved. □

E Proof of Theorem 2.4

Proof. Given n, α and Δ, let us denote $r_S(n, \alpha, \Delta)$ by r and v_S by v. As discussed in the sketch of the proof, we consider only those instances for which Algorithm S declines at least one job. Let L be an arbitrary instance from which Algorithm S declines at least one job and let $counter^*$ be the value of *counter* after all the jobs in L have been either accepted or declined. Recall from the sketch that if Algorithm S declines a job, it can be in two ways. First is when Algorithm S declines a job because no knapsack has enough free space for the job. Second is when although there is at least one knapsack with enough free capacity for the incoming job, the duration of the job belongs to $(1/(m+1), \theta_{counter})$. Therefore, after all the jobs in L have been either placed in knapsacks or declined by Algorithm S, one of the two cases can happen:

Case I: Algorithm S declined no job because of not having a knapsack with enough free capacity.

By the way it works, Algorithm S did not decline any job with duration $\leq \frac{1}{m+1}$. Let x denote the cumulative duration of jobs with duration $\leq \frac{1}{m+1}$ in L, let y denote the

cumulative duration of jobs with duration $> \frac{1}{m+1}$ in an optimal sub-sequence of L that the decision maker would accept if L was known in advance, and let z denote the cumulative duration of jobs with duration $> \frac{1}{m+1}$ in L that were accepted by Algorithm S. We have $OPT(L) \leq (x+y)\Delta$ and $v(L) \geq x+z$. Further, let $z = \sum_{j=1}^{counter^*-1} \theta_j + c$, where c is some non-negative constant. Then, $y \leq mn\theta_{counter^*} + c$ because no job with duration strictly greater than $\theta_{counter^*}$ could have been declined. Also, $counter^* \geq (I_S+1)$ because at least one job was declined and the interval $(1/(m+1), \theta_i)$ is empty for any $i \in [I_S]$. We have,

$$\frac{mn\theta_{counter^*}}{\sum_{j=1}^{counter^*-1} \theta_j} = \frac{mn\frac{w(n,\alpha)I_S}{mn(m+1)}\left(1+\frac{w(n,\alpha)}{mn}\right)^{counter^*-I_S-1}}{\frac{I_S}{m+1}\left(1+\frac{w(n,\alpha)}{mn}\right)^{counter^*-I_S-1}} = w(n,\alpha).$$

Thus,

$$\frac{OPT(L)}{v(L)} \leq \frac{(x+y)\Delta}{x+z} \leq \frac{x+mn\theta_{counter^*}+c}{x+\sum_{j=1}^{counter^*-1}\theta_j + c}\Delta \leq \frac{mn\theta_{counter^*}}{\sum_{j=1}^{counter^*-1}\theta_j}\Delta = w(n,\alpha)\Delta. \tag{21}$$

Case II: Algorithm S declined at least one job because of not having a knapsack with enough free capacity.

Let j be the smallest integer in the set $\{0,1,\ldots,n\}$ such that each of knapsacks $j+1, j+2, \ldots, n$ contains m jobs each with duration $> 1/(m+1)$. Because of the way in which Algorithm S selects and places jobs with duration $> 1/(m+1)$, the total duration of jobs in the last $n-j$ knapsacks must be $\geq \Theta_{n-j}$. The total duration of jobs in knapsack j is $> 1-\alpha$, because otherwise Algorithm S could not have declined any job because of lack of free space. Now, knapsack j must have a job with duration $\leq 1/(m+1)$ because otherwise, to have filled capacity $> 1-\alpha$, it must have m jobs each of duration $> 1/(m+1)$, and this is not possible because of the way in which we chose j.

Knapsacks $1, 2, \ldots, j-1$ must only have jobs of duration $\leq 1/(m+1)$. Otherwise, consider job k of duration $> 1/(m+1)$ that went to one of knapsack $1, 2, \ldots, j-1$. It must

have not been assigned to knapsack j because knapsack j did not have enough free space. In that case, since knapsack j had $< m$ jobs each with duration $> 1/(m+1)$ at any stage, it must also have had at least one job with duration $\leq 1/(m+1)$ to not have enough free space for job k when job k arrived. Because jobs with duration $\leq 1/(m+1)$ are assigned to the smallest-indexed knapsack with enough free space, the free space in each of knapsack $1, 2, \ldots, j-1$ must have been $< 1/(m+1)$ when job k arrived. Therefore, job k could not have been assigned any of knapsack $1, 2, \ldots, j-1$, which is a contradiction. Therefore, knapsacks $1, 2, \ldots, j-1$ must only have jobs of duration $\leq 1/(m+1)$. Therefore, if we focus on the first $j-1$ knapsacks only, it is as if we used Algorithm F to decide jobs, the maximum possible job duration was $1/(m+1)$ and at least one job was declined. Therefore, using intermediate results (Equations 8, 10, and 12) from the proof of Statement 1 in Theorem 2.3 (Appendix C) and setting the maximum job duration to $1/(m+1)$, the total duration of accepted jobs in first $(j-1)$ knapsacks is $\geq \min\left((j-1)\frac{m+1}{m+2}, (j-2)\left(1-\frac{m}{(m+1)^2}\right)+\frac{m}{m+1}\right)$ if $j > 1$ and 0 otherwise.

$v(L)$ is greater than or equal to the duration of jobs in the n knapsacks, which is greater than or equal to $u(j)\min\left((j-1)\frac{m+1}{m+2}, (j-2)\left(1-\frac{m}{(m+1)^2}\right)+\frac{m}{m+1}\right)+1-\alpha+\Theta_{n-j} = \gamma(j)$, where $u(j) = 1$ if $j > 1$ and 0 otherwise. Thus, $v(L) \geq \min_{j\in[n]} \gamma(j)$. Therefore,

$$\frac{OPT(L)}{v(L)} \leq n\Delta / \min_{j\in[n]} \gamma(j). \tag{22}$$

By Equations 21 and 22, $\frac{OPT(L)}{v(L)} \leq \max\left(w(n,\alpha), n/\min_{j\in[n]}\gamma(j)\right)\Delta$ holds for any arbitrary instance L from which Algorithm S declines at least one job. Therefore, $r \leq \max\left(w(n,\alpha), n/\min_{j\in[n]}\gamma(j)\right)\Delta$. Hence, proved. □

F Proof of Theorem 2.5

Proof. Let E_i denote the empty space in knapsack i, F_i the filled space in knapsack i, $E_{(i,j)}$ the empty space in segment (i,j), and $F_{(i,j)}$ the filled space in segment (i,j) after all the jobs from an instance have been either placed in knapsacks or declined (Figure F.1). We will call a segment to be partially-filled if some portion of at least one job lies strictly inside it.

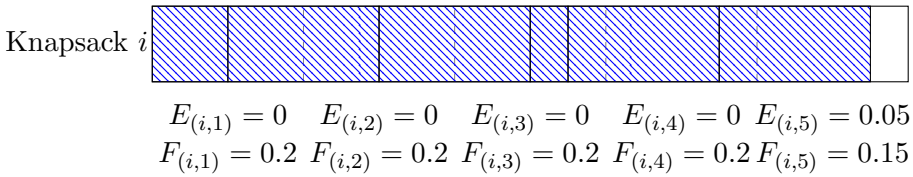

Figure F.1: Empty and filled spaces in segments of knapsack i after all the jobs in an arbitrary instance have been either placed in knapsacks or declined. Shaded rectangles represent jobs and dashed lines divide the knapsack into 5 segments.

Given n, α and Δ, let us denote $r_D(n, \alpha, \Delta)$ by r and v_D by v. Because of the reasons discussed in the sketch of the proof, we consider only those instances for which Algorithm D declines at least one job. Let L be an arbitrary instance from which Algorithm D declines at least one job. We will analyze the job assignments in the knapsacks after all the jobs in L have been either placed in knapsacks or declined. Note that $1/(m+1) < \alpha \leq 1/m$ from Equation 2.1.

As mentioned in the sketch of the proof, we begin by identifying the rightmost partially filled segment, (i^*, j^*), in the largest-indexed knapsack with at least one job assigned to it (Figure F.2). Because of the way Algorithm D works, this segment must belong to the set $\mathcal{I}$, defined as $\{(i,j) | i \in [n], j \in [m], (i-1)m + j \geq I_D\}$ in Definition 2.3.

To obtain a lower bound on $v(L)$, we find a lower bound on the total duration of accepted jobs in each knapsack, assume that the value-densities of portions of jobs that lie

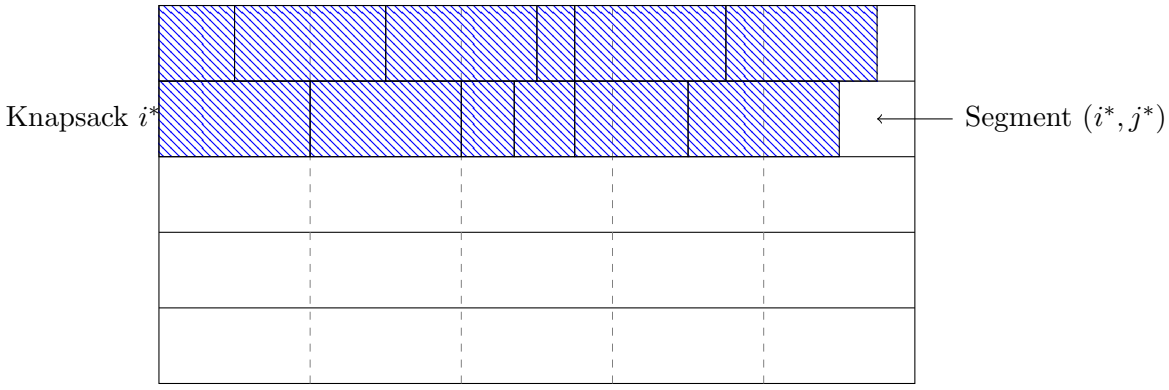

Figure F.2: Rightmost partially filled segment, (i^*, j^*), in the largest-indexed knapsack with at least one job assigned to it after all the jobs in an arbitrary instance have been either accepted or declined by Algorithm D.

in a particular segment of the knapsack are equal to the ϕ value of that segment, obtain the lower bound on total value of jobs assigned to each segment and sum this bound over all the mn segments. We will do this implicitly, without presenting the details. Now, assume that the lower bound we compute is $x > 0$, that is $v(L) \geq x$. Let the actual value of $v(L)$ be equal to $(x + c)$ for some non-negative c. Then, $OPT(L) \leq n\Phi(i^*, j^*) + c$ because no job with value density strictly greater than $\Phi(i^*, j^*)$ could have been declined. Since we must have $x \leq n\Phi(i^*, j^*)$, we have

$$\frac{OPT(L)}{v(L)} \leq \frac{n\Phi(i^*, j^*) + c}{x + c} \leq \frac{n\Phi(i^*, j^*)}{x}. \tag{23}$$

We will refer to the above argument as Equation 23 to use it repeatedly in the proof.

We begin with a proposition, the proof of which is omitted because it is similar to that for Proposition C.1 in the proof of Statement 1 in Theorem 2.3 (Appendix C).

Proposition F.1. *If we use Algorithm D for the OMKP, at most one partially-filled segment has free capacity* $\geq 1/(m + 1)$ *after all the jobs have been either placed in knapsacks or declined.*

The above proposition implies that only one of the following four cases can happen:

Case 1: There does not exist an partially-filled segment $(\widehat{i},\widehat{j})$ such that $1/(m+1) \leq E_{(\widehat{i},\widehat{j})}$.

$F_i > \frac{m-1}{m} + \left(\frac{1}{m} - \frac{1}{m+1}\right)$ for any knapsack $i \in [i^*-1]$. Further, $F_{i^*} > \frac{(j^*-1)}{m} + (\frac{1}{m} - \frac{1}{m+1})$. Finally, $F_i = 0$ for any knapsack $i \in [n] \setminus [i^*]$. Therefore,

$$v(L) \geq \sum_{i=1}^{i^*-1} \left[\sum_{j=1}^{m-1} \frac{\phi(i,j)}{m} + \frac{\phi(i,m)}{m(m+1)} \right] + \sum_{j=1}^{j^*-1} \frac{\phi(i^*,j)}{m} + \frac{\phi(i^*,j^*)}{m(m+1)},$$

and hence by Equation 23,

$$\frac{OPT(L)}{v(L)} \leq \frac{n\Phi(i^*,j^*)}{\sum_{i=1}^{i^*-1} \left[\sum_{j=1}^{m-1} \frac{\phi(i,j)}{m} + \frac{\phi(i,m)}{m(m+1)} \right] + \sum_{j=1}^{j^*-1} \frac{\phi(i^*,j)}{m} + \frac{\phi(i^*,j^*)}{m(m+1)}} = \tau(i^*,j^*). \quad (24)$$

Case 2: There exists a unique partially-filled segment $(\widehat{i},\widehat{j})$ such that $1/(m+1) \leq E_{(\widehat{i},\widehat{j})}$ and $\widehat{i} \neq i^*$.

This cannot happen for $i^* = 1$. So, consider $i^* \geq 2$. Choose an $i \in \{\widehat{i}+1, \widehat{i}+2, \ldots, i^*\}$ and let j be the last partially-filled segment in knapsack i. Note that $\widehat{j} = m$ because of the way Algorithm D works. Knapsack i has at least j jobs because each segment has a capacity of $\frac{1}{m}$, because largest possible job duration is $\alpha \leq \frac{1}{m}$, and because segments $(i,1),(i,2),\ldots,(i,j-1)$ have zero free space. Because we used Algorithm D to decide jobs, duration of each job in knapsack i is $> E_{(\widehat{i},\widehat{j})}$. Thus,

$$F_i > jE_{(\widehat{i},\widehat{j})} \implies F_{\widehat{i}} + F_i > F_{\widehat{i}} + jE_{(\widehat{i},\widehat{j})} = 1 + (j-1)E_{(\widehat{i},\widehat{j})} \geq 1 + \frac{j-1}{m+1} = \frac{m+j}{m+1}.$$

For the purpose deriving a lower bound on the total duration of accepted jobs, we hypothetically shift some potions of jobs from segment (i,j) to $(\widehat{i},\widehat{j})$ such that the total duration of these portions is equal to $F_i - \dfrac{j}{m} + 1/(m+1)$. Therefore, segment (i,j) now has free capacity $= 1/(m+1)$ and the total duration of jobs in knapsack $\widehat{i}$ becomes

$$F_{\widehat{i}} + F_i - \frac{j}{m} + \frac{1}{m+1} > \frac{m+j}{m+1} - \frac{j}{m} + \frac{1}{m+1} = \frac{m}{m+1}.$$

Further, since the portions of jobs that we shifted have value-densities greater than or equal to $\phi(\widehat{i},\widehat{j})$,

$$v(L) \geq \sum_{i=1}^{i^*-1}\left[\sum_{j=1}^{m-1}\frac{\phi(i,j)}{m}+\frac{\phi(i,m)}{m(m+1)}\right]+\sum_{j=1}^{j^*-1}\frac{\phi(i^*,j)}{m}+\frac{\phi(i^*,j^*)}{m(m+1)},$$

and hence by Equation 23,

$$\frac{OPT(L)}{v(L)} \leq \frac{n\Phi(i^*,j^*)}{\sum_{i=1}^{i^*-1}\left[\sum_{j=1}^{m-1}\frac{\phi(i,j)}{m}+\frac{\phi(i,m)}{m(m+1)}\right]+\sum_{j=1}^{j^*-1}\frac{\phi(i^*,j)}{m}+\frac{\phi(i^*,j^*)}{m(m+1)}} = \tau(i^*,j^*). \quad (25)$$

Case 3: There exists a unique partially-filled segment $(\widehat{i},\widehat{j})$ such that $1/(m+1) \leq E_{(\widehat{i},\widehat{j})}$, $\widehat{i}=i^*$ and $F_{(i^*,j^*)} > \frac{1}{m}-\alpha$.

Clearly, $\widehat{j}=j^*$. Knapsack i^* has at least j^* jobs because each segment has a capacity of $\frac{1}{m}$, because largest possible job duration is $\alpha \leq \frac{1}{m}$, and because segments $(i^*,1),(i^*,2),\ldots,(i^*,j^*-1)$ have zero free space. Further,

$$\frac{j^*}{m}-\alpha < F_{i^*} \leq \frac{j^*}{m}-\frac{1}{m+1} \implies \frac{1}{m}-\frac{\alpha}{j^*} < \frac{F_{i^*}}{j^*} \leq \frac{1}{m}-\frac{1}{(m+1)j^*},$$

which implies that the smallest job in knapsack i^* has duration $=\frac{1}{m}-\frac{\alpha}{j^*}+\epsilon$, for some ϵ such that $-\frac{1}{m}+\frac{\alpha}{j^*} < \epsilon \leq -\frac{1}{(m+1)j^*}+\frac{\alpha}{j^*}$. Because we used Algorithm D and this particular job did not go into any of the previous knapsacks $\implies E_i \leq \frac{1}{m}-\frac{\alpha}{j^*}+\epsilon$ for any knapsack $i \in [i^*-1]$. But if $\epsilon<0$, then $E_i \leq \frac{1}{m}-\frac{\alpha}{j^*}$ still holds. Therefore, $E_i \leq \frac{1}{m}-\frac{\alpha}{j^*}+\epsilon^+$ for some ϵ^+ such that $0 \leq \epsilon^+ \leq -\frac{1}{(m+1)j^*}+\frac{\alpha}{j^*}$. Thus, $F_i \geq 1-\frac{1}{m}+\frac{\alpha}{j^*}-\epsilon^+$ for any knapsack $i \in [i^*-1]$. Further, $F_{i^*} \geq \frac{j^*}{m}-\alpha+j^*\epsilon$ because knapsack i^* has at least j^* jobs, the smallest of which has duration $\frac{1}{m}-\frac{\alpha}{j^*}+\epsilon$. Because we are in case 3, $F_{i^*} > \frac{j^*}{m}-\alpha$. Thus, $F_{i^*} \geq \frac{j^*}{m}-\alpha+j^*\epsilon^+$. Further, because $v(L)$ must be greater than or equal to the duration of the jobs in the knapsacks,

$$v(L) \geq \sum_{i=1}^{i^*-1}\left[\sum_{j=1}^{m-1}\frac{\phi(i,j)}{m}+\phi(i,m)\left(\frac{\alpha}{j^*}-\epsilon^+\right)\right]+\sum_{j=1}^{j^*-1}\frac{\phi(i^*,j)}{m}+\phi(i^*,j^*)\left(\frac{1}{m}-\alpha+j^*\epsilon^+\right).$$

Because the above bound is linear in ϵ^+, and because $0 \leq \epsilon^+ \leq -\frac{1}{(m+1)j^*} + \frac{\alpha}{j^*}$, $v(L) \geq \mu(i^*, j^*)$, where

$$\mu(i,j) \doteq \min \left(\sum_{k=1}^{i-1} \left[\sum_{\ell=1}^{m-1} \frac{\phi(k,\ell)}{m} + \frac{\phi(k,m)}{m(m+1)} \right] + \sum_{\ell=1}^{j-1} \frac{\phi(i,\ell)}{m} + \frac{\phi(i,j)}{m(m+1)}, \right.$$
$$\left. \sum_{k=1}^{i-1} \left[\sum_{\ell=1}^{m-1} \frac{\phi(k,\ell)}{m} + \phi(k,m)\frac{\alpha}{j} \right] + \sum_{\ell=1}^{j-1} \frac{\phi(i,\ell)}{m} + \phi(i,j)\left(\frac{1}{m} - \alpha\right) \right).$$

Hence, by Equation 23,

$$\frac{OPT(L)}{v(L)} < \frac{n\Phi(i^*, j^*)}{\mu(i^*, j^*)} = \max\{\tau(i^*, j^*), \psi(i^*, j^*)\}. \tag{26}$$

Case 4: There exists a unique partially-filled segment $(\widehat{i}, \widehat{j})$ such that$1/(m+1) \leq E_{(\widehat{i},\widehat{j})}$, $\widehat{i} = i^*$ and $F_{(i^*,j^*)} \leq \frac{1}{m} - \alpha$.

Because of the way Algorithm D works and because at least one job is declined, $\widehat{j} = j^*$ and $m(i^*-1)+j^* \geq I_D+1$. Then using the same logic as Case 3 for deriving a lower bound on $v(L)$, we get $v(L) \geq \mu(i', j')$, where μ is as defined is the previous case, $(i', j') = (i^*-1, m)$ if $j = 1$ and $(i', j') = (i^*, j^*-1)$ otherwise. Now, if $v(L) = \mu(i', j')+c$ for some non-negative c, then $OPT(L) < n\Phi(i', j')+c$ because segment (i^*, j^*) could have accommodated at least one more job of any duration and thus no job with value density strictly greater than $\Phi(i', j')$ could have been declined. Therefore,

$$\frac{OPT(L)}{v(L)} < \frac{n\Phi(i', j')}{\mu(i', j')} = \max\left(\tau(i', j'), \psi(i', j')\right). \tag{27}$$

By Equations 24, 25, 26, and 27 from the above four cases, taking into account all the possible values of (i^*, j^*),

$$\frac{OPT(L)}{v(L)} \leq \max\left(\max_{(i,j)\in\mathcal{I}} \tau(i,j), \max_{(i,j)\in\mathcal{I}} \psi(i,j) \right).$$

Because L was an arbitrary instance from which Algorithm D declined at least one job,

$$r \leq \max\left(\max_{(i,j)\in\mathcal{I}} \tau(i,j), \max_{(i,j)\in\mathcal{I}} \psi(i,j) \right).$$

Hence, proved. □

G Proof of Theorem 2.6

Proof. We begin by defining some notation, proceed on to find a lower bound on the the total duration of accepted jobs when the sampled threshold is equal to x, and finally use this lower bound to prove the desired result.

Let L denote an arbitrary instance. Let B denote an optimal subset of jobs from L that could have been selected if L was known upfront. Let $d(x) \doteq$ total duration of jobs with value density equal to x in B, $L_x \doteq$ subsequence obtained by removing all jobs with value density $< x$ from L and setting the value density of remaining jobs equal to 1, $g(L_x) \doteq$ the total duration of jobs accepted by Algorithm A when deciding L_x, $h(x) \doteq xd(x)/OPT(L)$, and $C \doteq \{\text{distinct value-densities of jobs in } L\} \cup \{1\}$. Define $x_1, x_2, \ldots, x_k$ such that $C = \{x_1, x_2, \ldots, x_k\}$ and $x_1 \doteq 1 < x_2 < \ldots < x_k \leq \Delta$. Let $p_1 \doteq G(x_1)$, $p_i \doteq G(x_i) - G(x_{i-1})$ for $i = 2, \ldots, k$, and $\beta \doteq r_A(n, \alpha, 1)$.

For any $i \in [k]$, by the definition of r_A, $OPT(L_{x_i})/g(L_{x_i}) \leq \beta$. By definition of B, there exists a valid assignment of the jobs in B to the n knapsacks. So, a valid assignment of jobs with value density greater than or equal to x_i in B must also exist. Therefore, $OPT(L_{x_i})$ must be greater than or equal to the sum of durations of those jobs in B that have value density greater than or equal to x_i, i.e. $OPT(L_{x_i}) \geq \sum_{j=i}^{k} d(x_j)$. Thus, $\sum_{j=i}^{k} d(x_j)/g(L_{x_i}) \leq \beta \implies g(L_{x_i}) \geq \sum_{j=i}^{k} d(x_j)/\beta$.

Because $v_{R:A}(L) = \sum_{i=1}^{k} p_i x_i g(L_{x_i})$,

$$\frac{v_{R:A}(L)}{OPT(L)} \geq \frac{\sum_{i=1}^{k} p_i x_i \sum_{j=i}^{k} d(x_j)}{\beta OPT(L)} \implies \frac{OPT(L)}{v_{R:A}(L)} \leq \frac{\beta OPT(L)}{\sum_{i=1}^{k} d(x_i) \sum_{j=1}^{i} p_j x_j} = \frac{\beta OPT(L)}{p_1 \sum_{i=1}^{k} d(x_i) + \sum_{i=2}^{k} d(x_i) \sum_{j=2}^{i} p_j x_j}.$$

Now, $\sum_{j=2}^{i} p_j x_j = \sum_{j=2}^{i} \left(G(x_j) - G(x_{j-1})\right) x_j$ for any $i = 2, 3, \ldots, k$. Because $G(x)$ is increasing in x,

$\sum_{j=2}^{i} (G(x_j) - G(x_{j-1}))\, x_j \geq \int_1^{x_i} \frac{dG(x)}{dx} x\, dx = \int_1^{x_i} \frac{1}{x(1+\ln\Delta)} x\, dx = \frac{x_i - 1}{(1+\ln\Delta)}$. Therefore,

$$\frac{OPT(L)}{v_{R:A}(L)} \leq \frac{(1+\ln\Delta)\beta OPT(L)}{\sum_{i=1}^{k} d(x_i) + \sum_{i=2}^{k} d(x_i)(x_i - 1)} = \frac{(1+\ln\Delta)\beta OPT(L)}{\sum_{i=1}^{k} x_i d(x_i)} = \frac{(1+\ln\Delta)\beta}{\sum_{i=1}^{k} h(x_i)} = (1+\ln\Delta)\beta,$$

where the last equality holds because $\sum_{i=1}^{k} h(x_i) = 1$. Because L was arbitrary, $r_{R:A}(n, \alpha, \Delta) \leq \beta(1+\ln\Delta)$. □

H Proof of Statement 1 in Theorem 2.7

Proof. We construct a sequence of unit-length jobs, L. Let k be an arbitrary non-negative integer. Then L consists of $k+1$ subsequences, where the j-th subsequence consists of n jobs of value a_j and where $a_0 \doteq 1 < a_1 < a_2 < \ldots < a_k \doteq \Delta$, and $a_j \doteq a_0 \left(\frac{a_k}{a_0}\right)^{j/k}$. In particular, we can write the sequence L as follows:

$$\underbrace{a_0, a_0, \cdots, a_0}_{n \text{ jobs}},\ \underbrace{a_1, a_1, \cdots, a_1}_{n \text{ jobs}},\ \cdots, \underbrace{a_k, a_k, \cdots, a_k}_{n \text{ jobs}}. \tag{28}$$

Let A be an arbitrary randomized algorithm for the ORMP. Given A, we can calculate x_j, the expected number of jobs with value a_j accepted by A from sequence L. Next, we apply A to an instance that consists of only the first j subsequences. The n knapsacks can accommodate at most n jobs from this instance because the size of each job is equal to 1. Therefore, the offline optimal solution is to accept the n jobs with values equal to a_j from the instance. But, the expected value of jobs accepted by A is $\sum_{i=0}^{j} x_i a_i$. By the definition of CR, $r_A^{RM}(n, \Delta) \geq \zeta_j^A \doteq \frac{n a_j}{\sum_{i=0}^{j} x_i a_i}$, and also $r_A^{RM}(n, \Delta) \geq \max_{j=0,1,\cdots,k} \zeta_j^A$. Therefore, $\min_{A \in \mathcal{A}} \max_{j=0,1,\cdots,k} \zeta_j^A$, where $\mathcal{A}$ denotes the set of all randomized algorithms for the ORMP, is a valid lower bound on the CR of any randomized algorithm for the ORMP. We can translate the search for the minimum value of $\max_{j=0,1,\cdots,k} \zeta_j^A$ over $\mathcal{A}$ to a search over all

the possible values of x_j's that randomized algorithms induce. Equivalently, we determine $\max_{x_j's} \min_{j=0,1,\cdots,k} \frac{1}{\zeta_j^A}$ using LP formulation (LP2).

$$\begin{aligned} &\max\ z && \text{(LP2)}\\ &\text{subject to: } z \le \frac{\sum_{i=0}^{j} x_i a_i}{n a_j} \ \text{ for every } j = 0, 1, \cdots, k\\ &\sum_{i=0}^{k} x_i \le n \qquad \text{and}\\ &x_i \ge 0,\ i = 0, 1, \cdots, k \end{aligned}$$

The constraint $\sum_{i=0}^{k} x_i \le n$ follows from the fact that the n knapsacks can accommodate a maximum of n jobs from sequence 28. It can be shown that given $a_0, \ldots, a_k$, there exists an optimal solution $(x_0^*, x_1^*, \ldots, x_k^*, z^*)$ to the above LP such that

$$z^* = \frac{\sum_{i=0}^{0} x_i^* a_i}{n a_0} = \frac{\sum_{i=0}^{1} x_i^* a_i}{n a_1} = \ldots = \frac{\sum_{i=0}^{k} x_i^* a_i}{n a_k} \text{ and} \tag{29}$$

$$\sum_{i=0}^{k} x_i^* = n. \tag{30}$$

$$\text{Define } \zeta_j^* \doteq \frac{n a_j}{\sum_{i=0}^{j} x_i^* a_i},\ j = 0, 1, \ldots, k. \tag{31}$$

Because ζ_j^*'s are identical by Equation 29 for all j, we drop subscript j, i.e. $\zeta^* \doteq \zeta_j^*$. We know that for any a, b, c and d such that $b, d \neq 0$ and $b \neq d$, $a/b = c/d \implies a/b = c/d = (a-c)/(b-d)$. Therefore, using Equation 31, we obtain

$$\zeta^* = \frac{n a_j - n a_{j-1}}{\sum_{i=0}^{j} x_i^* a_i - \sum_{i=0}^{j-1} x_i^* a_i} = n \frac{a_j - a_{j-1}}{a_j x_j^*}. \tag{32}$$

From Equation 31, $x_0^* = \frac{n}{\zeta^*}$, and from Equation 32, $x_j^* = n \frac{a_j - a_{j-1}}{a_j \zeta^*}$ for all $j = 1, \ldots, k$.

Plugging these values in Equation 30, $\sum_{i=0}^{k} x_i^* = n$, gives us

$$\begin{aligned}\zeta^* &= 1 + \sum_{j=1}^{k} \left(1 - \frac{a_{j-1}}{a_j}\right) = k + 1 - \sum_{j=1}^{k} \frac{a_{j-1}}{a_j} \\ &= k + 1 - k\left(\frac{a_0}{a_k}\right)^{1/k} = 1 + k\left(1 - \left(\frac{1}{\Delta}\right)^{1/k}\right). \end{aligned} \tag{33}$$

As discussed previously, ζ^*, being the reciprocal of the optimal objective value to LP2, gives us a valid lower bound on the CR of any randomized algorithm for given n, and Δ. We note that this lower bound is increasing in k and that we may choose k arbitrarily. Therefore, we let $k \to \infty$ and still get a lower bound that does not depend on k. From Equation 33,

$$\zeta^* = 1 + k\left(1 - \left(\frac{1}{\Delta}\right)^{1/k}\right)$$

$$\implies \lim_{k\to\infty} \zeta^* = 1 + \ln \Delta. \text{ (Doing some algebra and using L'Hospital's Rule)}$$

□

I Proof of Statement 3 in Theorem 2.7

Proof. When $\Delta = 1$, any fair algorithm achieves CR equal to 1, which also a trivial lower bound on CR of any algorithm. Because $\sup\{x | f(x, n) \leq \Delta\} = 1$ when $\Delta = 1$, Statement 3 holds in that case. We focus on $\Delta > 1$ for rest of the proof. The proof is by contradiction. Let A be a deterministic algorithm such that $r_A^{RM}(n, \Delta) < \underline{r}_{det}(n, \Delta)$. Given n and Δ, denote $r_A^{RM}(n, \Delta)$ by r and $\underline{r}_{det}^{RM}(n, \Delta)$ by $\underline{r}_{det}$. Consider an instance L consisting only of unit-sized jobs. The sequence of values of jobs in L is one that is obtained by appending the jobs in sequences $s_0, s_1, \ldots, s_{n-I+1}$, presented in Table I.1, one after the other. Note that $I \doteq \lceil n/r \rceil$, $\epsilon > 0$ is arbitrarily small, $O_1(\epsilon) \doteq (1+\epsilon)rI\epsilon/n$, and $O_k(\epsilon) \doteq (1+\epsilon)r/n\left(I\epsilon + \sum_{\ell=1}^{k-1} O_\ell(\epsilon)\right)$ for $k = 2, 3, \ldots, n - I$.

Table I.1: Sequences $s_1, s_2, \ldots, s_{n-I+2}$. Each job is denoted by its value.

Notation	Sequence of Jobs
s_0	$\underbrace{1+\epsilon, \ldots, 1+\epsilon}_{j_0}$
s_1	$\underbrace{\frac{rI}{n}+O_1(\epsilon), \ldots, \frac{rI}{n}+O_1(\epsilon)}_{j_1}$
s_2	$\underbrace{\frac{rI}{n}\left(1+\frac{r}{n}\right)+O_2(\epsilon), \ldots, \frac{rI}{n}\left(1+\frac{r}{n}\right)+O_2(\epsilon)}_{j_2}$
$\vdots$	$\vdots$
s_{n-I}	$\underbrace{\frac{rI}{n}\left(1+\frac{r}{n}\right)^{n-I-1}+O_{n-I}(\epsilon), \ldots, \frac{rI}{n}\left(1+\frac{r}{n}\right)^{n-I-1}+O_{n-I}(\epsilon)}_{j_{n-I}}$
s_{n-I+1}	$\underbrace{\Delta, \ldots, \Delta}_{n}$

Integers $j_0, j_1, \ldots, j_{n-I}$ are such that when A is applied to L, I jobs are accepted from subsequence s_0, one job each is accepted from subsequences $s_1, s_2, \ldots, s_{n-I}$, and no job is accepted from subsequence s_{n-I+1}. In particular, the sequence of values of accepted jobs can be written as:

$$\underbrace{1+\epsilon, 1+\epsilon, \ldots, 1+\epsilon}_{I}, \underbrace{\frac{rI}{n}+O_1(\epsilon), \ldots, \frac{rI}{n}\left(1+\frac{r}{n}\right)^{n-I-1}+O_{n-I}(\epsilon)}_{n-I}.$$

We next prove the following two claims to show that such an instance L indeed exists:

1. The value of any job in L is less than or equal to Δ.

2. There exist integers $j_0, j_1, \ldots, j_{n-I}$ such that when Algorithm A is applied to L, the sequence of values of accepted jobs is as claimed.

Proof of Claim 1: The value of each job in L is less than or equal to Δ because the following statements are true.

- Value of each job in the subsequence s_{n-I+1} is no more than Δ.

- The value of any job in any one of the subsequence $s_0, s_1, \ldots, s_{n-I}$ is less than $(rI/n)\,(1 + r/n)^{n-I}$, which by Definition 2.1 is equal to $f(r, n)$. Because $r < \underline{r}_{det}$, Lemma 2.1 and Equation 2.8 imply that $f(r, n) \leq \Delta$. Therefore, the value of any job in any of subsequence $s_0, s_1, \ldots, s_{n-I}$ is less than or equal to Δ. □

Proof of Claim 2: The proof is by construction. Note that the n knapsacks can accommodate at most n jobs from L.

Consider an instance L_0 that has n jobs of the same kind as in sequence s_1. If we apply Algorithm A to this sequence and it accepts less than I jobs, then $\frac{OPT(L_1)}{v_A(L_1)} \geq \frac{n(1+\epsilon)}{(I-1)(1+\epsilon)} = \frac{n}{r}\frac{r}{\lceil n/r \rceil - 1} > r$, where the last inequality holds because $\frac{n}{r} > \lceil \frac{n}{r} \rceil - 1$. But this is not possible because A is r-competitive. Therefore, Algorithm A must accept at least I jobs from the sequence. Let job i_0 be the Ith job that A accepted from the sequence. Then, j_0 is the position of job i_0 in L_0.

Consider an instance L_1 with first j_0 jobs of the same kind as in sequence s_1, and then n jobs of the same kind as in sequence s_2. The offline optimal solution for L_1 is to accept the last n jobs. If we apply Algorithm A to L_1, it must accept I jobs from the first j_0 jobs. If it accepts no other job from the next n jobs, then $\frac{OPT(L_1)}{v_A(L_1)} = \frac{n\,(rI/n + O_1(\epsilon))}{I\,(1+\epsilon)} = \frac{rI + nO_1(\epsilon)}{I + I\epsilon} = \frac{rI + (1+\epsilon)rI\epsilon}{I + I\epsilon} > \frac{rI + rI\epsilon}{I + I\epsilon} = r$. But, this is not possible because A is r-competitive. Therefore, Algorithm A must accept at least one job from the last n jobs in L_1. Let job i_1 be the $(j_0 + 1)$th job that Algorithm A accepted from L_1. Then, $j_1 =$ position of job i_1 in L_1 minus j_0.

Similarly, we can prove the existence of $j_2, j_3, \ldots, j_{n-I}$ using the arguments described above for j_0 and j_1. □

By definition, $v_A(L)$ is the total value of jobs that Algorithm A will accept when we

apply it to L. Therefore,

$$\implies v_A(L) = I\,(1+\epsilon) + \sum_{k=1}^{n-I}\left(\frac{rI}{n}\left(1+\frac{r}{n}\right)^{k-1} + O_k(\epsilon)\right)$$

$$\implies v_A(L) = I\,(1+\epsilon) + \frac{rI}{n}\frac{\left(1+\frac{r}{n}\right)^{n-I}-1}{(1+\frac{r}{n})-1} + \sum_{k=1}^{n-I} O_k(\epsilon)$$

$$\implies \lim_{\epsilon\to 0}\frac{n}{v_A(L)} = \frac{1}{\frac{I}{n}\left(1+\frac{r}{n}\right)^{n-I}}. \tag{34}$$

If we knew L upfront, it was optimal to accept only the last n jobs of value Δ each. Therefore, $OPT(L) = n\Delta$. Further, there exists a $\delta > 0$ such that $r < \underline{r}_{det} - \delta$ because $r < \underline{r}_{det}$. Since A is r-competitive, we have

$$\frac{OPT(L)}{v_A(L)} \le r \text{ for any } \epsilon > 0$$

$$\implies \lim_{\epsilon\to 0}\frac{n\Delta}{v_A(L)} \le r$$

$$\implies r\frac{I}{n}\left(1+\frac{r}{n}\right)^{n-I} \ge \Delta \qquad \text{(substituting } \lim_{\epsilon\to 0}\frac{n}{v_A(L)} \text{ from Equation 34)}$$

$$\implies r\frac{I}{n}\left(1+\frac{r}{n}\right)^{n-I} \ge (\underline{r}_{det}-\delta)\frac{I}{n}\left(1+\frac{(\underline{r}_{det}-\delta)}{n}\right)^{n-I}$$

(from Equation 2.8 and Lemma 2.1)

$$\implies f(r,n) \ge f(\underline{r}_{det}-\delta, n) \qquad \text{(by Definition 2.1)}$$

$$\implies r \ge \underline{r}_{det} - \delta \text{ (by Lemma 2.1),}$$

which is a contradiction. Hence, proved. □

J Proof of Statement 4 in Theorem 2.7

Proof. Given n and Δ, let us denote $r_D^{RM}(n,\Delta)$ by r and v_D by v. We consider only those instances for which Algorithm D declines at least one job because the relative performance is equal to 1 for other instances. Let L be an arbitrary instance from which Algorithm D

declines at least one job. We will analyze the job assignments in the knapsacks after all the jobs in L have been either placed in knapsacks or declined.

As mentioned in Section 2.6, we set $\alpha = 1$ to apply Algorithm D to the ORMP. Therefore, $m = 1$ and each knapsack consists of a single cell. Let knapsack i^* be the largest-indexed knapsack that is filled. Since at least one job was declined, $i^* \geq I_D$. Then,

$$v(L) \geq \sum_{i=1}^{i^*} \phi(i, 1).$$

Further, no job with value greater than $\Phi(i^*, 1)$ could have been declined. Therefore, $OPT(L) \leq n\Phi(i^*, 1)$. Hence,

$$\frac{OPT(L)}{v(L)} \leq \frac{n\Phi(i^*, 1)}{\sum_{i=1}^{i^*} \phi(i, 1)} = t(n, 1, \Delta) = \inf\{x \geq 1 | f(x, n) \geq \Delta\}.$$

Since instance L was arbitrary, we have $r \leq \inf\{x \geq 1 | f(x, n) \geq \Delta\}$. Because f is strictly increasing in its first argument by Lemma 2.1, $\inf\{x \geq 1 | f(x, n) \geq \Delta\} = \sup\{x | f(x, n) \leq \Delta\} = \underline{r}_{det}^{RM}(n, \Delta)$. Further, since $r \geq \underline{r}_{det}^{RM}(n, \Delta)$, $r = \underline{r}_{det}^{RM}(n, \Delta)$. Hence, proved. □

K Proof of Theorem 3.2

Proof. We begin by defining some notation. Let $\beta > 0$ an arbitrarily large real number, k and b arbitrary positive integers, $\alpha \doteq \Delta^{\frac{1}{k}}$, and $\epsilon > 0$ arbitrarily small. Let $q_i \doteq \left(1/\alpha^{i-1}\right) - \left(1/\alpha^{i}\right)$ for $i \in [k]$ and $q_{k+i} \doteq 1/\left(b\alpha^{k}\right)$ for $i \in [b]$. Let $s_1, s_2, \ldots, s_{k+b}$ and $s'_{k+1}, s'_{k+2}, \ldots, s'_{k+b}$ be sequences of jobs, each with n jobs, as shown in Table K.1. Each 2-tuple in Table K.1 represents a job, with the first value in the tuple being the start time of the job and the second value being the job's duration. The jobs in the sequences are ordered by their arrival times. Appending these sequences one after another, we define sequences (instances) $L_1, L_2, \ldots, L_{k+b}$ as described in Table K.2. The arrival times of the jobs are such that the

Table K.1: Sequences $s_1, s_2, \ldots, s_{k+b}$ and $s'_{k+1}, s'_{k+2}, \ldots, s'_{k+b}$.

Notation	Sequence of Jobs
s_1	$(\beta, D_{min}), (\beta, D_{min}), \ldots, (\beta, D_{min})$
s_2	$(\beta, \alpha D_{min}), (\beta, \alpha D_{min}), \ldots, (\beta, \alpha D_{min})$
$\vdots$	$\vdots$
s_k	$(\beta, \alpha^{k-1} D_{min}), (\beta, \alpha^{k-1} D_{min}), \ldots, (\beta, \alpha^{k-1} D_{min})$
s_{k+1}	$(\beta+\epsilon, \alpha^k D_{min}), (\beta+\epsilon, \alpha^k D_{min}), \ldots, (\beta+\epsilon, \alpha^k D_{min})$
s'_{k+1}	$(\beta+\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), (\beta+\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), \ldots, (\beta+\epsilon-\alpha^k D_{min}, \alpha^k D_{min})$
s_{k+2}	$(\beta+2\epsilon, \alpha^k D_{min}), (\beta+2\epsilon, \alpha^k D_{min}), \ldots, (\beta+2\epsilon, \alpha^k D_{min})$
s'_{k+2}	$(\beta+2\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), (\beta+2\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), \ldots, (\beta+2\epsilon-\alpha^k D_{min}, \alpha^k D_{min})$
$\vdots$	$\vdots$
s_{k+b}	$(\beta+b\epsilon, \alpha^k D_{min}), (\beta+b\epsilon, \alpha^k D_{min}), \ldots, (\beta+b\epsilon, \alpha^k D_{min})$
s'_{k+b}	$(\beta+b\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), (\beta+b\epsilon-\alpha^k D_{min}, \alpha^k D_{min}), \ldots, (\beta+b\epsilon-\alpha^k D_{min}, \alpha^k D_{min})$

jobs in each of $L_1, L_2, \ldots, L_{k+b}$ are ordered by their arrival times. A specific example of $L_1, L_2, \ldots, L_{k+b}$ for the case when $b = k = 3$ and $n = 1$ is depicted in Figure K.1. Define $\mathcal{K} \doteq \{L_1, L_2, \ldots, L_{k+b}\}$. The probability distribution $\mathcal{Q}$ over the set $\mathcal{K}$ is shown in the Table K.2.

Then, $OPT(L_i) = nD_{min}\alpha^{i-1}$ if $i \in [k]$ and $OPT(L_i) = 2nD_{min}\alpha^k$ if $i \in [k+b] \setminus [k]$. Therefore,

$$\begin{aligned} E(OPT(L)) &= \sum_{i=1}^{k+b} q_i OPT(L_i) = \sum_{i=1}^{k} q_k nD_{min}\alpha^{i-1} + \sum_{i=1}^{b} q_{k+i} 2nD_{min}\alpha^k \\ &= nD_{min}\left(\sum_{i=1}^{k}\left(\frac{1}{\alpha^{i-1}} - \frac{1}{\alpha^i}\right)\alpha^{i-1} + 2\right) = nD_{min}\left(2 + k\left(1 - \frac{1}{\alpha}\right)\right). \quad (35) \end{aligned}$$

We next find an upper bound on $E(v_A(L))$ for any $A \in \mathcal{A}_D$. Consider an arbitrary deterministic Algorithm $A \in \mathcal{A}_D$. When the instance is $L_{k+b}(= \{s_1, \ldots, s_k, \ldots, s_{k+b}, s'_{k+b}\})$, let n_i denote the number of jobs accepted by Algorithm A from the subsequence s_i, $i \in [k+b]$.

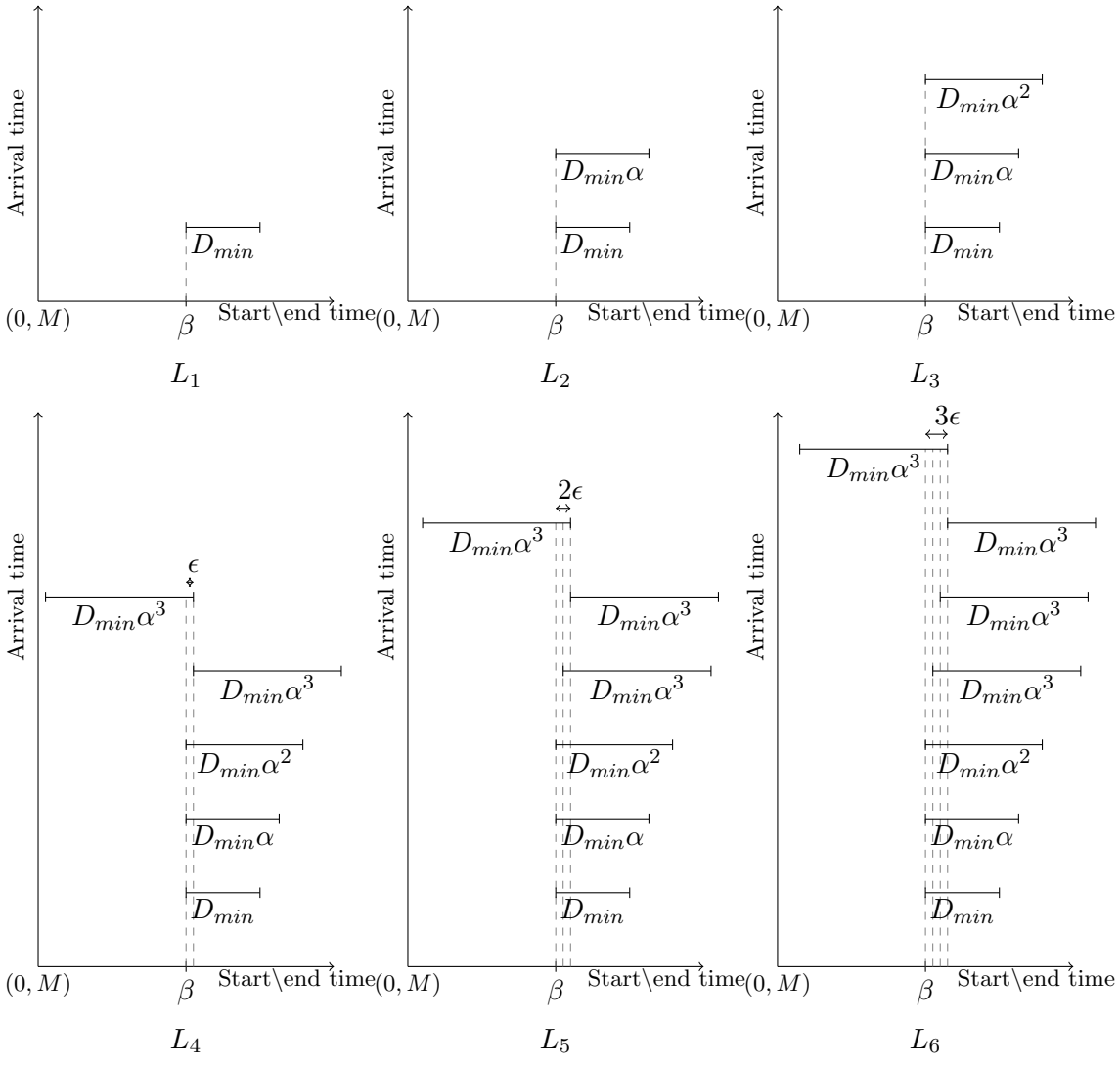

Figure K.1: $L_1, L_2, \ldots, L_{k+b}$ when $b = k = 3$ and $n = 1$. $M > 0$ is a very large real number.

Table K.2: Probability distribution $\mathcal{Q}$ over $\mathcal{K}$.

Instance	Probability
$L_1 = \{s_1\}$	q_1
$L_2 = \{s_1, s_2\}$	q_2
$\vdots$	$\vdots$
$L_k = \{s_1, s_2, \ldots, s_k\}$	q_k
$L_{k+1} = \{s_1, s_2, \ldots, s_k, s_{k+1}, s'_{k+1}\}$	q_{k+1}
$L_{k+2} = \{s_1, s_2, \ldots, s_k, s_{k+1}, s_{k+2}, s'_{k+2}\}$	q_{k+2}
$\vdots$	$\vdots$
$L_{k+b} = \{s_1, s_2, \ldots, s_k, s_{k+1}, s_{k+2}, s_{k+3}, \ldots, s_{k+b}, s'_{k+b}\}$	q_{k+b}

$\sum_{i=1}^{k+b} n_i \leq n$ holds because each job in the sequence $\{s_1, s_2, \ldots, s_k, s_{k+1}, s_{k+2}, s_{k+3}, \ldots, s_{k+b}\}$ overlaps with every other job in the same sequence and because only non-overlapping jobs can be assigned to the same server. Since A is a deterministic online algorithm, it will accept exactly n_1 jobs when faced with the instance $L_1(= \{s_1\})$. Similarly, when faced with $L_2(= \{s_1, s_2\})$, it will accept n_1 jobs from the subsequence s_1 and n_2 jobs from the subsequence s_2. And so on. Therefore, $v_A(L_i) = \sum_{j=1}^{i} n_j \alpha^{j-1} D_{min}$ for $i \in [k]$. Further, $v_A(L_i) \leq \sum_{j=1}^{k} n_j \alpha^{j-1} D_{min} + \sum_{j=k+1}^{i} n_j \alpha^k D_{min} + \left(n - \sum_{j=1}^{i-1} n_j\right) \alpha^k D_{min}$ for $i \in [k+b] \setminus [k]$ because $n - \sum_{j=1}^{i-1} n_j$ is an upper bound on the number of jobs that can be accepted from the last n jobs in instance L_i. $n - \sum_{j=1}^{i-1} n_j$ is a valid upper bound because every job in the subsequences $s_1, s_2, \ldots, s_{i-1}$ overlaps with each of the last n jobs in L_i when $i \in [k+b] \setminus [k]$ and because only non-overlapping jobs can be assigned to the same server. Since $E(v_A(L) = \sum_{i=1}^{k+b} q_i v_A(L_i)$,

we have

$$
\begin{aligned}
E(v_A(L) \leq & \sum_{i=1}^{k} q_i \sum_{j=1}^{i} n_j \alpha^{j-1} D_{min} + \\
& \sum_{i=k+1}^{k+b} q_i \left(\sum_{j=1}^{k} n_j \alpha^{j-1} D_{min} + \sum_{j=k+1}^{i} n_j \alpha^k D_{min} + \left(n - \sum_{j=1}^{i-1} n_j \right) \alpha^k D_{min} \right) \\
= & D_{min} \left(\sum_{i=1}^{k} n_i \alpha^{i-1} \left(\sum_{j=i}^{k} q_j \right) + \frac{1}{b\alpha^k} \left(\sum_{i=k+1}^{k+b} \sum_{j=1}^{k} n_j \alpha^{j-1} + \sum_{i=k+1}^{k+b} \left(\sum_{j=k+1}^{i} n_j + n - \sum_{j=1}^{i-1} n_j \right) \alpha^k \right) \right) \\
= & D_{min} \sum_{i=1}^{k} n_i \alpha^{i-1} \left(\frac{1}{\alpha^{i-1}} - \frac{1}{\alpha^k} \right) + \\
& \frac{D_{min}}{b\alpha^k} \left(b \sum_{j=1}^{k} n_j \alpha^{j-1} + \left(\sum_{i=k+1}^{k+b} n_i (k+b-i+1) + bn - b \sum_{i=1}^{k} n_i - \sum_{i=k+1}^{k+b} n_i (k+b-i) \right) \alpha^k \right) \\
= & D_{min} \sum_{i=1}^{k} n_i + D_{min} \left(\sum_{i=k+1}^{k+b} \frac{n_i}{b} + n - \sum_{i=1}^{k} n_i \right) \leq D_{min} \left(\frac{n}{b} + n \right) \left(\text{Using} \sum_{i=1}^{k+b} n_i \leq n \right) \\
= & n D_{min} \left(1 + 1/b \right). \qquad (36)
\end{aligned}
$$

Since Algorithm A was arbitrary, Inequality 36 holds for all algorithms in $\mathcal{A}_D$, and therefore, by Equation 35 and Theorem 3.1, for any randomized online Algorithm B for the ORP,

$$r_B(n, \mathcal{D}) \geq \inf_{A \in \mathcal{A}_D} \frac{E(OPT(L))}{E(v_A(L))} \geq \frac{n D_{min} \left(2 + k(1 - \frac{1}{\alpha}) \right)}{n D_{min} \left(1 + 1/b \right)} \qquad (37)$$

Since Equation 37 holds for any $k, b \in \mathbb{N}$,

$$r_B(n, \mathcal{D}) \geq \sup_{k,b \in \mathbb{N}} \frac{\left(2 + k(1 - \frac{1}{\alpha}) \right)}{(1 + 1/b)} = \sup_{k,b \in \mathbb{N}} \frac{\left(2 + k(1 - (1/\Delta)^{(1/k)}) \right)}{(1 + 1/b)}. \qquad (38)$$

Because the RHS in Equation 38 is increasing in both k and b,

$$r_B(n, \mathcal{D}) \geq \lim_{k,b \to \infty} \frac{\left(2 + k(1 - (1/\Delta)^{(1/k)}) \right)}{(1 + 1/b)} = 2 + \ln \Delta. \text{ (L'Hospital's Rule)}$$

Since Algorithm B was arbitrary, the result is proved. □

L Proof of Lemma 3.1

Proof. Let $L \in \mathcal{L}(n, \Delta)$ be an arbitrary instance of the ORP. By definition, $OPT(L)/v_A(L) = \sum_{c \in C_L} d_c / \sum_{b \in B_L} d_b$. By Definition 3.1, $\sum_{c \in C_L} w_L(b, c) \leq d_b$ for any job $b \in B_L$. Therefore,

$$\frac{OPT(L)}{v_A(L)} = \frac{\sum_{c \in C_L} d_c}{\sum_{b \in B_L} d_b} \leq \frac{\sum_{c \in C_L} d_c}{\sum_{b \in B_L} \sum_{c \in C_L} w_L(b, c)} = \frac{\sum_{c \in C_L} d_c}{\sum_{c \in C_L} \left(\sum_{b \in B_L} w_L(b, c) \right)} \leq \max_{c \in C_L} \left\{ \frac{d_c}{\sum_{b \in B_L} w_L(b, c)} \right\} \leq a.$$

Since L was arbitrary, $\sup_{L \in \mathcal{L}(n,\Delta)} (OPT(L)/v_A(L)) \leq a \implies r_A(n, \Delta) \leq a$. □

M Proof of Theorem 3.3

Before proceeding with the proof for Theorem 3.3, we present Lemma M.1 that we will use to substantiate some intermediate claims in the theorem's proof.

Lemma M.1. *Given an instance L, define B_L be the set of jobs accepted by Algorithm A and C_L be an offline optimal set. Then, following statements hold true:*

1. *If $n = 1$, any job $b \in B_L \cap C_L$ does not overlap with any other job in C_L except itself. Also, any job $c \in C_L \cap B_L$ does not overlap with any other job in B_L except itself.*
2. *If $D_{max} = D_{min}$, then any job $b \in B_L$ overlaps with at most $2n$ jobs in the set C_L.*
3. *If $D_{max} > D_{min}$, then for any job $b \in B_L$, the jobs in C_L that overlap with job b have total duration strictly less than $n\,(2D_{max} + d_b)$.*

Proof of Lemma M.1. Statement 1. Follows directly from the fact that set B_L contains only non-overlapping jobs and that the same holds for set C_L.

Statement 2. Set C_L contains jobs that would have been selected if we knew the instance L upfront, and therefore, it can be partitioned into n subsets such that each subset consists

only of non-overlapping jobs. Now, any job in B_L can overlap with at most 2 jobs in any one of these subsets. Therefore, any job $b \in B_L$ overlaps with at most $2n$ jobs in the set C_L.

Statement 3. Set C_L contains jobs that would have been selected if we knew the instance L upfront, and therefore, it can be partitioned into n subsets such that each subset consists only of non-overlapping jobs. Now, for any job $b \in B_L$, the total duration of jobs that overlap with job b in any one of the subsets must be strictly less than $2D_{max} + d_b$. Therefore, the total duration of jobs in C_L that overlap with a given job $b \in B_L$ is strictly less than $n\,(2D_{max} + d_b)$. □

Proof of Theorem 3.3. We begin by noting that although other well-known arguments may be used to prove the result for some specific values of n and Δ, but we prove all four statements using the common framework of allocation schemes and Lemma 3.1. Given an instance L, define B_L to be the set of jobs accepted by Algorithm A and C_L to be the offline optimal set. Then, the individual proofs of the four statements are as follows.

Statement 1. Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ D_{min}/2 & \text{if } b \text{ and } c \text{ overlap, } b \in B_L \setminus C_L, \text{ and } c \in C_L \setminus B_L, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 1.1: w_L is an allocation scheme from the set B_L to the set C_L.

Proof of Claim 1.1. If job $b \in B_L \cap C_L$, then it does not overlap with any other job in C_L except itself (Lemma M.1), and therefore, $\sum_{c \in C_L} w_L(b,c) = w_L(b,b) = D_{min} \leq d_b$. Otherwise if job $b \in B_L \setminus C_L$, it cannot overlap with more than 2 jobs in $C_L \setminus B_L$ (Lemma M.1), and therefore, $\sum_{c \in C_L} w_L(b,c) \leq 2\,(D_{min}/2) = D_{min} \leq d_b$. Therefore, by Definition 3.1, w_L is a

valid allocation scheme. □

Claim 1.2: $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq 2$ for any job c in C_L.

Proof of Claim 1.2. If job $c \in C_L \cap B_L$, then it does not overlap with any other job in B_L except itself (Lemma M.1), and therefore, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) = w_L(c,c)/d_c = D_{min}/D_{min} \leq 2$. Otherwise if job $c \in C_L \setminus B_L$, it must overlap with at least one job in $B_L \setminus C_L$ because Algorithm A could not have declined job c otherwise. Therefore, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq d_c / \left(D_{min}/2 \right) = D_{min} / \left(D_{min}/2 \right) = 2$. □

Since instance L was arbitrary, by Lemma 3.1 and Theorem 3.2, the result is proved. □

Statement 2. Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ \dfrac{d_c}{2\Delta + 1} & \text{if } b \text{ and } c \text{ overlap, } b \in B_L \setminus C_L, \text{ and } c \in C_L \setminus B_L, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 2.1: w_L is an allocation scheme from the set B_L to the set C_L.

Proof of Claim 2.1. If job $b \in B_L \cap C_L$, then it does not overlap with any other job in C_L except itself (Lemma M.1), and therefore, $\sum_{c \in C_L} w_L(b,c) = w_L(b,b) = D_{min} \leq d_b$. Otherwise if job $b \in B_L \setminus C_L$, the total duration of jobs in C_L that can overlap with it is strictly less than $2D_{max} + d_b$ (Lemma M.1), and therefore, $\sum_{c \in C_L} w_L(b,c) < \dfrac{2D_{max} + d_b}{2\Delta + 1} = \dfrac{2D_{max} + d_b}{2D_{max} + D_{min}} D_{min} \leq \dfrac{d_b}{D_{min}} D_{min} = d_b$. Therefore, by Definition 3.1, w_L is a valid allocation scheme. □

Claim 2.2: $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq 2\Delta + 1$ for any job c in C_L.

Proof of Claim 2.2. If job $c \in C_L \cap B_L$, then it does not overlap with any other job in B_L except itself (Lemma M.1), and therefore, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) = d_c / w_L(c,c) =$

$D_{min}/D_{min} \leq 2\Delta + 1$. Otherwise if job $c \in C_L \setminus B_L$, it must overlap with at least one job in $B_L \setminus C_L$ because Algorithm A could not have declined job c otherwise. Therefore, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq d_c / \left(\frac{d_c}{2\Delta+1} \right) = 2\Delta + 1$. □

Since instance L was arbitrary, by Lemma 3.1, $r_A \leq 2\Delta + 1$.

We will now prove a matching lower bound. Let $\beta > 0$ be arbitrarily large and $\epsilon > 0$ be arbitrarily small. Consider instance L in which the jobs arrive in following sequence: $(\beta, D_{min} + 2\epsilon), (\beta + \epsilon - D_{max}, D_{max}), (\beta + \epsilon, D_{min}), (\beta + \epsilon + D_{min}, D_{max})$. Each 2-tuple represents a job, with first value being the start time and the second being the duration. Then, $OPT(L)/v_A(L) = (D_{min}+2D_{max})/(D_{min}+2\epsilon)$. Therefore, $(D_{min}+2D_{max})/(D_{min}+2\epsilon) \leq r_A$. Since ϵ was arbitrary, this inequality holds for every ϵ, letting $\epsilon \to 0$ gives us the matching lower bound, $2\Delta + 1 \leq r_A$. □

<u>*Statement 3.*</u> Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min}/3 & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ D_{min}/(3n) & \text{if } b \text{ and } c \text{ overlap, and } b \not\equiv c, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 3.1: w_L is an allocation scheme from the set B_L to the set C_L.

Proof of Claim 3.1. For any job b in B_L, at most $2n$ jobs from $C_L \setminus \{b\}$ can overlap with it (Lemma M.1), and therefore, $\sum_{c \in C_L \setminus \{b\}} w_L(b,c) \leq 2nD_{min}/3n = 2D_{min}/3$. Further, if $b \in B_L \cap C_L$, $w_L(b,b) = D_{min}/3$. Therefore, for any job $b \in B_L$, $\sum_{c \in C_L} w_L(b,c) \leq 2D_{min}/3 + D_{min}/3 = D_{min} = d_b$. Hence, by Definition 3.1, w_L is an allocation scheme. □

Claim 3.2: $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq 3$ for any job c in C_L.

Proof of Claim 3.2. If job $c \in C_L \cap B_L$, then $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq d_c / w_L(c,c) = 3$. Otherwise if job $c \in C_L \setminus B_L$, it must overlap with at least n jobs in B_L because Algorithm A could

not have declined job c otherwise. Therefore, $d_c/\left(\sum_{b\in B_L} w_L(b,c)\right) \leq D_{min}/\left(nD_{min}/(3n)\right) = 3$. □

Since instance L was arbitrary, by Lemma 3.1, the result is proved. □

Statement 4. Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} \dfrac{d_c}{2\Delta+2} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ \dfrac{d_c}{n(2\Delta+2)} & \text{if } b \text{ and } c \text{ overlap, and } b \not\equiv c, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 4.1: w_L is an allocation scheme from the set B_L to the set C_L.

Proof of Claim 4.1. For any job b in B_L, jobs from $C_L \setminus \{b\}$ that overlap with it have total duration strictly less than $n(2D_{max} + d_i)$ (Lemma M.1), and therefore,

$$\begin{aligned} \sum_{c\in C_L\setminus\{b\}} w_L(b,c) < \frac{n(2D_{max}+d_b)}{n(2\Delta+2)} &= \frac{2\Delta+1}{2\Delta+2}\frac{2D_{max}+d_b}{2\Delta+1} = \frac{2\Delta+1}{2\Delta+2}\frac{2D_{max}+d_b}{2D_{max}+D_{min}}D_{min} \\ &\leq \frac{2\Delta+1}{2\Delta+2}\frac{d_b}{D_{min}}D_{min} = \frac{2\Delta+1}{2\Delta+2}d_b. \end{aligned}$$

Further, if $b \in B_L \cap C_L$, $w_L(b,b) = d_b/(2\Delta+2)$. Therefore, for any job $b \in B_L$, $\sum_{c\in C_L} w_L(b,c) \leq d_b$. Hence, by Definition 3.1, w_L is an allocation scheme. □

Claim 4.2: $d_c/\left(\sum_{b\in B_L} w_L(b,c)\right) \leq 2\Delta+2$ for any job c in C_L.

Proof of Claim 4.2. If job $c \in C_L \cap B_L$, then $d_c/\left(\sum_{b\in B_L} w_L(b,c)\right) \leq d_c/w_L(c,c) = 2\Delta + 2$. Otherwise if job $c \in C_L \setminus B_L$, it must overlap with at least n jobs in B_L because Algorithm A could not have declined job c otherwise. Therefore, $d_c/\left(\sum_{b\in B_L} w_L(b,c)\right) \leq d_c/\left(nd_c/\left(n(2\Delta+2)\right)\right) = 2\Delta+2$. □

Since instance L was arbitrary, by Lemma 3.1, $r_A \leq 2\Delta+2$.

We will now prove the lower bound. Let $\beta > 0$ be arbitrarily large and $\epsilon > 0$ be arbitrarily small. Consider instance L in which the jobs arrive in following sequence: $\underbrace{(\beta, D_{min} + 2\epsilon), \ldots, (\beta, D_{min} + 2\epsilon)}_{n}$, $\underbrace{(\beta + \epsilon - D_{max}, D_{max}), \ldots, (\beta + \epsilon - D_{max}, D_{max})}_{n}$, $\underbrace{(\beta + \epsilon, D_{min}), \ldots, (\beta + \epsilon, D_{min})}_{n}$, $\underbrace{(\beta + \epsilon + D_{min}, D_{max}), \ldots, (\beta + \epsilon + D_{min}, D_{max})}_{n}$. Each 2-tuple represents a job, with first value being the start time and the second being the duration. Then, $OPT(L)/v_A(L) = n(D_{min} + 2D_{max})/(n(D_{min} + 2\epsilon))$. Therefore, $(D_{min} + 2D_{max})/(D_{min} + 2\epsilon) \leq r_A$. Since ϵ was arbitrary, this inequality holds for every ϵ, letting $\epsilon \to 0$ gives us the lower bound, $2\Delta + 1 \leq r_A$. □

N Proof of Statement 4 in Theorem 3.4

Before presenting the proof for Theorem 3.4, we present Lemma N.1 that is required for supporting intermediate claims in the proof for Theorem 3.4 and other subsequent proofs.

Lemma N.1. *Consider job i that overlaps with $k > 0$ jobs, none of which overlap with each other. Let a be the duration of the smallest of the $k+1$ jobs under consideration, and let b be the duration of the largest. Then, $d_i/k \geq a/2$ if $a = b$ and $d_i/k \geq a/3$ if $a < b$.*

Proof of Lemma N.1. If $a = b$, then all the jobs are of the same duration, and therefore, job i cannot overlap with more than two jobs. Thus, $k \leq 2$ and $d_i/k \geq a/2$. Otherwise, $a < b$. When $k \leq 2$, $d_i \geq a \implies d_i/k \geq a/2 \geq a/3$. Otherwise, if $k \geq 3$, then at least $k-2$ jobs start after s_i, the start time of job i, and end before $s_i + d_i$, the end time of job i. Therefore, $d_i \geq (k-2)a \implies d_i/k \geq ((k-2)a)/k \geq a/3$. □

Proof of Statement 4 in Theorem 3.4. Given n and $\mathcal{D}$, denote $t(n, \Delta)$ by t. Given an instance L, define B_L to be the set of jobs accepted by Algorithm D and C_L to be the offline optimal set. Define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} d_c/(t+1) & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ t\phi(i)/((t+1)3n) & \text{if } b \text{ and } c \text{ overlap, } b \not\equiv c, \ b \text{ is assigned} \\ & \text{to server } i, \text{ and } \phi(i) \leq d_c, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 1: w_L is an allocation scheme from the set B_L to the set C_L.

Proof of Claim 1. For any given job $b \in B_L$ that is placed assigned to server i by Algorithm D, a non-zero allocation from job b was made to jobs in the set $C_L \setminus \{b\}$ that have duration greater than or equal to $\phi(i)$, and also to job b itself if it belongs to $B_L \cap C_L$. Set C_L contains jobs that would have been selected if we knew the instance L upfront, and therefore, the sets C_L and $C_L \setminus \{b\}$ can be partitioned into n subsets such that each subset consists only of non-overlapping jobs. Let D denote one of such n subsets of $C_L \setminus \{b\}$, and let M be the number of jobs in D with duration greater than or equal to $\phi(i)$. Then, $\sum_{d \in D} w_L(b,d) \leq Mt\phi(i)/((t+1)3n)$. If $M > 0$, by Lemma N.1, $d_b/M \geq \phi(i)/3 \implies \sum_{d \in D} w_L(b,d) \leq td_b/((t+1)n)$. This also holds true if $M = 0$. Since there are n subsets in the partition of $C_L \setminus \{b\}$, we have $\sum_{c \in C_L \setminus \{b\}} w_L(b,c) \leq td_b/(t+1)$. The only other non-zero allocation from job b could have been to itself if job $b \in B_L \cap C_L$, and this allocation is less than or equal to $d_b/(1+t)$. Thus, $\sum_{c \in C_L} w_L(b,c) \leq td_b/(t+1) + d_b/(t+1) = d_b$. Therefore, by Definition 3.1, w_L is an allocation scheme. □

Claim 2: $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq t+1$ for any job c in C_L.

Proof of Claim 2. Consider any job c in C_L and let m be the largest integer in $[n]$ such that $\phi(m) \leq d_c$. By definition of t, $d_c \leq \phi(m+1)$. If job c belongs to the set $C_L \setminus B_L$, then it was rejected by Algorithm D. Therefore, it must overlap with at least one job assigned

to each one of the servers $1, 2, \ldots, m$, where $I \leq m \leq n$. Therefore, $\sum_{b \in B_L} w_L(b,c) \geq \sum_{i=1}^{m} t\phi(i)/((t+1)3n) = \phi(m+1)/(t+1) \geq d_c/(t+1)$. Otherwise, if job c belongs to the set $C_L \cap B_L$, $\sum_{b \in B_L} w_L(b,c) \geq w_L(c,c) = d_c/(t+1)$. Therefore, for any job c in C_L, $d_c / \left(\sum_{b \in B_L} w_L(b,c) \right) \leq t+1$. □

Since instance L was arbitrary, by Lemma 3.1, the result is proved. □

O Proof of Statements 2 and 4 in Theorem 3.5

Proof. We begin by presenting some notation. L denotes an arbitrary instance, C_L denotes offline optimal set for L, $N_L(x)$ denotes the total number of jobs with duration equal to x in C_L, L_x denotes the sequence of jobs obtained by removing all jobs with duration strictly less than x from L, C_L^x denotes the subset of jobs obtained by removing all jobs with duration strictly less than x from C_L, B_L^x denotes the set of jobs accepted by Algorithm R from instance L when sampled threshold is x, $g(x)$ denotes the total duration of jobs in the set B_L^x, and E denotes the set $\{\text{distinct durations of jobs in } L\} \cup \{D_{min}\}$ such that $E = \{x_1, x_2, \ldots, x_k\}$ and $D_{min} = x_1 < x_2 < \ldots < x_k \leq D_{max}$. Define $p_1 \doteq \mu(x_1)$ and $p_i \doteq \mu(x_i) - \mu(x_{i-1})$ for $i = 2, \ldots, k$.

Statement 2. Algorithm R effectively chooses x_i as threshold with probability p_i for $i \in [k]$. Consider the run of algorithm when R chooses x_i as the threshold. Define function $w_L^{x_i} : B_L^{x_i} \times C_L^{x_i} \mapsto \mathbb{R}$ as follows:

$$w_L^{x_i}(b,c) \doteq \begin{cases} d_c & \text{if } b \in B_L^{x_i} \cap C_L^{x_i} \text{ and } b \equiv c, \\ x_i/3 & \text{if } b \text{ and } c \text{ overlap, } b \in B_L^{x_i} \setminus C_L^{x_i}, \text{ and } c \in C_L^{x_i} \setminus B_L^{x_i}, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 1: $w_L^{x_i}$ is an allocation scheme from the set $B_L^{x_i}$ to the set $C_L^{x_i}$.

Proof of Claim 1. If job $b \in B_L^{x_i} \cap C_L^{x_i}$, then it does not overlap with any other job in C_L except itself (Lemma M.1), and therefore, $\sum_{c \in C_L^{x_i}} w_L^{x_i}(b,c) = w_L^{x_i}(b,b) = d_b$. Otherwise if job $b \in B_L^{x_i} \setminus C_L^{x_i}$, the only non-zero allocations from job b are to jobs in $C_L^{x_i} \setminus B_L^{x_i}$. Let M be the number of jobs in $C_L^{x_i} \setminus B_L^{x_i}$ that overlap with job b. Therefore, $\sum_{c \in C_L^{x_i}} w_L^{x_i}(b,c) \leq M x_i/3$. If $M > 0$, by Lemma N.1, $d_b/M \geq x_i/3 \implies \sum_{c \in C_L^{x_i}} w_L^{x_i}(b,c) \leq d_b$. This also holds true if $M = 0$. Therefore, by Definition 3.1, $w_L^{x_i}$ is a valid allocation scheme. □

Now, any job $c \in C_L^{x_i}$ either belongs to $B_L^{x_i} \cap C_L^{x_i}$, in which case $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq w_L^{x_i}(c,c) = d_c \geq x_i \geq x_i/3$, or belongs to $C_L^{x_i} \setminus B_L^{x_i}$, in which case it is rejected because it overlaps with at least one job in $B_L^{x_i} \setminus C_L^{x_i}$ and hence, $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq x_i/3$. Therefore, for any job $c \in C_L^{x_i}$, $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq x_i/3$. Hence,

$$g(x_i) = \sum_{b \in B_L^{x_i}} d_b \geq \sum_{b \in B_L^{x_i}} \sum_{c \in C_L^{x_i}} w_L^{x_i}(b,c) = \sum_{c \in C_L^{x_i}} \sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq \sum_{c \in C_L^{x_i}} x_i/3 = \sum_{j=i}^{k} N(x_j) x_i/3$$

$$\implies \frac{\sum_{i=1}^{k} p_i g(x_i)}{OPT(L)} \geq \frac{\sum_{i=1}^{k} p_i x_i \sum_{j=i}^{k} N(x_j)}{3OPT(L)} = \frac{\sum_{i=1}^{k} N(x_i) \sum_{j=1}^{i} p_j x_j}{3OPT(L)} = \frac{\sum_{i=1}^{k} N(x_i) p_1 x_1 + \sum_{i=2}^{k} N(x_i) \sum_{j=2}^{i} p_j x_j}{3OPT(L)}$$

For any $i = 2, \ldots, k$, $\sum_{j=2}^{i} p_j x_j \geq \int_{x_1}^{x_i} \frac{d\mu}{dx} x dx = \int_{x_1}^{x_i} \frac{x\ dx}{x(1+\ln\Delta)} = \frac{x_i - x_1}{(1+\ln\Delta)}$. Further, $\sum_{i=1}^{k} p_i g(x_i) = v_R(L)$. Therefore,

$$\frac{OPT(L)}{v_R(L)} \leq \frac{3(1+\ln\Delta)OPT(L)}{\sum_{i=1}^{k} N(x_i) x_1 + \sum_{i=2}^{k} N(x_i)(x_i - x_1)} = \frac{3(1+\ln\Delta)}{\sum_{i=1}^{k} x_i N(x_i)/OPT(L)} = 3(1+\ln\Delta)$$

because $OPT(L) = \sum_{i=1}^{k} x_i N(x_i)$. Since L was arbitrary, the result is proved. □

Statement 4. Algorithm R effectively chooses x_i as threshold with probability p_i, $i \in [k]$. Consider the run of algorithm when R chooses x_i as the threshold. Define function $w_L^{x_i} : B_L^{x_i} \times C_L^{x_i} \mapsto \mathbb{R}$ as follows:

$$w_L^{x_i}(b,c) \doteq \begin{cases} d_c/4 & \text{if } b \in B_L^{x_i} \cap C_L^{x_i} \text{ and } b \equiv c, \\ x_i/(4n) & \text{if } b \text{ and } c \text{ overlap, and } b \not\equiv c, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 2: $w_L^{x_i}$ is an allocation scheme from the set $B_L^{x_i}$ to the set $C_L^{x_i}$.

Proof of Claim 2. For any given job $b \in B_L^{x_i}$, a non-zero allocation from job b was made to jobs in $C_L^{x_i} \setminus B_{x_i}$ and also to job b itself if it belongs to $B_L^{x_i} \cap C_L^{x_i}$. Now, the set of jobs in $C_{x_i} \setminus B_{x_i}$ that overlap job b can be partitioned into n subsets such that no two jobs overlap in each subset. This is possible because a valid assignment of jobs in $C \supseteq C_L^{x_i} \setminus B_L^{x_i}$ to the n servers must exist by definition of C. Let D denote one of these subsets, and let M be the number of jobs in it. Then $\sum_{d \in D} w_L^{x_i}(b,d) \leq M x_i/(4n)$. If $M > 0$, by Lemma N.1, $d_b/M \geq x_i/3 \implies \sum_{d \in D} w_L^{x_i}(b,d) \leq 3d_b/(4n)$. This also holds true if $M = 0$. Because there are n subsets in the partition, $\sum_{c \in c_L^{x_i} \setminus \{b\}} w_L^{x_i}(b,d) \leq n 3d_b/(4n) = 3d_b/4$. Also, if job $b \in B_L^{x_i} \cap C_L^{x_i}$, $w_L^{x_i}(b,b) = d_b/4$. Therefore, $\sum_{c \in c_L^{x_i}} w_L^{x_i}(b,d) \leq 3d_b/4 + d_b/4 = d_b$. Hence, by Definition 3.1, $w_L^{x_i}$ is a valid allocation scheme. □

Now, any job $c \in C_L^{x_i}$ either belongs to $B_L^{x_i} \cap C_L^{x_i}$, in which case $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq w_L^{x_i}(c,c) = d_c/4 \geq x_i/4$, or belongs to $C_L^{x_i} \setminus B_L^{x_i}$, in which case it is rejected because it overlaps with at least n jobs in $B_L^{x_i}$ and hence, $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq n x_i/(4n) = x_i/4$. Therefore, for any job $c \in C_L^{x_i}$, $\sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq x_i/4$. Hence,

$$g(x_i) = \sum_{b \in B_L^{x_i}} d_b \geq \sum_{b \in B_L^{x_i}} \sum_{c \in C_L^{x_i}} w_L^{x_i}(b,c) = \sum_{c \in C_L^{x_i}} \sum_{b \in B_L^{x_i}} w_L^{x_i}(b,c) \geq \sum_{c \in C_L^{x_i}} x_i/4 = \sum_{j=i}^{k} N(x_j) x_i/4.$$

We then follow steps similar to those for the proof of Statement 2 to get the desired result. □

Table P.1: Sequences $s_1, s_2, \ldots, s_{k+1}$.

Notation	Sequence of Jobs
s_1	$(\delta, D_{min}), (2\delta, D_{min}), \ldots, (n\delta, D_{min})$
s_2	$((n+1)\delta, \alpha D_{min}), ((n+2)\delta, \alpha D_{min}), \ldots, (2n\delta, \alpha D_{min})$
$\vdots$	$\vdots$
s_{k+1}	$((kn+1)\delta, \alpha^k D_{min}), ((kn+2)\delta, \alpha^k D_{min}), \ldots, ((kn+k)\delta, \alpha^k D_{min})$

Table P.2: Probability distribution $\mathcal{Q}$ over $\mathcal{K}$.

Instance	Probability
$L_1 = \{s_1\}$	q_1
$L_2 = \{s_1, s_2\}$	q_2
$\vdots$	$\vdots$
$L_{k+1} = \{s_1, s_2, \ldots, s_{k+1}\}$	q_{k+1}

P Proof of Theorem 3.6

In the proof for Theorem 3.2, just by changing the set $\mathcal{K}$ and the probability distribution $\mathcal{Q}$ and using similar algebraic arguments, we can prove Theorem 3.6. Therefore, we present only the changes that need to be made in the setup in the proof for Theorem 3.2. Let $\delta > 0$ be arbitrarily small, k be an arbitrary positive integer, and $\alpha \doteq \Delta^{\frac{1}{k}}$. Let $q_i \doteq \left(1/\alpha^{i-1}\right) - \left(1/\alpha^i\right)$ for $i \in [k]$ and $q_{k+1} \doteq 1/\left(\alpha^k\right)$. Let $s_1, s_2, \ldots, s_{k+1}$ be sequences, each with n jobs, as shown in Table P.1. Define $\mathcal{K} \doteq \{L_1, L_2, \ldots, L_{k+1}\}$. The probability distribution $\mathcal{Q}$ over the set $\mathcal{K}$ is shown in the Table P.2.

Q Proof of Theorem 3.7

We first present, without proof, Lemma Q.1 that is similar to Lemma M.1. Then, we present the allocation schemes for each of the four statements. Theorem 3.7 can be proved using Lemma Q.1, the above-mentioned allocation schemes, and and using similar algebraic arguments as in the proof of Theorem 3.3 in Appendix M. Therefore, the detailed proof is omitted.

Lemma Q.1. *Given an instance L, let B_L be the set of jobs accepted by Algorithm A and C_L be an offline optimal set. Then, following statements hold true: (1.) If $n = 1$, any job $b \in B_L \cap C_L$ does not overlap with any other job in C_L except itself. Also, any job $c \in C_L \cap B_L$ does not overlap with any other job in B_L except itself. (2.) If $D_{max} = D_{min}$, then any job $b \in B_L$ overlaps with at most n jobs in the set C_L that have start times greater than or equal to s_b, the start time of job b. (3.) If $D_{max} > D_{min}$, then for any job $b \in B_L$, the jobs in C_L that have start times greater than or equal to s_b and that overlap with job b have total duration strictly less than $n\,(D_{max} + d_b)$.*

Allocation Schemes for Proof of Theorem 3.7. <u>*Statement 1.*</u> Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ D_{min} & \text{if } b \text{ and } c \text{ overlap, } s_b < s_c,\ b \in B_L \setminus C_L, \text{ and } c \in C_L \setminus B_L, \\ 0 & \text{otherwise.} \end{cases}$$

<u>*Statement 2.*</u> Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ \dfrac{d_c}{\Delta + 1} & \text{if } b \text{ and } c \text{ overlap, } s_b < s_c,\ b \in B_L \setminus C_L, \text{ and } c \in C_L \setminus B_L, \\ 0 & \text{otherwise.} \end{cases}$$

Statement 3. Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} D_{min}/2 & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ D_{min}/(2n) & \text{if } b \text{ and } c \text{ overlap, and } s_b < s_c, \\ 0 & \text{otherwise.} \end{cases}$$

Statement 4. Given instance L, define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} \dfrac{d_c}{\Delta+2} & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ \dfrac{d_c}{n(\Delta+2)} & \text{if } b \text{ and } c \text{ overlap, and } s_b < s_c, \\ 0 & \text{otherwise.} \end{cases}$$

□

R Proof of Theorem 3.8

For first three cases, Algorithm D is a fair algorithm, and therefore, the results are same as Theorem 3.7. For the proof of Statement 4, we first present, without proof, Lemma R.1 that is similar to Lemma N.1. Then, we present an allocation scheme. Statement 4 can be proved using Lemma R.1, the above-mentioned allocation scheme, and using similar algebraic arguments as in the proof of Statement 4 in Theorem 3.4. Therefore, the detailed proof is omitted.

Lemma R.1. *Consider job i that overlaps with $k > 0$ jobs, all of which have start times greater than or equal to s_i and none of which overlap with each other. Let a be the duration of the smallest of the $k+1$ jobs under consideration, and let b be the duration of the largest. Then, $d_i/k \geq a$ if $a = b$ and $d_i/k \geq a/2$ if $a < b$.*

Allocation Scheme for Statement 4 in Theorem 3.8. Given n and $\mathcal{D}$, denote $t^*(n,\Delta)$ by t^*. Given an instance L, define B_L to be the set of jobs accepted by Algorithm D and C_L to be the offline optimal set. Define function $w_L : B_L \times C_L \mapsto \mathbb{R}$ as follows:

$$w_L(b,c) \doteq \begin{cases} d_c/(t^*+1) & \text{if } b \in B_L \cap C_L \text{ and } b \equiv c, \\ t^*\phi(i)/((t^*+1)2n) & \text{if } b \text{ and } c \text{ overlap, } s_b < s_c, b \text{ is assigned} \\ & \text{to server } i, \text{ and } \phi^*(i) \le d_c, \\ 0 & \text{otherwise.} \end{cases}$$ □

S Proof of Theorem 3.9

When $\Delta = 1$, Algorithm R is a fair algorithm, and therefore, the results for Statements 1 and 3 are same as the corresponding cases in Theorem 3.7. The proofs Statements 2 and 4 in Theorem 3.9 are similar to those for the corresponding statements in Theorem 3.5, except that we must use Lemma R.1 instead of Lemma N.1, and that the functions, $w_L^{x_i}$'s, must be defined as given below. The detailed proof is omitted.

Statement 2. Define function $w_L^{x_i} : B_L^{x_i} \times C_L^{x_i} \mapsto \mathbb{R}$ as follows:

$$w_L^{x_i}(b,c) \doteq \begin{cases} d_c & \text{if } b \in B_L^{x_i} \cap C_L^{x_i} \text{ and } b \equiv c, \\ x_i/2 & \text{if } b \text{ and } c \text{ overlap, } s_b < s_c, \ b \in B_L^{x_i} \setminus C_L^{x_i}, \text{ and } c \in C_L^{x_i} \setminus B_L^{x_i}, \\ 0 & \text{otherwise.} \end{cases}$$

Statement 4. Define function $w_L^{x_i} : B_L^{x_i} \times C_L^{x_i} \mapsto \mathbb{R}$ as follows:

$$w_L^{x_i}(b,c) \doteq \begin{cases} d_c/3 & \text{if } b \in B_L^{x_i} \cap C_L^{x_i} \text{ and } b \equiv c, \\ x_i/(3n) & \text{if } b \text{ and } c \text{ overlap, and } s_b < s_c, \\ 0 & \text{otherwise.} \end{cases}$$