Online Algorithms for Dynamic Resource Allocation Problems

Xinshang Wang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Online Algorithms for Dynamic Resource Allocation Problems

Xinshang Wang

Dynamic resource allocation problems are everywhere. Airlines reserve flight seats for those who purchase flight tickets. Healthcare facilities reserve appointment slots for patients who request them. Freight carriers such as motor carriers, railroad companies, and shipping companies pack containers with loads from specific origins to destinations.

We focus on optimizing such allocation problems where resources need to be assigned to customers in real time. These problems are particularly difficult to solve because they depend on random external information that unfolds gradually over time, and the number of potential solutions is overwhelming to search through by conventional methods.

In this dissertation, we propose viable allocation algorithms for industrial use, by fully leveraging data and technology to produce gains in efficiency, productivity, and usability of new systems. The first chapter presents a summary of major methodologies used in modeling and algorithm design, and how the methodologies are driven by the size of accessible data.

Chapters 2 to 5 present genuine research results of resource allocation problems that are based on Wang and Truong (2017); Wang et al. (2015); Stein et al. (2017); Wang et al. (2016). The algorithms and models cover problems in multiple industries, from a small clinic that aims to better utilize its expensive medical devices, to a technology giant that needs a cost-effective, distributed resource-allocation algorithm in order to maintain the relevance of its advertisements to hundreds of millions of consumers.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Van-Anh Truong. She has devoted tremendous amount of time to turning me from a PhD newbie to a mature researcher. It is an invaluable experience to work with her as she lets me understand that research is just research, nothing else.

I would like to sincerely thank my collaborators Prof. Guillermo Gallego and Prof. Cliff Stein, for the help with exploring new research topics together and for always adhering to high standards. Their deep knowledge continuously encourages me to pursue my academic journey.

My sincere appreciations also go to Prof. Donald Goldfarb, Prof. Garud Iyengar, Prof. Jay Sethuraman, Prof. Ward Whitt, and Prof. David Yao for being great teachers on various fundamental domains of Operations Research. It is my genuine fortune to be recognized and inspired by Prof. Ward Whitt before joining IEOR. I am also very grateful to Prof. Vineet Goyal, Prof. Adam Elmachtoub and Prof. Xi Chen for always being available for insightful discussions.

I specially thank my PhD colleague Anran Li, for helping me smoothly step into new research areas through friendly discussions and collaboration. I also thank many other PhD friends for supporting me throughout the journey. I thank all the IEOR staff members for sending lovely gifts to my children and for keeping the IEOR community warm and pleasant.

Last but not least, I would like to thank my parents, my wife, my daughter and my son, for giving me a wonderful home.

Xinshang Wang

August 2, 2017

To my parents, my wife and my children

Chapter 1

Introduction

Dynamic resource allocation problems are everywhere. Airlines reserve flight seats for those who purchase tickets. Freight carriers such as motor carriers, railroad companies, and shipping companies pack containers with loads from specific origins to destinations. Healthcare facilities reserve appointment slots for patients who request them. This dissertation focuses on such resource allocation applications where resources need to be assigned to customers in a dynamic environment.

Dynamic allocation problems are particularly difficult to solve because they depend on random external information that unfolds gradually over time, and the number of potential solutions is overwhelming to search through by conventional methods. The ability to solve dynamic allocation problems often determines the competitive edge of modern technology-based companies.

The research presented in this dissertation aims to design viable allocation algorithms for industrial use. We assess all the algorithms presented using the following criteria:

- **Analytical performance guarantees**. Having a performance guarantee means that the outcome of allocation cannot be arbitrarily bad compared with the best possible strategy. It is only with provable performance guarantees that one can safely automate allocation decisions, which is especially essential for Internet-based services and large-scale systems. Section 1.1 overviews different types of performance guarantees we use in the analysis of this dissertation.

- **Empirical performance**. For many resource allocation problems, simple heuristics may work very well. Through simulations based on industrial data, we ensure that the algorithms we have proposed, or some variants of our algorithms, outperform naive heuristics in practice. The empirical experiments in this dissertation are based on collaborations with Columbia University Medical Center (CUMC) and the Institute of Data Science and Technology (IDST) at Alibaba Group. Both institutions have provided valuable data for verifying the benefit of our allocation algorithms.

- **Ease of implementation**. We make sure that the types of data required by our algorithms and models are commonly available in industry, and that the computational cost of our algorithms is reasonable.

1.1 Overview of Analysis Techniques

In modeling resource allocation problems, the design of assumptions about dynamic elements mostly depends on what data are available. Data that are commonly useful in resource allocation problems include the arrivals, preferences and profiles of customers and resources

(i.e., agents, workers) recorded in the past. Typically, these data are used to estimate distributional information about future events. Depending on the amount of usable data, there are different practical requirements of designing a model. In this dissertation, we focus on the following three important domains of model design for dynamic resource allocation problems:

1. No (or not enough) reliable historical data. For example, at the start of a new business, there is very little past information that can be used to predict future demands. Another example is when demands are affected by an adversary, so that future adversarial events have little to do with past events. In these scenarios, it is advisable not to estimate model parameters based on historical data, and to focus on the worst-case analysis.

Mathematically, we use competitive analysis (Borodin and El-Yaniv, 2005) to evaluate the performance of algorithms in the worst scenario. (For an example of competitive analysis in revenue management, see Ball and Queyranne (2009).) We say that an algorithm is *offline* if it knows all future information upfront. An algorithm is *online* if at all points in time, the algorithm only knows past and current information.

Consider a dynamic allocation problem where the objective is to minimize total cost. Let I be an instance that contains all the information related to the problem. Given instance I, let ALG(I) and OFF(I) be the total cost of an online algorithm ALG and an optimal offline algorithm OFF, respectively. In competitive analysis, the type of performance guarantee that we analyze is called the *competitive ratio*. For an online algorithm ALG, its competitive

ratio is defined as

$$\max_{I} \frac{\mathrm{ALG}(I)}{\mathrm{OFF}(I)}. \tag{1.1.1}$$

That is, the maximum ratio between the cost achieved under the online algorithm and that under the optimal offline algorithm. An algorithm with a competitive ratio of α is said to be α-*competitive*.

Note that if the problem is to maximize revenue instead of minimizing cost, we should replace max with min in the definition (1.1.1) of competitive ratio, in order to analyze the minimum ratio of revenue earned by online algorithms.

In Chapter 2, we analyze the competitive ratios of online algorithms. Although the focus of analysis is on the worst-case scenario, we show that the algorithms we propose, and their variants, attain decent performance for average cases as well.

2. Plentiful historical data for small to medium systems. Scenarios in this domain can be, for example, a clinic that has been running for a decade, so it is easy to estimate future arrival rates of various types of patients. With the access to future distributional information such as arrival rates, we can mathematically define an optimal allocation algorithm that maximizes the expected performance for a given model. However, it is often difficult to actually compute such an optimal algorithm due to the curse of dimensionality. Therefore, for problems in this domain, we aim to find approximate algorithms with performance guarantees relative to an optimal algorithm.

Mathematically, let D be the set of all model parameters, which possibly contains distributional information such as customer arrival rates. Let I be a problem instance that

contains all the realized information such as the actual number and times of customer arrivals. We still keep the notion of online algorithms, and study online algorithms that make dynamic decisions based on D. For revenue-maximizing problems in this domain, we define the performance guarantee of an online algorithm ALG as

$$\min_{D} \frac{\mathbf{E}[\mathrm{ALG}(I)|D]}{\mathbf{E}[\mathrm{OPT}(I)|D]}, \tag{1.1.2}$$

where the expectations are taken over the problem instance I that is randomly drawn according to the distributional information in D, and $\mathrm{OPT}(I)$ is the total revenue earned by an optimal (but impossible-to-compute) online algorithm that maximizes $\mathbf{E}[\mathrm{OPT}(I)|D]$.

Note that for cost-minimizing problems, we should replace the min with max in the definition (1.1.2).

Compared with the worst-case ratio (1.1.1), the metric (1.1.2) considers the expected performance of algorithms. As a result, the metric (1.1.2) is less conservative as it assigns lower weights to bad cases that occur with small probabilities.

In Chapters 3 and 4, we actually relax the definition of OPT to allow any offline algorithms, so as to obtain theoretically stronger guarantees. Then, we still call (1.1.2) the (average-case) competitive ratio of ALG. Nevertheless, the purpose of studying the average-case competitive ratio is to approximate an optimal online algorithm.

3. Plentiful historical data for large systems (a.k.a., "big data"). When terabytes of data are being accumulated every day, there are two major difficulties in designing resource allocation algorithms. First, when customer volume is huge (e.g., a billion customers

per day), it must not take more than a few CPU operations to compute the allocation decision for each single customer. In other words, we cannot solve a new complex problem for each customer in real time; useful algorithms often rely heavily on precomputation, and make allocation decisions for batches of customers. Second, allocation algorithms must be implemented on distributed computing systems. This limits the scope of applicable optimization tools.

When the size of a system grows large, relative performance ratios such as (1.1.1) and (1.1.2) often approach 1. Thus, we focus on rate at which approximate algorithms approach the optimal one. Mathematically, let $I_1, I_2, I_3, \cdots$ be a sequence of problems with increasing size. We analyze the following *regret* of an algorithm ALG in the asymptotic regime

$$\lim_{t\to\infty}(\mathrm{OPT}(I_t) - \mathrm{ALG}(I_t)), \tag{1.1.3}$$

where $\mathrm{OPT}(I_t)$ is the performance of an optimal algorithm for the problem of size t.

In Chapter 5, we analyze the asymptotic regret of distributed algorithms that maximize revenue for a large-scale budget allocation problem.

1.2 Overview of Thesis Chapters

First, in Chapter 2, we study a fundamental model of resource allocation in which a finite amount of service capacity must be allocated to a stream of jobs of different priorities arriving online. Jobs incur costs and may also cancel while waiting for service. To increase the

rate of service, overtime capacity can be used at a cost. This model has application in healthcare scheduling, server applications, make-to-order manufacturing systems, general service systems, and green computing. We present an online algorithm that minimizes the total cost due to waiting, cancellations and overtime capacity usage. We prove that our scheduling algorithm has cost at most twice of an optimal online algorithm. We also provide extensive numerical experiments to test the performance of our algorithm and its variants. Our proofs of the competitive ratios use a cost-balancing approach in conjunction with the following new ideas:

- We construct a novel *distance function* which summarizes in a single number the difference between the history of the online algorithm OLN and the optimal offline algorithm OFF. The distance function $\phi_t(\text{OLN}, \text{OFF})$ has a nice physical interpretation. At any time t, if we immediately service $\phi_t(\text{OLN}, \text{OFF})$ additional jobs under the online algorithm, the remaining jobs will have lower priorities than the current remaining jobs under the offline algorithm. The distance function dynamically accounts for the difference in the number of scheduled and canceled jobs between the two algorithms.

- Depending on the sign of the distance function in each period, we *partition* the periods in the planning horizon into two sets. We show that in each type of the periods, one cost component of the online algorithm is dominated by the corresponding component of the offline algorithm. This result naturally leads to the proof of the competitive ratios.

- For the model with cancellations, we use *stochastic coupling* to compare the exoge-

nous cancellation events under the online and offline algorithms. When extended to the model with cancellations, our distance function incorporates the difference in the number of coupled cancellation events between the two algorithms.

- For the model with cancellations, we propose a new *cost-accounting scheme* which transforms cancellation costs into new waiting and overtime costs. This transformation allows the algorithms for the model without cancellations to be easily extended to capture cancellation behaviors.

Secondly, in Chapter 3, we study web and mobile applications that are used to schedule advance service, from medical appointments to restaurant reservations. We provide the first general, high-fidelity model of advance admission scheduling that captures customer preferences across different resources. We allow non-stationary arrivals and no-shows. We model the advance admission-scheduling problem as an online weighted bipartite matching problem with non-stationary arrivals and propose new algorithms with guarantees on the relative performance. We propose new algorithms with performance guarantees for this class of problems. The contributions of Chapter 3 are as follows:

- We prove the tightest known performance bound for the online matching problem with non-stationary stochastic arrivals. Specifically, we use the definition (1.1.2) of competitive ratio, and prove that a primitive algorithm, which we call the *Separation Algorithm*, has competitive ratio that is bounded by $\max(\frac{1}{2}, 1 - \sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}} + O(\frac{1}{k}))$, where k is the minimum capacity of a resource. Furthermore, we show that $\frac{1}{2}$ is the best constant competitive ratio that can be achieved.

We obtain our bound by analyzing a novel *bounded Poisson process*. This is a Poisson process to which we apply a sequence of reflecting barriers. The process arises in the dual of an optimization problem that characterizes our performance bound. The behavior of this process is very complex, with no known closed-form description. We managed to obtain a closed-form approximate characterization of the process.

- We improve on the Separation Algorithm by devising a novel bid-price-based algorithm, called the Marginal Allocation Algorithm, that is much more practical. First, the Marginal Allocation Algorithm is non-randomized, therefore more stable. Second, it is fair in the sense that it never rejects a high-priority customer but accept a low-priority customer, assuming that their arrival times and preferences are the same. We prove that the Marginal Allocation Algorithm has the same competitive ratio as the Separation Algorithm. In addition, in numerical experiments, we show that it achieves much better practical performance.

- Our model also has application in other important problems such as display-ad allocation and opaque revenue management. For the display-ad allocation problem, we give the tightest known performance bound for an algorithm, assuming non-stationary arrivals and arbitrary mean demand. For the opaque revenue-management problem, we are the first to study online allocation policies for model with an arbitrary number of products and time-varying arrival rates.

- We test the empirical performance of our algorithm against several well-known heuristics by using appointment-scheduling data from Columbia University Medical Center.

The results show that our scheduling algorithms perform the best among all tested policies. In particular, our algorithm is 21% more effective than the actual scheduling strategy used in the hospital system according to our performance metric.

Next, in Chapter 4, we study a resource allocation model in which a finite number of resources must be assigned in an online manner to customers with heterogeneous resource requirement. The system must find a feasible assignment of each customer to a resource or must reject the customer. The aim is to maximize the total expected capacity utilization of the resources over the horizon. This model has application in services, freight transportation, and online advertising. We present online algorithms with bounded average-case competitive ratios. The contributions of Chapter 4 are:

- We propose the first service reservation model with non-stationary arrivals and heterogeneous resource requirement. Our model generalizes Adwords problems in that we do not make the assumption of truncated bids, small bids, or i.i.d. demand.

- For our model, we propose the first online algorithms with constant competitive ratios defined in (1.1.2). Our algorithms are based on a smart reservation strategy. For each resource, we pre-calculate several groups of customer types such that when packing customers with types in each group in an arbitrary order, the resource utilization is bounded below. Our algorithms reserve each resource for a certain group of customer types so as to maximize the actual performance.

- We test our online algorithms as well as other simple heuristics using a real data set

from Columbia University Medical Center. We find that our online algorithms have the best empirical performance in most scenarios.

Finally, in Chapter 5, we study a novel resource-allocation problem faced by Alibaba Group. In this problem, mobile "push messages" must be sent over the course of a day to hundreds of millions of users. Each message can be sent to any number of users, and yields a reward when it generates a clickthrough, subject to a budget constraint on the total reward over all users for the message. This budget represents the maximum amount that an advertiser is willing to pay for clickthroughs for the message on a given day. Given users' diverse preferences, the problem aims to deliver the "right messages" to the "right users" to maximize ad revenues without overwhelming each user with too many messages. The main contributions of Chapter 5 are highlighted as follows:

- Due to the large size of the real application, we analyze algorithms for the above problem in an asymptotic regime. We consider a novel scaling of the problem "size," called big-data scaling. In this scaling, as the problem size grows, the number of users, as well as their diversity, grow. The scaling captures the fact that individual user information remains highly granular and distinctive even as the size of the user base increases.

- We first analyze a simple algorithm, called the *Static Algorithm*, which essentially sends out all push messages in one cycle based on a solution to a static assignment problem. We prove that the Static Algorithm has an asymptotic regret $O(\sqrt{t})$, where t is the parameter scaling the problem.

- We then propose a new algorithm, called the *Reservation Algorithm*, which adds a single recourse opportunity by sending push messages in two cycles over the course of a day and making use of information observed in the first cycle to adapt decisions in the second cycle. We prove that the Reservation Algorithm has asymptotic regret $O(t^{1/4} \log t)$.

- We code our algorithm using the MapReduce framework and test its performance on a distributed computing platform. Numerical experiments on three real data sets, each containing several hundred million users, show that our Reservation Algorithm improves the regret of the Static Algorithm by at least 10%-50%.

Chapter 2

Multi-priority Online Scheduling with Cancellations

2.1 Introduction

In many applications, a finite amount of a service resource must be allocated to a stream of jobs arriving randomly over time. Jobs are prioritized based on certain criteria such as profitability or urgency. When immediate service is not available, arriving jobs join a priority queue to be served at a later time. While waiting, jobs may cancel their requests and leave the queue randomly. A *cancellation* is any job that expires or leaves the system without being processed. To increase the rate of service, overtime resources can be used at a higher cost. The system must dynamically determine the service rate that minimizes the total cost due to waiting, cancellations and overtime resource usage.

The above problem is central to many applications in Operations Research. For example,

in many service systems, make-to-order manufacturing systems, retail stores and call centers, jobs correspond to customers arriving randomly over time. Depending on the application, customers may be served in order of their priorities. Backlogged customers may cancel their orders (Rubino and Ata, 2009; Blackburn, 1972), resulting in lost sales and even 'reshelving' costs (Martin et al., 1992). In these settings, the service rate can often be increased by using overtime work (Dellaert and Melo, 1998; Özdamar and Yazgaç, 1997), on-call workers (Greenhouse, 2012), or expedited procurement of parts. These strategies have the effect of temporarily increasing the service rate at an increased variable cost. The manager needs to dynamically determine the service policy so as to control the total system cost.

In healthcare facilities, jobs correspond to patient requests for resources such as diagnostic devices and operating rooms. Patients are often prioritized based on their urgency and served in order of priority (Min and Yih, 2010). Longer wait times, which result in lower quality of care, are represented as a cost on the system. Time is often slotted. With a limited number of time slots available each day, only a certain number of patients can be served on each day; the remaining patients must join a waitlist (Denton et al., 2010; Ayvaz and Huh, 2010; Gerchak et al., 1996). Patients in the waitlist may randomly cancel their requests, thus leaving the system. Often, patients can be served using surge capacity or overtime (Patrick et al., 2008) at an additional cost. The scheduler must select the number of patients to serve each day, using surge capacity or overtime as needed, to minimize the total cost, including waiting costs, lost revenue due to cancellations, and the cost of overtime work.

In the scaling of computer processing speed for minimizing energy usage (Bansal et al., 2009a; Yao et al., 1995), jobs correspond to sequences of CPU instructions that arrive ran-

domly. Jobs are often prioritized and processed in order of priority. With recent technologies, the processing speeds of CPUs can be dynamically raised at the cost of a higher rate of power usage. Such a speed-scaling technique often helps to save more power than the simple strategy of turning off a device during idle periods. The goal is to minimize the sum of some measure of quality of service, such as job completion time and total energy consumption (Bansal et al., 2009a).

Our model captures most, if not all, of these applications. Specifically, we consider a discrete-time planning horizon of T periods, where T is possibly infinite. Jobs are categorized into n ranked groups, or priority classes. Each class is associated with a waiting cost, a cancellation probability and a cancellation cost. Jobs are either processed in the current period or are added to a priority queue. In each period t, a number C_t of jobs of any priority can be processed. Additional jobs can be processed at an extra variable cost.

The above scheduling problem is difficult to analyze in real applications due to the difficulty in forecasting future information. On the demand side, future arrivals are often class-dependent and time-dependent (Huh et al., 2013), which requires an enormous amount of data to estimate the joint distribution of demand for multiple classes. For instance, a patient request often leads to subsequent periodic requests, resulting in the time correlation of demand. Also, in markets of new products or services, demand is often driven by intensive promotion campaigns, in which case future demand depends on promotional and social factors and is highly uncertain. On the supply side, processing capacities are often subject to occasional failures such as staff absenteeism, machine breakdown (Federgruen and So, 1990) and server crashes, which can be very hard to predict.

In viewing these difficulties, in this thesis chapter, we analyze the competitive ratio of online algorithms as defined by equation (1.1.1). For the scheduling problem without cancellations, we propose 2-competitive randomized and deterministic online algorithms. For the scheduling problem with cancellations, we relax the assumption of the online problem by making the 'offline' policy unaware of which jobs will cancel, i.e., the random cancellation events are exogenous to both the online and offline policies. Under this definition, we propose 2-competitive online algorithms for the model with cancellations. Further, we show that the competitive ratio of our deterministic algorithm is the best that can be achieved.

2.2 Literature Review

Our work is related to several streams of literature, including literature on appointment scheduling, make-to-order systems, machine scheduling, rent-or-lease problems, and other online algorithms in operations management.

2.2.1 Appointment scheduling

Our work is related to the literature on appointment scheduling, which has been studied intensively. For comprehensive reviews of the broader area, see Guerriero and Guido (2011); May et al. (2011); Cardoen et al. (2010) and Gupta (2007). A large part of the literature considers *intra-day* scheduling. In these problems, the number of patients to be served on each day is given or is exogenous, and the task is to set the sequence and the start time of each appointment so as to control patient wait time and provider idle time. Another part

of the literature models *multi-day* scheduling. In these problems, the allocation of patients to days is dynamically controlled. Some of this literature allows patients to be scheduled into future days at the time of arrival. This paradigm is called *advance scheduling*. See, for example, Truong (2015); Gocgun and Ghate (2012) and Patrick et al. (2008). In the rest of the multi-day literature, an intermediate problem in which only the number of patients to be scheduled to the current period is determined, and the rest of the patients are assumed to be added to a waitlist. This paradigm is called *allocation scheduling*. See, for example, Huh et al. (2013); Min and Yih (2010); Ayvaz and Huh (2010) and Gerchak et al. (1996). So far, very few works have studied the optimal advance-scheduling policy. Recently, Truong (2015) linked the solutions for the advance and allocation scheduling problems by showing that for a two-class model, their optimal scheduling polices are equivalent. This result points to the importance of allocation scheduling as a fundamental model.

Our model is an allocation-scheduling model. In allocation scheduling, past works have used dynamic programming to explore structural properties of the optimal scheduling policy. When there are one or two patient classes, the problem is easy to solve. For multi-class problems, some structural results are known but there is no policy with performance guarantees. Gerchak et al. (1996) and Huh et al. (2013) study scheduling problems with two patient classes. Patients in the emergent class require same-day service; patients in the elective class can wait. Gerchak et al. (1996) show that the optimal scheduling policy is not a cut-off policy; the optimal number of admissions increases in the size of waitlist. Huh et al. (2013) develop heuristics for a correlated and dynamic environment. Min and Yih (2010) and Ayvaz and Huh (2010) study the allocation-scheduling problem with multiple elective patient classes.

Min and Yih (2010) develop bounds on the optimal number of admissions. They show that priority-based discrimination results in as much as a 30% difference in the optimal number of admissions compared to an undiscriminated scheme. Ayvaz and Huh (2010) analyze the structural properties of an optimal scheduling policy and study numerical performance of a protect-constant heuristic. The heuristics presented in these works do not come with any performance guarantees. Moreover, for the static policies that they propose, such as the protect-constant policies, it is easy to search for the best protect-constant levels only when the number of demand classes is small. When the number of demand classes is large, it is much harder to search for the best set of protect-constant levels without additional structural properties. Thus, in multi-class settings, even heuristics with good empirical performance are hard to find.

2.2.2 Make-to-order systems

The scheduling system we consider is related to make-to-order manufacturing systems in that processing capacity is used to service realized demand. These make-to-order systems are usually modeled as queuing systems. In the framework of queuing systems, service times and inter-arrival times must be stationary, independent and most often, exponentially distributed to ensure that the model is tractable. Our approach differs from this literature in that we do not assume any joint distribution on future arrivals and service capacities. For reviews on admission control for make-to-order queues, see Stidham (1985) and more recently, Carr and Duenyas (2000). Blackburn (1972) studies the optimal strategies for turning on or off a server subject to reneging customers. Their work is related to ours in

that they consider the dynamic expansion of the service rate. While they only consider one type of jobs, we allow jobs to have multiple priorities, each with a different cancellation probability. Rubino and Ata (2009) consider a related problem in which customers can be outsourced and have chances to renege. They propose a heuristic based on the solution to the problem in the heavy-traffic regime.

Our model is related to the work of Keskinocak et al. (2001), who study single-server online scheduling problems with lead-time quotation, with application to make-to-order manufacturing systems. In their model, jobs can be rejected upon arrival, and waiting costs are incurred in each period before the jobs are finished. The rejection of jobs is similar to the use of overtime resource in our model. Our work can be seen as a multi-priority, multi-server extension of their model, and with further considerations for job reneging. We note that in their model, a job may span multiple periods, while in our model, every job can be finished in a single period. However, our model easily accommodates batched arrivals. A job that takes multiple periods to finish can be modeled as a batched arrival.

2.2.3 Machine scheduling

The class of machine and multiprocessor scheduling problems share some characteristics with our work. However, overtime usage and job cancellations are not common in the machine-scheduling literature. In a typical machine-scheduling problem, jobs must be assigned to one or more machines so as to minimize a chosen objective such as the make span, the total completion time or the total waiting time. Our model resembles a machine-scheduling problem in which (1) jobs have unit processing times, (2) jobs can be rejected or diverted

after being released, (3) each job has a specific release time, which is used to define a waiting cost, and (4) jobs may cancel randomly. However, an online version of this model has not been considered. We refer the reader to Chen et al. (1998) and Megow et al. (2006) for more detailed surveys of machine scheduling. Among the existing literature, the most relevant works include Noga and Seiden (2001) and Zhang et al. (2009). Noga and Seiden (2001) consider an online machine-scheduling problem where jobs have release times and the objective is to minimize the total waiting cost, but the service rate cannot be dynamically controlled. Zhang et al. (2009) study a deterministic offline scheduling problem where jobs can be rejected.

2.2.4 Ski-rental problem and extensions

Our problem extends the classical ski-rental problem first studied by Karlin et al. (1988). In this problem, a single job waits to be processed some time in the future, but the exact date that the job will be processed is unknown. A waiting cost of \$1 is incurred in each period that the job has to wait. The job can also be immediately processed at an additional cost of $\$B$ at any time. The ski-rental problem is online if the exact time that the job will be processed is unknown and is chosen by an adversary. The optimal competitive ratio of the ski-rental problem is 2 for deterministic algorithms (Karlin et al., 1988) and $e/(e-1)$ for randomized algorithms (Karlin et al., 1990). Many variants of the ski-rental problems that have been studied, including those with multiple renting options (Fujiwara et al., 2011; Lotker et al., 2012), time-varying rental cost (Bienkowski, 2008), and decisions that assign rented or bought capacity to edges in a network (Gupta et al., 2007).

In a variation of the ski-rental problem closest to our model, a computer system needs to decide whether to execute a task immediately, incurring a high power-consumption cost, or whether it should let the task wait and pay a waiting cost per unit time. These problems are called speed-scaling problems. One group of works considers the optimization problem of some energy related objective, subject to deadlines for job completion (Yao et al., 1995; Chan et al., 2007; Bansal et al., 2007a, 2009b, 2011). The first theoretical study of such model is given by Yao et al. (1995). They show that an optimal offline algorithm for any convex power function can be computed by a greedy method. They also give an online algorithm with constant competitive ratio when the power function is polynomial. Another group of works considers energy usage and job waiting time (Albers and Fujiwara, 2007; Bansal et al., 2007b, 2009a). Bansal et al. (2009a) propose an online algorithm that minimizes the sum of fractional waiting costs and energy usage for arbitrary power functions. When there are no cancellations, our model captures the tradeoff in Bansal et al. (2009a). However, Bansal et al. (2009a) and most works in speed scaling consider continuous-time models. Our model captures a discrete-time speed scaling problem in which processing speeds can only be changed in discrete periods. Cancellation behaviors are generally not considered in the speed-scaling literature.

2.2.5 Online algorithms in Operations Management

Our online algorithms and their performance guarantees are related to many other approximation and online algorithms developed in Operations Management. Levi et al. (2007) propose a cost-balancing technique for inventory control problems. They prove that this

cost-balancing algorithm is a 2-approximation. The cost-balancing technique is found to be very adaptable and is applied in approximation algorithms for many other supply-chain problems (see, for example, Levi et al. (2008a) and Levi et al. (2008b)). Recently, Truong (2014) develops an approximation algorithm for the stochastic inventory control problem by using a look-ahead optimization approach.

Ball and Queyranne (2009) consider an online version of a revenue management problem. They show that the simple protection-level policy gives the best possible competitive ratio. The ratio depends on the level of price discounts. Wagner (2010) considers the online economic lot-sizing problem. They model the online profit-maximizing problem as a min-max game, and provide conditions under which the competitive ratio is bounded. Buchbinder et al. (2013) study an online algorithm for a make-to-order variant of the joint-replenishment problem for which they proved a competitive ratio of three. Elmachtoub and Levi (2016) study a general class of customer-selection problems where decisions are made in two phases: In the first phase, arriving customers with different configurations are selected in an online manner. Then in the second phase, the cost of the service system is generated based on the set of selected customers. They develop a framework of analysis for this class of problems and apply it to various models.

The remainder of this thesis chapter is organized as follows.

In Section 2.3, we present our online algorithm for the scheduling model without cancellations. In Section 2.3.7, we generalize our results to the case that future costs are discounted. In Section 2.3.8, we discuss lower bounds on the competitive ratio and prove that our algo-

rithm is optimal. In Section 2.4, we extend the model and algorithm to capture cancellations. Finally, in Section 2.5, we report the numerical performance of our scheduling policies.

2.3 Model of Allocation Scheduling without Cancellations

In this section, we focus on a basic model without cancellation. The planning horizon has T periods, indexed from 1 to T, where T may be infinite. There are n groups, or priority classes. Each class i is associated with a waiting cost $w_i \geq 0$, which is incurred when a class i job stays in the waitlist for one period. Let the n classes be ordered in decreasing order of priority. We assume that the waiting costs satisfy $w_1 \geq w_2 \geq \cdots \geq w_n$. The scheduling policies we present in the thesis chapter do not depend on the total number n of classes, so n can be arbitrarily large and the collection of waiting costs can even approach a continuous distribution.

At the beginning of each period t, we observe the vector $s_t = (s_{t1}, s_{t2}, ..., s_{tn})$ representing the total number of jobs currently in the waitlist, where s_{ti} is the number of jobs in class i. Then, we observe the regular capacity C_t, which is the number of jobs, regardless of priority, that can be processed by regular resource in period t. Next we observe the number of new arrivals $\delta_t = (\delta_{t1}, \delta_{t2}, ..., \delta_{tn})$, where δ_{ti} stands for the number of arrivals of class i jobs. We have $s_t, \delta_t \in \mathbb{Z}_+^n$ and $C_t \in \mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of all non-negative integers. For an online algorithm, C_t and δ_t are completely unknown until period t, while for an offline algorithm the entire sample path $\{(C_t, \delta_t)\}_{t=1,2,...,T}$ is known at the beginning of period 1.

After the new arrivals have occurred, the number of jobs in system is represented by the vector $s_t + \delta_t$. From among the $\|s_t + \delta_t\|_1$ jobs in system, a scheduling policy determines the number $a_t \in \mathbb{Z}_+$ of jobs to service in period t. We restrict our attention to the class of policies that serve some number of highest-priorities jobs in each period. We will discuss the reason for this restriction presently.

If $a_t > C_t$, we assume that the additional $a_t - C_t$ jobs will be served by overtime resource incurring a total *overtime cost* of $(a_t - C_t)p$, where p is the cost of using an *overtime slot*. If $a_t \leq C_t$, no overtime cost will be incurred. Define $d_t \equiv (a_t - C_t)^+$ as the number of overtime slots used in period t. We normalize all cost values such that $p = 1$. Then the total overtime cost in period t is just d_t. The objective is the undiscounted total cost over a finite number T of periods, namely, $V_T^\Pi = \sum_{t=1}^{T}(d_t^\Pi + W_t^\Pi)$. We will discuss extensions to discounted-cost models later.

It is intuitive that once the number a_t is decided, it is optimal to serve the a_t jobs with the highest priorities. This property is proved in Ayvaz and Huh (2010) and Min and Yih (2010) for optimal stochastic policies. The same result holds here. However, we do not repeat the proof.

Because we only consider policies that schedule some number of highest priority jobs in each period, two scheduling policies differ only in the timing and number of jobs drawn from the waitlist. We introduce the following operator that extracts a certain number of jobs with the highest priorities from a given system state $s_t \in \mathbb{Z}_+^n$.

Definition 2.3.1. *For a vector $x \in \mathbb{Z}_+^n$ and a non-negative integer k, we define $h(x, k) \in \mathbb{Z}_+^n$*

as the vector that contains the k jobs with the highest priorities in x. Let $h_i(x,k)$ *be the ith element of* $h(x,k)$*. Let* $h(x,k)=0$ *for* $k<0$*, and* $h(x,k)=x$ *for* $k>\|x\|_1$*.*

Since the w_i's are decreasing in i, we have for $0 \le k \le \|x\|_1$,

$$\begin{cases} h_i(x,k) = x_i, & \text{for } i < i^* \\ h_i(x,k) = 0, & \text{for } i > i^* \\ h_{i^*}(x,k) = k - \sum_{i=1}^{i^*-1} x_i, & \text{otherwise,} \end{cases}$$

where

$$i^* = \min\{j | \sum_{i=1}^{j} x_i \ge k\}.$$

Using this operator, we can write the number of jobs remaining in the waitlist at the end of period t as

$$f_t = s_t + \delta_t - h(s_t + \delta_t, d_t + C_t).$$

Next, the waiting cost incurred in period t can be written as

$$W_t = f_t^\tau w,$$

where τ is the transpose operator. In the next period, the initial state of the system is $s_{t+1} = f_t$.

For each policy Π, we add a superscript Π to all the state and decision variables that result from Π. If Π is an online algorithm, the decision d_t^Π does not depend on any information to be realized later than period t. When we present our online algorithm in Section 2.3.1,

the objective is the undiscounted total cost over a finite number T of periods, namely, $V_T^{\Pi} = \sum_{t=1}^{T}(d_t^{\Pi} + W_t^{\Pi})$. We will show that our online algorithm gives a total cost which is at most 2 times the total cost under an optimal offline algorithm for any sample path $\{(C_t, \delta_t)\}_t$. In Section 2.3.7 we further show that the same result holds in discounted, finite and infinite-horizon settings.

2.3.1 Online algorithm and summary of main ideas

First, we will present a online algorithm for the allocation-scheduling problem without cancellations. We will sketch the main ideas that go into the proof that this algorithm is 2-competitive.

Define an online algorithm OLN as follows. In each period t, OLN balances the total cumulative waiting cost and total cumulative overtime cost by minimizing the maximum of the two. Mathematically, let $W(d) = (s_t^{\mathrm{OLN}} + \delta_t - h(s_t^{\mathrm{OLN}} + \delta_t, d + C_t))^{\tau} w$ be the waiting cost to be incurred in period t if d overtime slots are used in t. Then d_t^{OLN} is determined by (recall that the unit overtime cost is $p = 1$)

$$d_t^{\mathrm{OLN}} = \mathrm{argmin}_d \max(\sum_{i=1}^{t-1} d_i^{\mathrm{OLN}} + d, \sum_{i=1}^{t-1} W_i^{\mathrm{OLN}} + W(d)). \tag{2.3.1}$$

The idea of OLN is to keep these two cumulative costs as closely matched to each other as possible.

We will prove that OLN has a competitive ratio of 2. The proof is based on a new concept of a *distance function*, which summarizes the difference in state between two policies

in any period by a scalar. The distance function is an *analytical tool*. We will show that the distance function between an online policy OLN and an optimal offline policy OFF separates the horizon into consecutive intervals, depending on its value. Type-A intervals, where the distance is 0, are single-period. Type-B intervals, where the distance is positive, may be multi-period. We will show that in a type-A interval, the waiting cost of OLN is bounded by that of OFF. In a type-B interval, the cumulative overtime cost of OLN is bounded by that of OFF. By balancing the waiting and overtime cost, OLN ensures that the larger of its two costs in each interval is bounded by a cost of OFF.

2.3.2 Dominance relationship between two policies

We first define a dominance relationship and prove some of its implications. A dominance relation is a partial order between two system states such that one state is smaller than the other if the jobs in that state have lower priorities than in the other state. This dominance relation allows us to compare system states, thereby arriving at a bound on costs.

Definition 2.3.2. *For two vectors $x, x' \in \mathbb{Z}_+^n$, we say x is dominated by x' and write $x \preceq x'$ if*

$$\sum_{i=1}^{l} x_i \leq \sum_{i=1}^{l} x'_i \quad \forall l = 1, 2, .., n.$$

A result immediately following this definition is that, if x and x' represent different system states at the end of period t, and $x \preceq x'$, then the total waiting cost incurred by the jobs in x at t is no greater than that incurred by the jobs in x'.

Lemma 2.3.3. *For two vectors $x, x' \in \mathbb{Z}_+^n$, if $x \preceq x'$, then $x^\tau w \leq x'^\tau w$.*

Proof.

$$\begin{aligned}
&\sum_{i=1}^{l} x_i \le \sum_{i=1}^{l} x'_i \quad \forall l = 1, 2, ..., n \\
\Longrightarrow &\sum_{i=1}^{l} x_i \alpha \le \sum_{i=1}^{l} x'_i \alpha \quad \forall l = 1, 2, ..., n \text{ and } \forall \alpha > 0 \\
\Longrightarrow &\sum_{l=1}^{n-1} \sum_{i=1}^{l} x_i (w_l - w_{l+1}) + \sum_{i=1}^{n} x_i w_n \le \sum_{l=1}^{n-1} \sum_{i=1}^{l} x'_i (w_l - w_{l+1}) + \sum_{i=1}^{n} x'_i w_n \\
\Longrightarrow &\sum_{i=1}^{n} x_i w_i \le \sum_{i=1}^{n} x'_i w_i.
\end{aligned}$$

□

The following lemma states two simple operations that preserve a dominance relation:

Lemma 2.3.4. *Fix an integer $l \ge 0$ and two vectors $x, x' \in \mathbb{Z}_+^n$ satisfying*

$$x - h(x, l) \preceq x'.$$

1. *For any integer $l' \ge 0$,*

$$x - h(x, l + l') \preceq x' - h(x', l').$$

2. *For any vector $\delta \in \mathbb{N}_0^n$,*

$$x + \delta - h(x + \delta, l) \preceq x' + \delta.$$

Proof. This lemma is easily proved by directly checking the definition of dominance relation.

□

Table 2.1: Example of the distance function. Based on the numbers of scheduled overtime slots d_t^{Π} and d_t^{Θ} of two scheduling policies Π and Θ, respectively, the values of the distance function $\phi_t(\Pi, \Theta)$ are computed and listed in the bottom row.

Period t	1	2	3	4	5	6	7	8	9
d_t^{Θ}	0	2	0	0	0	1	0	0	1
d_t^{Π}	1	0	1	2	0	0	0	1	2
$\phi_t(\Pi, \Theta)$	0	2	1	0	0	1	1	0	0

2.3.3 Distance Function and Comparison of Scheduling Policies

The following *distance function* captures the difference in the cumulative overtime usage between two scheduling policies.

For two scheduling policies Π and Θ, the distance function $\phi_t(\Pi, \Theta)$ is defined recursively as

$$\begin{cases} \phi_0(\Pi, \Theta) = 0, \\ \phi_t(\Pi, \Theta) = \max\{\phi_{t-1}(\Pi, \Theta) - d_t^{\Pi} + d_t^{\Theta}, 0\} \quad \text{for } t \geq 1, \end{cases} \tag{2.3.2}$$

where recall that d_t^{Π} and d_t^{Θ} are the numbers of overtime slots used under Π and Θ in period t, respectively. That is, in a given period, the distance function changes by the number overtime slots that Θ uses in excess of Π. However, its value is always kept non-negative.

Table 2.1 provides an illustrative example. It lists the number of overtime slots used in periods from 1 to 9 on a sample path. The corresponding values of the distance function are shown in the bottom row of the table.

The above definition of the distance function is motivated by its use in Section 2.3.4 below. The distance function is used mainly to identify periods in which one policy is "behind" another policy (when the distance is positive) and periods in which it is "ahead" (when the distance is 0). When a policy is ahead in a given period, it has a lower waiting cost.

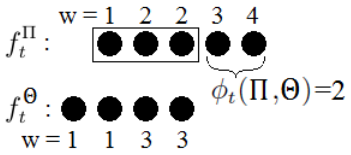


Figure 2.1: Illustration of the distance function. There are 4 priority classes with waiting costs $w = (4, 3, 2, 1)$. By the end of period t, $f_t^{\Pi} = (1, 1, 2, 1)$ and $f_t^{\Theta} = (0, 2, 0, 2)$. The figure displays all the jobs with their waiting costs marked. Assume that $\phi_t(\Pi, \Theta) = 2$. After $\phi_t(\Pi, \Theta) = 2$ jobs with the highest priorities are removed from f_t^{Π}, the remaining jobs, marked by the black box, have lower priorities than the jobs in f_t^{Θ}. Note that if we only removed 1 job with unit waiting cost of 4 from f_t^{Π}, the remaining jobs would not be 'dominated' by the jobs in f_t^{Θ}, as there would be 3 jobs with waiting costs of at least 2 remaining in f_t^{Π}, but only 2 such jobs in f_t^{Θ}.

When a policy is behind, we will show that it has used up fewer overtime slots cumulatively over a specific interval.

2.3.4 Invariance between policies in terms of the distance function

We will establish the following direct physical interpretation of the distance function. If we remove the number of jobs equal to the value of the function $\phi_t(\Pi, \Theta)$ from state f_t^{Π}, the rest of the jobs in f_t^{Π} will be dominated by the jobs in f_t^{Θ}. Therefore, when the distance is 0, Π is already ahead of Θ, in the sense that its state is dominated by that of Θ. When the distance is positive, Π is behind Θ because it needs to perform some positive number of jobs to get ahead. Figure 2.1 illustrates this interpretation.

We formalize the above statement as an invariance between two scheduling policies that can be stated in terms of the distance function and the dominance relation. This invariance will help us later to compare the cost of the policies.

Theorem 2.3.5. *For any two scheduling policies* Π *and* Θ*, we have*

$$f_t^{\Pi} - h(f_t^{\Pi}, \phi_t(\Pi, \Theta)) \preceq f_t^{\Theta}, \ \ \forall t = 1, 2, ..., T. \tag{2.3.3}$$

Proof. Recall that $s_t = f_{t-1}$, so equation (2.3.3) is equivalent to

$$s_t^{\Pi} - h(s_t^{\Pi}, \phi_{t-1}(\Pi, \Theta)) \preceq s_t^{\Theta}. \tag{2.3.4}$$

This equation (2.3.4) is clearly true for $t = 1$, as $s_1^{\Pi} = s_1^{\Theta}$ is the initial state.

Suppose that (2.3.4) holds up to period t, we next prove that it is also true for period $t + 1$.

In period t, after δ_t new jobs arrive, according to Lemma 2.3.4 we have

$$s_t^{\Pi} + \delta_t - h(s_t^{\Pi} + \delta_t, \phi_{t-1}(\Pi, \Theta)) \preceq s_t^{\Theta} + \delta_t.$$

Then after Θ removes $C_t + d_t^{\Theta}$ jobs, Lemma 2.3.4 gives us

$$s_t^{\Pi} + \delta_t - h(s_t^{\Pi} + \delta_t, \phi_{t-1}(\Pi, \Theta) + C_t + d_t^{\Theta}) \preceq s_t^{\Theta} + \delta_t - h(s_t^{\Theta} + \delta_t, C_t + d_t^{\Theta}).$$

Now we let $l = \phi_{t-1}(\Pi, \Theta) + d_t^{\Theta} - d_t^{\Pi}$ and rewrite the above equation as

$$s_t^{\Pi} + \delta_t - h(s_t^{\Pi} + \delta_t, l + C_t + d_t^{\Pi}) \preceq f_t^{\Theta}.$$

Depending on the value of l, there are two cases:

1. If $l < 0$, we have

$$s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, C_t + d_t^\Pi) \preceq s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, l + C_t + d_t^\Pi)$$

because the left hand side has more jobs removed from the vector $s_t^\Theta + \delta_t$. It is easy to check that the binary relation $\preceq$ is transitive, so the above equation leads to

$$s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, C_t + d_t^\Pi) \preceq f_t^\Theta$$

$$\Longrightarrow f_t^\Pi \preceq f_t^\Theta$$

$$\Longrightarrow f_t^\Pi - h(f_t^\Pi, 0) \preceq f_t^\Theta.$$

2. If $l \geq 0$, we have

$$\begin{aligned} s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, l + C_t + d_t^\Pi) &= s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, C_t + d_t^\Pi) \\ &\quad - h(s_t^\Pi + \delta_t - h(s_t^\Pi + \delta_t, C_t + d_t^\Pi), l) \\ &= f_t^\Pi - h(f_t^\Pi, l) \end{aligned}$$

$$\Longrightarrow f_t^\Pi - h(f_t^\Pi, l) \preceq f_t^\Theta.$$

In sum, we have

$$f_t^\Pi - h(f_t^\Pi, \max(l, 0)) \preceq f_t^\Theta$$

$$\Longrightarrow f_t^{\Pi} - h(f_t^{\Pi}, \phi_t(\Pi, \Theta)) \preceq f_t^{\Theta}$$

$$\Longrightarrow s_{t+1}^{\Pi} - h(s_{t+1}^{\Pi}, \phi_t(\Pi, \Theta)) \preceq s_{t+1}^{\Theta}.$$

Thus the theorem is proved. □

The above invariance also sheds light on the asymmetry of the distance function. The distance function $\phi_t(\Pi, \Theta)$ is merely a provable lower bound on the number of jobs that Π needs to perform to catch up to Θ. Many other lower bounds are possible. Since the invariance is a weak inequality, not an equality, it is not reversible. That is, $\phi_t(\Pi, \Theta) \neq -\phi_t(\Pi, \Theta)$ because a rearrangement of terms in the invariance does not produce the opposite inequality.

2.3.5 Partition of the horizon

The next theorem shows that the distance function separates all periods into two types, depending on the sign of the distance function. In one case, the current waiting cost incurred under policy Π is bounded by that under Θ. In the other case, the cumulative overtime cost incurred under Π is bounded by that under Θ.

Theorem 2.3.6. *In any period t,*

1. *if $\phi_t(\Pi, \Theta) = 0$, then $W_t^{\Pi} \leq W_t^{\Theta}$;*

2. *if $\phi_t(\Pi, \Theta) > 0$, let $t_0 = \max\{k : \phi_k(\Pi, \Theta) = 0, k < t\}$. Then*

$$\sum_{k=t_0+1}^{t} d_k^{\Pi} < \sum_{k=t_0+1}^{t} d_k^{\Theta}.$$

Proof. If $\phi_t(\Pi, \Theta) = 0$, we know from Theorem 2.3.5 that $f_t^{\Pi} \leq f_t^{\Theta}$. Then Lemma 2.3.3 gives $W_t^{\Pi} \leq W_t^{\Theta}$.

The case $\phi_t(\Pi, \Theta) > 0$ can be proved by directly checking the definition of the distance function. □

Using Table 2.1, we illustrate the two types of periods distinguished in Theorem 2.3.6.

1. Periods in which $\phi_t(\Pi, \Theta) = 0$. These are periods $t = 1, 4, 5, 8, 9$ in Table 2.1. From the first statement of Theorem 2.3.6 we know that for this type of periods, the waiting costs under Π is bounded by the waiting costs under Θ.

2. Periods in which $\phi_t(\Pi, \Theta) > 0$. We can divide these periods into intervals of consecutive periods, e.g., interval $[2, 3]$ and interval $[6, 7]$ in Table 2.1. During each of these intervals, the total number of overtime slots used in Π is no greater than the number of overtime slots used in Θ. Hence, in these intervals the total overtime cost under Π is bounded by that under Θ.

In sum, in any type of periods, one cost component of Π, either the waiting cost or the overtime cost, is bounded by the corresponding cost of Θ. Since Θ can be any scheduling policy including the optimal offline policy, Π will have a competitive ratio of 2 if it can balance the two cost components evenly in an online manner.

Similar to the distance function, a *potential function* is a commonly used mapping from the history of two policies to a scalar. By definition, the change in potential in each period must satisfy a generic inequality involving the one-period costs for the two policies. These inequalities can be simply summed to produce the competitive bound desired, if the potential

function also satisfies certain boundary conditions. The distance function, in contrast, is used to identify intervals in which OLN is ahead of OFF, and intervals in which OLN is behind OFF. In each type of interval, a cost of OLN is shown to be upper bounded by a cost of OFF. The intervals may be multi-period. The distance function does not satisfy the generic inequality required for potential functions. Its usage is also distinct from that of potential functions. The sign of the distance function is used (whether positive or 0) but not its numerical value.

2.3.6 Proof of performance

We will show that OLN has a competitive ratio of 2 by using the distance function above.

Theorem 2.3.7. *For any policy* Π *and any sample path,*

$$\max(\sum_{i=1}^{t} d_i^{OLN}, \sum_{i=1}^{t} W_i^{OLN}) \leq \sum_{i=1}^{t} (d_i^{\Pi} + W_i^{\Pi}), \ \ \forall t = 1, 2, ..., T. \tag{2.3.5}$$

Proof. When $t = 0$ the condition (2.3.5) is trivially true. Suppose that (2.3.5) is true up to period $t-1$. We next prove that it also holds for period t.

Let $g_t = \max(\sum_{i=1}^{t} d_i^{\mathrm{OLN}}, \sum_{i=1}^{t} W_i^{\mathrm{OLN}})$ be the maximum of the two cumulative costs up to period t.

- Case 1: $\phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - d_t^{\mathrm{OLN}} < 0$. We immediately have $d_t^{\mathrm{OLN}} > 0$ and

$$\phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - (d_t^{\mathrm{OLN}} - 1) \leq 0.$$

Then from Theorem 2.3.6 we know that

$$W(d_t^{\mathrm{OLN}} - 1) \leq W_t^{\Pi}.$$

In other words, even if we schedule one fewer job in period t under OLN, the resulting waiting cost for this period is still less than or equal to W_t^{Π}. The decision criterion for OLN in (2.3.1) gives us

$$g_t \leq g_{t-1} + W(d_t^{\mathrm{OLN}} - 1)$$

because otherwise using $d_t^{\mathrm{OLN}} - 1$ overtime slots instead of d_t^{OLN} in period t would reduce the maximum component of cumulative costs. Connecting the above two equations we get

$$g_t \leq g_{t-1} + W(d_t^{\mathrm{OLN}} - 1) \leq g_{t-1} + W_t^{\Pi} \leq \sum_{i=1}^{t-1}(d_i^{\Pi} + W_i^{\Pi}) + W_t^{\Pi} \leq \sum_{i=1}^{t}(d_i^{\Pi} + W_i^{\Pi}),$$

where the third inequality follows from induction on the $(t-1)$-th period.

- Case 2: $\phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - d_t^{\mathrm{OLN}} > 0$. Again let

$$t_0 = \max\{k : \phi_k(\mathrm{OLN}, \Pi) = 0, k < t\} \tag{2.3.6}$$

be the last period in which the distance function was equal to 0. Since in this case

$\phi_t(\mathrm{OLN}, \Pi) = \phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - d_t^{\mathrm{OLN}} > 0$, from Theorem 2.3.6 we know that

$$\sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} < \sum_{i=t_0+1}^{t} d_i^{\Pi}$$

$$\Longrightarrow \sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} + 1 \leq \sum_{i=t_0+1}^{t} d_i^{\Pi}.$$

On the other hand, definition (2.3.1) gives us

$$g_t \leq g_{t_0} + (\sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} + 1)$$

because otherwise we could use one more overtime slot to reduce g_t. Combining the above two equations we get

$$g_t \leq g_{t_0} + (\sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} + 1) \leq g_{t_0} + \sum_{i=t_0+1}^{t} d_i^{\Pi} \leq \sum_{i=1}^{t_0} (d_i^{\Pi} + W_i^{\Pi}) + \sum_{i=t_0+1}^{t} d_i^{\Pi} \leq \sum_{i=1}^{t} (d_i^{\Pi} + W_i^{\Pi}),$$

where the third inequality comes from induction on the t_0-th period.

- Case 3a: $\phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - d_t^{\mathrm{OLN}} = 0$, $g_t = \sum_{i=1}^{t} d_i^{\mathrm{OLN}}$. Let t_0 be defined as in (2.3.6). From the definition of the distance function we know that

$$\sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} = \sum_{i=t_0+1}^{t} d_i^{\Pi}.$$

Then we have

$$g_t \leq g_{t_0} + \sum_{i=t_0+1}^{t} d_i^{\mathrm{OLN}} \leq \sum_{i=1}^{t_0}(d_i^{\Pi} + W_i^{\Pi}) + \sum_{i=t_0+1}^{t} d_i^{\Pi} \leq \sum_{i=1}^{t}(d_i^{\Pi} + W_i^{\Pi}),$$

where the first inequality comes from the condition for this case, namely that $g_t = \sum_{i=1}^{t} d_i^{\mathrm{OLN}}$.

- Case 3b: $\phi_{t-1}(\mathrm{OLN}, \Pi) + d_t^{\Pi} - d_t^{\mathrm{OLN}} = 0$, $g_t = \sum_{i=1}^{t} W_i^{\mathrm{OLN}}$. From Theorem 2.3.6 we have

$$W_t^{\mathrm{OLN}} \leq W_t^{\Pi}$$

$$\implies g_t \leq g_{t-1} + W_t^{\mathrm{OLN}} \leq g_{t-1} + W_t^{\Pi} \leq \sum_{i=1}^{t-1}(d_i^{\Pi} + W_i^{\Pi}) + W_t^{\Pi} \leq \sum_{i=1}^{t}(d_i^{\Pi} + W_i^{\Pi}),$$

where the first inequality comes from the condition for this case, namely that $g_t = \sum_{i=1}^{t} W_i^{\mathrm{OLN}}$.

□

Finally using Theorem 2.3.7 we can show that OLN is 2-competitive, by letting Π be the optimal offline algorithm OFF.

Corollary 2.3.8. *On every sample path,*

$$\sum_{i=1}^{T}(d_i^{OLN} + W_i^{OLN}) \leq 2\sum_{i=1}^{T}(d_i^{OFF} + W_i^{OFF}).$$

Proof.

$$\sum_{i=1}^{T}(d_i^{\text{OLN}} + W_i^{\text{OLN}}) \leq 2\max(\sum_{i=1}^{T} d_i^{\text{OLN}}, \sum_{i=1}^{T} W_i^{\text{OLN}}) \leq 2\sum_{i=1}^{T}(d_i^{\text{OFF}} + W_i^{\text{OFF}}).$$

□

2.3.7 Generalization to Discounted Costs

Now we generalize our previous results to the case of discounted future costs. Given a discount factor $\gamma \in (0, 1)$, let the total discounted cost from period 1 to T be $V_T^{\Pi}(\gamma)$,

$$V_T^{\Pi}(\gamma) = \sum_{t=1}^{T}(d_t^{\Pi} p + \phi_t^{\Pi})\gamma^{t-1}.$$

The following theorem ensures that the competitive ratio of our online algorithm is still 2 in the discounted-cost case.

Theorem 2.3.9. *For any policy* Π *and any horizon* T*, where* T *is possibly infinite, we have*

$$V_T^{OLN}(\gamma) \leq 2V_T^{\Pi}(\gamma).$$

Proof. We already know from Corollary 2.3.8 that for any length t of the horizon and any sample path we have

$$V_t^{\text{OLN}} \leq 2V_t^{\Pi},$$

where V_t^{Π} is the undiscounted cost from periods 1 to t. Then for any policy Π,

$$\begin{aligned}
V_T^{\mathrm{OLN}}(\gamma) &= \sum_{t=1}^{T} (d_t^{\mathrm{OLN}} p + \phi_t^{\mathrm{OLN}}) \gamma^{t-1} \\
&= \sum_{t=1}^{T} (V_t^{\mathrm{OLN}} - V_{t-1}^{\mathrm{OLN}}) \gamma^{t-1} \\
&= \sum_{t=1}^{T-1} V_t^{\mathrm{OLN}} \cdot (\gamma^{t-1} - \gamma^t) + V_T^{\mathrm{OLN}} \cdot \gamma^{T-1} \\
&\leq \sum_{t=1}^{T-1} 2V_t^{\Pi} \cdot (\gamma^{t-1} - \gamma^t) + 2V_T^{\Pi} \cdot \gamma^{T-1} \\
&= 2V_T^{\Pi}(\gamma).
\end{aligned}$$

□

2.3.8 Lower Bounds

We prove that our online algorithm achieves the optimal competitive ratio by reducing our scheduling problem into a *ski-rental problem*, and concluding that the competitive ratios for the ski-rental problem apply to our model.

The classical ski-rental problem, which is first studied by Karlin et al. (1988), is a simplified version of our allocation-scheduling problem. In the ski-rental problem, a single job waits to be processed some time in the future, but the exact date that the job will be processed is unknown. A waiting cost of \$1 is incurred in each period that the job has to wait. The job can also be immediately processed at an additional cost of $\$B$ at any time. If we know that the job has to wait at least B periods, then it is optimal to immediately process

the job in the current period. If the job needs to wait no more than B periods, then it is optimal to let it wait. This ski-rental problem is online if the exact time that the job will be processed is unknown and is chosen by an adversary. It is well known that the optimal competitive ratio of the ski-rental problem is 2 for deterministic algorithms (Karlin et al., 1988) and $e/(e-1)$ for randomized algorithms (Karlin et al., 1990).

Theorem 2.3.10. *OLN is an optimal online algorithm for the allocation-scheduling model.*

Proof. In our allocation scheduling model, if there is only one job in the system and we always let $C_t = 0$ until some future period chosen by an adversary, then the problem reduces to the ski-rental problem. Thus, the ski-rental problem is a subclass of the allocation-scheduling problem. Therefore, its lower bounds on the competitive ratio also apply to the algorithms for the allocation-scheduling problem. From this, we can conclude that our 2-competitive deterministic algorithm has the lowest possible competitive ratio. □

2.4 Model of Allocation Scheduling with Cancellations

In this section, we consider the allocation-scheduling problem with cancellations. The online algorithm we propose in this section is adapted from the cost-balancing algorithm of the previous section. The algorithm in this section is a deterministic one. We will only prove the competitive ratio over an undiscounted and finite horizon, but similar to the results in Section 2.3.7, our competitive analysis can be easily generalized to a discounted and infinite horizon.

Starting from the model without cancellations, we assume that a class i job has a cancellation probability of $q_i \in [0, 1]$, and a cancellation cost of $r_i \geq 0$. We assume that the cancellation cost dominates the overtime cost for each class, i.e., $r_i \geq p = 1$ for all i. We further assume that the cancellation probabilities and costs are higher for higher-priority classes, i.e., $r_1 \geq r_2 \geq \cdots \geq r_n$, and $q_1 \geq q_2 \geq \cdots \geq q_n$. This assumption makes sense in most applications. In healthcare, higher priority patients have a higher need to be seen quickly, less willingness to wait, and higher tendency to leave for other care arrangements if they are made to wait for too long. In server applications, higher priority jobs have shorter deadlines. In service systems, higher priority customers are more impatient to wait, and often bring higher profits to the system which would be lost if they leave the queue.

In each period, the following events happen in sequence

1. At the beginning of period t, $s_t = (s_{t1}, s_{t2}, ..., s_{tn})$ is the total number of jobs in system, where s_{ti} is the number of jobs of class i.

2. Each job in class i independently leaves the system with probability q_i. The remaining jobs form a state m_t, $m_t \leq s_t$. The total cancellation cost incurred in period t is

$$R_t = (s_t - m_t)^\tau r.$$

3. The capacity C_t and new arrivals δ_t are observed. The system state becomes $m_t + \delta_t$.

4. The scheduling decision d_t for period t is made. The number of jobs remaining in the queue is f_t, $f_t = m_t + \delta_t - h(m_t + \delta_t, d_t + C_t)$. The overtime cost incurred in period t

is d_t, and the waiting cost incurred is

$$W_t = f_t^\tau w.$$

5. In the next period we have $s_{t+1} = f_t$.

For the competitive analysis of the online algorithm with cancellations, we assume that an offline algorithm sees future arrivals and capacities, δ_t, C_t, $t = 1, 2, ..., T$, but does not see which jobs will cancel. Let $\mathcal{F} = \sigma(\delta_1, \delta_2, ..., \delta_T, C_1, C_2, ..., C_T)$ contain the information that an offline algorithm can see. The objective is

$$\mathbf{E}[V_T|\mathcal{F}] = \mathbf{E}[\sum_{t=1}^{T}(R_t + d_t + W_t)|\mathcal{F}],$$

where the expectation is taken over the random cancellation events.

Before presenting the online algorithm, it is necessary to reexamine the question of, in the presence of job cancellations, whether it is still optimal for the offline algorithm to serve jobs with the highest priorities first, i.e., whether we can still use the $h(\cdot, \cdot)$ operator to represent an optimal offline scheduling decision. The following theorem ensures that this service rule is still optimal.

Theorem 2.4.1. *The optimal offline algorithm OFF always schedules jobs with the highest priorities in each period.*

Proof. As an offline algorithm, OFF knows all the arrivals and capacities upfront. However, since the cancellation events are exogenous to offline algorithms, OFF faces a stochastic

setting in which jobs cancel randomly in each period. In this stochastic decision process, let $u_t^1(s)$ be the expected cost of OFF from t to T when the system state at t is s. That is, let

$$u_t^1(s) = \mathbf{E}[\sum_{i=t}^{T}(R_i^{\text{OFF}} + d_i^{\text{OFF}} + W_i^{\text{OFF}})|\mathcal{F}, s_t = s].$$

Let $u_t^2(s)$ be the cost of OFF from t to T immediately after cancellations have occurred in period t, and when the system state at t is s,

$$u_t^2(s) = \mathbf{E}[d_t^{\text{OFF}} + W_t^{\text{OFF}} + \sum_{i=t+1}^{T}(R_i^{\text{OFF}} + d_i^{\text{OFF}} + W_i^{\text{OFF}})|\mathcal{F}, m_t = s].$$

We next show by induction that for any $s_1 \preceq s_2$,

$$u_t^1(s_1) \leq u_t^1(s_2) \tag{2.4.1}$$

$$\text{and } u_t^2(s_1) \leq u_t^2(s_2). \tag{2.4.2}$$

These two results will naturally lead to the proof of this theorem.

First, it is clear that (2.4.2) holds in the last period T, as no cancellation will ever happen starting at that time, and hence the result reduces to the case without cancellations. Suppose that (2.4.2) holds starting from period t. We next prove that (2.4.1) also holds for period t and that (2.4.2) holds for period $t-1$.

Let e_i be the unit vector with 1 for the ith element and 0 for all other elements. Since

adding more jobs to the system only imposes a larger cost, we must have

$$u_t^1(s) \leq u_t^1(s + e_i)$$

for any $i = 1, 2, ..., n$. Then to prove (2.4.1) it suffices to prove that for any $i < j$,

$$u_t^1(s + e_j) \leq u_t^1(s + e_i).$$

For any $\tilde{s} \leq s$, let $P(s, \tilde{s})$ be the probability that all the jobs in $\tilde{s}$ remain while all the jobs in $s - \tilde{s}$ cancel. Then the offline cost value can be written as

$$u_t^1(s + e_i) = \sum_{\tilde{s} \leq s} P(s, \tilde{s}) \left[(s - \tilde{s})^\tau r + q_i(r_i + u_t^2(\tilde{s})) + (1 - q_i) u_t^2(\tilde{s} + e_i) \right],$$

where $r_i + u_t^2(\tilde{s})$ is the total cost value under the condition that the additional job e_i cancels, and $u_t^2(\tilde{s} + e_i)$ is the cost value under the condition that the additional job does not cancel.

By induction we know that $u_t^2(\tilde{s} + e_j) \leq u_t^2(\tilde{s} + e_i)$ if $i < j$. Moreover, the marginal cost of $u_t^2(\cdot)$ must be bounded by the overtime cost, namely,

$$u_t^2(\tilde{s} + e_i) - u_t^2(\tilde{s}) \leq p \leq r_i$$

because otherwise the offline policy would service the additional job e_i by overtime and

reduce the marginal cost to p. Then for any $i < j$,

$$\begin{aligned} u_t^1(s+e_j) &= \sum_{\tilde{s}\le s} P(s,\tilde{s})\left[(s-\tilde{s})^\tau r + q_j(r_j + u_t^2(\tilde{s})) + (1-q_j)u_t^2(\tilde{s}+e_j)\right] \\ &\le \sum_{\tilde{s}\le s} P(s,\tilde{s})\left[(s-\tilde{s})^\tau r + q_j(r_i + u_t^2(\tilde{s})) + (1-q_j)u_t^2(\tilde{s}+e_i)\right] \\ &\le \sum_{\tilde{s}\le s} P(s,\tilde{s})\left[(s-\tilde{s})^\tau r + q_i(r_i + u_t^2(\tilde{s})) + (1-q_i)u_t^2(\tilde{s}+e_i)\right] \\ &= u_t^1(s+e_i). \end{aligned}$$

where the last inequality follows from the fact that $q_i > q_j$ and that $r_i + u_t^2(\tilde{s}) \ge u_t^2(\tilde{s}+e_i)$.

Thus we have proved (2.4.1) for period t. Now it is immediately clear that the optimal offline scheduling rule in period $t-1$ always services the jobs with the highest priorities, because (2.4.1) states that it is better to have lower priority jobs in the system at the beginning of period t, and that the costs to serve any two jobs are the same. It also follows that (2.4.2) holds for period $t-1$, as having lower-priority jobs in system leads to lower waiting costs and, at the same time, lower-priority jobs at the beginning of the next period.

□

2.4.1 Online algorithm and summary of main ideas

The extension of the online algorithm OLN that we develop in this section works similarly to the version we presented earlier. This algorithm OLN balances the waiting cost with and the sum of overtime costs and cancellation costs, by minimizing the maximum of the

cumulative cost components. Some redefinition of the costs is necessary. Therefore, we will defer a precise description of the online algorithm until we have described this redefinition.

Two ideas are necessary in the development of a performance bound. First we need to show that the dominance relation previously developed holds analogously under cancellations. In general, it needs not hold, but we will show that if cancellation events are stochastically coupled, then the dominance relation can be preserved. Second, we need to incorporate into our analysis a new cancellation cost. We do this by developing a new cost-accounting scheme. In this scheme, we treat a cancellation as a job that is forced to be served in overtime. With this change, we can apply the proof of the performance bound in the previous section with few changes.

2.4.2 Coupling of two scheduling policies

Our goal in this section is to compare the system states under two policies Π and Θ by redefining our distance function. Suppose that the policies start with the same initial state and experience the same capacity C_t and arrivals δ_t for each period t. Since both online and offline algorithms do not know which jobs will cancel, we can couple the cancellation events under Π and Θ. We show that a new distance function can be defined based on a coupling of cancellations.

Let $o_t^{\Pi} = \|s_t^{\Pi} - m_t^{\Pi}\|_1$ be the total number of canceled jobs in period t for policy Π. We

define a new distance function $\bar{\phi}(\Pi, \Theta)$ for any two policies Π and Θ as follows

$$\begin{cases} \bar{\phi}_0(\Pi, \Theta) = 0, \\ \bar{\phi}_t(\Pi, \Theta) = \max\{\bar{\phi}_{t-1}(\Pi, \Theta) - d_t^{\Pi} - o_t^{\Pi} + d_t^{\Theta} + o_t^{\Theta}, 0\} \quad \text{for } t \geq 1. \end{cases} \tag{2.4.3}$$

This new distance function takes both the number of overtime slots and the number of canceled jobs into account.

Suppose that at the beginning of period t we have

$$s_t^{\Pi} - h(s_t^{\Pi}, \phi_{t-1}(\Pi, \Theta)) \preceq s_t^{\Theta}. \tag{2.4.4}$$

Then we can always simulate the cancellations in period t in three phases as follows (see Figure 2.2 for an illustration):

1. Let the $\bar{\phi}_{t-1}(\Pi, \Theta)$ jobs with the highest priorities in state s_t^{Π}, i.e., those counted in $h(s_t^{\Pi}, \bar{\phi}_{t-1}(\Pi, \Theta))$, make their cancellation decisions.

2. Let $l = (\|s_t^{\Pi}\|_1 - \bar{\phi}_{t-1}(\Pi, \Theta))^+$ be the number of remaining jobs in state s_t^{Π} that have not made their cancellation decisions yet. Let $U_1, U_2, ..., U_l$ be i.i.d. $[0, 1]$ uniform random variables. For each of the l jobs, going from the highest priority to the lowest priority, if the ith job is in class j, let the ith job cancel if and only if $q_j \geq U_i$. Then, for the l jobs with the highest priorities in state s_t^{Θ}, let them cancel similarly, by using the same sequence of uniform random variables $U_1, U_2, ..., U_l$ (but using possibly different cancellation probabilities). In this way we have coupled the cancellation events between

the l jobs with the lowest priorities under Π and the l jobs with the highest priorities under Θ.

3. Let the other jobs in s_t^Θ make their cancellation decisions.

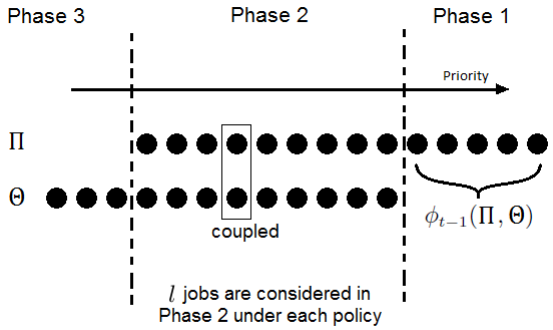


Figure 2.2: Stochastic coupling of cancellation events. After removing $\phi_{t-1}(\Pi, \Theta)$ jobs with the highest priorities from the state under Π, the remaining l jobs are dominated by the jobs under Θ, in that the priority of each remaining job under Π is at most that of the job with the same priority ranking under Θ. In Phase 2, the cancellation events of each pair of jobs having the same priority ranking under the two policies are coupled together.

The following theorem shows that, under the above coupling of cancellation events, the distance function still enables us to set up a dominance relationship between Π and Θ.

Theorem 2.4.2. *Suppose (2.4.4) holds in period t. After the above coupled cancellation process, we have on every sample path,*

$$m_t^\Pi - h(m_t^\Pi, \bar{\phi}_{t-1}(\Pi, \Theta) - o_t^\Pi + o_t^\Theta) \preceq m_t^\Theta. \tag{2.4.5}$$

In particular,

$$\bar{\phi}_{t-1}(\Pi, \Theta) - o_t^\Pi + o_t^\Theta \geq 0. \tag{2.4.6}$$

By the end of period t,

$$f_t^{\Pi} - h(f_t^{\Pi}, \bar{\phi}_{t-1}(\Pi, \Theta) - d_t^{\Pi} - o_t^{\Pi} + d_t^{\Theta} + o_t^{\Theta}) \preceq f_t^{\Theta}. \tag{2.4.7}$$

Proof. Recall that $l = (\|s_t^{\Pi}\|_1 - \bar{\phi}_{t-1}(\Pi, \Theta))^+$. Let x_i^{Π} and x_i^{Θ} be the vectors of jobs considered in the ith coupling phase under Π and Θ, respectively, for $i = 1, 2, 3$ (see Figure 2.2). In particular, we have in Phase 1,

$$x_1^{\Pi} = h(s_t^{\Pi}, \bar{\phi}_{t-1}(\Pi, \Theta)),$$

in Phase 2,

$$x_2^{\Pi} = s_t^{\Pi} - x_1^{\Pi} \quad \text{and} \quad x_2^{\Theta} = h(s_t^{\Theta}, l),$$

and in Phase 3,

$$x_3^{\Theta} = s_t^{\Theta} - x_2^{\Theta}.$$

Let $\bar{x}_i^{\Pi}$ and $\bar{x}_i^{\Theta}$ be the vectors of remaining jobs in x_i^{Π} and x_i^{Θ}, respectively, after cancellations have occurred. Let $y_i^{\Pi} = \|x_i^{\Pi}\|_1 - \|\bar{x}_i^{\Pi}\|_1$ and $y_i^{\Theta} = \|x_i^{\Theta}\|_1 - \|\bar{x}_i^{\Theta}\|_1$ be the number of canceled jobs in phase i under Π and Θ, respectively.

Under coupling, the ith job in x_2^{Π}, ranked by priority, is coupled with the ith job in x_2^{Θ}. According to the initial condition (2.4.4), we have $x_2^{\Pi} \preceq x_2^{\Theta}$, i.e., the ith job in x_2^{Π} has equal or lower priority than the ith job in x_2^{Θ}. According to the coupling process, if the ith job in

x_2^Π cancels, then the ith job in x_2^Θ cancels. So we must have

$$y_2^\Pi \leq y_2^\Theta.$$

Since $x_2^\Pi \preceq x_2^\Theta$, by removing $y_2^\Theta - y_2^\Pi$ jobs with the highest priorities from $\bar{x}_2^\Pi$, the resulting state must be dominated by $\bar{x}_2^\Theta$, i.e.,

$$\bar{x}_2^\Pi - h(\bar{x}_2^\Pi, y_2^\Theta - y_2^\Pi) \preceq \bar{x}_2^\Theta.$$

In phase 1, there are $\bar{\phi}_{t-1}(\Pi, \Theta) - y_1^\Pi$ jobs in $\bar{x}_1^\Pi$. Plugging these jobs into the dominance relation, we get

$$\bar{x}_1^\Pi + \bar{x}_2^\Pi - h(\bar{x}_1^\Pi + \bar{x}_2^\Pi, \bar{\phi}_{t-1}(\Pi, \Theta) - y_1^\Pi + y_2^\Theta - y_2^\Pi) \preceq \bar{x}_2^\Theta.$$

By further adding the jobs in phase 3, we get

$$\bar{x}_1^\Pi + \bar{x}_2^\Pi - h(\bar{x}_1^\Pi + \bar{x}_2^\Pi, \bar{\phi}_{t-1}(\Pi, \Theta) - y_1^\Pi + y_2^\Theta - y_2^\Pi + y_3^\Theta) \preceq \bar{x}_2^\Theta + x_3^\Theta,$$

which is just (2.4.5).

To prove (2.4.6), note that $\bar{\phi}_{t-1}(\Pi, \Theta) \geq y_1^\Pi$ because no more than $\bar{\phi}_{t-1}(\Pi, \Theta)$ jobs can cancel in phase 1, and $y_2^\Theta \geq y_2^\Pi$ due to the coupling process. Hence

$$\bar{\phi}_{t-1}(\Pi, \Theta) - o_t^\Pi + o_t^\Theta = \bar{\phi}_{t-1}(\Pi, \Theta) - y_1^\Pi + y_2^\Theta - y_2^\Pi + y_3^\Theta \geq \bar{\phi}_{t-1}(\Pi, \Theta) - y_1^\Pi + y_2^\Theta - y_2^\Pi \geq 0,$$

proving (2.4.6).

To see that (2.4.5) and (2.4.6) lead to (2.4.7), we treat m_t^{Π} and m_t^{Θ} as states in an intermediate period, and treat $\bar{\phi}_{t-1}(\Pi,\Theta) - o_t^{\Pi} + o_t^{\Theta} \geq 0$ as the distance function value for the intermediate period. Then (2.4.7) follows according to Theorem 2.3.5.

□

Corollary 2.4.3. *For any two scheduling policies* Π *and* Θ*, we have on every sample path,*

$$f_t^{\Pi} - h(f_t^{\Pi}, \bar{\phi}_t(\Pi,\Theta)) \preceq f_t^{\Theta}, \ \ \forall t = 1, 2, ..., T.$$

Proof. This statement is equivalent to equation (2.4.7), which is also the same as

$$s_{t+1}^{\Pi} - h(s_{t+1}^{\Pi}, \bar{\phi}_t(\Pi,\Theta)) \preceq s_{t+1}^{\Theta}.$$

This finishes the proof that (2.4.4) holds for all period t by induction. Therefore, (2.4.7) holds for all period t. □

In the remainder of this thesis chapter, we use the coupling of cancellations whenever we compare two scheduling policies. We will directly use Theorem 2.4.2 and Corollary 2.4.3 without each time specifying that cancellations are coupled.

2.4.3 New Cost-Accounting Scheme

Next we present a new cost-accounting scheme. The idea of the new cost-accounting scheme is to treat a cancellation as a job that is forced to be served in overtime, by moving around

some components of the cost of a cancellation. With this change, we can apply the proof of the performance bound in Theorem 2.3.7 with few changes.

The new cost-accounting scheme separates each cancellation cost into two parts: $r_i - p$ and p (recall that $r_i > p$ for each i). The online algorithm incorporates the two parts of the cancellation cost into the original waiting cost and overtime cost respectively. It achieves a competitive ratio of two by rebalancing the two components.

In the new cost-accounting scheme, let the new cancellation cost be $\tilde{r}_i = p = 1$ for all class i, and let the new waiting cost in period t for class i jobs be

$$\tilde{w}_{t,i} = \begin{cases} w_i + \gamma(r_i - 1)q_i & \text{for period } t < T, \\ w_i & \text{for period } t = T < \infty, \end{cases}$$

where γ is the discount factor. Note that the new waiting cost in the last period is different from that in other periods. It is also easy to check that $\tilde{w}_{t,1} \geq \tilde{w}_{t,2} \geq \cdots \geq \tilde{w}_{t,n}$, and thus $f_t^{\Pi} \preceq f_t^{\Theta}$ implies $\tilde{w}_t^{\tau} f_t^{\Pi} \leq \tilde{w}_t^{\tau} f_t^{\Theta}$ for all t.

Now the total waiting cost in period t is

$$\tilde{W}_t = f_t^{\tau} \tilde{w}_t.$$

And the total cost in period t can be written as

$$\tilde{\Omega}_t = (o_t + d_t) + \tilde{W}_t.$$

The following theorem states that the new cost-accounting scheme is equivalent to the original one.

Theorem 2.4.4. *For any horizon T, where T can be infinite, the total cost for any scheduling policy differs only by a constant between the original and new cost-accounting scheme.*

Proof. Let Ω_t and $\tilde{\Omega}_t$ be the cost incurred in period t under the original and new cost-accounting schemes, respectively. We have for any scheduling policy,

$$
\begin{aligned}
&E[\sum_{t=1}^{T}\gamma^{t-1}\Omega_t|\mathcal{F}] \\
&\quad= E[\sum_{t=1}^{T}\gamma^{t-1}((s_t-m_t)^\tau r + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= E[\sum_{t=1}^{T}\gamma^{t-1}((s_t-m_t)^\tau (r-\mathbf{1}) + \|s_t-m_t\|_1 + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= E[\sum_{t=1}^{T}\gamma^{t-1}((s_t-m_t)^\tau (r-\mathbf{1}) + o_t + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= E[\sum_{t=1}^{T}\gamma^{t-1}(E[(s_t-m_t)^\tau (r-\mathbf{1})|\mathcal{F}, s_t] + o_t + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= E[\sum_{t=1}^{T}\gamma^{t-1}(\sum_{i=1}^{n} s_{ti}q_i(r_i-1) + o_t + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= \sum_{i=1}^{n} s_{1,i}q_i(r_i-1) + E[\sum_{t=2}^{T}\gamma^{t-1}\sum_{i=1}^{n} s_{ti}q_i(r_i-1) + \sum_{t=1}^{T}\gamma^{t-1}(o_t + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= \sum_{i=1}^{n} s_{1,i}q_i(r_i-1) + E[\sum_{t=1}^{T-1}\gamma^{t-1}\sum_{i=1}^{n} f_{t,i}\gamma q_i(r_i-1) + \sum_{t=1}^{T}\gamma^{t-1}(o_t + d_t + f_t^\tau w)|\mathcal{F}] \\
&\quad= \sum_{i=1}^{n} s_{1,i}q_i(r_i-1) + E[\sum_{t=1}^{T}\gamma^{t-1}(o_t + d_t + f_t^\tau \tilde{w}_t)|\mathcal{F}] \\
&\quad= \sum_{i=1}^{n} s_{1i}q_i(r_i-1) + E[\sum_{t=1}^{T}\gamma^{t-1}\tilde{\Omega}_t|\mathcal{F}].
\end{aligned}
$$

Note that the second term on the last line is the total cost value under the new cost-accounting scheme, and the first term is a constant that depends only on the initial state.

□

This theorem implies that the optimal policy remains the same under the new cost-accounting scheme. Moreover, since the total cost value decreases by a constant $\sum_{i=1}^{n} s_{1i}q_i(r_i - 1)$ when new costs are applied, the online algorithms under the new cost-accounting scheme are also online algorithms for the original costs, with the same competitive ratios. We next construct the online algorithms under the new cost-accounting scheme.

2.4.4 Online algorithm and proof of performance

In the presence of cancellations, our online algorithm OLN balances the waiting cost $\tilde{W}_t^{\text{OLN}}$ and the sum of overtime cost and cancellation cost by minimizing the maximum of the cumulative cost components. Mathematically, let $W(d) = (m_t^{\text{OLN}} + \delta_t - h(m_t^{\text{OLN}} + \delta_t, d + C_t))^{\tau}\tilde{w}_t$ be the waiting cost to be incurred in period t if d overtime slots are used in t. Then d_t^{OLN} is determined by

$$d_t^{\text{OLN}} = \text{argmin}_d \ \max(\sum_{i=1}^{t} o_i^{\text{OLN}} + \sum_{i=1}^{t-1} d_i^{\text{OLN}} + d, \sum_{i=1}^{t-1} \tilde{W}_i^{\text{OLN}} + W(d)). \tag{2.4.8}$$

Theorem 2.4.5. *For any policy* Π *and any sample path,*

$$\max(\sum_{i=1}^{t}(o_i^{OLN} + d_i^{OLN}), \sum_{i=1}^{t} \tilde{W}_i^{OLN}) \leq \sum_{i=1}^{t}(o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}), \ \ \forall t = 1, 2, ..., T. \tag{2.4.9}$$

Proof. The proof is similar to the proof for the deterministic algorithm without cancellations. When $t = 0$ the condition (2.4.9) is trivially true. Suppose that (2.4.9) is true up to period $t-1$. We next prove that it also holds in period t.

Let $g_t = \max(\sum_{i=1}^{t} o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}), \sum_{i=1}^{t} \tilde{W}_i^{\mathrm{OLN}})$ be the maximum of the two cumulative costs up to period t.

- Case 1: $\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} + d_t^{\Pi} - o_t^{\mathrm{OLN}} - d_t^{\mathrm{OLN}} < 0$. According to Theorem 2.4.2 we have $\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} - o_t^{\mathrm{OLN}} \geq 0$. So we must have $d_t^{\mathrm{OLN}} > 0$ and

$$\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} + d_t^{\Pi} - o_t^{\mathrm{OLN}} - (d_t^{\mathrm{OLN}} - 1) \leq 0,$$

  which means that the distance function in period t will be 0 even if we schedule one fewer overtime slot. Then according to Corollary 2.4.3 we know that

$$W(d_t^{\mathrm{OLN}} - 1) \leq \tilde{W}_t^{\Pi}.$$

  On the other hand, the definition of OLN (2.4.8) gives us

$$g_t \leq g_{t-1} + W(d_t^{\mathrm{OLN}} - 1)$$

  because otherwise using $d_t^{\mathrm{OLN}} - 1$ overtime slots instead of d_t^{OLN} in period t would reduce the maximum component of cumulative costs. Connecting the above two equations we

get

$$g_t \le g_{t-1} + W(d_t^{\mathrm{OLN}} - 1) \le g_{t-1} + \tilde{W}_t^{\Pi} \le \sum_{i=1}^{t-1}(o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}) + \tilde{W}_t^{\Pi} \le \sum_{i=1}^{t}(o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}),$$

where the third inequality follows from induction on the $(t-1)$th period.

- Case 2: $\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} + d_t^{\Pi} - o_t^{\mathrm{OLN}} - d_t^{\mathrm{OLN}} > 0$. Let

$$t_0 = \max\{k : \bar{\phi}_k(\mathrm{OLN}, \Pi) = 0, k < t\} \tag{2.4.10}$$

be the last period for which the distance function equals 0. According to the definition of the new distance function we know that

$$\sum_{i=t_0+1}^{t}(o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) < \sum_{i=t_0+1}^{t}(o_i^{\Pi} + d_i^{\Pi})$$

$$\Longrightarrow \sum_{i=t_0+1}^{t}(o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) + 1 \le \sum_{i=t_0+1}^{t}(o_i^{\Pi} + d_i^{\Pi}).$$

On the other hand, definition (2.4.8) gives us

$$g_t \le g_{t_0} + \sum_{i=t_0+1}^{t}(o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) + 1$$

because otherwise we could use one more overtime slot to reduce g_t. Combining the

above two equations we get

$$\begin{aligned} g_t \leq & g_{t_0} + \sum_{i=t_0+1}^{t} (o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) + 1 \\ \leq & g_{t_0} + \sum_{i=t_0+1}^{t} (o_i^{\Pi} + d_i^{\Pi}) \\ \leq & \sum_{i=1}^{t_0} (o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}) + \sum_{i=t_0+1}^{t} (o_i^{\Pi} + d_i^{\Pi}) \\ \leq & \sum_{i=1}^{t} (o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}), \end{aligned}$$

where the third inequality comes from induction on the t_0-th period.

- Case 3a: $\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} + d_t^{\Pi} - o_t^{\mathrm{OLN}} - d_t^{\mathrm{OLN}} = 0$, $g_t = \sum_{i=1}^{t} (o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}})$. Let t_0 be defined as in (2.4.10). From the definition of the distance function we know that

$$\sum_{i=t_0+1}^{t} (o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) = \sum_{i=t_0+1}^{t} (o_i^{\Pi} + d_i^{\Pi}).$$

Then we have

$$g_t \leq g_{t_0} + \sum_{i=t_0+1}^{t} (o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}) \leq \sum_{i=1}^{t_0} (o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}) + \sum_{i=t_0+1}^{t} (o_i^{\Pi} + d_i^{\Pi}) \leq \sum_{i=1}^{t} (o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}),$$

where the first inequality comes from the condition of this case, namely that $g_t = \sum_{i=1}^{t} (o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}})$.

- Case 3b: $\bar{\phi}_{t-1}(\mathrm{OLN}, \Pi) + o_t^{\Pi} + d_t^{\Pi} - o_t^{\mathrm{OLN}} - d_t^{\mathrm{OLN}} = 0$, $g_t = \sum_{i=1}^{t} \tilde{W}_i^{\mathrm{OLN}}$. From Corollary

2.4.3 we have

$$f_t^{\mathrm{OLN}} \preceq f_t^{\Pi}$$

$$\Longrightarrow \tilde{W}_t^{\mathrm{OLN}} \leq \tilde{W}_t^{\Pi}$$

$$\Longrightarrow g_t \leq g_{t-1} + \tilde{W}_t^{\mathrm{OLN}} \leq g_{t-1} + \tilde{W}_t^{\Pi} \leq \sum_{i=1}^{t-1}(o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}) + \tilde{W}_t^{\Pi} \leq \sum_{i=1}^{t}(o_i^{\Pi} + d_i^{\Pi} + \tilde{W}_i^{\Pi}),$$

where the first inequality comes from the condition of this case, namely that $g_t = \sum_{i=1}^{t} \tilde{W}_i^{\mathrm{OLN}}$.

□

Finally using Theorem 2.4.5 we can show that OLN is 2-competitive in the new cost-accounting scheme, by letting Π be the optimal offline algorithm OFF.

Corollary 2.4.6. *On every sample path,*

$$\sum_{i=1}^{T}(o_i^{OLN} + d_i^{OLN} + \tilde{W}_i^{OLN}) \leq 2\sum_{i=1}^{T}(o_i^{OFF} + d_i^{OFF} + \tilde{W}_i^{OFF}).$$

Proof.

$$\begin{aligned}\sum_{i=1}^{T}(o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}} + \tilde{W}_i^{\mathrm{OLN}}) \leq & 2\max(\sum_{i=1}^{T}(o_i^{\mathrm{OLN}} + d_i^{\mathrm{OLN}}), \sum_{i=1}^{T}\tilde{W}_i^{\mathrm{OLN}}) \\ \leq & 2\sum_{i=1}^{T}(o_i^{\mathrm{OFF}} + d_i^{\mathrm{OFF}} + \tilde{W}_i^{\mathrm{OFF}}).\end{aligned}$$

□

Using Theorem 2.4.4, we can establish the performance guarantee of our cost-balancing algorithm in the original cost-accounting scheme.

Corollary 2.4.7.

$$E[\sum_{t=1}^{T}((s_t^{OLN} - m_t^{OLN})^\tau r + d_t^{OLN} + w^\tau f_t^{OLN})|\mathcal{F}] \\ \leq 2E[\sum_{t=1}^{T}((s_t^{OFF} - m_t^{OFF})^\tau r + d_t^{OFF} + w^\tau f_t^{OFF})|\mathcal{F}].$$

Proof.

$$\begin{aligned} &E[\sum_{t=1}^{T}((s_t^{\mathrm{OLN}} - m_t^{\mathrm{OLN}})^\tau r + d_t^{\mathrm{OLN}} + w^\tau f_t^{\mathrm{OLN}})|\mathcal{F}] \\ &\quad = \sum_{i=1}^{n} s_{1i} q_i (r_i - 1) + E[\sum_{t=1}^{T}((o_t^{\mathrm{OLN}} + d_t^{\mathrm{OLN}}) + \tilde{w}_t^\tau f_t^{\mathrm{OLN}})|\mathcal{F}] \\ &\quad \leq 2\sum_{i=1}^{n} s_{1i} q_i (r_i - 1) + 2E[\sum_{t=1}^{T}((o_t^{\mathrm{OFF}} + d_t^{\mathrm{OFF}}) + \tilde{w}_t^\tau f_t^{\mathrm{OFF}})|\mathcal{F}] \\ &\quad = 2E[\sum_{t=1}^{T}((s_t^{\mathrm{OFF}} - m_t^{\mathrm{OFF}})^\tau r + d_t^{\mathrm{OFF}} + w^\tau f_t^{\mathrm{OFF}})|\mathcal{F}], \end{aligned}$$

where the first and last equality follows from Theorem 2.4.4. □

Similar to the result in Section 2.3.8, we can show that in the allocation-scheduling model with cancellations, OLN has the best competitive ratio for any online algorithms.

Theorem 2.4.8. *OLN is an optimal online algorithm for the allocation-scheduling model with cancellations.*

Proof. Theorem 2.3.10 implies that 2 is an upper bound of the optimal competitive ratio for

the model with cancellations. Thus OLN is an optimal online algorithm for the model with cancellations. □

Note that we have allowed both demand and capacity to vary in this model because the algorithms we propose are flexible enough to deal with uncertainty on both sides. In systems with stable or predictable capacity, there is still uncertainty on the demand side and in the worst case, the theoretical performance guarantee remains the same. However, as the uncertainty reduces the practical performance of our algorithms will tend to improve.

2.5 Numerical Performance

In this section we test the numerical performance of our online algorithm OLN. We vary multiple parameters to test the sensitivity of its performance. We compare the results against those of an optimal offline policy, an optimal stochastic policy which knows about the distribution of future arrivals and capacities, and other heuristics. The results show that the gap between our online algorithm and the offline algorithm is within 16% in most cases when the total arrival rate is close to the daily capacity. The gap is larger when capacity exceeds demand or when demand exceeds capacity by a large amount. However, in such extreme cases certain naive policies perform very well and these policies should be used. For example, when there is a large surplus of capacity, no overtime resource should be used. When the number of arrivals is overwhelming, we always want to schedule as many jobs as possible in each period.

We present two sets of experiments. In the first set of experiments, we populate the

problem with synthetic parameters. In the second set of experiments, we simulate the algorithms on a real appointment-scheduling data set from Columbia University Medical Center. We report the *expected performance* of our algorithm and other heuristics under a range of parameters .

2.5.1 Experiments with synthetic data

We generate synthetic model parameters as follows. Our base test case is a two-class problem with parameters $w = (0.3, 0.1)$, $r = (2, 1.2)$ and $q = (0.1, 0.05)$. Arrivals are set to be stationary independent Poisson random variables with mean $\mathbf{E}[\delta_t] = (2, 3)$ for each period t. Capacity is constant $C_t = 5$ for all periods. The planning horizon has $T = 60$ periods and the discount factor is $\gamma = 0.95$. Each cost value is simulated by at least 10000 replicates. For each test case, we report the relative performance of the following five policies:

- V^{OLN} denotes the total cost of our online algorithm OLN.
- V^{OFF} denotes the optimal offline cost. Recall that OFF knows which jobs will arrive in every future period, but does not know which jobs will cancel in each period. In each period, OFF makes an optimal dynamic decision as to how many overtime slots to use knowing the cancellation probabilities of all jobs. OFF is computed using dynamic programming. In the presence of cancellations, computing OFF is costly due to the huge state space we need to store. Thus, we only report the performance of OFF when the total number of classes is small.
- V^{OPT} denotes the total cost of a stochastic optimal policy OPT that knows the dis-

tribution of future demands. OPT is computed using dynamic programming. We also report the performance of OPT only in small test cases.

- V^{OLN^*} denotes the total cost of a variant OLN* of our cost-balancing policy. OLN* balances two cost components by using a different balancing ratio. An optimal balancing ratio is chosen for each test case. More precisely, for each test case, we estimate the expected total cost of our online algorithm under various values of balancing ratios, and use a line search method to find a balancing ratio that locally minimizes the cost.

- V_c denotes the total cost of a cutoff policy. We define a cutoff policy $C(k)$ as follows. $C(k)$ performs k extra overtime slots per week. The i-th overtime slot, $i = 1, \ldots, k$, is always performed on the $(i \mod 5) - th$ day of the week. For each test case, we find the optimal cutoff value k^* and let V_c be the total cost of $C(k^*)$.

Table 2.2 shows the test results for different values of C_t. Our online scheduling policy OLN performs the best when demands and capacities are balanced, i.e., $C_t = \delta_1 + \delta_2 = 5$. It is interesting to notice that the gap between OPT and OFF is very small when C_t is below the total arrival rate, but increases to around 10% when C_t equals it. As mentioned earlier, this is because it is easy to carry out a near-optimal policy when C_t exceeds or is less than the total arrival rate by a large amount. Thus, it is most valuable and difficult to study the case when the daily capacity C_t is close to the expected daily number of arrivals $\|\delta\|_1$. The result of our online policy is satisfactory in such situations. Its gap against OPT is only around 5%.

Tables 2.3 to 2.5 show the results when parameters of the higher priority class are varied. Generally all the scheduling policies we consider are not sensitive to these parameters. This is because most often all the higher priority jobs are served by regular capacities and thus do not affect the overtime and waiting costs. In Tables 2.6 and 2.7, the waiting costs of both classes are varied. In these cases, the performance of all the scheduling policies change broadly. Nevertheless, the variant OLN* of our online policy always outperforms the cutoff heuristic.

In Table 2.8, we allow the arrivals to be non-stationary and test different patterns of arrival rates. In the case of cyclic arrivals, the arrival rates are set to be $(4, 7)$, $(1, 1)$ and $(1, 1)$ for every three consecutive periods. The case of declining arrivals has rates dropping from $(3, 5)$ to $(1, 1)$ linearly, whereas the case of growing rates has the reversed pattern. In all these cases, the gap of our online policy is within 30%, and the gap of its variant OLN* is smaller than that of the cutoff policy by as much as 5%. Table 2.9 shows the results when there are more priority classes. In these settings, the waiting costs are uniformly distributed between 0.8 and 0.2, the cancellation probabilities are uniformly distributed between 0.2 and 0.05, and the cancellation costs are uniformly distributed between 8 and 1. The arrival rates for all the classes are set to be the same with $C_t = \|\delta\|_1$. We compare cost values against the cutoff heuristic. Both OLN and OLN* outperform the cutoff policy. Moreover, the online policy performs better when there are more priority classes.

Table 2.2: Performance results under different values of C_t.

C_t	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
2	101.3%	137.1%	102.3%	102.3%
3	104.2%	135.2%	106.7%	107.6%
4	108.9%	128.0%	112.8%	120.3%
5	110.5%	115.5%	115.8%	118.1%
6	102.5%	115.3%	102.6%	102.6%
7	100.2%	116.5%	100.5%	100.2%

Table 2.3: Performance results under different values of w_1.

w_1	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
0.1	110.8%	115.4%	115.7%	117.6%
0.2	110.2%	115.5%	115.4%	117.4%
0.3	109.7%	115.6%	115.0%	117.3%
0.4	110.4%	115.6%	115.9%	118.5%
0.5	109.9%	115.6%	115.4%	118.5%
0.6	110.2%	115.6%	116.1%	119.2%
0.7	109.5%	115.7%	115.2%	118.9%
0.8	109.5%	115.7%	115.2%	119.5%
0.9	110.0%	115.6%	115.8%	120.5%

Table 2.4: Performance results under different values of q_1.

q_1	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
0.1	110.8%	115.5%	116.1%	118.5%
0.2	109.6%	115.5%	115.0%	117.5%
0.3	110.3%	115.6%	116.0%	118.7%
0.4	109.8%	115.6%	115.5%	118.6%
0.5	110.2%	115.8%	116.1%	119.4%
0.6	109.8%	115.7%	115.6%	119.4%
0.7	109.7%	115.7%	115.5%	119.7%
0.8	110.2%	115.7%	116.1%	120.7%
0.9	110.4%	115.7%	116.5%	121.3%

Table 2.5: Performance results under different values of r_1.

r_1	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
2	110.2%	115.6%	115.5%	117.8%
3	110.2%	115.5%	115.8%	118.2%
4	109.9%	115.7%	115.5%	118.4%
5	109.9%	115.6%	115.7%	118.8%
6	110.3%	115.6%	116.0%	119.7%
7	109.9%	115.7%	115.6%	119.8%
8	109.7%	115.6%	115.5%	120.0%

Table 2.6: Performance results when w_1 and w_2 are both increasing.

w_1	w_2	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
0.3	0.1	110.8%	115.5%	116.2%	118.5%
0.4	0.2	116.9%	127.4%	123.2%	133.9%
0.5	0.3	119.5%	132.9%	123.2%	129.3%
0.6	0.4	118.5%	136.7%	120.2%	121.2%
0.7	0.5	114.6%	138.7%	114.7%	114.7%
0.8	0.6	110.9%	141.6%	111.0%	111.0%
0.9	0.7	107.1%	143.0%	107.2%	107.2%

Table 2.7: Performance results when w_1 is increasing and w_2 is decreasing.

w_1	w_2	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
0.5	0.5	114.3%	138.8%	114.5%	114.7%
0.6	0.4	117.9%	136.7%	119.7%	120.6%
0.7	0.3	120.2%	132.9%	123.9%	130.0%
0.8	0.2	116.9%	127.4%	123.1%	134.3%
0.9	0.1	109.7%	115.7%	115.5%	120.2%

Table 2.8: Performance results when demand is non-stationary.

Demand Pattern	$V^{\mathrm{OPT}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}}/V^{\mathrm{OFF}}$	$V^{\mathrm{OLN}^*}/V^{\mathrm{OFF}}$	V_c/V^{OFF}
Cyclic	104.4%	111.7%	110.3%	111.0%
Declining	106.3%	130.2%	110.6%	115.9%
Growing	106.6%	121.5%	121.3%	121.6%

Table 2.9: Performance results under larger dimensions of state space.

Number of job classes	V^{OLN}/V_c	V^{OLN^*}/V_c
4	98.4%	91.6%
8	90.6%	88.0%

2.5.2 Experiments with Real Data

In this section, we test our cost balancing algorithm and other heuristics using a real data set that contains medical information of patients who seek treatment at Columbia University Medical Center (CUMC). The data is collected from the Congenital Heart Center at CUMC, and includes information on all children ≤ 21 years of age undergoing cardiac surgery in the pediatric operating rooms in 2014.

There are two types of surgeries, urgent and elective. We assume that urgent patients have a waiting cost of $w_1 = 10$ per day. We categorize elective patients according to their STAT categories. Following this classification method, elective patients are divided into five groups according to five different risk levels for mortality, with STAT Category 1 being the lowest in risk and STAT Category 5 being the highest in risk of mortality (Society of Thoracic Surgeons, 2016). We assume that higher-risk elective cases have higher waiting costs. This assumption is reasonable because these cases are elective cases that can afford to wait for some time. Among elective cases, higher risk tends to correspond to more serious health conditions that can be remedied by surgery, thus higher health costs if delayed. Thus, we assign the waiting costs of these categories to be $(w_2, w_3, ..., w_6) = (5, 4, 3, 2, 1)$. We vary the overtime cost over a range $[5, 32]$.

Most often, each OR can serve 2 patients on a given day during regular hours. There are two ORs in the medical center. On Tuesdays, Wednesdays and the 1st and 3rd Mondays of every month, two ORs are open and thus the regular capacity is 4. On Thursdays, Fridays

Table 2.10: Empirical performance of algorithms under different values of unit overtime cost.

Overtime Cost	V^{OLN}/V_c	V^{OLN^*}/V_c
5	162.50%	100.00%
8	148.22%	100.00%
11	136.48%	99.99%
14	126.57%	99.98%
17	118.29%	99.98%
20	110.88%	100.47%
23	104.62%	99.14%
26	98.70%	96.65%
29	93.61%	93.13%
32	89.35%	89.09%

and every 2nd and 4th Monday of every month, only one OR is open and thus the regular capacity is 2. The regular capacity is 0 during weekends.

We estimate directly from data the arrival rates of urgent patients as a function of the day of week. For elective patients, however, since the data record only the dates of surgery, but not of arrival, we estimate the arrival rates based on the daily number of scheduled surgeries, assuming that the arrival rates are constant over weekdays. We simulate a horizon of 60 days for 1000 replicates in each test case. We do not consider cancellations in this scenario.

We report the results in Table 2.10. We can see that OLN performs better than the cutoff policy when the overtime cost is large, which is the case in real settings. Furthermore, if the right balancing ratio is chosen, OLN* always outperforms the cutoff policy.

2.6 Conclusions

We study an important resource allocation model in which arriving jobs wait in a priority queue to be allocated to resources that renew in every period. A scheduling algorithm determines how many additional resources to purchase in each period at a higher cost in order to reduce the length of the queue. The model generally captures the tradeoff between reducing queue length and minimizing overtime/outsourcing cost.

We propose cost-balancing algorithms that achieve optimal competitive ratios for this problem. Our analysis involves constructing a novel *distance function* which captures the difference between the decisions made by two arbitrary algorithms in the past. By applying the distance function $\phi_t(\text{OLN}, \text{OFF})$ to an online algorithm OLN and an offline algorithm OFF, we obtain the dynamic difference in the number of scheduled and canceled jobs between the two algorithms. This property of the distance function directly leads to the proof competitive ratios of our cost-balancing algorithms.

It is extremely easy to implement our scheduling algorithms as their decisions only depend on cumulative cost values in the past. We test our algorithms and their variants in the appointment scheduling application where the goal is to minimize the sum of patient waiting cost and total overtime cost. Although our online algorithms are designed to optimize their performance in the worst scenario, we find that they achieve satisfactory performance for average cases as well. Moreover, we find that one of the variants of our algorithms, which picks the best cost-balancing ratio for each setting, has near-optimal performances for all scenarios. The managerial implication is that, if the decision maker is able to predict the

correct ratio of demand and capacity in future, a simple balancing algorithm that makes no other assumption about future events can achieve near-optimal performance.

Chapter 3

Online Advance Admission Scheduling for Services with Customer Preferences

3.1 Introduction

We study advance admission scheduling decisions in service systems. Advance admission scheduling decisions are those that determine specific times for customers' arrival to a facility for service. Advance admission scheduling is used in many service industries. Airlines reserve flight seats for those who purchase flight tickets. Restaurants reserve tables for customers who call in advance. Healthcare facilities reserve appointment slots for patients who request them. Advance admission scheduling enables service providers to better match capacity with demand because they control customers' actual arrivals to service facilities.

We formulate and analyze a resource-allocation model that generally captures such admission scheduling systems. Our model is essentially an online weighted bipartite matching problem. The resources in our model, when partitioned into units, can be seen as nodes on one side of a bipartite graph. All the customers correspond to nodes on the other side that are arriving online. The type of each arriving customer is determined by a time-varying distribution. Our model can be found in many applications in Operations Research. We summarize three applications related to revenue management below.

Single-leg revenue management. A special case of our model is the classic single-leg revenue-management problem in which all resources to be allocated are available at the same time. Customers who bring a higher benefit correspond to higher-fare classes. The decision is how to admit or reject customers, given the time remaining until the flight and the current inventory of available seats.

Ad allocation. In a typical display-ad allocation problem, e-commerce companies aim at tailoring display ads for each type of customer. Each ad, which corresponds to a unit of a resource, is often associated with a maximum number of times to be displayed. Knowing the arrival rates of future customers, the task is to make the most effective matching between ads and customers.

Management of opaque products. Internet retailers such as Hotwire or Priceline often offer a buyer an under-specified or *opaque product*, such as a flight ticket, with certain details such as the exact flight timing or the name of the airline withheld until after purchase. This enables the retailer to more flexibly manage their inventory.

Opaque products are often sold at a discount compared to specific products, making them attractive to wider segments of the market. These products are common in internet advertising, tour operations, property management (Gallego et al., 2004) and e-retailing. Customers purchase an opaque product if the declared characteristics fit their preferences. The buyer agrees to accept any specific product that meets the opaque description. In our model, a specific product corresponds to a node on the right side of a bipartite graph. A unit of demand for an opaque product corresponds to a node on the left that connects to all of the specific products contained in the opaque product. The weight of an edge corresponds to the revenue earned by selling the opaque product. Demands arrive randomly over time. We assume that demand for each opaque product is exogenous and independent of the availability of other products. This follows the traditional assumption of independent demand for revenue-management problems. When demand occurs, a decision is made to assign a specific product to that demand unit. Knowing the arrival rates of all demands, we want to maximize the total expected revenue by strategically assigning specific products.

For concreteness, we explain further details of our model based on an actual appointment scheduling problem in the healthcare context. Take the example of MyChart, a digital admission-scheduling application developed by Epic System. Epic is an electronic medical records company that is managing the records of millions of health care providers and more than half of the patient population in the U.S. (Husain, 2014). Epic deploys MyChart to perform online scheduling of appointments through internet portals. The use of applications

like MyChart is part of a general trend in healthcare towards providing electronic access to service through web and mobile applications (Publication, 2015).

When a patient schedules an appointment over a web portal, MyChart first asks the patient for the type of visit desired, whether it is for a physical exam, a consultation, a flu shot, etc. Next, it asks for the beginning and end of the range of preferred dates. It then shows a menu with a check box for morning and afternoon session for each day in the preferred date range. Patients can select one or more preferred sessions. Finally, MyChart either offers the patient one or more appointments, or states that no appointment can be found. We can conceive of many variations over this basic interface.

Consider the following model of advance admission scheduling that captures MyChart as an example. Consider a continuous time, finite horizon. There are multiple service providers. Each provider offers a number of service sessions over the horizon, some in regular hours and some in overtime. We call a session associated with a single provider a *resource.* There are n resources available over the horizon. All the resources are known. Each resource j can serve C_j customers. We call C_j the *capacity* of resource j. Each resource j must be booked by time t_j or it perishes at time t_j. There are m customer *types*. Patients of type i, $i = 1, \ldots, m$, arrive according to some known non-homogeneous Poisson process and make reservations through any of the modes made available by the provider, web, phone, or mobile. A patient of type i generates a *benefit* of r_{ij} when served with a unit of resource j. We assume that the type of customer can be observed at the time that they arrive to make an appointment, through the pattern of preferences that they indicate and any data stored in the system on their profiles. We require that customers arriving at time t have weight 0

for all resources j that perish at time $t_j < t$. The number of customer types can be kept finite by discretizing the horizon but this number can be very large. We will discuss this point shortly. When a customer arrives, a unit of an available resource must be assigned to her, or she must be rejected. Each unit of a resource can be assigned to at most one customer. We allow no-shows and the practice of overbooking to compensate for the effect of no-shows. The objective of the problem is to allocate the resources to the customers to maximize the expected total benefit of the allocation.

A salient feature of our model is that the resources may be perishable. This feature makes the model especially appropriate for service applications. More specifically, each resource may be associated with a known expiry date that falls within or beyond the horizon. The way we capture an expiry date for a specific resource is to make that resource infeasible for all customer types that arrive after its expiry date. That is, to capture the perishability of resources, we equivalently force the composition of customer types that arrive over time to change over time. For this reason, the non-stationarity of arrivals in our model is of especial importance.

Another significant advantage of non-stationary arrivals is the ability to better capture real applications. In real applications, demands can be highly non-stationary, changing with the time of day, time of week, seasons and longer-term trends (Huh, Liu and Truong 2012). Kim and Whitt (2014) have shown, for example, that call-center and hospital demands are well-modeled by non-homogeneous Poisson processes. For a problem that essentially aims to match demand with supply over time, capturing this non-stationarity in demand arrivals can lead to significant improvements in performance over stationary models.

3.2 Literature Review

3.2.1 Revenue Management

Our work is related to the revenue-management literature. We refer to Talluri and Van Ryzin (2004) for a comprehensive review of this literature. Traditional works in this area assume that demands for products are exogenous and independent of the availability of other products (Lautenbacher and Stidham, 1999; Lee and Hersh, 1993; Littlewood, 1972). The decision is whether to admit or reject a customer upon her arrival. Our model reduces to this admission-control problem in the special case that the resources are identical and are available at the same time.

When customers are open to purchase one among a set of different resources, our model controls which resource to assign to each customer. Thus, our model captures the problem of managing opaque products. Sellers of an opaque product conceal part of the products' information from customers. Sellers have the ability to select which specific product to offer after the purchase of opaque product. Previous works related to opaque products include Gallego and Phillips (2004), Fay and Xie (2008), Petrick et al. (2010), Chen et al. (2010), Lee et al. (2012), Gönsch et al. (2014) and Fay and Xie (2015). Due to the problem of large state space, most analyses focus on models with very few product types. For systems with many product types, some pricing and allocation heuristics are known. There is numerical evidence that much of the benefit of opaque products can be obtained by having two or three alternatives (Elmachtoub and Wei, 2013). However, when a retailer has a large number of alternative products, it is unclear how to design such an opaque product. Our work is

the first to study online allocation policies with constant performance guarantees for the management of an opaque product with an arbitrary number of alternatives.

Our model assumes independent demands, i.e., the demand for each product is exogenous and independent of the availability of other products. Many recent works in revenue management consider endogenous demands, which means that customers who find their most preferred product unavailable might turn to other products. Examples of works on dependent demands include Gallego et al. (2004), Zhang and Cooper (2005), Liu and van Ryzin (2008) and Gallego et al. (2015). One of the main characteristics of these models is that customer preferences cannot be observed until purchase decisions are made. In such situation, sellers only have a distributional information of customer preferences. This phenomenon does not apply to admission scheduling systems. In these systems, customer preference can be revealed before a unit of a resource is assigned. In MyChart, for example, the system is able to customize the appointment to offer to each patient after knowing the patient's profile and availability. We assume that each customer's preference is observed before a resource is assigned. Knowledge of preferences gives service providers the ability to improve the efficiency of the resource-allocation process by tailoring the service offered to each customer.

The work in this chapter is also related to the still limited literature on designing robust policies for revenue management, which we reviewed in Section 2.2.5. In this area, the most related work to this thesis chapter is Ball and Queyranne (2009). They analyze online algorithms for the single-leg revenue-management problem using the definition (1.1.1) of competitive ratio. They prove that this competitive ratio cannot be bounded by any constant when there are arbitrarily many customer types. In our work, we focus on the average-case

competitive ratio (1.1.2), and show that our algorithms have such competitive ratios bounded by a constant for any number of customer types and for a more general multi-resource model.

3.2.2 Appointment Scheduling

Our work is related to the literature on appointment scheduling, which we reviewed in detail in Section 2.2.1. We contribute to this literature by proposing the first online scheduling policy with performance guarantees for a very general multi-class advance-scheduling problem.

Our model captures the preferences of patients in a general way. Patient preferences are an important consideration in most out-patient scheduling systems. In the literature considering patient preferences, Gupta and Wang (2008) considers a single-day scheduling model where each arriving patient picks a single slot with a particular physician, and the clinic accepts or rejects the request. Our model can be seen as a multi-period generalization of their work. We also characterize the theoretical performance in an online setting, whereas they use stochastic dynamic programming as the modeling framework and develop heuristics. Feldman et al. (2014) study how to offer sets of open appointment slots to a stream of arriving patients over a finite horizon of multiple days, given that patients have preferences for slots that can be captured by the multinomial logit model. Their work is strongly influenced by assortment-planning problems. An important observation, which was first made by Gupta and Wang (2008), is that there is a fundamental difference between many advance admission-scheduling problems and assortment-planning problems. In admission scheduling, we can often work with revealed preferences, whereas in assortment-planning problems, decisions

are made with knowledge only of a distribution of customer preferences. Working with revealed preferences allows for a more efficient allocation of service compared to working with opaque preferences. It also leads to more analytically tractable models.

3.2.3 Online Resource Allocation

Our work is also closely related to works on online matching problems. Traditionally, the online bipartite matching problem studied by Karp et al. (1990) is known to have a best competitive ratio of 0.5 for deterministic algorithms and $1 - 1/e$ for randomized algorithms. For the online weighted bipartite matching problem that we consider, the worst-case competitive ratio cannot be bounded by any constant. Many subsequent works have tried to improve performance ratios under relaxed definitions of competitiveness.

Specifically, three types of assumptions are commonly used. The first type of assumption is that each demand node is independently and identically (i.i.d.) picked from a *known* set of nodes. Under this assumption, Jaillet and Lu (2013); Manshadi et al. (2012); Bahmani and Kapralov (2010); Feldman et al. (2009) propose online algorithms with competitive ratios higher than $1 - 1/e$ for the cardinality matching problem, in which the goal is to maximize the total number of matched pairs. Haeupler et al. (2011) study online algorithms with competitive ratios higher than $1 - 1/e$ for the weighted bipartite matching problem. Our definition of competitive ratio is the same as theirs. Our model is also similar, but we allow a more general arrival process of demand nodes in which the distribution of nodes can change over time. Previous analyses depend crucially on the fact that demand nodes are i.i.d. in order to simplify the expression for the probability that any demand node is matched to any

resource node. The expression becomes much more complex, and the arguments break down in the case that demand arrivals are no longer i.i.d.

The second type of assumption is that the sequence of demand nodes is a random permutation of an *unknown* set of nodes. This random permutation assumption has been used in the secretary problem (Kleinberg, 2005; Babaioff et al., 2008), adword problem (Goel and Mehta, 2008) and the bipartite matching problem (Mahdian and Yan, 2011; Karande et al., 2011). Kesselheim et al. (2013) study the weighted bipartite matching problem with extension to combinatorial auctions. Our work is different from all of these in that the non-stationarity of arrivals in our model cannot be captured by the random permutation assumption.

The third type of assumption made is that each demand node requests a very small amount of resource. The combination of this assumption and the random-permutation assumption often leads to polynomial-time approximation schemes (PTAS) for problems such as adword (Devanur and Hayes, 2009), stochastic packing (Feldman et al., 2010), online linear programming (Agrawal et al., 2014), and packing problems (Molinaro and Ravi, 2014). Typically, the PTAS proposed in these works use dual prices to make allocation decisions. Under this third assumption, Devanur et al. (2011) study a resource allocation problem in which the distribution of nodes is allowed to change over time, but still needs to follow a requirement that the distribution at any moment induce a small enough offline objective value. They then study the asymptotic performance of their algorithm. In our model, the amount capacity requested by each customer is not necessarily small relative to the total

amount of capacity available. Therefore, the analysis in these previous works does not apply to our problem.

In our model, the arrival rates, or the distribution of demand nodes, are allowed to change over time. This non-stationarity poses new challenges, because it cannot be analyzed with existing methods. At the same time, it is an essential feature in our model because it allows us to capture the perishability of service capacity in the applications that we consider. When a resource perishes within the horizon, the demand for that resource drops to 0. Such a demand process must be time-varying. This important feature has received only limited attention so far. Ciocan and Farias (2012) consider an allocation model with a very general arrival process, but their allocation policy has performance guarantee only when the arrival rates are uniform. In this thesis chapter, we allow arrival processes to be non-homogeneous Poisson processes with arbitrary rates.

Our algorithms solves a linear program and uses its optimal solution to make matching decisions. The idea of using optimal solutions to a linear program is natural and has been used by several previous works mentioned above. For example, Feldman et al. (2009), Manshadi et al. (2012), Haeupler et al. (2011), and Kesselheim et al. (2013) have used similar algorithms to obtain constant competitive ratios, albeit for different demand models.

The paper of Alaei, Hajiaghayi and Liaghat (2012) solves an online matching problem with non-stationary arrivals in a discrete-time setting. They propose an algorithm similar to one of our primitive algorithms, called the Separation Algorithm. They prove that this algorithm achieves a competitive ratio of at least $1-\frac{1}{\sqrt{k-3}}$ and at most approximately $1-\frac{1}{\sqrt{2\pi k}}$, where k is the minimum capacity of a resource. Compared to Alaei, Hajiaghayi and Liaghat

(2012), we prove a stronger lower bound of $\max(1/2, 1-\sqrt{\frac{2}{\pi k}}+O(\frac{1}{k}))$ on the competitive ratio for our Separation Algorithm, using a few of the same ideas but largely different techniques, as we will elaborate on in Section 3.5. Thus, our lower bound is more similar to their upper bound. We also point out that the Separation Algorithm is not practical because it might reject high-priority customers, while there are still other open resources. More importantly, because of randomization, it might reject a high-priority customer, but accept a low-priority customer at nearly the same time. For this reason, we propose a new "bid-pricing" algorithm, based on the Separation Algorithm, that avoids all of the above problems. We prove that the improved algorithm has the same theoretical performance guarantee, and has much better computational performance as tested on real data.

3.3 Problem Formulation

3.3.1 Model

There are n resources known to be available at specific instants over a continuous horizon $[0, 1]$. There are m customer types. Customers of type i randomly arrive over the horizon according to a known non-homogeneous Poisson process with rate $\lambda_i(t)$, for $t \in [0, 1]$. Let $\Lambda_i \equiv \int_0^1 \lambda_i(t)\, dt$ be the expected total number of arrivals of type-i customers.

Each resource j has a capacity of C_j units. Each resource j must be booked by time t_j or it perishes at time t_j.

When a customer arrives, one unit of capacity of an available resource must be assigned to the customer, or the customer must be rejected. A customer of type i earns a benefit

r_{ij} if assigned to resource j. The objective is to allocate the resources to the customers to maximize the expected total benefit from all of the allocated resources.

3.3.2 Definition of Competitive Ratios

We apply the definition (1.1.2) of competitive ratio. Let δ_i be the actual total number of arrivals of type i customers. We must have $\mathbf{E}[\delta_i] = \Lambda_i$. An offline algorithm knows $\delta = (\delta_1, \delta_2, ..., \delta_m)$ at the beginning of the horizon. Let $\text{OPT}(\delta)$ be the optimal offline benefit given the number of arrivals δ. Note that an optimal offline algorithm essentially solves a maximum weighted matching problem, between the customers and resources. An online algorithm, however, does not know the entire sample path of future arrivals, but only knows the arrival rates $\lambda_i(t),\ i = 1, 2, ..., m$. We aim at designing online algorithms with total benefit ALG with bounded competitive ratio $\frac{\mathbf{E}[\text{ALG}]}{\mathbf{E}[\text{OPT}(\delta)]}$, where the expectation is taken over the random arrivals δ.

3.4 Online Resource Allocation Algorithms

Computing an optimal dynamic allocation policy for our problem by dynamic programming is intractable due to the curse of dimensionality. Let $c_j(t)$ indicate the amount of resource j that remains at time t. Let $V(c(t), t)$ be the optimal expected total benefit to go starting with state $c(t) = (c_1(t), c_2(t), ..., c_n(t))$ and at time t.

In this section, we propose two online algorithms that approximate the total benefit

$V(c(t), t)$ by a sum of single-variable functions

$$\sum_{j=1}^{n} f_j(t, c_j(t)),$$

where $f_j(t, c_j(t))$ is a *benefit function* that approximates the optimal benefit that can be obtained from resource j from time t to the end of the horizon.

The first algorithm, which we call the *Separation Algorithm*, separates and optimizes the decisions for each single resource. We show that the Separation Algorithm is 0.5-competitive using a simple and innovative Lagrangian duality approach. We will further show that if k is the minimum capacity of any resource, then the competitive ratio of the Separation Algorithm can be improved to $\max(\frac{1}{2}, 1 - \sqrt{\frac{2}{\pi k}} + O(\frac{1}{k}))$. Finally, we prove that 0.5 is the best possible constant competitive ratio for our model.

The second algorithm, which we call the *Marginal Allocation Algorithm*, improves on the Separation Algorithm by converting it to a bid-price algorithm, which can be easily applied to a real admission scheduling system.

Before presenting the two online algorithms, we first characterize an optimal offline algorithm and an upper bound on the optimal offline benefit.

3.4.1 Offline Algorithm and Its Upper Bound

In the offline case, the total number of arrivals δ_i of each customer type i is known, and the exact arrival time is irrelevant. Given the δ_i's, the maximum offline benefit $\mathrm{OPT}(\delta)$ is given

by a maximum weighted matching problem, which can be formulated as the following LP:

$$\begin{aligned} \text{OPT}(\delta) = \max \ & \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} r_{ij} \\ \text{s.t.} \ & \sum_{j=1}^{n} x_{ij} \le \delta_i, \quad \text{for } i = 1, 2, ..., m \\ & \sum_{i=1}^{m} x_{ij} \le 1, \quad \text{for } j = 1, 2, ..., n \\ & x_{ij} \ge 0, \quad \text{for } i = 1, 2, ..., m; \ j = 1, 2, ..., n. \end{aligned} \tag{3.4.1}$$

where the decision x_{ij} is the number of type-i customers who are assigned to resource j. Let $\bar{x}(\delta)$ be an optimal solution to this LP. Then $\text{OPT}(\delta) = \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}\bar{x}_{ij}(\delta)$.

We are interested in finding an upper bound on the expected optimal offline benefit $\mathbf{E}[\text{OPT}(\delta)]$. We next show that LP (3.4.2), which uses $\mathbf{E}[\delta]$ instead of δ as the total demand, gives such an upper bound:

$$\begin{aligned} \max \ & \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} r_{ij} \\ \text{s.t.} \ & \sum_{j=1}^{n} x_{ij} \le \Lambda_i, \quad \text{for } i = 1, 2, ..., m \\ & \sum_{i=1}^{m} x_{ij} \le 1, \quad \text{for } j = 1, 2, ..., n \\ & x_{ij} \ge 0. \end{aligned} \tag{3.4.2}$$

Theorem 3.4.1. *The optimal objective value of (3.4.2) is an upper bound on* $\mathbf{E}[OPT(\delta)]$.

Proof. Since $\sum_{j=1}^{n} \bar{x}_{ij}(\delta) \le \delta_i$ and $\sum_{i=1}^{m} \bar{x}_{ij}(\delta) \le 1$, we must have $\sum_{j=1}^{n} \mathbf{E}[\bar{x}_{ij}(\delta)] \le \mathbf{E}[\delta_i] = \Lambda_i$ and $\sum_{i=1}^{m} \mathbf{E}[\bar{x}_{ij}(\delta)] \le 1$. Thus, $\mathbf{E}[\bar{x}_{ij}(\delta)]$ is a feasible solution to the LP (3.4.2). It follows

that the optimal objective value of (3.4.2) is an upper bound on

$$\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}\mathbf{E}[\bar{x}_{ij}(\delta)] = \mathbf{E}[\mathrm{OPT}(\delta)].$$

Similar techniques have been used in revenue management to prove similar results (Gallego and Van Ryzin, 1997). □

3.4.2 Separation Algorithm and Constant Competitive Ratio

The Separation Algorithm works by solving the LP (3.4.2) once, routing the customers to the resources according to an optimal solution to the LP (3.4.2). Then, for each resource separately, the algorithm optimally controls the admission of customers who have been routed to that resource. Using the LP information with respect to the expected number of arrivals (or sometimes, an estimate of the expected number of arrivals) is natural and has been used in several previous results (for example, Feldman et al. (2009), Manshadi et al. (2012), Haeupler et al. (2011), and Kesselheim et al. (2013)).

For the rest of this section, we assume without loss of generality that the capacity of each resource is 1. Our bound and algorithm will be independent of the capacity. In the next section, we shall show that the algorithm and bound can be improved as the minimum capacity increases beyond 1.

Let x^* be an optimal solution to the linear program (3.4.2). Whenever a customer of type i arrives, the Separation Algorithm randomly and independently picks a candidate resource $j \in \{1, 2, ..., n\}$ with probability x^*_{ij}/Λ_i, regardless of the availability of resources. We say

that this customer is *routed* to resource j. Then based on a further decision, the algorithm may either assign resource j to the customer or reject the customer.

According to the Poisson thinning property, the arrival process of type-i customers who will be routed to resource j is a non-homogeneous Poisson process with rate

$$\lambda_{ij}(t) \equiv \lambda_i(t) x^*_{ij}/\Lambda_i, \quad \text{for } 0 \le t \le 1. \tag{3.4.3}$$

Viewing the random routing process as exogenous, each resource j receives an independent arrival process with split rate $\lambda_{ij}(t)$ from each customer type i. Then for each resource j, the Separation Algorithm optimally controls the admission of customers who are routed to resource j. That is, when a type-i customer is routed to resource j at time t, the algorithm compares the benefit r_{ij} of this customer with the optimal expected future benefit $f_j(t)$ that can be earned from customers who will be routed to resource j after time t. The algorithm assigns resource j to the customer if r_{ij} is greater than $f_j(t)$. Given the split rates $\lambda_{ij}(t)$'s, the optimal expected future benefit $f_j(t)$ can be computed by solving the well-known Hamilton-Jacobi-Bellman equation

$$f'_j(t) = -\sum_{i=1}^{m} \lambda_i(t) x^*_{ij}/\Lambda_i \cdot (r_{ij} - f_j(t))^+ \tag{3.4.4}$$

with boundary condition $f_j(1) = 0$.

We call $f_j(t)$ the *benefit function* of resource j. Although the HJB equation is in continuous time, in practice, it can be computed by discretizing the horizon into periods. Further-

more, according to properties of the HJB equation, $f_j(t)$ is decreasing in t, which captures the fact that resources are expiring over time.

Below are the detailed steps of the Separation Algorithm.

1. Solve for an optimal solution x^* to the linear program (3.4.2).

2. For each resource $j \in \{1, 2, ..., n\}$, compute the benefit function $f_j(t)$ according to (3.4.4).

3. Upon an arrival of a type-i customer at time t, randomly pick a number $j \in \{1, 2, ..., n\}$ with probability x^*_{ij}/Λ_j. Assign resource j to the customer if resource j is still available and $r_{ij} \geq f_j(t)$.

The following lemma gives the total expected benefit of the Separation Algorithm.

Lemma 3.4.2. *The expected total benefit of the Separation Algorithm is* $\sum_{j=1}^{n} f_j(0)$ *at time* 0. *If resource* j *is available at time* t, *the Separation Algorithm earns benefit* $f_j(t)$ *from resource* j *in time* $[t, 1]$ *in expectation.*

Proof. According to the HJB equation (3.4.4), the Separation Algorithm earns $f_j(0)$ from resource j in expectation. Therefore, the total expected benefit of the Separation Algorithm is $\sum_{j=1}^{n} f_j(0)$.

□

The Separation Algorithm has the appeal that, at any time and any given state, we can easily compute the total expected benefit of remaining resources by summing up the values of benefit functions of all available resources. More importantly, the marginal benefit of having

an additional resource is exactly equal to the value of the benefit function. As a result of the convenience of computing marginal benefit values, we can significantly improve the empirical performance of the Separation Algorithm by converting it into a bid-price algorithm, which we will discuss in the next section.

The following theorem states that the Separation Algorithm has constant performance guarantee.

Theorem 3.4.3. *The Separation Algorithm is* 0.5*-competitive.*

To prove this theorem, we first analyze the competitive ratio for a single-resource benefit-maximization problem. Specifically, we want to study the following performance ratio for resource j.

$$f_j(0)/\sum_{i=1}^{m} r_{ij}x_{ij}^*, \tag{3.4.5}$$

where $\sum_{i=1}^{m} r_{ij}x_{ij}^*$ can be seen as an upper bound on the expected optimal offline benefit for resource j (see Theorem 3.4.1), and $f_j(0)$ is the expected benefit of the Separation Algorithm for resource j.

In order to determine a lower bound of (3.4.5), we first normalize $\sum_{i=1}^{m} r_{ij}x_{ij}^*$ to 1, which is helpful because all benefit values can be scaled by an arbitrary constant without affecting performance ratios. We must then search for a lower bound on $f_j(0)$ by examining all possible combinations of problem data, including the arrival rates (3.4.3), benefit values r_{ij}, and all possible optimal solutions x^* to the LP (3.4.2), subject to the normalization condition $\sum_{i=1}^{m} r_{ij}x_{ij}^* = 1$.

Since both x^* and the arrival rates $\lambda_i(t)$ can be expressed in terms of the split rates

$$\lambda_i(t) = \sum_{j=1}^{n} \lambda_{ij}(t), \ \forall i = 1, 2, ..., m,$$

$$x^*_{ij} = \int_0^1 \lambda_{ij}(t)dt, \ \forall i = 1, 2, ..., m, j = 1, 2, ..., n,$$

we will instead examine all values of r_{ij}'s and $\lambda_{ij}(t)$'s. This problem can be formulated as follows

$$\inf_{\lambda_{ij}(t), r_{ij}, i=1,...,m} f_j(0) \tag{3.4.6}$$

$$\text{s.t.} \quad f_j'(t) = -\sum_{i=1}^{m} \lambda_{ij}(t) \cdot (r_{ij} - f_j(t))^+ \tag{3.4.7}$$

$$\int_0^1 \sum_{i=1}^{m} \lambda_{ij}(t)dt \leq 1 \tag{3.4.8}$$

$$\int_0^1 \sum_{i=1}^{m} \lambda_{ij}(t) r_{ij} dt = 1 \tag{3.4.9}$$

$$\lambda_{ij}(t), r_{ij} \geq 0, \ \ i = 1, 2, ..., m \tag{3.4.10}$$

$$f(1) = 0, \tag{3.4.11}$$

In this optimization problem, the decisions are $\lambda_{ij}(t)$'s and r_{ij}'s. Constraint (3.4.7) is the dynamic programming equation (3.4.4). Constraint (3.4.8) is equivalent to the capacity constraint in LP (3.4.2), namely

$$\sum_{i=1}^{m} x^*_{ij} \leq 1,$$

which requires that the average number of customers who are routed to resource j is at most

1 (recall that the capacity of each resource is 1 and each customer requires a unit of resource according to our model). Constraint (3.4.9) is the normalization condition $\sum_{i=1}^{m} r_{ij} x_{ij}^* = 1$.

Theorem 3.4.4. *The optimal objective value of problem (3.4.6) is at least 0.5.*

Proof. Consider a fixed resource j. Define $g(t) = -f_j'(t)$. Replacing the function $(\cdot)^+$ by inequalities, we can rewrite problem (3.4.6) as

$$\inf_{g(t),\lambda_{ij}(t),z_i(t),r_{ij},i=1,...,m} \int_0^1 g(t)dt \tag{3.4.12}$$

$$\text{s.t.} \quad g(t) = \sum_{i=1}^{m} \lambda_{ij}(t) z_i(t) \tag{3.4.13}$$

$$g(t) \geq 0$$

$$z_i(t) \geq r_{ij} - \int_t^1 g(s)ds, \;\; i = 1, 2, ..., m \tag{3.4.14}$$

$$z_i(t) \geq 0, \;\; i = 1, 2, ..., m$$

$$\int_0^1 \sum_{i=1}^{m} \lambda_{ij}(t) dt \leq 1$$

$$\int_0^1 \sum_{i=1}^{m} \lambda_{ij}(t) r_{ij} dt = 1 \tag{3.4.15}$$

$$\lambda_{ij}(t), r_{ij} \geq 0, \;\; i = 1, 2, ..., m.$$

Note that because $g(t) = \sum_{i=1}^{m} \lambda_{ij}(t) z_i(t)$ and the objective function minimizes $\int_0^1 g(t)dt$, we want $z_i(t)$ as small as possible. Therefore, the optimal solution must have $z_i(t) = (r_{ij} - \int_t^1 g(s)ds)^+$.

Then, we dualize the constraints (3.4.13), (3.4.14) and (3.4.15) using their Lagrangian

multipliers, and obtain the following problem.

$$\begin{aligned}\inf_{g(t),\lambda_{ij}(t),z_i(t),r_{ij},i=1,...,m} & \int_0^1 g(t)dt + \int_0^1 \gamma(t)[g(t) - \sum_{i=1}^m \lambda_{ij}(t)z_i(t)]dt \\ & + \sum_{i=1}^m \int_0^1 \theta_i(t)[r_{ij} - \int_t^1 g(s)ds - z_i(t)]dt \\ & + \omega(\int_0^1 \sum_{i=1}^m \lambda_{ij}(t)r_{ij}dt - 1) \\ \text{s.t.} \quad & g(t) \geq 0 \\ & z_i(t) \geq 0, \;\; i = 1, 2, ..., m \\ & \int_0^1 \sum_{i=1}^m \lambda_{ij}(t)dt \leq 1 \\ & \lambda_{ij}(t), r_{ij} \geq 0, \;\; i = 1, 2, ..., m.\end{aligned} \tag{3.4.16}$$

As long as $\theta_i(t) \geq 0$ for all $t \in [0, 1]$ and $i = 1, 2, ..., m$, the optimal objective value of problem (3.4.16) is a lower bound of (3.4.12), because every feasible solution of problem (3.4.12) is also feasible in (3.4.16), and the objective value of every feasible solution of problem (3.4.12) is greater than or equal to the corresponding objective value in (3.4.16). To find a lower bound on the optimal value of (3.4.12), we will instead find a lower bound on the optimal value of (3.4.16).

Next, we choose the following values for the dual variables.

$$\gamma(t) = -0.5,$$

$$\theta_i(t) = 0.5\lambda_{ij}(t), \text{ for } i = 1, 2, ..., m$$

$$\omega = -0.5,$$

Since the constraint of (3.4.16) requires $\lambda_{ij}(t) \geq 0$, we have $\theta_i(t) \geq 0$. Thus, when using these values of dual variables, the optimal objective value of (3.4.16) is a lower bound of (3.4.12), and hence also a lower bound of problem (3.4.6). Plugging in these values, the objective function of (3.4.16) can be reduced to

$$\begin{aligned}
&\int_0^1 g(t)dt - \int_0^1 0.5[g(t) - \sum_{i=1}^m \lambda_{ij}(t) z_i(t)]dt \\
&\quad + \sum_{i=1}^m \int_0^1 0.5\lambda_{ij}(t)[r_{ij} - \int_t^1 g(s)ds - z_i(t)]dt \\
&\quad - 0.5(\int_0^1 \sum_{i=1}^m \lambda_{ij}(t) r_{ij} dt - 1) \\
&= 0.5 \int_0^1 g(t)dt - 0.5 \sum_{i=1}^m \int_0^1 \lambda_{ij}(t) \left(\int_t^1 g(s)ds \right) dt + 0.5 \\
&= 0.5 \int_0^1 g(t)dt - 0.5 \sum_{i=1}^m \int_0^1 g(t) \left(\int_0^t \lambda_{ij}(s)ds \right) dt + 0.5 \\
&= 0.5 \int_0^1 g(t) \left(1 - \int_0^t \sum_{i=1}^m \lambda_{ij}(s)ds \right) dt + 0.5
\end{aligned} \tag{3.4.17}$$

Since we know from the constraint of (3.4.16) that $\int_0^1 \sum_{i=1}^m \lambda_{ij}(t)dt \leq 1$, we must have

$$1 - \int_0^t \sum_{i=1}^m \lambda_{ij}(s)ds \geq 0$$

for all $t \leq 1$. Thus, the infimum of (3.4.17) is achieved by setting $g(t) = 0$. The corresponding optimal objective value is 0.5. Therefore, 0.5 is a lower bound of problem (3.4.6). □

Now that we have proved $f_j(0)/\sum_{i=1}^{m} r_{ij}x_{ij}^* \geq 0.5$ for every resource j, we can readily show the competitive ratio of the Separation Algorithm.

Proof of Theorem 3.4.3.

Since $f_j(0)/\sum_{i=1}^{m} r_{ij}x_{ij}^* \geq 0.5$ for all j, we must have

$$\frac{\sum_{j=1}^{n} f_j(0)}{\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}x_{ij}^*} \geq 0.5.$$

From Theorem 3.4.1 we know that

$$\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}x_{ij}^* \geq \mathbf{E}[\mathrm{OPT}(\delta)],$$

which gives

$$\sum_{j=1}^{n} f_j(0) \geq 0.5\mathbf{E}[\mathrm{OPT}(\delta)].$$

From Lemma 3.4.2 we know that $\sum_{j=1}^{n} f_j(0)$ is the expected total benefit of the Separation Algorithm, so this algorithm is 0.5-competitive. □

3.5 Capacity-Dependent Competitive Ratio

When a resource has greater than unit capacity, the algorithm presented in the previous section treats each unit of the resource as a separate resource, and does not exploit the fact that these units are interchangeable. In this section, we show that we can improve the Separation Algorithm when resources have greater than unit capacities.

Let C_j denote the capacity of resource j, for $j = 1, 2, ..., n$. We assume that all capacity values are positive integers. The following linear program incorporates resource capacities into LP (3.4.2).

$$\begin{aligned} \max \ & \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} r_{ij} \\ \text{s.t.} \ & \sum_{j=1}^{n} x_{ij} \leq \Lambda_i, \quad \text{for } i = 1, 2, ..., m \\ & \sum_{i=1}^{m} x_{ij} \leq C_j, \quad \text{for } j = 1, 2, ..., n \\ & x_{ij} \geq 0. \end{aligned} \tag{3.5.1}$$

Let x^* be an optimal solution to (3.5.1). For each resource j, the following is a dynamic program that optimally controls the admissions of the fraction x^*_{ij}/Λ_i of type i customers, for $i = 1, 2, ..., m$, who are routed to resource j.

$$f'_j(t, c) = -\sum_{i=1}^{m} \lambda_i(t) x^*_{ij}/\Lambda_i \cdot (r_{ij} - f_j(t, c) + f_j(t, c-1))^+, \tag{3.5.2}$$

where $f_j(t, c)$ is the expected total future benefit that can be earned from resource j starting at time t when there are c units still available. The boundary conditions are $f_j(1, c) = 0$ and $f_j(t, 0) = 0$.

Let $c_j(t)$ be the remaining capacity of resource j at time t. When a customer of type i arrives at time t and is routed to resource j, the (generalized) Separation Algorithm compares

r_{ij} with $f_j(t, c_j(t)) - f_j(t, c_j(t) - 1)$. It offers resource j if $c_j(t) \geq 1$ and

$$r_{ij} \geq f_j(t, c_j(t)) - f_j(t, c_j(t) - 1),$$

and rejects the customer otherwise.

Since the dynamic program (3.5.2) optimally integrates the decisions for all the C_j units of resource j, the value of the new benefit function $f_j(t, c)$ must be at least the sum of benefit functions that the original Separation Algorithm uses for each available unit of the resource. Then by a similar argument as the proof of Theorem 3.4.3, we can easily check that the total expected benefit of the generalized Separation Algorithm must be at least $\sum_{j=1}^{n} f(0, C_j)$. Therefore the algorithm is still 0.5-competitive.

More importantly, we expect that this generalized Separation Algorithm will have better performance when the capacity values are large, due to the integrated decisions made for each entire resource. To prove an improved bound for the case of large capacities, we focus on a single resource j. We will prove that the Separation Algorithm achieves a better competitive ratio for this resource. We will suppress the index j of the resource in the rest of the section except when needed to avoid confusion.

Assume that the capacity of the resource is k. Let $V_l(t) = f_j(t, k - l)$ be the optimal expected future benefit at time t when the remaining capacity is $k - l$ (this notation will be more convenient for analysis), for $t \in [0, 1]$ and $l = 0, 1, ..., k - 1$. The HJB equation defining

$V_l(\cdot)$ is

$$\frac{dV_l(t)}{dt} = -\sum_{i=1}^{m} \lambda_i(t) x_{ij}^*/\Lambda_i \cdot (r_{ij} - V_l(t) + V_{l+1}(t))^+ = -\sum_{i=1}^{m} \lambda_{ij}(t) \cdot (r_{ij} - V_l(t) + V_{l+1}(t))^+$$

with boundary conditions $V_k(t) = 0$ and $V_l(1) = 0$.

We are interested in the performance ratio

$$\frac{f_j(0,k)}{\sum_{i=1}^m x_{ij}^* r_{ij}} = \frac{V_0(0)}{\int_0^1 \sum_i r_{ij}\lambda_{ij}(t)dt},$$

where $\sum_{i=1}^m x_{ij}^* r_{ij} = \int_0^1 \sum_i r_{ij}\lambda_{ij}(t)dt$ is an upper bound on the optimal expected offline benefit. We want to find the smallest such ratio by examining all possible inputs r and $\lambda(\cdot)$.

Note that at any time t, the performance ratio can be *lowered* by replacing the problem instance with one in which there is only one type of customer arrival with rate $\lambda(t) = \sum_{i=1}^m \lambda_{ij}(t)$ and reward $r(t) = \frac{\sum_{i=1}^m r_{ij}\lambda_{ij}(t)}{\sum_{i=1}^m \lambda_{ij}(t)}$, so that the worst-case instance has one customer type, and time-dependent reward function $r(\cdot)$. This observation has also been made by Alaei, Hajiaghayi and Liaghat (2012). Thus, to characterize the worst-case performance ratio, we only need to bound the ratio

$$\frac{V_0(0)}{\int_0^1 r(t)\lambda(t)dt}, \tag{3.5.3}$$

for V defined as

$$\frac{dV_l(t)}{dt} = -\lambda(t)(r(t) - V_l(t) + V_{l+1}(t))^+,$$

over all reward functions $r(\cdot)$ and arrival rate functions $\lambda(\cdot)$ such that the second constraint of (3.5.1) is satisfied, i.e.,

$$\int_0^1 \lambda(t)dt \leq k.$$

Note that the first constraint of (3.5.1) is implicitly satisfied after we set

$$\lambda(t) = \sum_{i=1}^{m} \lambda_{ij}(t) = \sum_{i=1}^{m} \lambda_i(t) x_{ij}^*/\Lambda_i.$$

3.5.1 Homogenizing time

Without loss of generality, we can change the horizon length, $\lambda(\cdot)$ and $r(\cdot)$ as follows, while keeping the ratio (3.5.3) unchanged:

1. If $\int_0^1 \lambda(t)dt < k$, we can extend the horizon to length $T > 1$ by adding more arrivals with benefit 0. Thus, we can equivalently assume $\int_0^T \lambda(t)dt = k$.

2. Define a (virtual) time variable as

$$\bar{t} = \bar{t}(t) \equiv \int_0^t \lambda(s)ds.$$

   Note that $\bar{t} \in [0, k]$. Using this new time variable, we can define new benefit functions as

$$\bar{V}_i(s) = V_i(\bar{t}^{-1}(s)),$$

where we interpret $\bar{t}^{-1}(\cdot)$ as the first time t that satisfies $\bar{t}(t) = s$. Similarly, we can define $\bar{r}(s) = r(\bar{t}^{-1}(s))$. Then we can equivalently transform the HJB equation for $V_i(t)$ as follows

$$\frac{dV_i(t)}{dt} = \frac{d\bar{V}_i(\bar{t})}{d\bar{t}}\frac{d\bar{t}}{dt} = \frac{d\bar{V}_i(\bar{t})}{d\bar{t}}\lambda(t)$$

$$\implies \frac{d\bar{V}_i(\bar{t})}{d\bar{t}}\lambda(t) = -\lambda(t)(r(t) + V_{i+1}(t) - V_i(t))^+ = -\lambda(t)(\bar{r}(\bar{t}) + \bar{V}_{i+1}(\bar{t}) - \bar{V}_i(\bar{t}))^+$$

$$\implies \frac{d\bar{V}_i(\bar{t})}{d\bar{t}} = -(\bar{r}(\bar{t}) + \bar{V}_{i+1}(\bar{t}) - \bar{V}_i(\bar{t}))^+, \ \ \forall \bar{t} \in [0, k]$$

This equation can be viewed as another HJB equation with arrival rate 1 and revenue function $\bar{r}(\cdot)$, with boundary conditions $V_k(t) = 0$ for $t \in [0, k]$ and $V_i(k) = 0$ for $i = 0, 1, ..., k-1$. Furthermore, the upper bound on the expected offline benefit can be transformed as

$$\int_0^T r(t)\lambda(t)dt = \int_0^k \bar{r}(\bar{t})d\bar{t}.$$

In summary, we can equivalently transform the problem into one whose arrival rate is uniformly 1 and whose time horizon is $[0, k]$.

3.5.2 Bound-revealing optimization problem

After applying the above transformations, we can write an optimization problem that reveals the competitive ratio as follows

$$\min_{r(t), V_i(t), i=0,1,...,k-1; t\in[0,k]} V_0(0) \tag{3.5.4}$$

$$\text{s.t. } \frac{dV_i(t)}{dt} = -(r(t) + V_{i+1}(t) - V_i(t))^+, \ \forall i = 0, 1, ..., k-1; t \in [0, k]$$

$$\int_0^k r(t)dt = 1$$

$$V_i(t) \geq 0, \ \forall i = 0, 1, ..., k-1; t \in [0, k]$$

$$r(t) \geq 0.$$

Here the second constraint $\int_0^k r(t)dt = 1$ normalizes the upper bound on the expected offline benefit. By using $g_i(t) = -dV_i(t)/dt$ and replacing $(\cdot)^+$ with linear constraints, we can write the above problem equivalently as (note that $g_k(t) = 0, \forall t \in [0, k]$)

$$\min_{r(t), g_i(t), i=0,1,...,k-1; t\in[0,k]} \int_0^k g_0(s)ds$$

$$\text{s.t. } g_i(t) \geq r(t) + \int_t^k g_{i+1}(s)ds - \int_t^k g_i(s)ds, \ \forall i = 0, 1, ..., k-1; \forall t \in [0, k]$$

$$\int_0^k r(t)dt = 1$$

$$g_i(t) \geq 0, \ \forall i = 0, 1, ..., k-1; t \in [0, k]$$

$$r(t) \geq 0.$$

Let $\alpha_i(t)$ be a dual variable for the first constraint, for all $i = 0, 1, ..., k-1$ and $t \in [0, k]$. Let β be a dual variable for the second constraint. The dual problem is

$$
\begin{aligned}
&\max_{\alpha_i(t),\beta} \ \beta \\
&\text{s.t. } \alpha_0(t) + \int_0^t \alpha_0(s)ds \le 1, \ \forall t \in [0,k] \\
&\qquad \alpha_i(t) + \int_0^t \alpha_i(s)ds \le \int_0^t \alpha_{i-1}(s)ds, \ \forall i = 1,2,...,k-1; \forall t \in [0,k] \qquad (3.5.5) \\
&\qquad \beta \le \sum_{i=0}^{k-1} \alpha_i(t), \ \forall t \in [0,k] \\
&\qquad \alpha_i(t) \ge 0.
\end{aligned}
$$

This dual problem tries to maximize the minimum value of $\sum_{i=0}^{k-1} \alpha_i(t)$ with respect to t. The optimal β is a lower bound on the competitive ratio.

3.5.3 A dual-feasible solution for the bound-revealing problem

We first show that a feasible solution to the dual problem (3.5.5) can be constructed based on a bounded Poisson process. We will use this dual feasible solution to obtain a lower bound on the optimal value of the bound-revealing optimization problem (3.5.4). Alaei, Hajiaghayi and Liaghat (2012) also prove their bound by working with a dual-feasible solution. However, we construct our dual-feasible solution differently. Because our bound has to be tighter, our analysis of this solution is also much more involved.

Let $t_0, t_1, t_2, ..., t_k$ be a sequence of time points such that $0 = t_0 < t_1 < \cdots < t_{k-1} < t_k = k$.

Let $\{N(t)\}_{t \ge 0}$ be a (counting) Poisson process with rate 1. We apply an upper barrier

to $N(t)$ to obtain a new bounded process $\{R(t)\}_{t\geq 0}$. Starting with an initial value 0 at time $t_0 = 0$, the barrier increases by 1 at times $t_1, t_2, ..., t_{k-1}$. At these time points, the new bounded process has values

$$R(t_i) = \max(i-1, R(t_{i-1}) + N(t_i) - N(t_{i-1})),\ \forall i = 1, 2, ..., k-1,$$

with $R(t_0) = R(0) = 0$. And for $t \in [t_i, t_{i+1}]$, we have

$$R(t) = \max(i, R(t_i) + N(t) - N(t_i)),\ \forall i = 1, 2, ..., k-1.$$

Eventually,

$$R(t_k) = R(k) = \max(k-1, R(t_{k-1}) + N(k) - N(t_{k-1})).$$

Theorem 3.5.1. *There exists a feasible dual solution β^*, $\alpha_i^*(t)$ for $t \in [0, k]$, $i = 0, 1, 2, ..., k-1$, such that*

$$\alpha_i^*(t) = P(R(t) = i)\beta^*,\ \ \forall t \in [0, k], i = 0, 1, ..., k-1,$$

$$k(1-\beta^*) = \beta^* \left[k - \sum_{i=0}^{k-1} iP(R(k) = i) \right] \tag{3.5.6}$$

for the bounded Poisson process $R(t)$ as constructed above.

Proof. Note that the distribution of $\{R(t)\}_{t\geq 0}$ is determined by the time points $t_1, t_2, ..., t_{k-1}$. In particular, for $t \in (t_i, t_{i+1})$, the value of $P(R(t) = i)$ is only determined by $t_1, t_2, ..., t_i$.

Given any value $\beta \in (0, 1)$, we can construct a sequence of those time points $t_1, t_2, ..., t_{k-1}$

recursively based on the following condition

$$\int_{t_i}^{t_{i+1}} P(R(t) = i)dt = \frac{1}{\beta} - 1, \ \forall i = 0, 1, ..., k-2. \tag{3.5.7}$$

Here t_i is when the barrier is increased to position i, and is thus the first time that $P(R(t) = i)$ becomes positive. Given $t_1, t_2, ..., t_i$, this condition sets the value for $t_{i+1} = t_{i+1}(\beta)$ by requiring that the area under the function $P(R(t) = i)$ for $t \in [t_i, t_{i+1}]$ is exactly $1/\beta - 1$.

According to the above construction, since $P(R(t) = i)$ is a continuous function of t, the time points $t_1, t_2, ..., t_{k-1}$ must change continuously in β.

Furthermore, when $\beta \to 1$, i.e., the area under the function $P(R(t) = i)$ for $t \in [t_i, t_{i+1}]$ tends to 0 for each $i = 0, 1, ..., k-2$, we must have $t_{i+1} - t_i \to 0$ for each $i = 0, 1, ..., i-2$. This implies that $t_{k-1} \to t_0 = 0$. On the other hand, when $\beta \to 0$, we have $1/\beta^* - 1 \to \infty$, so the area under $P(R(t) = i)$ for $t \in [t_i, t_{i+1}]$ can be arbitrarily large. In other words, by tuning the value of β, we can set t_{k-1} to be any value within $(0, k)$.

Therefore, there must exist some $\beta \in (0, 1)$ such that t_{k-1} satisfies

$$\int_{t_{k-1}}^{t_k} P(R(t) = k-1)dt = \frac{1}{\beta} - 1.$$

Let β^* be such a value that satisfies this condition. Set $\alpha_i^*(t) = P(R(t) = i)\beta^*$. We next prove that this construction of β^* and $\alpha_i^*(t)$, for $i = 0, 1, ..., k-1$ and $t \in [0, k]$, satisfies the constraints of (3.5.5).

First of all, for $t \leq t_1$, we have

$$\begin{aligned}
\alpha_0^*(t) + \int_0^t \alpha_0^*(s)ds &= \beta^* P(R(t) = 0) + \int_0^t \beta^* P(R(s) = 0)ds \\
&= \beta^* \cdot 1 + \beta^* \int_0^t P(R(s) = 0)ds \\
&\leq \beta^* + \beta^* \int_0^{t_1} P(R(s) = 0)ds \\
&= \beta^* + \beta^*(1/\beta^* - 1) \\
&= 1.
\end{aligned}$$

Note that the inequality is tight when $t = t_1$. For $t > t_1$, since the barrier is above position 0, the value of the random process $R(t)$, when at position 0, is randomly jumping to position 1 at (hazard) rate equal to $P(R(t) = 0)$. Thus, we must have, for $t > t_1$,

$$\begin{aligned}
&\frac{\partial P(R(t) = 0)}{\partial t} = -R(t) \\
\Longrightarrow &P(R(t) = 0) - P(R(t_1) = 0) = -\int_{t_1}^t P(R(s) = 0)ds \\
\Longrightarrow &P(R(t) = 0) + \int_0^t P(R(s) = 0)ds = P(R(t_1) = 0) + \int_0^{t_1} P(R(s) = 0)ds \\
\Longrightarrow &\alpha_0^*(t) + \int_0^t \alpha_0^*(s)ds = \alpha_0^*(t_1) + \int_0^{t_1} \alpha_0^*(s)ds = 1.
\end{aligned}$$

Therefore, the first constraint of (3.5.5) holds and is tight for $t \geq t_1$.

To prove that the second constraint also holds, we recursively look at $i = 1, 2, ..., k-1$. Recall that t_i is the first time that $P(R(t) = i)$ becomes positive. Thus for $t \leq t_i$ we have

$P_i(R(t) = i) = 0$ and thus

$$\alpha_i^*(t) + \int_0^t \alpha_i^*(s)ds = \beta^* P(R(t) = i) + \int_0^t \beta^* P(R(s) = i)ds = 0.$$

For $t \in [t_i, t_{i+1}]$, we have

$$\begin{aligned}
\alpha_i^*(t) + \int_0^t \alpha_i^*(s)ds &= \beta^* P(R(t) = i) + \int_0^t \beta^* P(R(s) = i)ds \\
&= \beta^* P(R(t) = i) + \beta^* \int_{t_i}^t P(R(s) = i)ds \\
&\leq \beta^* P(R(t) = i) + \beta^* \int_{t_i}^{t_{i+1}} P(R(s) = i)ds \\
&= \beta^* P(R(t) = i) + \beta^*(1/\beta^* - 1) \\
&= \beta^* P(R(t) = i) + \beta^* \int_{t_{i-1}}^{t_i} P(R(s) = i - 1)ds.
\end{aligned}$$

When $t \in [t_i, t_{i+1}]$ and $R(t) = i$, the random process is actively bounded by the barrier, so the probability $P(R(t) = i)$, as a function of t, can only increase due to the transition from state $R(t) = i - 1$ to $R(t) = i$. The rate at which $P(R(t) = i)$ increases is $P(R(t) = i - 1)$.

Thus, we have $P(R(t)=i)=\int_{t_i}^{t} P(R(s)=i-1)ds$, which leads to

$$\begin{aligned}\alpha_i^*(t)+\int_0^t \alpha_i^*(s)ds \leq & \beta^* P(R(t)=i)+\beta^*\int_{t_{i-1}}^{t_i} P(R(s)=i-1)ds \\ =&\beta^*\int_{t_i}^{t} P(R(s)=i-1)ds+\beta^*\int_{t_{i-1}}^{t_i} P(R(s)=i-1)ds \\ =&\beta^*\int_0^t P(R(s)=i-1)ds \\ =&\int_0^t \alpha_{i-1}^*(s)ds.\end{aligned} \tag{3.5.8}$$

Note that the inequality is tight when $t=t_{i+1}$.

For $t>t_{i+1}$, the barrier is above i, so the random process $R(t)$, if still at state $R(t)=i$, is not actively bounded by the barrier. Thus the state $R(t)=i$ is involved in two transitions: from state i to $i+1$, and from $i-1$ to i. More precisely, we have for $t>t_{i+1}$,

$$\frac{\partial P(R(t)=i)}{\partial t}=P(R(t)=i-1)-P(R(t)=i)$$

$$\begin{aligned}\Longrightarrow & P(R(t)=i)+\int_0^t P(R(s)=i)ds-\int_0^t P(R(s)=i-1)ds \\ &=0.\end{aligned}$$

$$\Longrightarrow \alpha_i^*(t)+\int_0^t \alpha_i^*(s)ds=\int_0^t \alpha_{i-1}^*(s)ds. \tag{3.5.9}$$

This proves that the second constraint of (3.5.5) holds (and is tight for $t\geq t_{i+1}$, for each $i=1,2,...,k-1$, respectively). Finally, the last constraint of (3.5.5) trivially holds because $\sum_{i=0}^{k-1} P(R(t)=i)=1 \Longrightarrow \beta^*=\sum_{i=0}^{k-1}\alpha_i^*(t)$.

To prove (3.5.6), we can deduce that

$$
\begin{aligned}
&\beta^* \sum_{i=0}^{k-1} iP(R(k)=i) \\
=&\sum_{i=0}^{k-1} i\alpha_i^*(k) \\
=&\sum_{i=0}^{k-1} i\left[\int_0^k \alpha_{i-1}^*(s)ds - \int_0^k \alpha_i^*(s)ds\right] \text{ (by (3.5.9))} \\
=&\sum_{i=0}^{k-1} \int_0^k \alpha_i^*(s)ds - k\int_0^k \alpha_{k-1}^*(s)ds \text{ (by canceling identical terms)} \\
=&\int_0^k \left(\sum_{i=0}^{k-1} \alpha_i^*(s)\right) ds - k\beta^* \int_0^k P(R(s)=k-1)ds \\
=&\int_0^k \beta^* ds - k\beta^*(1/\beta^* - 1) \\
=&2k\beta^* - k.
\end{aligned}
$$

We can then easily obtain (3.5.6) by rearranging terms. □

Given that β^* and α^* are dual-feasible, we will next attempt to bound objective β^* by analyzing the process $R(\cdot)$.

First we show that the times at which the barriers are applied are bounded by $1, 2, \ldots, k-1$.

Theorem 3.5.2. *The time points $t_1, t_2, ..., t_{k-1}$ constructed in the proof of Theorem 3.5.1 satisfy $t_i \leq i$, for $i = 1, 2, ..., k-1$.*

Before proving Theorem 3.5.2, we first prove Lemmas 3.5.3 to 3.5.7, which characterize further the behavior of the process $R(t)$. These lemmas collectively show that when the

barriers are applied at regular points starting at some time of the horizon, i.e., $t_i = i \ \ \forall i \geq l$ for some integer l, the time spent at the barriers must be monotone decreasing in the index i for all $i \geq l$.

For ease of notation, let

$$I_i \equiv \int_{t_i}^{t_{i+1}} \mathbf{1}(R(s) = i)ds$$

be the total time that the bounded process $R(t)$ stays at the barrier i during the interval $[t_i, t_{i+1}]$, for $i = 0, 1, ..., k-1$. Note that $\mathbf{E}[I_i] = \int_{t_i}^{t_{i+1}} P(R(s) = i)ds$. Let $P_i(\lambda)$ be the probability that a Poisson random variable with mean λ is equal to i. Let $P_{\geq i}(\lambda)$ and $P_{\leq i}(\lambda)$ denote $\sum_{j=i}^{\infty} P_j(\lambda)$ and $\sum_{j=0}^{i} P_j(\lambda)$, respectively.

First, assuming that the barriers are applied at regular points $0, 1, \ldots, k-1$, we can quantify the difference in expected time spent at each barrier, given different starting points for the process $R(\cdot)$,

Lemma 3.5.3. *Given any $l \in \{1, 2, ..., k-1\}$, if $t_l = l$ and $t_{l+1} = l+1$, we must have*

$$\mathbf{E}[I_l | R(l) = l - j] - \mathbf{E}[I_l | R(l) = l - j - 1] = P_{\geq j+1}(1)$$

for all $j = 0, 1, ..., l-1$.

Proof.

$$\begin{aligned} &\mathbf{E}[I_l | R(l) = l - j] \\ &= \int_l^{l+1} P(R(s) = l | R(l) = l - j)ds \end{aligned}$$

$$= \int_0^1 P_{\geq j}(s)ds.$$

Similarly, $\mathbf{E}[I_l|R(l) = l - j - 1] = \int_0^1 P_{\geq j+1}(s)ds$. Thus,

$$\begin{aligned}
&\mathbf{E}[I_l|R(l) = l - j] - \mathbf{E}[I_l|R(l) = l - j - 1] \\
=&\int_0^1 P_{\geq j}(s)ds - \int_0^1 P_{\geq j+1}(s)ds \\
=&\int_0^1 P_j(s)ds \\
=&\int_0^1 e^{-s}\frac{s^j}{j!}ds \\
=&\sum_{\nu=j+1}^{\infty} e^{-1}\frac{1}{\nu!} \\
=&P_{\geq j+1}(1).
\end{aligned}$$

□

Next, assuming that the barriers are applied at regular points $0, 1, \ldots, k-1$, we can bound differences in time spent at each barrier for successive pairs of starting points.

Lemma 3.5.4. *Given any $l \in \{2, 3, ..., k-1\}$, if $t_i = i$ for all $i = l, l+1, ..., k-1$, we must have*

$$\mathbf{E}[I_i|R(l) = l] - \mathbf{E}[I_i|R(l) = l - 1] \geq e^{-1}(\mathbf{E}[I_i|R(l) = l - 1] - \mathbf{E}[I_i|R(l) = l - 2])$$

for all $i = l, l+1, ..., k-1$.

Proof. Fix any $i \in \{l, l+1, ..., k-1\}$. For ease of notation, define

$$\Delta_{d,j} \equiv \mathbf{E}[I_i | R(d) = d - j] - \mathbf{E}[I_i | R(d) = d - j - 1]$$

to be the decrease in the expected time that $R(t)$ stays at the barrier during $[t_i, t_{i+1}]$, when the state at time $t = d$ changes from $R(d) = d - j$ down to $R(d) = d - j - 1$, for all $d = l, l+1, ..., i$ and $j = 0, 1, ..., d-1$.

From Lemma 3.5.3 we know that $\Delta_{i,j} = P_{\geq j+1}(1)$. Furthermore, for $d < i$ and $d \geq l$, the value of $\mathbf{E}[I_i | R(d) = d - j]$ can be recursively computed by conditioning on $R(d+1)$, i.e., on the movement of the random process during time $[d, d+1]$. Precisely,

$$\mathbf{E}[I_i | R(d) = d - j] = \sum_{\nu=0}^{j} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d - j + \nu] + \sum_{\nu=j+1}^{\infty} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d].$$

Here, for example, $R(d+1) = d - j + \nu$ represents the condition where the random process $R(t)$ moves ν steps upwards during time $[d, d+1]$; $R(d+1) = d$ represents the condition where the random process hits the barrier at time $t = d + 1$.

Similarly,

$$\begin{aligned}
\mathbf{E}[I_i | R(d) = d - j - 1] &= \sum_{\nu=0}^{j+1} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d - j - 1 + \nu] + \sum_{\nu=j+2}^{\infty} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d] \\
&= \sum_{\nu=0}^{j} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d - j - 1 + \nu] + \sum_{\nu=j+1}^{\infty} P_\nu(1) \mathbf{E}[I_i | R(d+1) = d].
\end{aligned}$$

The above two equations lead to the following recursion for $\Delta_{d,j}$

$$\begin{aligned}\Delta_{d,j} &= \mathbf{E}[I_i|R(d) = d-j] - \mathbf{E}[I_i|R(d) = d-j-1] \\ &= \sum_{\nu=0}^{j} P_\nu(1)[\mathbf{E}[I_i|R(d+1) = d-j+\nu] - \mathbf{E}[I_i|R(d+1) = d-j-1+\nu]] \\ &= \sum_{\nu=0}^{j} P_\nu(1)\Delta_{d+1,j-\nu+1}.\end{aligned} \tag{3.5.10}$$

Note that in order to prove the lemma, we need to show $\Delta_{l,0}/\Delta_{l,1} \geq 1/e$. To this end, we prove a stronger result by constructing a bound on $\Delta_{d,j}/\Delta_{d,j+1}$ for all $d = l, l+1, ..., i$, and $j = 0, 1, ..., d$. We construct the bounds using a sequence of 'stationary' values $\Delta_{*,0}, \Delta_{*,1}, ...$, which are defined based on the recursion (3.5.10) and are independent of d:

$$\Delta_{*,0} = 1;$$

$$\Delta_{*,j} = \sum_{\nu=0}^{j} P_\nu(1)\Delta_{*,j-\nu+1}, \quad \forall j = 0, 1, 2, ... \tag{3.5.11}$$

$$\Longrightarrow \begin{cases} \Delta_{*,1} = e\Delta_{*,0}, \\ \Delta_{*,j+1} = (e-1)\Delta_{*,j} - \sum_{\nu=2}^{j} \frac{1}{\nu!}\Delta_{*,j-\nu+1}, \ \forall j \geq 1. \end{cases} \tag{3.5.12}$$

We next prove that

$$\frac{\Delta_{d,j}}{\Delta_{d,j+1}} \geq \frac{\Delta_{*,j}}{\Delta_{*,j+1}} \tag{3.5.13}$$

using induction on d.

- First, we prove that (3.5.13) holds when $d = i$ by showing that $\Delta_{i,j}$ is decreasing in j but $\Delta_{*,j}$ is increasing in j.

  By Lemma 3.5.3, $\Delta_{i,j} = P_{\geq j+1}(1) > P_{\geq j+2}(1) = \Delta_{i,j+1}$, which means $\Delta_{i,j}$ is decreasing in j.

  From (3.5.12) we know that $\Delta_{*,0} = e^{-1}\Delta_{*,1} < \Delta_{*,1}$. Provided that $\Delta_{*,\nu} \leq \Delta_{*,j}$ for all $\nu \leq j$ and some $j \geq 1$, we can deduce from (3.5.12),

$$\frac{\Delta_{*,j+1}}{\Delta_{*,j}} \geq e - 1 - \sum_{\nu=2}^{j} \frac{1}{\nu!} \geq e - 1 - \sum_{\nu=2}^{\infty} \frac{1}{\nu!} = e - 1 - (e - 2) = 1.$$

  Thus, $\Delta_{*,j}$ is increasing in j, which finishes the proof that (3.5.13) holds when $d = i$.

- When $d < i$,

$$\begin{aligned}
&\Delta_{d,j}\Delta_{*,j+1} - \Delta_{d,j+1}\Delta_{*,j} \\
&= \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{d+1,j-\nu_1+1}\right)\left(\sum_{\nu_2=0}^{j+1} P_{\nu_2}(1)\Delta_{*,j-\nu_2+2}\right) \\
&\qquad - \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{*,j-\nu_1+1}\right)\left(\sum_{\nu_2=0}^{j+1} P_{\nu_2}(1)\Delta_{d+1,j-\nu_2+2}\right) \text{ (by (3.5.10) and (3.5.11))} \\
&= \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{d+1,j-\nu_1+1}\right) P_0(1)\Delta_{*,j+2} - \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{*,j-\nu_1+1}\right) P_0(1)\Delta_{d+1,j+2} \\
&\quad + \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{d+1,j-\nu_1+1}\right)\left(\sum_{\nu_2=0}^{j} P_{\nu_2+1}(1)\Delta_{*,j-\nu_2+1}\right) \\
&\quad - \left(\sum_{\nu_1=0}^{j} P_{\nu_1}(1)\Delta_{*,j-\nu_1+1}\right)\left(\sum_{\nu_2=0}^{j} P_{\nu_2+1}(1)\Delta_{d+1,j-\nu_2+1}\right) \\
&= \sum_{\nu_1=0}^{j} P_{\nu_1}(1)P_0(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j+2} - \Delta_{*,j-\nu_1+1}\Delta_{d+1,j+2}\right)
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{\nu_1=0}^{j}\sum_{\nu_2=0}^{\nu_1-1} P_{\nu_1}(1)P_{\nu_2+1}(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1}\right)\\
&+\sum_{\nu_2=0}^{j}\sum_{\nu_1=0}^{\nu_2-1} P_{\nu_1}(1)P_{\nu_2+1}(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1}\right)\\
&+\sum_{\nu_1=0}^{j} P_{\nu_1}(1)P_{\nu_1+1}(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_1+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_1+1}\right)\\
=&\sum_{\nu_1=0}^{j} P_{\nu_1}(1)P_0(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j+2}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j+2}\right)\\
&+\sum_{\nu_1=0}^{j}\sum_{\nu_2=0}^{\nu_1-1} P_{\nu_1}(1)P_{\nu_2+1}(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1}\right)\\
&+\sum_{\nu_2=0}^{j}\sum_{\nu_1=0}^{\nu_2-1} P_{\nu_1}(1)P_{\nu_2+1}(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1}\right)\\
=&\sum_{\nu_1=0}^{j} P_{\nu_1}(1)P_0(1)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j+2}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j+2}\right) \qquad (3.5.14)\\
&+\sum_{\nu_1=0}^{j}\sum_{\nu_2=0}^{\nu_1-1}\left(P_{\nu_1}(1)P_{\nu_2+1}(1)-P_{\nu_2}(1)P_{\nu_1+1}(1)\right)\left(\Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1}\right)
\end{aligned}
$$

(3.5.15)

Now using induction on $d+1$, we know that for $\nu_1 \geq 0$,

$$\frac{\Delta_{d+1,j-\nu_1+1}}{\Delta_{d+1,j+2}} \geq \frac{\Delta_{*,j-\nu_1+1}}{\Delta_{*,j+2}},$$

and thus (3.5.14)≥ 0. In (3.5.15), since $\nu_1 > \nu_2$, we can use induction on $d+1$ again to obtain

$$\frac{\Delta_{d+1,j-\nu_1+1}}{\Delta_{d+1,j-\nu_2+1}} \geq \frac{\Delta_{*,j-\nu_1+1}}{\Delta_{*,j-\nu_2+1}}$$

$$\implies \Delta_{d+1,j-\nu_1+1}\Delta_{*,j-\nu_2+1}-\Delta_{*,j-\nu_1+1}\Delta_{d+1,j-\nu_2+1} \geq 0.$$

Furthermore, since $\nu_1 > \nu_2$,

$$P_{\nu_1}(1)P_{\nu_2+1}(1) - P_{\nu_2}(1)P_{\nu_1+1}(1) = P_{\nu_1}(1)P_{\nu_2}(1)\left(\frac{1}{\nu_2+1} - \frac{1}{\nu_1+1}\right) \geq 0.$$

Thus, (3.5.15)≥ 0. In sum, we have shown $\Delta_{d,j}\Delta_{*,j+1} - \Delta_{d,j+1}\Delta_{*,j} \geq 0$, which finishes the induction proof for condition (3.5.13).

This result (3.5.13) directly leads to the statement of the Lemma

$$\Delta_{l,0} \geq \Delta_{l,1}\frac{\Delta_{*,0}}{\Delta_{*,1}} = \Delta_{l,1}/e.$$

□

Using the previous result, we relax the assumption that all the barriers are applied at regular points $0, 1, \ldots, k-1$. We assume now that the barriers are applied at regular times beyond a point. Under this condition, we show that the differences in time spent at successive barriers are increasing with the starting point of the process.

Lemma 3.5.5. *Given any $l \in \{1, 2, ..., k-2\}$, if $t_i \leq i$ for $i = 1, 2, ..., l$, and $t_i = i$ for $i = l+1, l+2, ..., k-1$, we must have*

$$\mathbf{E}[I_i|R(l) = l] - \mathbf{E}[I_{i+1}|R(l) = l] \geq \mathbf{E}[I_i|R(l) = l-1] - \mathbf{E}[I_{i+1}|R(l) = l-1]$$

for all $i = l, l+1, ..., k-2$.

Proof. Fix any $i \in \{l, l+1, ..., k-2\}$. By symmetry, we have $\mathbf{E}[I_i|R(l) = l] = \mathbf{E}[I_{i+1}|R(l+1) = l+1]$ and $\mathbf{E}[I_i|R(l) = l-1] = \mathbf{E}[I_{i+1}|R(l+1) = l]$.

Thus,

$$\begin{aligned}
&\mathbf{E}[I_i|R(l) = l] - \mathbf{E}[I_{i+1}|R(l) = l] - (\mathbf{E}[I_i|R(l) = l-1] - \mathbf{E}[I_{i+1}|R(l) = l-1]) \\
=&\mathbf{E}[I_{i+1}|R(l+1) = l+1] - \mathbf{E}[I_{i+1}|R(l+1) = l] - (\mathbf{E}[I_{i+1}|R(l) = l] - \mathbf{E}[I_{i+1}|R(l) = l-1])) \\
=&\mathbf{E}[I_{i+1}|R(l+1) = l+1] - \mathbf{E}[I_{i+1}|R(l+1) = l] - e^{-1}(\mathbf{E}[I_{i+1}|R(l+1) = l] - \mathbf{E}[I_{i+1}|R(l+1) = l-1])) \\
\geq&0,
\end{aligned}$$

where the last inequality follows from Lemma 3.5.4. □

Next, assuming that the barriers are applied at regular points $0, 1, \ldots, k-1$, we show that the time spent by the process at each barrier is decreasing with the index of the barrier.

Lemma 3.5.6. *If $t_i = i$ for all $i = 1, 2, ..., k-1$, we must have $\mathbf{E}[I_i] \geq \mathbf{E}[I_{i+1}]$ for all $i = 1, 2, ..., k-2$.*

Proof. It is obvious that for any $i \geq 1$,

$$\mathbf{E}[I_i|R(1) = 1] \geq \mathbf{E}[I_i|R(1) = 0],$$

because when the starting position becomes lower, it is harder for the random process $R(t)$ to reach the barrier at any later time. Since $\mathbf{E}[I_i|R(1) = 0] = \mathbf{E}[I_i]$, and by symmetry, $\mathbf{E}[I_i|R(1) = 1] = \mathbf{E}[I_{i-1}]$, we have $\mathbf{E}[I_{i-1}] \geq \mathbf{E}[I_i]$ for all $i \geq 1$. □

Finally, we relax the requirement of Lemma 3.5.6. We require only that the barriers be applied at regular points only after some time. We show that the time spent at the barriers are still decreasing.

Lemma 3.5.7. *Given any $l \in \{1, 2, ..., k-2\}$, if $t_i \leq i$ for $i = 1, 2, ..., l$, and $t_i = i$ for $i = l+1, l+2, ..., k-1$, we must have*

$$\mathbf{E}[I_i] \geq \mathbf{E}[I_{i+1}]$$

for all $i = l, l+1, ..., k-2$.

Proof. Given any sequence of times points $\bar{t}_1 \leq \bar{t}_2 \leq \cdots \leq \bar{t}_l$ such that $\bar{t}_j \leq j$, $\forall j = 1, 2, ..., l$, we want to prove the lemma when $t_j = \bar{t}_j$ for $j \leq l$ and $t_j = j$ for $j > l$.

Fix any $i \in \{l, l+1, ..., k-2\}$. Initially, set $t_j = j$ for all $j = 1, 2, ..., k-1$, and we know that $\mathbf{E}[I_i] \geq \mathbf{E}[I_{i+1}]$ by Lemma 3.5.6. We next prove that $\mathbf{E}[I_i] \geq \mathbf{E}[I_{i+1}]$ always holds when we reduce the value of t_j from j to $\bar{t}_j$ *sequentially* for $j = 1, 2, ..., l$.

Define

$$I_i' = \begin{cases} I_i, & \text{if } i > l \\ \int_l^{l+1} \mathbf{1}(R(s) = l) ds, & \text{if } i = l. \end{cases}$$

By this definition, I_i is different from I_i' only if $i = l$ and $t_l < l$ (we create the definition of I_i' so as to facilitate the proof for the case of $i = l$). Note that we always have $I_i \geq I_i'$, and in particular, $I_i = I_i'$ when initially $t_j = j$ for all $j = 1, 2, ..., l$.

Consider the result of reducing t_j from j to $\bar{t}_j$, when $t_d = d$ for all $d = j+1, j+2, ..., k-1$. Let $\hat{R}(t), \hat{I}_i$ and $\hat{I}_i'$ be the value of $R(t), I_i$ and I_i', respectively, before reducing t_j, i.e., when

$t_j = j$. Let $\bar{R}(t), \bar{I}_i$ and $\bar{I}'_i$ be the new value of $R(t), I_i$ and I'_i, respectively, after reducing t_j to $\bar{t}_j$.

Suppose $\mathbf{E}[\hat{I}_i] \geq \mathbf{E}[\hat{I}_{i+1}]$ holds, we next prove that $\mathbf{E}[\bar{I}_i] \geq \mathbf{E}[\bar{I}_{i+1}]$ also holds.

- When $t_j = j$, since $j \leq l$, we must have $t_l = l$ and thus $\mathbf{E}[\hat{I}_i] = \mathbf{E}[\hat{I}'_i]$.
- We must have $P(\bar{R}(j) = \nu) = P(\hat{R}(j) = \nu)$ for all $\nu \leq j-2$, because if $R(t) \leq j-2$, the random process is not affected by the barrier after time t_{j-1}.
- We must have $P(\bar{R}(j) = j-1) \leq P(\hat{R}(j) = j-1)$ and $P(\bar{R}(j) = j) \geq P(\hat{R}(j) = j)$ because when t_j becomes smaller, there is more time for the random process $R(t)$ to jump from state $j-1$ up to j. Moreover,

$$P(\bar{R}(j) = j) - P(\hat{R}(j) = j) = P(\hat{R}(j) = j-1) - P(\bar{R}(j) = j-1) \geq 0. \tag{3.5.16}$$

Based on the above results, we can then deduce that ($\mathbf{E}[\bar{I}_i]$ is defined as $\mathbf{E}[I_i]$ given $t_j = \bar{t}_j$. Similarly, $\mathbf{E}[\hat{I}_i]$ is defined as $\mathbf{E}[I_i]$ given $t_j = j$)

$$\begin{aligned}
&\mathbf{E}[\bar{I}_i] - \mathbf{E}[\bar{I}_{i+1}] \\
\geq &\mathbf{E}[\bar{I}'_i] - \mathbf{E}[\bar{I}_{i+1}] \\
= &\sum_{\nu=0}^{j} \mathbf{E}[\bar{I}'_i | \bar{R}(j) = \nu] P(\bar{R}(j) = \nu) - \sum_{\nu=0}^{j} \mathbf{E}[\bar{I}_{i+1} | \bar{R}(j) = \nu] P(\bar{R}(j) = \nu) \\
= &\sum_{\nu=0}^{j} \mathbf{E}[\hat{I}'_i | \hat{R}(j) = \nu] P(\bar{R}(j) = \nu) - \sum_{\nu=0}^{j} \mathbf{E}[\hat{I}_{i+1} | \hat{R}(j) = \nu] P(\bar{R}(j) = \nu)
\end{aligned}$$

$$\left(\begin{array}{l}\text{Given } R(j)=\nu\text{, reducing } t_j \text{ does not affect the random process after } t \geq j,\\ \\ \text{due to the memoryless property.}\end{array}\right)$$

$$=\sum_{\nu=0}^{j-2}\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]P(\hat{R}(j)=\nu)+\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]P(\bar{R}(j)=\nu)$$

$$-\sum_{\nu=0}^{j-2}\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu]P(\hat{R}(j)=\nu)-\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu]P(\bar{R}(j)=\nu)$$

$$=\sum_{\nu=0}^{j}\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]P(\hat{R}(j)=\nu)+\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu](P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$-\sum_{\nu=0}^{j}\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu]P(\hat{R}(j)=\nu)-\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu](P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$=\mathbf{E}[\hat{I}_i']+\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu](P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$-\mathbf{E}[\hat{I}_{i+1}]-\sum_{\nu=j-1}^{j}\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu](P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$=\mathbf{E}[\hat{I}_i']-\mathbf{E}[\hat{I}_{i+1}]+\sum_{\nu=j-1}^{j}(\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]-\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu])(P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$=\mathbf{E}[\hat{I}_i]-\mathbf{E}[\hat{I}_{i+1}]+\sum_{\nu=j-1}^{j}(\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]-\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu])(P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$\geq\sum_{\nu=j-1}^{j}(\mathbf{E}[\hat{I}_i'|\hat{R}(j)=\nu]-\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=\nu])(P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))$$

$$=(P(\bar{R}(j)=\nu)-P(\hat{R}(j)=\nu))\times$$

$$(\mathbf{E}[\hat{I}_i'|\hat{R}(j)=j]-\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=j]-\mathbf{E}[\hat{I}_i'|\hat{R}(j)=j-1]+\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=j-1]).\ \text{(by (3.5.16))}$$

Now Lemma 3.5.5 gives

$$\mathbf{E}[\hat{I}_i'|\hat{R}(j)=j]-\mathbf{E}[\hat{I}_i'|\hat{R}(j)=j-1]-\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=j]+\mathbf{E}[\hat{I}_{i+1}|\hat{R}(j)=j-1]\geq 0.$$

This proves that $\mathbf{E}[\bar{I}_i] \geq \mathbf{E}[\bar{I}_{i+1}]$. Therefore, we always have $\mathbf{E}[I_i] \geq \mathbf{E}[I_{i+1}]$ when we change t_j from j to $\bar{t}_j$ for all $j = 1, 2, ..., l$. □

The idea of the proof of Theorem 3.5.2 is as follows. We will start by setting the barriers at times $0, 1, \ldots, k-1$. We then successively reduce the values t_i, $i = 0, 1, \ldots$, until the time spent at each barrier is no more than $1/\beta - 1$. By the monotonicity shown in Lemma 3.5.7, this procedure must stop with the time spent at each barrier bounded above by $1/\beta - 1$.

If we change the value of β, the time points $t_1, t_2, ..., t_{k-1}$ that result from the above procedure must change continuously in β. We simply choose β such that, when the procedure ends, the time spent at the last barrier is $1/\beta - 1$, which implies that the time spent at all barriers is exactly $1/\beta - 1$.

Proof of Theorem 3.5.2.

We give a new and more detailed construction of the same set of times points as constructed in the proof of Theorem 3.5.1.

Fix any $\beta \in (0, 1)$. Starting with $t_i = i, \forall i = 1, 2, ..., k-1$, we run the following algorithm:

For $i = 0, 1, ..., k-2$:

(a) If $\mathbf{E}[I_i] > 1/\beta - 1$, reduce t_{i+1} such that $\mathbf{E}[I_i] = 1/\beta - 1$.

(b) Stop if $\mathbf{E}[I_i] < 1/\beta - 1$.

If the algorithm stops at (b) when $i = l$, we must have

$$\mathbf{E}[I_0] = \mathbf{E}[I_1] = \cdots = \mathbf{E}[I_{l-1}] = 1/\beta - 1$$

and, according to Lemma 3.5.7,

$$1/\beta - 1 = \mathbf{E}[I_{l-1}] \geq \mathbf{E}[I_l] \geq \cdots \geq \mathbf{E}[I_{k-1}]. \tag{3.5.17}$$

On the other hand, if the algorithm never stops at (b), we must have

$$\mathbf{E}[I_0] = \mathbf{E}[I_1] = \cdots = \mathbf{E}[I_{k-2}] = 1/\beta - 1. \tag{3.5.18}$$

If we change the value of β, the time points $t_1, t_2, ..., t_{k-1}$ as the result of the algorithm must change continuously in β. This implies that $\mathbf{E}[I_{k-1}] = \int_{t_{k-1}}^{k} P(R(s) = k-1)ds$ must change continuously in β. When β is close to 0, we must have $\mathbf{E}[I_{k-1}] < 1/\beta - 1$; when β is close to 1, we must have $\mathbf{E}[I_{k-1}] > 1/\beta - 1$. Therefore, there must exist a β such that, when the algorithm ends,

$$\mathbf{E}[I_{k-1}] = 1/\beta - 1.$$

Let β^* be such a value.

Now the time points have met all desired conditions if (3.5.18) holds (i.e., the algorithm never stops at (b)). If the algorithm stops at some step (b), then according to (3.5.17),

$$1/\beta^* - 1 = \mathbf{E}[I_{l-1}] \geq \mathbf{E}[I_l] \geq \cdots \geq \mathbf{E}[I_{k-1}] = 1/\beta^* - 1$$

$$\Longrightarrow \mathbf{E}[I_0] = \mathbf{E}[I_1] = \cdots = \mathbf{E}[I_{k-1}] = 1/\beta^* - 1,$$

which gives all desired conditions of the time points as well. □

Lemma 3.5.8.

$$\beta^* = \frac{1}{2} + \frac{1}{2k} \sum_{i=0}^{k-1} i\, \alpha_i^*(k).$$

Proof. Starting from Theorem 3.5.1 we can deduce that

$$k(1-\beta^*) = \beta^* \left[k - \sum_{i=0}^{k-1} iP(R(k)=i) \right]$$

$$\implies 2k\beta^* = k + \beta^* \sum_{i=0}^{k-1} iP(R(k)=i) = k + \sum_{i=0}^{k-1} i\alpha_i^*(k)$$

$$\implies \beta^* = \frac{1}{2} + \frac{1}{2k} \sum_{i=0}^{k-1} i\alpha_i^*(k).$$

□

3.5.4 Computing the bound

First, we prove an inequality, which will be useful in computing our bound.

Lemma 3.5.9. *For any $x, y \in \mathbb{Z}$ and $\lambda \in [0,k]$ such that $x \geq y \geq k-1-\lambda$, we must have for any $l = 0, 1, ..., k-1$,*

$$\sum_{i=-l}^{l} P_{k-1+i-x}(\lambda) \leq \sum_{i=-l}^{l} P_{k-1+i-y}(\lambda).$$

Proof. If suffices to prove the case when $x = y+1$. We have

$$\sum_{i=-l}^{l} P_{k-1+i-(y+1)}(\lambda) - \sum_{i=-l}^{l} P_{k-1+i-y}(\lambda)$$

$$=P_{k-2-l-y}(\lambda) - P_{k-1+l-y}(\lambda).$$

If $k-2-l-y < 0$, the lemma trivially holds because $P_{k-2-l-y}(\lambda) = 0$ and thus $P_{k-2-l-y}(\lambda) - P_{k-1+l-y}(\lambda) \leq 0$.

Now suppose $k-2-l-y \geq 0$. Then

$$\begin{aligned}
&\frac{P_{k-2-l-y}(\lambda)}{P_{k-1+l-y}(\lambda)} \\
=&\frac{\lambda^{k-2-l-y}}{(k-2-l-y)!} \cdot \frac{(k-1+l-y)!}{\lambda^{k-1+l-y}} \\
=&\frac{1}{\lambda^{2l+1}} \prod_{i=-l}^{l} (k-1-y+i).
\end{aligned}$$

Since $y \geq k-1-\lambda$, we must have $k-1-y \leq \lambda$ and $(k-1-y+i)(k-1-y-i) \leq \lambda^2$. This shows that $P_{k-2-l-y}(\lambda)/P_{k-1+l-y}(\lambda) \leq 1$ and thus $P_{k-2-l-y}(\lambda) - P_{k-1+l-y}(\lambda) \leq 0$. □

Lemma 3.5.10. *For any $l = 0, 1, ..., k-1$, we have*

$$\sum_{i=-l}^{l} P_{k-1+i}(k) \leq \frac{1}{\beta^*} \sum_{i=0}^{l} \alpha^*_{k-1-i}(k).$$

Proof. Define

$$R^{(i)} \equiv R(t_i) + N(k) - N(t_i), \ \forall i = 0, 1, 2, ..., k.$$

Note that since the bounded process $R(t)$ is determined by $N(t)$, the random variables $R^{(i)}$'s are also determined by $N(t)$.

Since

$$P(R^{(0)} = i) = P(R(t_0) + N(k) - N(t_0) = i) = P(N(k) = i) = P_i(k)$$

and

$$P(R^{(k)} = i) = P(R(k) = i) = \alpha_i^*(k)/\beta^*,$$

it suffices to show that

$$\sum_{i=-l}^{l} P(R^{(j-1)} = k - 1 + i) \leq \sum_{i=-l}^{l} P(R^{(j)} = k - 1 + i) \tag{3.5.19}$$

for all $l = 0, 1, ..., k-1$ and $j = 1, 2, ..., k$.

According to the definition of the bounded process $R(t)$, if $R(t_{j-1}) + N(t_j) - N(t_{j-1}) \leq j - 1$, then $R(t_j) = R(t_{j-1}) + N(t_j) - N(t_{j-1})$ and thus

$$\begin{aligned} R^{(j)} &= R(t_j) + N(k) - N(t_j) \\ &= R(t_{j-1}) + N(t_j) - N(t_{j-1}) + N(k) - N(t_j) \\ &= R(t_{j-1}) + N(k) - N(t_{j-1}) \\ &= R^{(j-1)}. \end{aligned}$$

Therefore,

$$\sum_{i=-l}^{l} P(R^{(j-1)} = k - 1 + i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) \leq j - 1)$$

$$= \sum_{i=-l}^{l} P(R^{(j)} = k-1+i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) \leq j-1)$$

for all $l = 0, 1, ..., k-1$ and $j = 1, 2, ..., k$.

Now consider the case $x = R(t_{j-1}) + N(t_j) - N(t_{j-1}) > j-1$. We must have

$$R^{(j-1)} = x + N(k) - N(t_j),$$

$$R^{(j)} = j - 1 + N(k) - N(t_j).$$

Recall that Lemma 3.5.2 gives $t_j \leq j$, so $x > j-1 \geq t_j - 1 = k-1-(k-t_j)$. We can then apply Lemma 3.5.9 by further setting $y = j-1$ and $\lambda = k - t_j$ and obtain (for $x > j-1$)

$$\begin{aligned}
&\sum_{i=-l}^{l} P(R^{(j-1)} = k-1+i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) = x) \\
=&\sum_{i=-l}^{l} P(x + N(k) - N(t_j) = k-1+i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) = x) \\
=&\sum_{i=-l}^{l} P(N(k) - N(t_j) = k-1+i-x | R(t_{j-1}) + N(t_j) - N(t_{j-1}) = x) \\
=&\sum_{i=-l}^{l} P(N(k-t_j) = k-1+i-x) \\
\leq&\sum_{i=-l}^{l} P(N(k-t_j) = k-1+i-(j-1)) \\
=&\sum_{i=-l}^{l} P(j-1+N(k) - N(t_j) = k-1+i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) = x) \\
=&\sum_{i=-l}^{l} P(R^{(j)} = k-1+i | R(t_{j-1}) + N(t_j) - N(t_{j-1}) = x)
\end{aligned}$$

In sum, we have shown (3.5.19), which proves the lemma. □

Finally, we derive our bound. The bound is simply a reduction of the equation

$$k(1-\beta^*) = \beta^* \left[k - \sum_{i=0}^{k-1} iP(R(k)=i) \right],$$

which follows from Theorem 3.5.1. β^* is strictly greater than 0.5 for $k \geq 2$. For example, when $k=2$, β^* satisfies

$$3\beta + \beta e^{1/\beta - 3} = 2,$$

from which we can obtain $\beta^* \approx 0.615$.

Theorem 3.5.11.

$$\beta^* \geq \frac{1}{1 + \frac{1}{k}[\sum_{i=2k-1}^{\infty} iP_i(k) + 2\sum_{i=1}^{k-1} iP_{k+i-1}(k)]}.$$

Proof. Combining Lemma 3.5.8 and Lemma 3.5.10, we obtain

$$\begin{aligned}
\beta^* =& \frac{1}{2} + \frac{1}{2k} \sum_{i=0}^{k-1} i\, \alpha_i^*(k) \\
=& \frac{1}{2} + \frac{1}{2k} \sum_{l=0}^{k-2} \sum_{i=0}^{l} \alpha_{k-1-i}^*(k) \\
\geq& \frac{1}{2} + \frac{1}{2k} \sum_{l=0}^{k-2} \beta^* \sum_{i=-l}^{l} P_{k-1+i}(k) \\
=& \frac{1}{2} + \frac{\beta^*}{2k} \left[\sum_{l=0}^{k-2} \sum_{i=-l}^{0} P_{k-1+i}(k) + \sum_{l=0}^{k-2} \sum_{i=1}^{l} P_{k-1+i}(k) \right] \\
=& \frac{1}{2} + \frac{\beta^*}{2k} \left[\sum_{i=1}^{k-1} iP_i(k) + \sum_{i=k}^{2k-2} (2k-i-2)P_i(k) \right]
\end{aligned}$$

$$
\begin{aligned}
=&\frac{1}{2}+\frac{\beta^*}{2k}\left[\sum_{i=1}^{2k-2} iP_i(k)+\sum_{i=k}^{2k-2}(2k-2i-2)P_i(k)\right]\\
=&\frac{1}{2}+\frac{\beta^*}{2k}\left[\sum_{i=1}^{2k-2} iP_i(k)-2\sum_{i=1}^{k-1} iP_{k-1+i}(k)\right].
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow \beta^* \geq&\frac{k}{2k-\left[\sum_{i=1}^{2k-2} iP_i(k)-2\sum_{i=1}^{k-1} iP_{k-1+i}(k)\right]}\\
=&\frac{k}{k+\left[k-\sum_{i=1}^{2k-2} iP_i(k)\right]+2\sum_{i=1}^{k-1} iP_{k-1+i}(k)}\\
=&\frac{k}{k+\sum_{i=2k-1}^{\infty} iP_i(k)+2\sum_{i=1}^{k-1} iP_{k-1+i}(k)}\\
=&\frac{1}{1+\frac{1}{k}\left[\sum_{i=2k-1}^{\infty} iP_i(k)+2\sum_{i=1}^{k-1} iP_{k-1+i}(k)\right]}
\end{aligned}
$$

□

Corollary 3.5.12. *Assuming that the minimum capacity for each resource is k, the competitive ratio for the Separation Algorithm is at least*

$$\beta^* \geq \frac{1}{1+2\left[\frac{P_{\geq k}(k)}{k}+\frac{e^{-k}k^k}{k!}\right]} = 1-\sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}}+O(\frac{1}{k}).$$

Proof.

$$
\begin{aligned}
\sum_{i=2k-1}^{\infty} iP_i(k)+2\sum_{i=1}^{k-1} iP_{k+i-1}(k) \leq & \quad 2\sum_{i=1}^{\infty} iP_{k+i-1}(k)\\
= & \quad 2\sum_{i=1}^{\infty} i\frac{k^{k+i-1}}{(k+i-1)!}e^{-k}\\
= & \quad 2\left[\frac{k^k e^{-k}}{(k-1)!}+\sum_{i=k}^{\infty} P_i(k)\right].
\end{aligned}
$$

$$\begin{aligned}\Longrightarrow \beta^* \geq & \quad \frac{1}{1+\frac{2}{k}\left[\frac{k^k e^{-k}}{(k-1)!}+\sum_{i=k}^{\infty} P_i(k)\right]} \\ = & \quad \frac{1}{1+2\left[\frac{e^{-k}k^k}{k!}+\frac{P_{\geq k}(k)}{k}\right]}.\end{aligned}$$

By Stirling's formula,

$$\frac{e^{-k}k^k}{k!} = \frac{1}{\sqrt{2\pi k}} + o(1/k).$$

Furthermore, since $P_{\geq k}(k) \leq 1$, we have

$$\frac{P_{\geq k}(k)}{k} = O(1/k).$$

Thus,

$$\frac{1}{1+2\left[\frac{e^{-k}k^k}{k!}+\frac{P_{\geq k}(k)}{k}\right]} = \frac{1}{1+2\left[\frac{1}{\sqrt{2\pi k}}+o(1/k)+O(1/k)\right]} = 1-\sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}}+O(1/k).$$

In sum, we have proved that

$$\frac{V_0(0)}{\int_0^1 r(t)\lambda(t)dt} \geq \beta^* \geq 1-\sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}}+O(1/k).$$

Thus, if $C_j \geq k$ for all resource j,

$$\frac{f_j(t,C_j)}{\sum_{i=1}^m x_{ij}^* r_{ij}} = \frac{V_0(0)}{\sum_{i=1}^m \int_0^1 r_{ij}\lambda_{ij}(t)dt} \geq 1-\sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}}+O(1/k)$$

$$\Longrightarrow \frac{\sum_{j=1}^{n} f_j(t, C_j)}{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^* r_{ij}} \geq 1 - \sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}} + O(1/k).$$

This proves the competitive ratio of the Separation Algorithm. □

3.6 Marginal Allocation Algorithm

The Separation Algorithm, when carried out in practice, has several problems. First, it might route customers to unavailable resources when they can be better matched to other resources. Second, because of the random routing, it might unfairly accept a lower-priority customer after rejecting a higher-priority customer. In this section, we present the *Marginal Allocation Algorithm* which resolves these issues by converting the Separation Algorithm into a bid-price algorithm. We will prove that the Marginal Allocation Algorithm has theoretical performance no worse than that of the Separation Algorithm.

The Marginal Allocation Algorithm uses the marginal benefit $f_j(t, c_j(t)) - f_j(t, c_j(t) - 1)$ as a bid price for resource j. When a customer of type i arrives, the Marginal Allocation Algorithm rejects the customer if $r_{ij} < f_j(t, c_j(t)) - f_j(t, c_j(t) - 1)$ for all available resource j; otherwise, it assigns this customer to resource

$$\text{argmax}_j\{r_{ij} - f_j(t, c_j(t)) + f_j(t, c_j(t) - 1) | j \text{ is available at time } t\}.$$

To carry out this algorithm, we only need to compute the n benefit functions at the beginning of the horizon, thus reducing the space requirement to polynomial size. At any

time t, we only need to know the n benefit functions $f_j(t, c_j(t))$, for $j = 1, 2, ..., n$, so as to make a decision.

The following theorem states that the Marginal Allocation Algorithm performs at least as well as the Separation Algorithm:

Theorem 3.6.1. *The expected total benefit of the Marginal Allocation Algorithm is no less than that of the Separation Algorithm.*

Proof. Let $h(t, c)$ be the expected future benefit that can be obtained by the Marginal Allocation Algorithm starting at time t when $c = (c_1, c_2, ..., c_n)$ is the vector of resources available at t. Let $f(t, c) = \sum_{j=1}^{n} f_j(t, c_j)$ be the expected future benefit that can be obtained by the Separation Algorithm starting at time t when c is the vector of resources available at t. We will show that

$$h(t, c) \geq f(t, c) \tag{3.6.1}$$

for every given state (t, c).

Define an algorithm $\Pi^{(i)}$ as follows. For the first i customers, apply the Marginal Allocation Algorithm. Afterward, for the $(i+1)$-th, $(i+2)$-th,..., customers, apply the Separation Algorithm. Let $h^{(i)}(t, c)$ be the expected future benefit when algorithm $\Pi^{(i)}$ is applied starting at time t with remaining inventory c, and *assuming that no customers have arrived prior to time* t. We must have

$$h^{(0)}(t, c) = f(t, c),$$

$$\lim_{i \to \infty} h^{(i)}(t, c) = h(t, c).$$

The HJB equation for computing the expected benefit of algorithm $\Pi^{(1)}$ is

$$\frac{\partial h^{(1)}(t,c)}{\partial t} = -\sum_{i=1}^{m} \lambda_i(t) \left(\max_{j\in\{1,2,...,n\}} (r_{ij} - f_j(t,c_j) + f_j(t,c_j-1))^+ - \Delta^{(1)}(t,c) \right), \quad (3.6.2)$$

where

$$\Delta^{(1)}(t,c) \equiv h^{(1)}(t,c) - f(t,c).$$

The boundary conditions for (3.6.2) are $h^{(1)}(1,c) = 0$ (the length of the horizon is 1) and $f_j(t,-1) = -\infty, \forall j = 1,2,...,n$ and $t \in [0,1]$. Note that according to the boundary condition $f_j(t,-1) = -\infty$, the 'bid price' of any resource j that has no remaining inventory becomes infinity, as $f_j(t,0) - f_j(t,-1) = \infty$.

To see why (3.6.2) is true, consider the discrete version of (3.6.2). During any small period $(t, t+\delta t)$, one of the following three events will take place.

- No customer arrives during $(t, t+\delta t)$. Then the expected future benefit $h^{(1)}(t,c)$ turns into $h^{(1)}(t+\delta t, c)$.

- A customer of some type i arrives, but $\Pi^{(1)}$ (which applies the Marginal Allocation Algorithm to the customer) rejects the customer. We must have

$$\max_{j\in\{1,2,...,n\}} (r_{ij} - f_j(t+\delta t,c_j) + f_j(t+\delta t,c_j-1))^+ = 0,$$

because r_{ij} must be smaller than the 'bid price' $f_j(t+\delta t,c_j) - f_j(t+\delta t,c_j-1)$ of all

available resources j; we must also have that $h^{(1)}(t,c)$ turns into $f(t+\delta t,c)$, as $\Pi^{(1)}$ turns into the Separation Algorithm.

- A customer of some type i arrives and the customer is assigned to a resource j. In this case, the system collects benefit r_{ij}, and $h^{(1)}(t,c)$ turns into $f(t+\delta t,c-e_j)$ as $\Pi^{(1)}$ turns into the Separation Algorithm, where e_j is the unit vector with the j-th position being 1. Note that

$$f(t+\delta t,c-e_j)=f(t+\delta t,c)+f_j(t+\delta t,c_j-1)-f_j(t+\delta t,c_j).$$

Then mathematically, we can combine the second and the third bullet points, and say that when a customer of type i arrives, the expected future benefit $h^{(1)}(t,c)$ turns into total *current and future* benefit

$$f(t+\delta t,c)+\max_{j\in\{1,2,\ldots,n\}}\left(r_{ij}-f_j(t+\delta t,c_j)+f_j(t+\delta t,c_j-1)\right)^+.$$

In sum, the recursive equation for $h^{(1)}(t,c)$ is

$$\begin{aligned}h^{(1)}(t,c)=&(1-\sum_{i=1}^{m}\lambda_i(t)\delta t)h^{(1)}(t+\delta t,c)\\&+\sum_{i=1}^{m}\lambda_i(t)\delta t\left(f(t+\delta t,c)+\max_{j\in\{1,2,\ldots,n\}}\left(r_{ij}-f_j(t+\delta t,c_j)+f_j(t+\delta t,c_j-1)\right)^+\right).\end{aligned}$$

Letting $\delta t\to 0$ leads to (3.6.2).

Therefore,

$$\frac{\partial h^{(1)}(t,c)}{\partial t} \leq \frac{\partial f(t,c)}{\partial t} + \sum_{i=1}^{m} \lambda_i(t)\Delta^{(1)}(t,c). \tag{3.6.3}$$

This equation implies that, if at some time t_0 we have $\Delta^{(1)}(t_0,c) < 0$ or equivalently

$$h^{(1)}(t_0,c) - f(t_0,c) < 0, \tag{3.6.4}$$

then we must have

$$\frac{\partial h^{(1)}(t,c)}{\partial t} < \frac{\partial f(t,c)}{\partial t}, \ \ \forall t \in (t_0,1] \tag{3.6.5}$$

and

$$h^{(1)}(t,c) < f(t,c), \ \ \forall t \in (t_0,1]. \tag{3.6.6}$$

However, since we know that $h^{(1)}(1,c) = f(1,c) = 0$, (3.6.6) cannot be true, and thus (3.6.4) cannot be true. Therefore, we have proved

$$h^{(1)}(t,c) \geq f(t,c), \ \ \forall t \in [0,1]. \tag{3.6.7}$$

Next, we show that

$$h^{(i)}(t,c) \geq h^{(i-1)}(t,c), \ \ \forall t \in [0,1] \tag{3.6.8}$$

by induction on i.

Equation (3.6.7) already proves the base case $i = 1$. Suppose for some $\bar{i} > 1$, (3.6.8) holds for all $i < \bar{i}$. Now we show that it also holds for $i = \bar{i}$. By definition, for any $\bar{i} > 1$, algorithms $\Pi^{(\bar{i})}$ and $\Pi^{(\bar{i}-1)}$ must allocate the first customer in the same way. Thus, $\Pi^{(\bar{i})}$ and

$\Pi^{(\bar{i}-1)}$ earn the same benefit from the first customer, and then transit into the same state. After that first customer, $\Pi^{(\bar{i})}$ continues to apply $\Pi^{(\bar{i}-1)}$ pretending that no customer has ever arrived, while $\Pi^{(\bar{i}-1)}$ continues to apply $\Pi^{(\bar{i}-2)}$. By induction, the expected future benefit of $\Pi^{(\bar{i}-1)}$ is at least that of $\Pi^{(\bar{i}-2)}$. Therefore, the expected future benefit of $\Pi^{(\bar{i})}$ is at least that of $\Pi^{(\bar{i}-1)}$.

Thus, we have proved (3.6.8). It immediately follows that

$$h^{(\infty)}(t,c) \geq h^{(0)}(t,c).$$

□

3.7 Asymptotic performance

We can show that the Marginal-Allocation Algorithm is asymptotically optimal as the system size tends to infinity. Talluri and Van Ryzin (1998) are the first to study asymptotic behavior of bid-price control in network revenue management problems. Our proof follows theirs and subsequent proofs of similar results. Let $C_j = \theta \bar{C}_j$, for $j = 1, 2, ..., n$, $\lambda_i(t) = \theta \bar{\lambda}_i(t)$, for $i = 1, 2, ..., m$, where θ is a system scaling parameter and the barred quantities are fixed. Let $\bar{x}^*$ be an optimal solution for the system $(\bar{C}, \bar{\lambda}(t))$, then $x^* = \theta \bar{x}^*$ is an optimal solution for the system $(C, \lambda(t))$. The following theorem guarantees that the performance of the algorithm approaches the optimal objective value of the LP (3.5.1) when θ goes to infinity.

Theorem 3.7.1.

$$\lim_{\theta\to\infty} \frac{\sum_{j=1}^{n} f_j(0, C_j)}{r'x^*} = \lim_{\theta\to\infty} \frac{\sum_{j=1}^{n} f_j(0, \theta\bar{C}_j)}{\theta r'\bar{x}^*} = 1,$$

where x^* is an optimal solution to (3.5.1).

Proof. It suffices to prove that for each j,

$$\lim_{\theta\to\infty} \frac{f_j(0, C_j)}{\sum_{i=1}^{m} r_{ij}x_{ij}^*} = \lim_{\theta\to\infty} \frac{f_j(0, \theta\bar{C}_j)}{\theta\sum_{i=1}^{m} r_{ij}\bar{x}_{ij}^*} = 1.$$

Consider the single-resource benefit-maximization problem for resource j. Let N be the total number of customers who will come to resource j. N is a Poisson random variable with mean

$$\mathbf{E}[N] = \sum_{i=1}^{m} x_{ij}^* = \theta\sum_{i=1}^{m} \bar{x}_{ij}^*.$$

Let $M = \max_{ij} r_{ij}$ be an upper bound on all of the benefits. A first-come, first-served algorithm for resource j will admit $\min(N, C_j)$ customers, and obtain an expected total benefit of at least

$$\sum_{i=1}^{m} x_{ij}^* r_{ij} - \mathbf{E}[N - \min(N, C_j)]M,$$

where $x_{ij}^* r_{ij}$ is the total expected benefit from all type i customers, and $N - \min(N, C_j)$ is the number of customers who cannot fit into the C_j units of capacity.

Since the dynamic programming algorithm must perform at least as well as the first-in,

first-served algorithm, we have

$$f_j(0, C_j) \geq \sum_{i=1}^{m} x_{ij}^* r_{ij} - \mathbf{E}[N - \min(N, C_j)]M.$$

When θ goes to infinity, both x^* and $\mathbf{E}[N]$ grow in proportion to θ. Thus,

$$\begin{aligned}
\lim_{\theta\to\infty} \frac{f_j(0, C_j)}{\sum_{i=1}^{m} r_{ij} x_{ij}^*} &\geq \lim_{\theta\to\infty} \frac{\sum_{i=1}^{m} x_{ij}^* r_{ij} - \mathbf{E}[N - \min(N, C_j)]M}{\sum_{i=1}^{m} r_{ij} x_{ij}^*} \\
&\geq 1 - \lim_{\theta\to\infty} \frac{\mathbf{E}[N - \min(N, \sum_{i=1}^{m} x_{ij}^*)]M}{\sum_{i=1}^{m} r_{ij} x_{ij}^*} \\
&\geq 1 - \lim_{\theta\to\infty} \frac{0.4\sqrt{\sum_{i=1}^{m} x_{ij}^*}M}{\sum_{i=1}^{m} r_{ij} x_{ij}^*} \\
&= 1 - \lim_{\theta\to\infty} \frac{\sqrt{\theta}}{\theta} \cdot \frac{0.4\sqrt{\sum_{i=1}^{m} \bar{x}_{ij}^*}M}{\sum_{i=1}^{m} r_{ij} \bar{x}_{ij}^*} \\
&= 1,
\end{aligned}$$

where the last inequality uses the fact that, when the mean of Poisson random variable N is very large, $\mathbf{E}[(N - \mathbf{E}[N])^+]$ is bounded by $0.4\sqrt{\mathbf{E}[N]}$.

□

3.8 Upper Bound on the Competitive Ratio

In the above analysis we have shown that 0.5 is a lower bound on the best competitive ratio. Next, we show that 0.5 is also an upper bound on competitive ratio of any online algorithm. That is, our algorithms achieve the best constant competitive ratio.

Theorem 3.8.1. *The competitive ratio of any online algorithm is at most* 0.5.

Proof. Consider a situation in which a single resource is available to be allocated. There are two types of customers who want to be matched to that resource.

- Type-1 customers arrive in time $[0, 0.5]$. Their arrival rate is very large in the period $[0, 0.5]$. In particular, $\Lambda_1 = \int_0^{0.5} \lambda_1(t)dt \gg 1$, so that we can ignore the event that no type-1 customer arrives. Their benefit for the resource is $r_1 = 1$.

- Type-2 customers arrive in time $[0.5, 1]$. They have a very small arrival rate. In particular, $\Lambda_2 = \int_{0.5}^{1} \lambda_2(t)dt \ll 1$. Their benefit for that resource is $r_2 = 1/\Lambda_2 \gg 1$.

Since $r_2 \gg r_1$, the offline algorithm will allocate the resource to a type-2 customer, if there is one. The probability that at least one type-2 customer arrives is $1 - e^{-\Lambda_2} = \Lambda_2 + o(\Lambda_2^2)$. With probability $1 - o(\Lambda_2)$, no type-2 customer will arrive, in which case the optimal offline algorithm will assign a type-1 customer (there are plenty of type-1 customers) and earn benefit $r_1 = 1$. In sum, the expected total offline benefit is

$$\begin{aligned} & r_2(\Lambda_2 + o(\Lambda_2^2)) + r_1(1 - o(\Lambda_2)) \\ = \; & 1/\Lambda_2(\Lambda_2 + o(\Lambda_2^2)) + 1 \cdot (1 - o(\Lambda_2)) \\ = \; & 1 + o(\Lambda_2) + 1 - o(\Lambda_2) \\ = \; & 2 + o(\Lambda_2). \end{aligned}$$

The decision of an online algorithm is whether to allocate the resource to a type-1 customer during the first half of the horizon. If it does allocate the resource to a type-1 customer, the online algorithm earns benefit $r_1 = 1$. Otherwise, with probability $\Lambda_2 + o(\Lambda_2^2)$ it earns r_2,

which equals $1 + o(\Lambda_2)$ in expectation. In sum, the expected benefit obtained by an online algorithm cannot exceed $1 + o(\Lambda_2)$. Thus, an upper bound of the competitive ratio is

$$(1 + o(\Lambda_2))/(2 + o(\Lambda_2)),$$

which tends to 0.5 in the limit as $\Lambda_2 \to 0$. □

3.9 Overbooking

Another issue that is common to all advance admission-scheduling systems is the issue of no-shows. When customers book in advance, events may transpire between the date of the booking and the planned date of service that cause customers to miss their appointments. Due to the frequent occurrence of no-shows, overbooking is commonly used in service industries. Suppose each customer has a no-show probability of p_j when assigned to resource j, and incurs a cost of D_j when being denied getting resource j. Then we can model the overbooking strategy by expanding capacities at additional costs. Assume that the no-show events are exogenous to both online and offline algorithm. For resource j, the kth overbooked unit of capacity incurs an expected marginal cost of

$$o_j(k) = D_j \cdot (1 - p_j) \cdot \left[\sum_{l=0}^{k-1} \binom{C_j + k - 1}{l} p_j^l (1 - p_j)^{C_j + k - 1 - l} \right], \tag{3.9.1}$$

where the value in the brackets represents the probability that, among the C_j+k-1 customers who have already booked resource j, at most $k-1$ of them do not show up. The additional $1-p_j$ in the product represents the probability that the kth overbooked customer does show up. Note that the marginal cost $o_j(k)$ is independent of customer type, and is increasing in k.

Assuming that the benefit r_{ij} is earned whether a customer of type i actually takes resource j, the marginal benefit of allocating the kth overbooked unit of resource j to a type i customer is

$$\tilde{r}_{i,j,k} = r_{ij} - o_j(k).$$

When using this benefit value $\tilde{r}_{i,j,k}$, we are treating each overbooked unit of resource j as a virtual slot to be allocated. Then, the theoretical bound of our algorithms still applies, with $\tilde{r}_{i,j,k}$ being the benefit of expanded units.

Since $\tilde{r}_{i,j,k} \leq r_{i,j}$ and $\tilde{r}_{i,j,k}$ decreases in k, an optimal offline algorithm, when allocating resource j, will first fill in the C_j units of regular capacity and then assign customers to those virtual slots with lower values of k. It will not use virtual slots with non-positive marginal benefit. Then, when b overbooked units of resource j are used under the optimal offline algorithm by the end, the total cumulative cost

$$\sum_{k=1}^{b} o_j(k) \tag{3.9.2}$$

is just the actual expected overbooking cost for resource j.

3.10 Computing Algorithms

In some applications, the number of customer types m can be exponentially large such that the size of LP (3.4.2) is too large to be dealt with in practice. For example, in the display-ad allocation problem, customers can have hundreds of different attributes (Ciocan and Farias, 2014) and thus the dimension of customer type space can be huge. In such cases, it is hard to compute the benefit functions $f_j(t)$ by directly solving the LP (3.4.2), due to the huge number of constraints.

In this section, we propose an alternative method that estimates the benefit functions $f_j(t)$ by simulation, using only a subset of all customer types. The simulation algorithm works if the average number of arrivals Λ_i is very small for every customer type i. (If Λ_i is large for certain type i, one can randomly split the customers into multiple types, such that the arrival rate of each type is smaller.) The algorithm requires the ability to

- Randomly select a set S of customer types. Each of the m customer types has the same probability to be selected into S, and is independent of the selection of other types. This can be realized by generating customer types with random attributes.

- Estimate the expected number of arrivals Λ_i for any given customer type i, by using historical data and possibly certain assumptions on customer preference.

- Estimate the total arrival rate $\lambda(t) = \sum_{i=1}^{m} \lambda_i(t)$ of all customers at time t. In practice, one can often use a discrete-time horizon, and then $\lambda(t)$ is just the average number of arrivals in period t.

- Generate a random customer who arrives at time t. The probability that the customer is of type i is $\lambda_i(t)/\lambda(t)$, which is the probability that an actual arrival at time t belongs to type i. This can be easily achieved by drawing arrivals from data.

The algorithm simulates the derivative of benefit function $f_j'(t)$ in the following steps.

1. Select a random set $S \subseteq \{1, 2, ..., m\}$ of customer types. Every customer type has an equal probability to be chosen into S. Let $\epsilon = |S|/m$.

2. Estimate Λ_i for each type $i \in S$.

3. Solve the following small LP

$$\begin{aligned} \max \quad & \sum_{i\in S} \sum_{j=1}^{n} x_{ij} r_{ij} && (3.10.1)\\ \text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \le \Lambda_i, \quad \text{for } i \in S && (3.10.2)\\ & \sum_{i\in S} x_{ij} \le \epsilon, \quad \text{for } j = 1, 2, ..., n && (3.10.3)\\ & x_{ij} \ge 0. && (3.10.4) \end{aligned}$$

Let $p = (p_1, p_2, ..., p_n)$ be the optimal dual variables corresponding to the constraints (3.10.3). Then we define a primal solution $x(p)$ to the original LP (3.4.2) as

$$x_{ij}(p) = \mathbf{1}_{ij}(p) \cdot \Lambda_i,$$

where

$$\mathbf{1}_{ij}(p) = \begin{cases} 1 & \text{if } j = \text{argmax}_k\{r_{ik} - p_k\} \text{ and } r_{ij} \geq p_j, \\ 0 & \text{otherwise.} \end{cases}$$

We assume that there is no tie in determining the index k that maximizes $r_{ik} - p_k$. This can be achieved by adding a small perturbation to the benefit values r_{ij} (Feldman et al., 2010; Agrawal et al., 2014).

4. Generate a number of random arrivals at time t. Let $(b_1, b_2, ..., b_m)$ be the vector containing sample points of random arrivals, where b_i is the number of arrival instances for type i customers.

5. Estimate the total arrival rate $\lambda(t)$ at time t.

6. Finally, $f_j'(t)$ is estimated by

$$\hat{f}_j'(t) = -\frac{\lambda(t)}{\|b\|_1} \sum_{i=1}^{m} b_i \cdot x_{ij}(p)/\Lambda_i \cdot (r_{ij} - f_j(t))^+ \tag{3.10.5}$$

$$= -\frac{\lambda(t)}{\|b\|_1} \sum_{i=1}^{m} b_i \cdot \mathbf{1}_{ij}(p) \cdot (r_{ij} - f_j(t))^+. \tag{3.10.6}$$

Note that although the summation has m elements, at most $\|b\|_1$ of them are non-zero.

The idea of this simulation process is that when the dual prices p are approximately optimal for the original LP (3.4.2), the induced primal solution $x(p)$ will also be a near-optimal solution to the LP (3.4.2).

The following result is first given by Feldman et al. (2010). Recall that $\epsilon = |S|/m$. The result says that when the number m of customer types is large, the above sampling procedure

with a fixed sample size $|S|$ yields a solution x, which is close in value to the optimal solution x^* with high probability, as long as the average total arrival rate and the average relative total expected benefit of each demand type is not too large.

Theorem 3.10.1. *(Feldman et al., 2010) With high probability,*

$$\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}x_{ij}(p) \geq (1-O(\epsilon))\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij}x^*_{ij},$$

given

$$\max_{i,j}\{\frac{r_{ij}\Lambda_i}{\sum_{kl} r_{kl}x^*_{kl}}\} \leq \frac{\epsilon}{(n+1)(\ln m+\ln n)}$$

and

$$\max_{i}\{\Lambda_i\} \leq \frac{\epsilon^3}{(n+1)(\ln m+\ln n)}.$$

The following theorem guarantees that, if the primal solution $x(p)$ used to compute the benefit functions is near-optimal, then $\mathbf{E}[\hat{f}'_j(t)|S]$ performs well when used in our algorithms, where the expectation is taken over the random sample $(b_1, b_2, ..., b_m)$ of arrivals in step 4 above. According to the central limit theorem, $\hat{f}'_j(t)$ converges to $\mathbf{E}[\hat{f}'_j(t)|S]$ when the sample size $\|b\|_1$ tends to infinity. Thus, the number of samples $\|b\|_1$ should be chosen accordingly.

Theorem 3.10.2. *Suppose $x(p)$ is $1-O(\epsilon)$ optimal for the LP (3.4.2). If we use $\mathbf{E}[\hat{f}'_j(t)|S]$ as the derivative of the benefit function, our algorithms are $0.5(1-O(\epsilon))$-competitive.*

Proof. Let

$$g_j(t) \equiv -\mathbf{E}[\hat{f}'_j(t)|S] = \sum_{i=1}^{m}\lambda_i(t)\cdot\mathbf{1}_{ij}(p)\cdot(r_{ij}-\hat{f}_j(t))^+.$$

When we use $x(p)$ to route customers to resources in the Separation Algorithm, the optimal expected benefit for resource j is just $\int_t^1 g_j(s)ds$. Then we can apply Theorem 3.4.4 to resource j to get

$$\int_0^1 g_j(t)dt \geq 0.5 \sum_{i=1}^{m} r_{ij} x_{ij}(p),$$

which leads to

$$\sum_{j=1}^{n} \int_0^1 g_j(t)dt \geq 0.5 \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} x_{ij}(p).$$

Thus,

$$\sum_{j=1}^{n} \int_0^1 g_j(t)dt \geq 0.5 \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} x_{ij}(p) \geq 0.5(1 - O(\epsilon)) \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} x^*_{ij}.$$

This proves that our algorithms are $0.5(1 - O(\epsilon))$-competitive.

□

3.11 Numerical Studies

We compare our Marginal Allocation Algorithm against the outcome of the actual scheduling practices used in the Division of Clinical Genetics within the Department of Pediatrics at Columbia University Medical Center (CUMC). One of our collaborators at CUMC oversees appointment scheduling practice at the medical center. We estimate our model parameters, including patient preferences, arrival rates and hospital processing capacities, by using historical appointment-scheduling data from the outpatient clinics. We also test the performance of our algorithm against some simple heuristics. We find that our Marginal Allocation

Algorithm performs the best among all heuristics considered, and is 21% more efficient than current practice, according to our performance metric, which we will explain below.

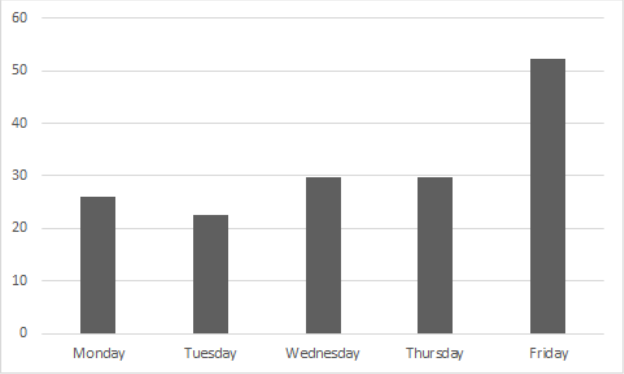
Specifically, we used data from the Division of Clinical Genetics at CUMC. Clinical Genetics is a field of medicine where adults are assessed for the risk of having offsprings with heritable conditions and children are assessed for genetic disorders. Geneticists use physical exams, chromosome testing and DNA analysis to diagnose patients suspected of having genetic abnormalities. The data we used contain more than 9000 appointment entries recorded in the year 2013. Each entry in the data records information about one appointment. The entry includes the date that the patient makes the appointment, the exact time of the appointment, whether the patient eventually showed up to the original appointment, canceled the appointment some time later, or missed the appointment. Canceled appointment slots are offered to new patients when possible.

The average number of patients who arrive to make appointments on each day is shown in Figure 3.1. During the week, there tend to be more patients who initiate requests for appointments on Thursday and Friday than on Monday and Tuesday. The actual arrival (of requests) pattern is highly non-stationary, as the average number of arrivals on Friday is about twice that on other days. Our Marginal Allocation Algorithm gracefully handles this inherent non-stationarity.

We assume that there are two sessions on each day, a morning and an afternoon session. Each session on each day corresponds to a resource in our model. About 98% appointments were scheduled into sessions on Monday through Thursday. We ignore the 2% of appointments scheduled into other sessions because there are insufficient data to perform accurate

analysis for these sessions. In other words, we set the capacity of sessions on Friday, Saturday and Sunday to be 0. The capacity of sessions from Monday to Thursday are set based on the actual number of appointments made on these days, which is about 23 appointments per session. We will vary the capacity values in some of our experiments.

Figure 3.1: Average number of arrivals in a week.



In this numerical experiment we do not model rescheduling, and treat each rescheduled appointment as an independent request. We also do not model the reuse of canceled appointment slots. Canceled slots are reused in practice, resulting in more efficient use of capacity. In this way, our algorithm are at a disadvantage compared to actual practice because it has less capacity at its disposal.

We assume that the higher the probability that a patient will show up for a session, the more preferred the session is. Thus, we use *show probabilities* as a proxy for patient preferences for each session in a week. Specifically, we define the *benefit* of assigning a

patient who arrives in period i to a session j as

$$r_{ij} = \begin{array}{l}\text{Probability that the patient arriving in period } i \text{ will show up in} \\ \text{session } j \text{ without canceling the appointment some time later or} \\ \text{missing the appointment eventually.}\end{array} \tag{3.11.1}$$

This definition of benefit value does not capture all practical concerns, but it gives a good sense of scheduling effectiveness. The higher the measure is, the fewer no-shows and cancellations are likely to result, and the fewer appointments slots are potentially wasted. One of our collaborators at CUMC oversees appointment scheduling practice at CUMC. In practice, operators try to subjectively assign appointments to accommodate patient preferences while maintaining a high level of utilization of capacity. Thus, our definition of benefit is compatible with the goals of the actual system.

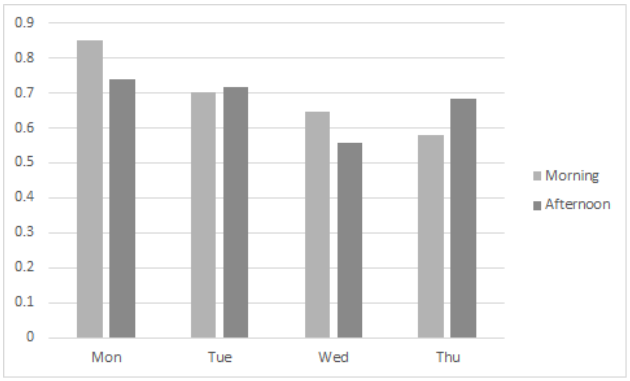
We estimate the show probabilities as a function of 3 factors: the day of the week, the time of day (morning/afternoon) and the number of days of wait starting from the patient's arrival to the actual appointment. In the first part of our experiment, we assume that patients have identical preferences in the sense that any two patients arriving on the same day will have the same benefit values for each open session. Thus, patients differ only in their time of arrival.

Both of the above assumptions regarding the homogeneity of preferences and the usefulness of show probabilities as indicators of preferences are strong assumptions. We are aware that the show probabilities are imperfect substitute for actual preferences. They also only express an average measure of preference. A finer experiment would take into account ac-

tual preferences and variability of preferences among patients. However, we believe that our experiment is still valuable in indicating the value of using online algorithms. In a sense, our online algorithms are at a disadvantage compared to real practice because in practice, appointments were made taking into account actual preferences, whereas our online algorithms "know" only the show probabilities.

Figure 3.2 illustrates the show probabilities of patients who arrive on a Thursday to make appointments for the following week. We can see that, in general, the shorter the wait is in days, the higher the show probability is. Figure 3.3 illustrates the show probability as a function of number of days to wait before getting service. The show probabilities range from as low as 27%, for appointments made more than two months into the future, to as high as 97%, for same-day visits. Table 3.1 shows more show probabilities as a function of waiting time and day of week of the appointment.

Figure 3.2: Show probabilities of appointment slots assigned to patients who arrived on the previous Thursday.



We used a 12-week period from March to May in 2013 as our time horizon. An appointment reminder system was in use during this time. There are 2032 patients scheduled

Figure 3.3: Show probabilities as functions of number of days to wait before getting service.

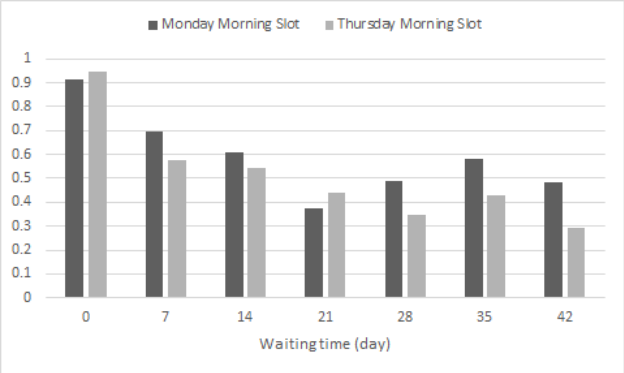


Table 3.1: Show probabilities for morning sessions, as a function waiting time and day of week of the appointment. Some cells are NA because there is no patient arrival during weekends.

	Number of days waiting								
Day of Week of Appointment	0	1	2	3	4	5	6	7	8
Mon	91%	NA	NA	81%	85%	78%	82%	70%	69%
Tue	78%	83%	NA	NA	62%	70%	73%	58%	53%
Wed	97%	61%	46%	NA	NA	57%	65%	52%	50%
Thur	95%	67%	41%	50%	NA	NA	60%	58%	57%

during this horizon according to our data. We use the sample consisting of these 2032 patient arrivals to simulate the performance of the following scheduling policies.

- The Marginal Allocation Algorithm (MAA). The arrival rates, which are inputs of the algorithm, are estimated using our one-year data in 2013. The average number of arrivals in each day of week has been shown earlier in Figure 3.1.

- The Marginal Allocation Algorithm with estimation error $\alpha\%$ (MAA-$\alpha\%$). This algorithm uses benefit values (3.11.1) that are each randomly and independently perturbed by $\alpha\%$. The total benefit earned by this algorithm is computed using the unperturbed benefit values. We include these algorithms to test the impact of our parameter estimation errors on the performance comparison with actual practice.

- The Separation Algorithm with larger than unit capacity.
- The outcome of actual practice used in hospitals. The total benefit earned by the actual strategy is also calculated using the benefit values defined in (3.11.1).
- The greedy policy, which always assigns a patient to the available session that is most preferred by the patient, as indicated by the show probability of the session. It captures a naive but easily implementable policy when a scheduler is aware of patient preferences.
- The bid-price policy, which uses the optimal dual variables of LP (3.4.2) corresponding to the capacity constraints as the bid prices. It assigns an arriving customer to the resource with the lowest price smaller than or equal to the revenue that the customer brings. This heuristic is a widely used heuristic in resource-allocation problems.

In our first experiment, we do not consider overbooking and cancellations. The capacity of each session is set to be the number of appointments made in practice. In other words, we assume that the actual practice fully utilizes the capacity of all resources. Furthermore, we assume that patients arriving on the same day have homogeneous benefit values.

Since we use show probability as the benefit of scheduling a patient, the total benefit that a scheduling policy earns from the total 2032 patients is equal to the expected number of patients, among 2032, who will show up to the original appointments. In particular, since the show probabilities are themselves estimated based on the scheduling of the actual practice, the total benefit earned by the actual practice is just the actual number of patients, out of the total 2032, that showed up during the horizon.

For each scheduling policy, we report as its performance the ratio of total benefit to the total number 2032 of arrivals. This ratio represents the overall percentage of patients who will show up. Table 3.2 summarizes the performance of all scheduling policies we consider. We can see that our Marginal Allocation Algorithm performs the best, and in particular, gives more than 30% improvement over the actual practice, according to our performance measure. It is noteworthy that the greedy and bid-price policies do not have performance guarantees and can perform arbitrarily badly. In contrast, our Marginal Allocation Algorithm has not only a provable performance guarantee, but also good empirical performance.

The strength of our Marginal Allocation Algorithm is more directly reflected in comparison with the greedy policy. The greedy policy can be carried out by anyone as long as the patient preferences are exploited. Our Marginal Allocation Algorithm, which does smart reservation, gives 12.9% empirical improvement in scheduling efficiency over this heuristic. Note that in this experiment, all patients have the same priority. Our Marginal Allocation Algorithm is likely to exhibit much higher benefits when there are more patient types to consider because it can make more intelligent tradeoffs among the types than the greedy policy can. Remarkably, our Marginal Allocation Algorithm can be implemented as easily as the greedy policy. In the greedy approach, the scheduler has to be given a number representing estimated patient preference for each session. In our Marginal Allocation Algorithm, the scheduler also needs to be given only one number, namely the marginal value of benefit function, for each session.

Table 3.2: The empirical performance of different scheduling policies.

Scheduling Policy	Performance of scheduling policies relative to LP upper bound
Actual Strategy	67%
Greedy	81%
Bid-Price Heuristic	89%
Separation Algorithm	80%
MAA	92%
MAA-5%	91%
MAA-10%	88%
MAA-20%	83%
MAA-40%	74%

3.11.1 Consideration for Overbooking

Starting from the numerical settings in the previous section, we study the practice of overbooking. Let A_j be the actual number of patients who are assigned to session j. We assume that the actual strategy overbooks each session by a constant ratio, and thereby treat $C_j = \alpha A_j$ as the actual capacity of session j, where $\alpha \in [0, 1]$ is a scaling parameter that we vary in the numerical experiment.

We define the no-show probability as

$$P_{NS} = \frac{\begin{array}{c}\text{Total number of no-shows} \\ + \\ \text{Total number of appointments that are canceled no more than} \\ \text{2 days prior to the appointment time}\end{array}}{\text{Total number of appointments}}.$$

The number is 26.89% as estimated from the data for Clinical Genetics.

A common practice is to take advantage of such high no-show probability by scheduling more patients to a session than its actual capacity can handle. Using terminology defined in Section 3.9, we use P_{NS} as the no-show probability for every session. We also vary the

no-show penalty D in our experiments in the range $[2, 10]$. In this way, the pair (α, D) tunes the cost (3.9.1) of overbooking each session. The previous experiment corresponds to the case $\alpha = 1, D = \infty$.

Now the total benefit of a scheduling policy is equal to the sum of all benefit values (3.11.1), i.e., show probabilities, earned from patients less the overbooking costs (3.9.1). In particular, we apply the function (3.9.1) of overbooking cost to the actual practice as well. That is, in our experiment the total overbooking costs incurred under the actual practice does not depend on the actual overbooked number of patients, but rather on the expected costs (3.9.1) estimated a priori. The performance of each scheduling policy is reported as its total benefit relative to the total benefit of the actual practice.

Table 3.3 summarizes the performance of scheduling policies when $\alpha = 0.75$ and D ranges from 2 to 15. Generally the performance of all policies decreases as the penalty D increases because of the reduced benefit of overbooking.

Table 3.4 reports the performance of scheduling policies when $D = 3$ and α increases from 70% to 100%. The performance of all the scheduling policies reaches a limit for large values of α. This is because when α is large, there is a large surplus of capacity associated with low overbooking costs. In such cases, scheduling policies virtually cannot see any capacity constraint, and thus have very good performance. Overall, for all values of α, our Marginal Allocation Algorithm performs at least 30% better than actual practice.

Table 3.3: The total benefit of scheduling policies relative to LP upper bound under different values of penalty D. $\alpha = 0.75$.

D	Actual Strategy	Greedy	Bid-Price Heuristic	Separation Alg.	MAA
2	70.1%	81.5%	89.0%	82.1%	93.2%
3	68.7%	80.6%	86.7%	82.0%	92.4%
4	66.8%	80.0%	86.6%	82.5%	92.3%
5	64.5%	79.5%	87.2%	82.8%	92.3%
6	62.2%	79.2%	88.7%	82.5%	92.0%
7	59.7%	78.9%	88.0%	82.6%	92.0%
8	57.1%	78.7%	88.6%	82.5%	92.0%
9	54.5%	78.2%	88.4%	82.4%	92.0%
10	51.8%	77.8%	88.4%	82.1%	91.5%

Table 3.4: The total benefit of scheduling policies relative to LP upper bound under different values of α. $D = 3$.

α	Actual Strategy	Greedy	Bid-Price Heuristic	Separation Alg.	MAA
70%	62.7%	77.2%	88.3%	82.9%	92.2%
75%	68.7%	80.6%	86.7%	82.0%	92.4%
80%	70.8%	83.9%	88.9%	81.8%	93.4%
85%	71.4%	88.9%	92.5%	82.6%	94.5%
90%	71.1%	91.4%	94.4%	83.3%	94.8%
95%	70.7%	92.5%	95.9%	84.6%	95.8%
100%	70.5%	93.2%	95.7%	85.7%	96.2%

3.11.2 Consideration for Patient Availability

In the previous numerical experiments, patients who arrive in the same periods are treated as identical. However, in reality there is variability among patients' availability. In this section, we capture this variability by simulating a particular chosen patient's availability for a particular session of the week as being drawn from a given distribution. This experiment tests whether more complex heterogeneous patient types affect the comparative performance of our algorithm.

We model the heterogeneity of patient availability as follows. A patient cannot be assigned to a session if he is unavailable for it. Otherwise, the benefit for the session is still the show probability as modeled in the previous sections. We assume that each patient has the same availability pattern for every week. A patient is available for any session with probability P_A, and this event is independent of the availability for other sessions in the same week. We vary P_A from 15% to 100% to test the performance of all the scheduling policies we consider. When $P_A = 100\%$, the problem is reduced to the one in the last section, in which a patient can be assigned to any session.

Since we model 8 sessions in a week, one in the morning and one in the afternoon from Monday to Thursday (recall that there were very few appointments scheduled for Friday), each patient's availability can be represented by an 8-dimension binary vector. Then, patients arriving in each period are further divided into 2^8 patient types, with $r_{i,k,j} = 0$ if a patient of type $k \in \{1, 2, ..., 2^8\}$ arriving in period i is not available for session j.

We assume that the sessions offered by actual practice to patients were all available, so

Table 3.5: The total benefit of scheduling policies relative to LP upper bound under different values of P_A. $D = 3$, $\alpha = 0.7$.

P_A	Actual Strategy	Greedy	Bid-Price Heuristic	Separation Alg.	MAA
15.00%	88.8%	89.6%	96.0%	93.1%	96.4%
20.00%	77.6%	86.0%	93.5%	90.9%	95.1%
25.00%	72.5%	83.4%	92.1%	89.5%	94.2%
30.00%	70.0%	81.5%	91.4%	88.9%	93.7%
35.00%	68.4%	80.2%	91.4%	88.5%	93.5%
40.00%	67.2%	79.3%	90.8%	87.9%	93.2%
45.00%	66.3%	78.7%	91.5%	87.6%	93.0%
50.00%	65.7%	78.2%	90.7%	86.8%	92.7%
55.00%	65.1%	77.8%	90.5%	86.0%	92.7%
60.00%	64.7%	77.6%	89.3%	85.5%	92.7%
65.00%	64.3%	77.5%	88.7%	85.0%	92.8%
70.00%	64.0%	77.5%	88.9%	84.5%	92.5%
75.00%	63.7%	77.6%	88.6%	84.0%	92.4%
80.00%	63.4%	77.6%	87.7%	83.7%	92.3%
85.00%	63.2%	77.6%	88.3%	83.5%	92.4%
90.00%	63.0%	77.5%	88.4%	83.3%	92.5%
95.00%	62.9%	77.5%	88.0%	83.1%	92.4%
100.00%	62.7%	77.2%	88.3%	82.9%	92.2%

that the total benefit of actual practice is not affected by this newly modeled feature. The performance of each of the remaining scheduling policies is the averaged total benefit over $10,000$ runs of simulation. In each simulation we draw the same 2032 number of arrivals from data, but we randomly generate patient availability. For P_A ranging from 15% to 100%, Table 3.5 shows the performance of scheduling policies relative to the performance of actual practice. The relative performance is better for higher values of P_A, as there is more flexibility in scheduling when patients are available to more sessions. Even when P_A is as small as 15%, our Marginal Allocation Algorithm still performs 8% better than actual practice. The gap gradually increases to more than 40% as P_A increases.

3.12 Conclusions

We study a resource allocation model where arriving customers are assigned to resources that expire at given times. The reward of an assignment depends on both the type of customer and the resource. Our model is essentially an online edge-weighted bipartite matching problem with non-stationary arrivals.

The allocation algorithms we propose rely on solving both a linear program and a dynamic program. First, the algorithms solve a linear program to obtain an optimal static allocation. Then based on this static allocation, the algorithms divide the original dynamic problem into multiple dynamic subproblems. Each subproblem contains exactly one resource, as well as the types of demands that are assigned to the resource in the static allocation. For each subproblem, its allocation process is essentially a single-resource Markov decision process, so we can optimally solve it using dynamic programming. Our Marginal Allocation algorithm uses the marginal reward values given by the dynamic programs as the bid-prices for the corresponding resources. Upon an arrival of a customer, the algorithm allocates a resource that maximizes the marginal reward based on these bid prices. Unlike the conventional bid-price heuristic that uses constant bid prices, our Marginal Allocation algorithm computes bid prices that depend on both the remaining inventory of resources and the remaining time for allocation. In particular, our bid prices decrease in time, which captures the depreciation of unmatched resources.

For this problem, we prove the tightest known competitive ratios defined in (1.1.2). Specifically, we prove that our algorithms have competitive ratios bounded by $\max(\frac{1}{2}, 1 -$

$\sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{k}} + O(\frac{1}{k})$), where k is the minimum capacity of a resource. Furthermore, we show that $\frac{1}{2}$ is the best constant competitive ratio that can be achieved.

As part of our collaboration with Columbia University Medical Center, we simulate the empirical performance of our algorithms using a 10-year dataset from CUMC. Specifically, we apply our allocation algorithms in the appointment scheduling problem, in which the goal is to reduce the overall no-show probability by better matching patients to appointment slots they prefer. We find that our algorithms perform better than simple heuristics in most test scenarios. In particular, our Marginal Allocation algorithm is 21% more effective than the actual scheduling strategy used in the hospital.

Chapter 4

Advance Service Reservations with Heterogeneous Customers

4.1 Introduction

We study a fundamental model of resource allocation in which a finite number of resources must be assigned in an online manner to a heterogeneous stream of customers. The customers arrive randomly over time according to known stochastic processes. Each customer requires a specific amount of capacity and has a specific preference for each of the resources, with some resources being feasible for the customer and some not. The system must find a feasible assignment of each customer to a resource or must reject the customer. The aim is to maximize the total expected capacity utilization of the resources over the time horizon.

Compared with the model in Chapter 3, the model we study in this chapter assumes that customers have heterogeneous capacity requirements. As we shall show, much of the effort

in our algorithms and their analysis is directed towards taking care of these differences in capacity requirement.

This model has application in multiple areas, including services, online advertising, and freight transportation. We now explain a few of the applications.

Service Reservation. In services such as healthcare, the resources can correspond to service sessions. For example, a resource might be a Monday afternoon session from 1 to 5 PM with Dr. Smith. The customers are patients who arrive to book appointments over time. Based on a patient's urgency, type of visit, arrival time, and preferences, the patient might require a specific length of visit and might be preferably assigned only to a subset of sessions. Upon the arrival of a patient, the system has to reserve a part of a session for the patient. This appointment decision typically takes place immediately. If an appointment cannot be found, the system must reject the patient.

Generalized Adwords. In online advertizing, the resources correspond to advertisers. The capacity of each resource corresponds to the budget of the corresponding advertiser. Ad impressions arrive randomly over time. Each impression, depending on its characteristics, commands a known non-negative bid from each of the advertisers. When an impression occurs, the ad platform must allocate it to an advertiser for use. The ad platform earns the bid, and the budget of the advertiser is depleted by the same amount. The aim of the ad platform is to maximize the expected revenue earned. Our model is more general than adwords models, as we allow bids to have arbitrary sizes, whereas adwords model tend to assume that bid sizes must be very small relative to

the budgets, or that each bid must be truncated by the remaining budget (Mehta, 2012).

Freight Allocation. Freight carriers such as motor carriers, railroad companies, and shipping companies have fleets of containers that can be deployed to move loads from specific origins to destinations. The assignment of containers to routes are tactical decisions that are performed on a larger time scale. Suppose that we focus on a single route. Each container, with its specific departure and arrival time, corresponds to a resource. Customer demands for the route arrive randomly over time. Each demand unit has a specific size and delivery time line. As each demand unit arrives, the operational decision is how to assign the demand unit to a specific container in the fleet (Spivey and Powell, 2004). This assignment generates a quoted time of delivery for the customer, reduces the available capacity in the container, and earns the system an amount that can be roughly proportional to the amount of capacity consumed.

Our model captures most, if not all, of the features of the above applications. Specifically, we consider a continuous-time planning horizon. There are m resources with known capacities. There are n customer types. Each customer type is associated with a known stochastic arrival process. Each customer can be assigned to a known subset of the resources, and consumes a known amount of each resource that it is assigned to. The system aims to assign customers to resources immediately and irrevocably as they arrive in order to maximize the total expected amount of resources used.

This model allows any non-homogeneous customer arrival processes, so it shares several

advantages of assuming non-stationary arrivals with the model in Chapter 3. Among those advantages explained in Chapter 3, the most important one is that the non-stationarity of arrivals captures the expiration of resources.

In this thesis chapter, we aim to develop near-optimal algorithms that are robust and easy to compute. We apply the definition (1.1.2) of competitive ratio, and propose 0.321-competitive online algorithms. Further, we show that an upper bound on the competitive ratio of any algorithm is 1/2. Ours are the first algorithms with performance guarantees for the advance reservation of service with heterogeneous customer needs and preferences. They are also the first algorithms with constant competitive ratios for the adwords problem without any assumption on the bid size and on the stationarity of the arrival process. Despite the conservative performance characterization, we show that our algorithms perform extremely well compared to common heuristics as demonstrated on a real data set from a large hospital system in New York City.

4.2 Literature Review

Our model falls within the literature on online resource allocation, which we reviewed in Section 3.2.3. In the following sub-sections, we provide more detailed reviews on two subclasses of this literature, the Adwords problem and the dynamic knapsack problem, which are more closely related to the model in this thesis chapter.

In addition, our work falls into the *advance scheduling* paradigm of the appointment scheduling literature, which we reviewed in Section 2.2.1.

4.2.1 Adwords problems

Our model generalizes adwords problems. Considerable work has been done in this area. If each bid is truncated by the remaining budget, it was shown by Mehta (2012) that a greedy algorithm achieves a worst-case competitive ratio of 1/2 in the adversarial-demand model. For adwords models in which demands arrive in random orders and bids are small, Goel and Mehta (2008) prove that a greedy algorithm achieves a worst-case ratio of $1-1/e$. Mirrokni, Gharan, and Zadimoghaddam (2012) later improve this ratio to 0.76. If demands are i.i.d., but bids are not necessarily small, Devanur, Jain, Sivan, and Wilkens (2011) show that a greedy algorithm achieves the worst-case ratio of $1-1/e$. Later, Devanur, Sivan, and Azar (2012) show that under stochastic demands, if the bid to budget ratio is at most $1/d$, $d \geq 2$, and if bids can be truncated, then there is an algorithm that achieves a worst-case ratio of $1-1/\sqrt{2\pi d}$. If the bid to budget ratios at most ϵ^2, then the algorithm achieves a worst-case ratio of $1-O(\epsilon)$. Finally, no algorithm can achieve a worst-case ratio that is better than $1-o(1/\sqrt{d})$ when the bid to budget ratios are as large as $1/d$. The main difference between our work and this literature is that we do not make the assumption of truncated bids, small bids, or i.i.d. demand. Furthermore, we study the ratio of expected performance between the online and optimal offline algorithm, rather than the worst-case ratio.

4.2.2 Dynamic knapsack problems

Our problem is related to multi-constrained dynamic knapsack problems (MKP). In these problems, a set of randomly arriving items must be packed into one or more knapsacks,

Table 4.1: Results on adwords models.

Reference	Lower bound achieved	Assumption
Our work	0.321	stochastic demand
	$1 - 1/\sqrt{2\pi d} + O(1/d)$	stochastic demand, bid to budget ratio at most $1/d$
Mehta (2012)	0.5	adversarial demand, truncated bids
Goel and Mehta (2008)	$1 - 1/e \approx 0.63$	randomly ordered demand, small bids
Mirrokni et al. (2012)	0.76	randomly ordered demand, small bids
Devanur et al. (2012)	$1 - 1/\sqrt{2\pi d}$	stochastic demands, bid to budget ratio at most $1/d$, truncated bids
Devanur et al. (2011)	$1 - 1/e \approx 0.63$	i.i.d. demand, truncated bids

respecting the capacity constraints of the knapsacks. The goal is to maximize the value of the items packed. Note that our problem is different from these dynamic multi-knapsack problems. In our problem, each customer can be satisfied using one of a subset of resources, rather than any resource, due to preferences, urgency, priorities, etc. These feasibility constraints must be accounted for in the assignment decision. In contrast, a knapsack problems, an object can be placed into any knapsack, as long as the capacity constraints are satisfied.

Dynamic-programming characterizations have been studied in the case of one knapsack Papastavrou, Rajagopalan, and Kleywegt (1996), Kleywegt and Papastavrou (1998); Van Slyke and Young (2000), Lin, Lu, and Yao (2008), and Chen and Ross (2014). Some results generalize to the MKP but these results are not sufficient to yield provable approximations (Van Slyke and Young, 2000). Many authors have studied online algorithms for the MKP. It is shown in Marchetti-Spaccamela and Vercellis (1995) that no online algorithm for MKP exists with a constant worst-case competitive ratio. Therefore, Marchetti-Spaccamela and Vercellis (1995) and Lueker (1998) study algorithms with bounded additive differences away from the offline optimal. Finally, Chakrabarty, Zhou, and Lukose (2013) design an

algorithm with a bounded worst-case competitive ratio, assuming that the size of each item is very small relative to the capacity, and the value-to-size ratio of each item is upper and lower bounded by two constants. Our model is different from the MKP model because our resources are not interchangeable, as customer preferences for them might be different. Our approach also differs from existing MKP approaches in that we seek to bound the ratio of expected performance between the online and optimal offline algorithm, rather than the worst-case ratio.

This thesis chapter is organized as follows. We specify the model and performance metric in Section 4.3. In Section 4.4, we prove that 0.5 is an upper bound on the competitive ratio of any online algorithm for this problem. We derive an upper bound on the optimal offline objective in Section 4.5. In Section 4.6, we design a basic online algorithm with a competitive ratio of $0.5(1 - 1/e) \approx 0.316$, which serves to illustrate our key ideas. In Section 4.7, we refine the algorithm to employ resource sharing in order to obtain an improved competitive ratio of 0.321, as well as an improved empirical performance. In Section 4.8, we compare the empirical performance of our algorithms against two commonly used heuristics by simulating the algorithms on appointment-scheduling data obtained from a large hospital system in New York City.

4.3 Model and Performance Metric

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and consider a continuous horizon $[0, T]$. There are m resources and n customer types. Resource $j \in [m]$ has capacity $c_j \in \mathbb{R}_+$. Customers

of type $i \in [n]$ arrive according to a non-homogeneous Poisson process with rate $\lambda_i(t)$, for $t \in [0, T]$. The arrival rates of all the customer types are known. When a customer arrives, one of the m resources needs to be immediately allocated to the customer, or the customer must be rejected. If resource j is allocated to a customer of type i, exactly u_{ij} units of resource j must be provided. We assume that the $u_{ij} \in [0, c_j]$, $\forall i \in [n]$ and $j \in [m]$, are known. The reward earned for the assignment of customer type i to resource j is also u_{ij}. The objective is to maximize the total expected reward over the horizon, which equivalently maximizes total resource utilization.

We apply the definition (1.1.2) of competitive ratio. More specifically, in this model, the problem instance I is a sample path of customer arrivals over the entire horizon. Let $\mathrm{ALG}(I)$ be the total amount of resources allocated by an online algorithm ALG. Let $\mathrm{OPT}(I)$ be the total amount of resources allocated by an optimal offline algorithm OPT. We are interested in analyzing online algorithms ALG with bounded ratio $\frac{\mathbf{E}[\mathrm{ALG}(I)]}{\mathbf{E}[\mathrm{OPT}(I)]}$.

4.4 Upper Bound on the Competitive Ratio

In this section, we show that 0.5 is an upper bound on competitive ratio of any online algorithm for this problem.

Proposition 4.4.1. *The competitive ratio of any online algorithm is at most* 0.5.

Proof. Consider an input with two customer types and a single resource. Assume that the horizon is $[0, 1]$. The capacity of the resource is $c_1 = 1$.

- Type-1 customers have a very large arrive rate in time $[0, 0.5]$, but their arrival rate is 0 after time 0.5. In particular, $\Lambda_1 = \int_0^{0.5} \lambda_1(t)dt \gg 1$, so that we can ignore the event that no type-1 customer arrives. Their utilization for the single resource is $u_{11} = \epsilon/\Lambda_1$ for some very small value ϵ.

- Type-2 customers arrive in time $(0.5, 1]$. They have a very small arrival rate $\Lambda_2 = \int_{0.5}^{1} \lambda_2(t)dt = \epsilon$. Their utilization for the resource is $u_{21} = c_1 = 1$.

Since customers of type 2 request the entire resource, the offline algorithm will allocate the resource to a type-2 customer if there is one. The probability that at least one type-2 customer arrives is $1 - e^{-\Lambda_2} = \Lambda_2 + o(\Lambda_2^2) = \epsilon + o(\epsilon^2)$. With probability $1 - o(\Lambda_2) = 1 - o(\epsilon)$, no type-2 customer will arrive, in which case the optimal offline algorithm will accept as many type-1 customers as possible. The expected total utilization of all type-1 customers is $u_{11} \cdot \Lambda_1 = \epsilon$. Suppose $\epsilon \ll c_1 = 1$. Then all type-1 customer can be accepted. In sum, the expected amount of resource allocated by an optimal offline algorithm is

$$\begin{aligned} & 1 \cdot (\epsilon + o(\epsilon^2)) + \epsilon \cdot (1 - o(\epsilon)) \\ =& 2\epsilon + o(\epsilon^2). \end{aligned}$$

The decision of an online algorithm is whether to accept type-1 customers during time $[0, 0.5]$. If it does accept type-1 customers, the online algorithm earns $u_{11} \cdot \Lambda_1 = \epsilon$ in expectation. Otherwise, with probability $\Lambda_2 + o(\Lambda_2^2)$ it earns u_{21}, which is $u_{21}(\Lambda_2 + o(\Lambda_2^2)) = \epsilon + o(\epsilon^2)$ in expectation. In sum, an online algorithm cannot allocate more than $\epsilon + o(\epsilon^2)$ in

expectation. Thus, an upper bound on the competitive ratio is

$$(\epsilon + o(\epsilon^2))/(2\epsilon + o(\epsilon^2)),$$

which tends to 0.5 in the limit as $\epsilon \to 0$.

□

4.5 Upper Bound on the Optimal Offline Objective

We derive an upper bound on the optimal offline objective, namely $\mathbf{E}[\mathrm{OPT}(I)]$. Since $\mathbf{E}[\mathrm{OPT}(I)]$ is very hard to analyze due to its complex offline properties, we are interested in developing an upper bound on $\mathbf{E}[\mathrm{OPT}(I)]$, which is more tractable. We will later compare the performance of our online algorithms against this upper bound, rather than directly with $\mathbf{E}[\mathrm{OPT}(I)]$.

Our upper bound can be formulated as a solution to a static LP as follows. For customers of type i, let

$$\Lambda_i = \int_0^T \lambda_i(t)dt$$

be the expected total number of arrivals over the horizon. Consider a static LP (LP) that allocates the expected demands Λ_i, $i \in [n]$, to the capacities c_j, $j \in [m]$. The decision variable x_{ij} of the LP stands for the average number of customers of type i to be allocated

to resource j. The LP produces a fractional assignment.

$$\begin{aligned} V^{LP} = \max_{x_{ij}} & \sum_{i\in[n]}\sum_{j\in[m]} x_{ij}u_{ij} \\ \text{s.t.} & \sum_{i\in[n]} x_{ij}u_{ij} \le c_j, \quad \forall j \in [m] \\ & \sum_{j\in[m]} x_{ij} \le \Lambda_i, \quad \forall i \in [n]. \end{aligned} \tag{4.5.1}$$

By the linearity of assignment problems, it can be shown easily that

Proposition 4.5.1. *V^{LP} is an upper bound on $\mathbf{E}[OPT(I)]$.*

Proof. Let $a_i(I)$ be the actual number of arrivals of type-i customers in sample path I. Let $\tilde{x}(I)$ be a corresponding optimal offline (fractional) assignment. Then $\tilde{x}(I)$ must satisfy

$$\sum_{i\in[n]} \tilde{x}_{ij}(I)u_{ij} \le c_j, \quad \forall j \in [m],$$

$$\sum_{j\in[m]} \tilde{x}_{ij}(I) \le a_i(I), \quad \forall i \in [n].$$

Taking expectation on both sides, we obtain

$$\sum_{i\in[n]} \mathbf{E}[\tilde{x}_{ij}(I)]u_{ij} \le c_j, \quad \forall j \in [m],$$

$$\sum_{j\in[m]} \mathbf{E}[\tilde{x}_{ij}(I)] \le \mathbf{E}[a_i(I)] = \Lambda_i, \quad \forall i \in [n].$$

These inequalities imply that $\mathbf{E}[\tilde{x}(I)]$ is a feasible solution to (4.5.1). Thus V^{LP} must be an upper bound on $\sum_{i\in[n],j\in[m]} \mathbf{E}[\tilde{x}_{ij}(I)]u_{ij}$, which proves the proposition.

□

4.6 Basic Online Algorithm

As a warm up, we design an online algorithm which we prove to have a competitive ratio of at least $0.5(1 - 1/e) \approx 0.316$. This algorithm serves to illustrate the following two main ideas, which we will later refine to obtain an improved bound.

- *LP-based random routing.* We make use of an optimal solution x^* to the static LP (4.5.1) to route customers to resources. Note that this solution assigns demand to supply at an aggregate level, in the expected sense. Given a solution x^*, for each arriving customer of type $i \in [n]$, we randomly route the customer to each candidate resource $j \in [m]$ independently with probability x^*_{ij}/Λ_i. We say a customer is *routed* to resource j if resource j is chosen as a candidate resource for the customer. By random routing, we can conclude that the arrival process of type-i customers who are routed to resource j is a non-homogeneous Poisson process with rate $\lambda_i(t)\frac{x^*_{ij}}{\Lambda_i}$, for $t \in [0, T]$.

- *Reservation by customer type.* After the random routing stage, we make binary admission decisions about whether to commit each resource j to each customer i who is routed to j. If the decision is 'no', we reject the customer. We make this admission decision as follows. For each resource j, we divide the candidate customer types who will potentially be routed to j into two sets based on utilization u_{ij}. Set $L_j \subseteq [n]$ consists

of customer types of which the utilizations u_{ij} are larger than $c_j/2$. Mathematically,

$$L_j = \{i \in [n] : u_{ij} > c_j/2\}.$$

The other set $S_j = [n] - L_j$ consists of customer types with utilization u_{ij} that are at most $c_j/2$.

For each resource j, our algorithm chooses one set, either S_j or L_j, whichever has the higher expected total utilization for resource j. The algorithm exclusively reserves resource j for customers whose types are in the chosen set. The algorithm rejects all customer types in the complementary set. This step is meant to resolve conflict in resource usage among different customer types by restricting use of the resource to the most promising subset of customer types.

Large-or-Small (LS) Algorithm:

1. (Pre-processing step) Solve the LP (4.5.1). Let x^* be an optimal solution. For each resource j, define

$$U_j^L \equiv \sum_{i \in L_j} x_{ij}^* u_{ij}$$

as the amount of resource j allocated to customer types in L_j by the static LP. Similarly, define

$$U_j^S \equiv \sum_{i \in S_j} x_{ij}^* u_{ij}$$

as the amount of resource j allocated to customer types in S_j by the LP.

2. (Reservation step) Reserve the resource j for customer types in the set L_j if $U_j^L \geq U_j^S$. Otherwise, reserve resource j for customer types in the set S_j.

3. (Random routing step) Upon an arrival of a type-i customer, randomly pick a resource j with probability x_{ij}^*/Λ_i.

4. (Admission step) If the remaining capacity of resource j is at least u_{ij} and i belongs to the set that is reserved for j in the pre-processing step then accept the customer. Otherwise, reject the customer.

As a consequence of the random routing process, we can separate the analysis for every resource $j \in [m]$. Define

$$U_j \equiv U_j^L + U_j^S$$

as the total amount of resource j allocated by the LP. We will show that in expectation, at least

$$\frac{1}{2}\left(1 - \frac{1}{e}\right) U_j$$

units of resource j will be occupied in LS.

We will use the following technical lemma, which bounds the tail expectation of demands following a compound Poisson distribution.

Lemma 4.6.1. *Let $X_1, X_2, X_3, ...$ be a sequence of i.i.d. random variables that take values from $[0, \beta]$, for some given $\beta \in [0, \frac{1}{l}]$ with $l \geq 2$ being an integer. Let N be a Poisson random*

variable. For any given $\alpha \in [0,1]$*, if*

$$\mathbf{E}\left[\sum_{k=1}^{N} X_k\right] = \alpha,$$

we must have

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{N} X_k, 1-\beta\right)\right] \geq \frac{1-\beta}{l-1}\mathbf{E}[\min(N', l-1)],$$

where N' *is a Poisson random variable with mean* $\alpha(l-1)/(1-\beta)$*. In particular, when* $l = 2$*,*

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{N} X_k, 1-\beta\right)\right] \geq (1-\beta)\left(1-e^{-\alpha/(1-\beta)}\right).$$

Proof. Let $Z_1, Z_2, Z_3, ...,$ be a sequence of i.i.d. random variables each following a uniform distribution over $[0, (1-\beta)/(l-1)]$. For every $k = 1, 2, ...,$ define a function

$$\tilde{X}_k(x) \equiv \frac{1-\beta}{l-1}\mathbf{1}(Z_k < x),$$

where $\mathbf{1}(\cdot)$ denotes an indicator function.

Since $\beta \in [0, 1/l]$, we must have $\beta \leq (1-\beta)/(l-1)$. It is then easy to check that for any $x \in [0, \beta]$, we have

$$\mathbf{E}[\tilde{X}_k(x)] = \frac{1-\beta}{l-1} \cdot \frac{x}{(1-\beta)/(l-1)} = x.$$

Thus, we have for every $k = 1, 2, ...,$

$$\mathbf{E}[\tilde{X}_k(X_k)|X_k] = X_k.$$

According to Jensen's inequality, we must have

$$\min\left(\sum_{k=1}^{N} X_k, 1-\beta\right) \geq \mathbf{E}\left[\min\left(\sum_{k=1}^{N} \tilde{X}_k(X_k), 1-\beta\right) | X_1, X_2, \ldots\right]$$

$$\Longrightarrow \mathbf{E}\left[\min\left(\sum_{k=1}^{N} X_k, 1-\beta\right)\right] \geq \mathbf{E}\left[\min\left(\sum_{k=1}^{N} \tilde{X}_k(X_k), 1-\beta\right)\right].$$

Since $\tilde{X}_k(X_k)$ is either 0 or $(1-\beta)/(l-1)$, the term $\sum_{k=1}^{N} \tilde{X}_k(X_k)$ has the same distribution as $N'(1-\beta)/(l-1)$ where N' is a Poisson random variable with mean

$$\mathbf{E}[N'] = \frac{l-1}{1-\beta}\mathbf{E}\left[\sum_{k=1}^{N} \tilde{X}_k(X_k)\right] = \frac{l-1}{1-\beta}\mathbf{E}\left[\sum_{k=1}^{N} X_k\right] = \frac{l-1}{1-\beta}\alpha.$$

Therefore,

$$\begin{aligned}\mathbf{E}[\min(\sum_{k=1}^{N} X_k, 1-\beta)] \geq & \mathbf{E}\left[\min\left(\sum_{k=1}^{N} \tilde{X}_k(X_k), 1-\beta\right)\right] \\ = & \mathbf{E}[\min(N'(1-\beta)/(l-1), 1-\beta)] \\ = & \frac{1-\beta}{l-1}\mathbf{E}[\min(N', l-1)].\end{aligned}$$

When $l = 2$, this equals $(1-\beta)\left(1-e^{-\alpha/(1-\beta)}\right)$.

□

We are now ready to prove the competitive ratio of LS. The idea is to compare the utilization of each resource j under LS with the utilization of resource j under OPT. The latter is given by $U_j^S + U_j^L$. The former depends on the choice of the set reserved for j, either

L_j or S_j. With either choice, we can gauge the total expected utilization, in some cases using Lemma 4.6.1, to obtain a lower bound. We then repeat this comparison for all resources j to arrive at a global bound.

Theorem 4.6.2. *LS is at least* $(1 - 1/e)/2$*-competitive.*

Proof. For each resource $j \in [m]$ there are two cases.

- Case 1: $U_j^L \geq U_j^S$. Let Y_j^L be the total number of customers who are routed to resource j and whose types are in L_j. Y_j^L is a Poisson random variable with mean

$$\mu_j^L \equiv \mathbf{E}[Y_j^L] = \sum_{i \in L_j} x_{ij}^*.$$

Conditional on the value of Y_j^L, the amount of resource j requested by each of the Y_j^L customers is i.i.d. and has mean

$$\bar{u}_j^L \equiv \frac{\sum_{i \in L_j} x_{ij}^* u_{ij}}{\sum_{i \in L_j} x_{ij}^*} = \frac{U_j^L}{\mu_j^L}.$$

If $Y_j^L = 1$, the expected amount of resource j taken by that only customer is just $\bar{u}_j^L$. Thus, we get an expected reward $P(Y_j^L = 1)\bar{u}_j^L$ from the event $Y_j^L = 1$.

If $Y_j^L > 1$, only the first customer can take resource j, and all the other $Y_j^L - 1$ customers will be rejected due to lack of remaining capacity. The expected amount of resource taken by the first customer may not be $\bar{u}_j^L$ since arrivals are non-homogeneous,

but must be still greater than $c_j/2$. Thus, we get an expected reward $P(Y_j^L > 1)c_j/2$ from the event $Y_j^L > 1$.

In sum, the expected amount of resource taken by these Y_j^L customers is at least

$$\begin{aligned} & P(Y_j^L = 1)\bar{u}_j^L + P(Y_j^L > 1)c_j/2 \\ =& \mu_j^L e^{-\mu_j^L}\bar{u}_j^L + \left(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}\right) c_j/2 \\ =& U_j^L e^{-\mu_j^L} + \left(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}\right) c_j/2. \end{aligned} \tag{4.6.1}$$

We obtain a lower bound on (4.6.1) by minimizing its value with respect to μ_j^L. We can deduce that

$$\begin{aligned} & \frac{d}{d\mu_j^L}\left[U_j^L e^{-\mu_j^L} + \left(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}\right) c_j/2\right] = 0 \\ \Longrightarrow & -U_j^L e^{-\mu_j^L} + \left(1 + e^{-\mu_j^L} - e^{-\mu_j^L} + \mu_j^L e^{-\mu_j^L}\right) c_j/2 = 0 \\ \Longrightarrow & \mu_j^L = 2U_j^L/c_j. \end{aligned} \tag{4.6.2}$$

It is easy to check that (4.6.1) is minimized at solution (4.6.2), and the corresponding minimum value of (4.6.1) is

$$\begin{aligned} & U_j^L e^{-2U_j^L/c_j} + \left(1 - e^{-2U_j^L/c_j} - 2U_j^L/c_j e^{-2U_j^L/c_j}\right) c_j/2 \\ =& \left(1 - e^{-2U_j^L/c_j}\right) c_j/2 \\ \geq& \left(1 - e^{-U_j/c_j}\right) c_j/2 \end{aligned}$$

$$\geq \left(1 - e^{-1}\right) U_j/2.$$

The last step follows since $U_j/c_j \leq 1$.

- Case 2: $U_j^L < U_j^S$. Let Y_j^S be the total number of customers who are routed to resource j and whose types are in S_j. Y_j^S is a Poisson random variable with mean

$$\mathbf{E}[Y_j^S] = \sum_{i \in S_j} x_{ij}^*.$$

Let $W_1, W_2, W_3, ...$, be a sequence of i.i.d. random variables each having distribution

$$P(W_k \leq x) = \sum_{i \in S_j} \mathbf{1}(u_{ij} \leq x) \frac{x_{ij}^*}{\sum_{l \in S_j} x_{lj}^*}, \quad \forall k = 1, 2, ...$$

Here each W_k can be seen as the random amount of resource j requested by one of the Y_j^S customers conditional on the value of Y_j^S. Then $\sum_{k=1}^{Y_j^S} W_k$ represents the total random amount of resource j requested by all the Y_j^S customers. It is easy to check that

$$\mathbf{E}\left[\sum_{k=1}^{Y_j^S} W_k\right] = U_j^S \geq \frac{U_j}{2}$$

$$\Longrightarrow \mathbf{E}\left[\sum_{k=1}^{Y_j^S} \frac{W_k}{c_j}\right] = \frac{U_j^S}{c_j} \geq \frac{U_j}{2c_j}.$$

If $\sum_{k=1}^{Y_j^S} W_k \leq c_j$, all the Y_j^S customers will be accepted, and we will get total reward $\sum_{k=1}^{Y_j^S} W_k$ from resource j.

If $\sum_{k=1}^{Y_j^S} W_k > c_j$, some of the Y_j^S customers must be rejected due to lack of capacity. But whenever a customer is rejected, the remaining available capacity of resource j must be strictly less than $0.5c_j$, since $W_k/c_j \in [0, 0.5]$ w.p.1 for every k.

In sum, the total reward we get from resource j is at least

$$\begin{aligned}&\sum_{k=1}^{Y_j^S} W_k \cdot \mathbf{1}(\sum_{k=1}^{Y_j^S} W_k \le c_j) + 0.5c_j \cdot \mathbf{1}(\sum_{k=1}^{Y_j^S} W_k > c_j)\\ \ge &\min(\sum_{k=1}^{Y_j^S} W_k, 0.5c_j).\end{aligned}$$

Its expected value can be written as

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j^S} W_k, \frac{c_j}{2}\right)\right] = c_j \mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j^S} \frac{W_k}{c_j}, \frac{1}{2}\right)\right].$$

We then apply Lemma 4.6.1 to obtain

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j^S} \frac{W_k}{c_j}, \frac{1}{2}\right)\right] \ge \frac{1}{2}\left(1 - e^{-2U_j^S/c_j}\right) \ge \frac{1}{2}\left(1 - e^{-\frac{U_j}{c_j}}\right)$$

$$\implies \mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j^S} \frac{W_k}{c_j}, \frac{1}{2}\right)\right] \ge \frac{U_j}{2c_j}\left(1 - \frac{1}{e}\right)$$

$$\implies \mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j^S} W_k, \frac{c_j}{2}\right)\right] \ge \frac{U_j}{2}\left(1 - \frac{1}{e}\right).$$

In sum, in both cases the expected amount of resource j allocated to customers is at least $U_j(1 - 1/e)/2$. Summing over every resource $j \in [m]$, we can obtain the performance

guarantee of our algorithm

$$\sum_{j\in[m]} U_j \cdot \frac{1}{2}\left(1-\frac{1}{e}\right) = V^{LP} \cdot \frac{1}{2}\left(1-\frac{1}{e}\right) \geq \mathbf{E}\left[OPT(I)\right] \cdot \frac{1}{2}\left(1-\frac{1}{e}\right).$$

□

As a corollary, we can obtain simple bounds on the performance of the LS algorithm in the case that every u_{ij} is bounded away from c_j. The bounds follow directly from Lemma 4.6.1.

Corollary 4.6.3. *If there is some integer $d \geq 2$ for which $u_{ij} \leq \frac{c_j}{d}$ for all i and j, then the competitive ratio of LS is at least*

$$1 - e^{-d}\sum_{i=d}^{\infty}(i-d+1)\frac{d^{i-1}}{i!} = 1 - \frac{1}{\sqrt{2\pi d}} + O(1/d).$$

Proof. Similar to the proof of Theorem 4.6.2, since the first case can be eliminated, the second case remains. Let Y_j be the total number of customers who are routed to resource j. Then Y_j is a Poisson random variable with mean

$$\mathbf{E}[Y_j] = \sum_{i\in[n]} x^*_{ij}.$$

Let $W_1, W_2, W_3, ...$, be a sequence of i.i.d. random variables each having distribution

$$P(W_k \leq x) = \sum_{i\in[n]} \mathbf{1}(u_{ij} \leq x)\frac{x^*_{ij}}{\sum_{l\in[n]} x^*_{lj}}, \ \ \forall k = 1, 2, ...$$

Here each W_k can be seen as the random amount of resource j requested by one of the Y_j customers conditional on the value of Y_j. Then $\sum_{k=1}^{Y_j} W_k$ represents the total random amount of resource j requested by all the Y_j customers. It is easy to check that

$$\mathbf{E}\left[\sum_{k=1}^{Y_j} W_k\right] = U_j.$$

$$\Longrightarrow \mathbf{E}\left[\sum_{k=1}^{Y_j} \frac{W_k}{c_j}\right] = \frac{U_j}{c_j}.$$

Since $W_k/c_j \in [0, 1/d]$ for every k, the expected amount of resource j taken by these Y_j customers is at least

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j} W_k, c_j - c_j/d\right)\right] = c_j \mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j} \frac{W_k}{c_j}, 1 - 1/d\right)\right].$$

We then apply Lemma 4.6.1 to obtain

$$\mathbf{E}\left[\min\left(\sum_{k=1}^{Y_j} \frac{W_k}{c_j}, 1 - 1/d\right)\right] \geq \frac{1 - 1/d}{d-1}\mathbf{E}[\min(N', d-1)] = \frac{1}{d}\mathbf{E}[\min(N', d-1)],$$

where N' is a Poisson random variable with mean $U_j/c_j \cdot (d-1)/(1-1/d) = U_j/c_j \cdot d$. Let N be a Poisson random variable with mean $\mathbf{E}[N] = d$ that is independent of N'. We have $\mathbf{E}[N] \geq \mathbf{E}[N']$ since $U_j/c_j \leq 1$. We can further deduce that

$$\frac{1}{d}\mathbf{E}[\min(N', d-1)]$$

$$
\begin{aligned}
=&\frac{1}{d}\left[\sum_{i=1}^{d-1} i\cdot e^{-\mathbf{E}[N']}\frac{(\mathbf{E}[N'])^i}{i!}+\sum_{i=d}^{\infty}(d-1)e^{-\mathbf{E}[N']}\frac{(\mathbf{E}[N'])^i}{i!}\right]\\
=&\frac{1}{d}\mathbf{E}[N']\left[\sum_{i=0}^{d-2} e^{-\mathbf{E}[N']}\frac{(\mathbf{E}[N'])^i}{i!}+\sum_{i=d-1}^{\infty}\frac{d-1}{i+1}\cdot e^{-\mathbf{E}[N']}\frac{(\mathbf{E}[N'])^i}{i!}\right]\\
=&\frac{1}{d}\mathbf{E}[N']\mathbf{E}[\min(1,\frac{d-1}{N'+1})]\\
\geq&\frac{1}{d}\mathbf{E}[N']\mathbf{E}[\min(1,\frac{d-1}{N+1})]\\
=&\frac{1}{d}\frac{\mathbf{E}[N']}{\mathbf{E}[N]}\cdot\mathbf{E}[N]\mathbf{E}[\min(1,\frac{d-1}{N+1})]\\
=&\frac{1}{d}\frac{\mathbf{E}[N']}{\mathbf{E}[N]}\cdot\mathbf{E}[N]\left[\sum_{i=0}^{d-2} e^{-\mathbf{E}[N]}\frac{(\mathbf{E}[N])^i}{i!}+\sum_{i=d-1}^{\infty}\frac{d-1}{i+1}\cdot e^{-\mathbf{E}[N]}\frac{(\mathbf{E}[N])^i}{i!}\right]\\
=&\frac{1}{d}\frac{\mathbf{E}[N']}{\mathbf{E}[N]}\cdot\left[\sum_{i=1}^{d-1} i\cdot e^{-\mathbf{E}[N]}\frac{(\mathbf{E}[N])^i}{i!}+\sum_{i=d}^{\infty}(d-1)e^{-\mathbf{E}[N]}\frac{(\mathbf{E}[N])^i}{i!}\right]\\
=&\frac{1}{d}\frac{\mathbf{E}[N']}{\mathbf{E}[N]}\mathbf{E}[\min(N,d-1)]\\
=&\frac{U_j}{c_j}\cdot\frac{1}{d}\mathbf{E}[\min(N,d-1)]\\
=&\frac{U_j}{c_j}\cdot\frac{1}{d}(\mathbf{E}[N]-\mathbf{E}[\max(N-d+1,0)])\\
=&\frac{U_j}{c_j}\cdot\left(1-e^{-d}\sum_{i=d}^{\infty}(i-d+1)\frac{d^{i-1}}{i!}\right).
\end{aligned}
$$

Since U_j/c_j is an upper bound on the performance of the offline policy, the competitive ratio attained by the LS algorithm for resource j, hence overall, is $1-e^{-d}\sum_{i=d}^{\infty}(i-d+1)\frac{d^{i-1}}{i!}$. When d tends to infinity, this competitive ratio behaves as

$$1-e^{-d}\sum_{i=d}^{\infty}(i-d+1)\frac{d^{i-1}}{i!}$$

$$
\begin{aligned}
=&1 - e^{-d}\frac{d^d}{d!} - \frac{1}{d}e^{-d}\sum_{i=d}^{\infty}\frac{d^i}{i!} \\
=&1 - \frac{1}{\sqrt{2\pi d}} + O(1/d).
\end{aligned}
$$

□

For $d = 2$, the above bound is approximately 0.43.

Theorem 4.6.4. *The competitive ratio* $(1 - 1/e)/2$ *is tight for LS.*

Proof. We prove the theorem by constructing a special case where the total expected reward of LS is $\mathbf{E}[OPT(I)] \cdot (1 - 1/e)/2$.

Let $n = 2$. Let $u_{1j} = 0.1$ and $u_{2j} = 0.5 + \epsilon$ for every j, for some given small ϵ. Let $c_j = 1$ for every j. The expected number of arrivals is $\Lambda_1 = 5$ and $\Lambda_2 = 0.5/(0.5 + \epsilon)$. We set m to be very large such that (with probability very close to 1) each demand unit can find a distinct resource to be assigned to. Thus we have

$$\mathbf{E}[OPT(I)] = \Lambda_1 u_{1j} + \Lambda_2 u_{2j} = 5 \cdot 0.1 + 0.5/(0.5 + \epsilon) \cdot (0.5 + \epsilon) = 1.$$

On the other hand, suppose when we apply the LS algorithm, the LP solution is $x^*_{11} = \Lambda_1$, $x^*_{21} = \Lambda_2$ and $x^*_{ij} = 0$ for all other i, j. Then LS reserves resource 1 only for type-2 customers. In this way, the probability that LS accepts one customer of type 2 is $1 - e^{-\Lambda_2}$. Thus, the

expected total reward of LS is

$$u_{21} \cdot (1 - e^{-\Lambda_2}) = (0.5 + \epsilon) \cdot (1 - e^{-0.5/(0.5+\epsilon)}).$$

It approaches $(1 - 1/e)/2$ when ϵ tends to 0. In sum, the competitive ratio of LS tends to $(1 - 1/e)/2$ when ϵ tends to 0.

□

4.7 Improving the Bound

In this section, we derive an algorithm with an improved competitive ratio compared to LS. This algorithm also groups customer types based on the utilization u_{ij}, but in a more sophisticated way than LS. Moreover, this algorithm also relaxes the random routing step in order to allow customers more opportunities to be assigned to resources. This strategy of allowing greater resource sharing among customer types greatly improves the empirical performance of the algorithm.

We will prove that the competitive ratio of the new algorithm is

$$r^* = \max \left\{ r \in (0, 0.5) : r \leq \max_{z \in (0,0.5)} h(z, r) \right\}, \tag{4.7.1}$$

where

$$h(z, r) \equiv z - \left[z - \frac{1}{2} \left(1 - \frac{1}{1 - 2r} \cdot \frac{1}{e^2} \right) \right] (1 - 2r) \left(\frac{1 - z}{1 - z - r} \right)^{2(1-z)}. \tag{4.7.2}$$

We can numerically solve (4.7.1) to find that $r^* \approx 0.321$.

For every resource j, we first divide all customer types into two sets S_j and L_j in the same way that LS does. Then, we further partition the customers in S_j into two sets, "medium small" and "tiny", depending on their utilization of resource j. Let

$$M_j = \{i \in S_j : u_{ij} \geq z^* \cdot c_j\}, \tag{4.7.3}$$

$$T_j = \{i \in S_j : u_{ij} < z^* \cdot c_j\}, \tag{4.7.4}$$

where

$$z^* \equiv \arg\max_{z \in (0,0.5)} h(z, r^*) \approx 0.42. \tag{4.7.5}$$

It is easy to check that there is only one maximizer.

Recall that $U_j = \sum_{i \in [n]} x^*_{ij} u_{ij}$, $U^L_j = \sum_{i \in L_j} x^*_{ij} u_{ij}$ and $U^S_j = \sum_{i \in S_j} x^*_{ij} u_{ij}$, where x^* is an optimal solution to the LP (4.5.1). We further define $U^M_j \equiv \sum_{i \in M_j} x^*_{ij} u_{ij}$ and $U^T_j \equiv \sum_{i \in T_j} x^*_{ij} u_{ij}$ analogously.

Intuitively, the load values $U^T_j, U^M_j, U^S_j, U^L_j$ serve as estimates for how much capacity of resource j is expected to be utilized by customers of types in the sets T_j, M_j, S_j and L_j, respectively. For a given resource j, if any of the load values dominates the others, it might be a good strategy to reserve resource j exclusively for customers in the corresponding set.

We next categorize every resource j into one of two types based on the load values.

Definition 4.7.1. *Resource j is a type-A resource if*

$$U_j^S \geq -0.5c_j \log(1 - 2r^* U_j/c_j)$$

$$or \quad U_j^T \geq -(1 - z^*)c_j \log\left(1 - \frac{r^* U_j}{c_j(1 - z^*)}\right).$$

Otherwise, resource j is a type-B resource.

The motivation for the above definition is as follows. If resource j is of type A, then U_j^S or U_j^T are relatively large compared to other load values, which implies that customers that are routed to resource j by the LP (4.5.1) tend to have relatively small utilization u_{ij}. On the other hand, if resource j is of type B, then U_j^L is relatively larger, which implies that customers that are routed to resource j by the LP tend to have relatively large utilization u_{ij}.

Depending on the type of resource, we will reserve each resource wholly for a certain set of customer types. We say that a customer of type i is *admissible* to resource j if this customer can be assigned to resource j by our algorithm. The following definition defines the reservation criteria of the algorithm.

Definition 4.7.2. *A customer of type i, $i \in [n]$, is admissible to resource j, $j \in [m]$, if and only if at least one of the following criteria holds:*

- *Resource j is of type A,*
- *or $i \in M_j \cup L_j$.*

We are now ready to specify the improved algorithm.

Refined Large-or-Small Algorithm (RLS):

1. (Pre-processing step) Same as for LS.

2. (Random routing step) Same as for LS.

3. (Admission step) If a customer is admissible to resource j and there is enough remaining capacity in j, then assign the customer to resource j.

4. (Resource-sharing step) If a customer is rejected in the Admission Step, but there is another resource with enough remaining capacity and to which the customer is admissible, then assign the customer to any such resource. Otherwise, reject the customer.

The idea of the algorithm is as follows. If a resource j is of type A, then we can bound from above the left-over capacity of the resource, because a type-A resource tends to be used by a sufficiently high number of customers in sets S_j and T_j. Whenever one such customer is rejected, we know that the remaining capacity is small. Furthermore, since it is not disadvantageous to turn away customers of other types, their utilization being higher than those in S_j and T_j, we will admit customers of all types to a type-A resource. On the other hand, if resource j is of type B, then customers who are admissible to the resource have large utilization values. We can allocate a large enough amount of the resource as soon as one such customer arrives. We do not admit customers with small utilization values to type-B resources in order to leave enough space for customer types in M_j and L_j.

For each resource j, let N_j denote the total number of customers who are routed to resource j in Step (2) of RLS. Note that N_j does not include customers who are assigned

resource j in Step (4) of RLS. Let $W_{j1}, W_{j2}, ...,$ be a sequence of i.i.d. random variables each having a distribution that is given by

$$P(W_{j1} \leq x) = \sum_{i \in [n]} \mathbf{1}(u_{ij} \leq x) \frac{x^*_{ij}}{\sum_{k \in [n]} x^*_{kj}}.$$

That is, each variable W_{j1} can be seen as the utilization u_{ij} of a single random customer who is routed to resource j during the horizon. Since the probability that such a customer has type i is $\frac{x^*_{ij}}{\sum_{k \in [n]} x^*_{kj}}$, W_{j1} takes value u_{ij} with this probability. The following lemma gives a lower bound on the expected amount of capacity of a type-A resource that will be allocated by RLS.

Lemma 4.7.3. *If resource j is of type-A, then the expected amount of resource j allocated by RLS is at least*

$$\max \left\{ \mathbf{E} \left[\min(\sum_{k=1}^{N_j} W_{jk} \mathbf{1}(W_{jk} \leq 0.5c_j), 0.5c_j) \right], \mathbf{E} \left[\min(\sum_{k=1}^{N_j} W_{jk} \mathbf{1}(W_{jk} \leq z^* c_j), (1 - z^*)c_j) \right] \right\}.$$

Proof. $\sum_{k=1}^{N_j} W_{jk} \mathbf{1}(W_{jk} \leq 0.5c_j)$ has the same distribution as the total (random) amount of resource j requested by customers who are routed to resource j and whose types are in S_j. If the actual amount of resource j allocated to customers is less than $\sum_{k=1}^{N_j} W_{jk} \mathbf{1}(W_{jk} \leq 0.5c_j)$, it must be that at least one customer with type in S_j is rejected due to lack of remaining capacity of resource j. In such a case, the actual amount of resource j allocated to customers must be at least $0.5c_j$, because otherwise the customer with type in S_j would not have

been rejected. Thus, the total amount of resource j allocated by our algorithm is at least $\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \leq 0.5c_j), 0.5c_j)$.

A similar argument applies to customers with types in T_j. $\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \leq z^*c_j)$ has the same distribution as the total amount of resource j requested by customers who are routed to resource j and whose types are in T_j. If at least one of these requests is rejected, the remaining capacity of resource j must be at most z^*c_j. Thus, the total amount of resource j allocated to customers is at least $\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \leq z^*c_j), (1-z^*)c_j)$.

The proof follows when we take expectation of the lower bounds.

□

Let $\mu_j^M = \sum_{i\in M_j} x_{ij}^*$ and $\mu_j^L = \sum_{i\in L_j} x_{ij}^*$ be the expected number of customers who are routed to resource j and whose types are in M_j and L_j, respectively. The following lemma gives a lower bound on the expected amount of capacity of a type-B resource that will be allocated by *RLS*.

Lemma 4.7.4. *If resource j is of type B, then the expected amount of resource j allocated to customers in RLS is at least*

$$\min\{z^*c_j, e^{-\mu_j^M}[U_j^L e^{-\mu_j^L} + 0.5c_j(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L})] + (1 - e^{-\mu_j^M})z^*c_j\}.$$

Proof. If any customer is assigned to resource j in Step (4) of the algorithm, then at least z^*c_j of resource j is allocated, since every customer type i that is admissible to resource j satisfies $u_{ij} \geq z^*c_j$.

If no customer is assigned to resource j by Step (4), then resource j can only be allocated

to customers who are directly routed to this resource, i.e. in Step (3) of RLS. We consider three cases:

- At least one customer with type in M_j is routed to resource j. This event occurs with probability $1 - e^{-\mu_j^M}$. In such a case, we use $z^* c_j$ as the lower bound on the amount of resource j taken by customers.

- No customer with type in M_j is routed to resource j, and exactly one customer with type in L_j is routed to resource j. This event occurs with probability $e^{-\mu_j^M} \cdot \mu_j^L e^{-\mu_j^L}$. Conditional on this event, the expected amount of resource j taken by the only customer with type in L_j is
$$\frac{\sum_{i \in L_j} x_{ij}^* u_{ij}}{\sum_{i \in L_j} x_{ij}^*} = \frac{U_j^L}{\mu_j^L}.$$

- No customer with type in M_j is routed to resource j, and more than one customer with type in L_j are routed to resource j. This event occurs with probability $e^{-\mu_j^M}(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L})$. In this event, we use $0.5c_j$ as the lower bound on the amount of resource j taken by the customer in L_j, by definition of L_j.

In summary, if no customer is assigned to resource j in Step (3), the expected amount of resource j taken by customers routed to the resource is at least

$$\begin{aligned}
&(1 - e^{-\mu_j^M}) z^* c_j + e^{-\mu_j^M} \cdot \mu_j^L e^{-\mu_j^L} \cdot \frac{U_j^L}{\mu_j^L} + e^{-\mu_j^M}(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}) 0.5 c_j \\
=&e^{-\mu_j^M}[U_j^L e^{-\mu_j^L} + 0.5 c_j (1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L})] + (1 - e^{-\mu_j^M}) z^* c_j.
\end{aligned}$$

We complete the proof by combining this result with the lower bound $z^* c_j$ for the case that a customer is assigned to resource j by Step (4) of the algorithm.

□

We combine the previous two lemmas to prove the performance guarantee of the algorithm.

Theorem 4.7.5. *For each resource j, the expected amount of resource j allocated to customers is at least $r^* U_j$.*

Proof. First consider the case that resource j is of type A. Since $\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le 0.5c_j)$ and $\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le z^* c_j)$ has compound Poisson distribution with mean

$$\mathbf{E}\left[\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le 0.5c_j)\right] = U_j^S, \quad \mathbf{E}\left[\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le z^* c_j)\right] = U_j^T,$$

we can apply Lemma 4.6.1 to get

$$\begin{aligned}\mathbf{E}\left[\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le 0.5c_j), 0.5c_j)\right] =& c_j \mathbf{E}\left[\min(\sum_{k=1}^{N_j} \frac{W_{jk}}{c_j}\mathbf{1}(W_{jk} \le 0.5c_j), 0.5)\right] \\ \ge& c_j 0.5\left(1 - e^{-2U_j^S/c_j}\right),\end{aligned}$$

and

$$\begin{aligned}\mathbf{E}\left[\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \le z^* c_j), (1-z^*)c_j)\right] =& c_j \mathbf{E}\left[\min(\sum_{k=1}^{N_j} \frac{W_{jk}}{c_j}\mathbf{1}(W_{jk} \le z^* c_j), 1-z^*)\right] \\ \ge& c_j(1-z^*)\left(1 - e^{-\frac{U_j^T}{c_j(1-z^*)}}\right).\end{aligned}$$

Then according to Definition 4.7.1, we have by the definition of type-A resources

$$U_j^S \geq -0.5c_j \log(1 - 2r^* U_j/c_j) \quad \Longrightarrow \quad c_j 0.5(1 - e^{-2U_j^S/c_j}) \geq r^* U_j,$$

or

$$U_j^T \geq -(1-z^*)c_j \log(1 - \frac{r^* U_j}{c_j(1-z^*)}) \quad \Longrightarrow \quad c_j(1-z^*)(1 - e^{-\frac{U_j^T}{c_j(1-z^*)}}) \geq r^* U_j.$$

In sum, we have

$$\max\left\{ \mathbf{E}\left[\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \leq 0.5c_j), 0.5c_j)\right], \mathbf{E}\left[\min(\sum_{k=1}^{N_j} W_{jk}\mathbf{1}(W_{jk} \leq z^* c_j), (1-z^*)c_j)\right]\right\}$$
$$\geq \max\left\{ c_j 0.5(1 - e^{-2U_j^S/c_j}), c_j(1-z^*)(1 - e^{-\frac{U_j^T}{c_j(1-z^*)}})\right\}$$
$$\geq r^* U_j.$$

This proves the theorem for type-A resources.

Now we consider the case that resource j is of type B. Starting from this point, we will assume without loss of generality that $c_j = 1$. Based on Lemma 4.7.4, we need to show

$$\min\{z^*, e^{-\mu_j^M}[U_j^L e^{-\mu_j^L} + 0.5(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L})] + (1 - e^{-\mu_j^M})z^*\} \geq r^* U_j.$$

Since $z^* > r^*$ as we numerically checked, it suffices to show

$$e^{-\mu_j^M}\left[U_j^L e^{-\mu_j^L} + 0.5\left(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}\right)\right] + \left(1 - e^{-\mu_j^M}\right) z^* \geq r^* U_j$$

based on Lemma 4.7.4.

By examining the first and second derivatives of $U_j^L e^{-\mu_j^L} + 0.5(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L})$ with respect to μ_j^L, it is easy to check that

$$\begin{aligned} e^{-\mu_j^M}\left[U_j^L e^{-\mu_j^L} + 0.5\left(1 - e^{-\mu_j^L} - \mu_j^L e^{-\mu_j^L}\right)\right] + \left(1 - e^{-\mu_j^M}\right) z^* \geq & \quad e^{-\mu_j^M} \cdot 0.5\left(1 - e^{-2U_j^L}\right) + \left(1 - e^{-\mu_j^M}\right) z^* \\ = & \quad z^* - e^{-\mu_j^M}\left[z^* - 0.5\left(1 - e^{-2U_j^L}\right)\right]. \end{aligned}$$

If $z^* < 0.5(1 - e^{-2U_j^L})$, we must have $z^* - e^{-\mu_j^M}[z^* - 0.5(1 - e^{-2U_j^L})] > z^* > r^* = r^* c_j \geq r^* U_j$, which proves the theorem for this case.

Now suppose $z^* \geq 0.5(1 - e^{-2U_j^L})$. Since $\mu_j^M = \sum_{i \in M_j} x_{ij}^* \geq \sum_{i \in M_j} x_{ij}^* \cdot 2u_{ij} = 2U_j^M$, we have

$$\begin{aligned} z^* - e^{-\mu_j^M}\left[z^* - 0.5\left(1 - e^{-2U_j^L}\right)\right] \geq & z^* - e^{-2U_j^M}\left[z^* - 0.5\left(1 - e^{-2U_j^L}\right)\right] \\ = & z^* - e^{-2(U_j^S - U_j^T)}\left[z^* - 0.5\left(1 - e^{-2(U_j - U_j^S)}\right)\right]. \quad (4.7.6) \end{aligned}$$

It is easy to see that (4.7.6) is decreasing in U_j^T. We next show that, given U_j and U_j^T,

(4.7.6) is also decreasing in U_j^S.

$$\begin{aligned}
&\frac{\partial}{\partial U_j^S}\left[z^* - e^{-2(U_j^S - U_j^T)}[z^* - 0.5(1 - e^{-2(U_j - U_j^S)})]\right] \\
=&2e^{-2(U_j^S - U_j^T)}[z^* - 0.5(1 - e^{-2(U_j - U_j^S)})] - e^{-2(U_j^S - U_j^T)}e^{-2(U_j - U_j^S)} \\
=&e^{-2(U_j^S - U_j^T)}(2z^* - 1) \\
<&0.
\end{aligned}$$

According to Definition 4.7.1, we must have

$$U_j^S < -0.5\log(1 - 2r^*U_j) \tag{4.7.7}$$

and

$$U_j^T < -(1 - z^*)\log\left(1 - \frac{r^*U_j}{1 - z^*}\right). \tag{4.7.8}$$

Since (4.7.6) is decreasing in U_j^S and U_j^T, we can plug in (4.7.7) and (4.7.8) and obtain

$$\begin{aligned}
&z^* - e^{-2(U_j^S - U_j^T)}[z^* - 0.5(1 - e^{-2(U_j - U_j^S)})] \\
\geq&z^* - e^{-2(-0.5\log(1-2r^*U_j) + (1-z^*)\log(1 - \frac{r^*U_j}{1-z^*}))}[z^* - 0.5(1 - e^{-2(U_j + 0.5\log(1-2r^*U_j))})] \\
=&z^* - \left[z^* - \frac{1}{2}\left(1 - \frac{1}{1 - 2r^*U_j}\cdot\frac{1}{e^{2U_j}}\right)\right](1 - 2r^*U_j)\left(\frac{1 - z^*}{1 - z^* - r^*U_j}\right)^{2(1-z^*)}.
\end{aligned} \tag{4.7.9}$$

Since z^* and r^* are constants, (4.7.9) is a function of a single variable U_j. It is easy to check that this function is increasing and concave in U_j for $U_j \leq c_j = 1$. Moreover, it equals

0 at $U_j = 0$. Therefore,

$$
\begin{aligned}
& z^* - \left[z^* - \frac{1}{2} \left(1 - \frac{1}{1-2r^*U_j} \cdot \frac{1}{e^{2U_j}} \right) \right] (1 - 2r^*U_j) \left(\frac{1-z^*}{1-z^*-r^*U_j} \right)^{2(1-z^*)} \\
\geq & U_j \left[z^* - \left[z^* - \frac{1}{2} \left(1 - \frac{1}{1-2r^*} \cdot \frac{1}{e^2} \right) \right] (1 - 2r^*) \left(\frac{1-z^*}{1-z^*-r^*} \right)^{2(1-z^*)} \right] \\
= & U_j h(z^*, r^*) \\
= & U_j r^*.
\end{aligned}
$$

This completes the proof for the theorem.

□

4.8 Numerical Study

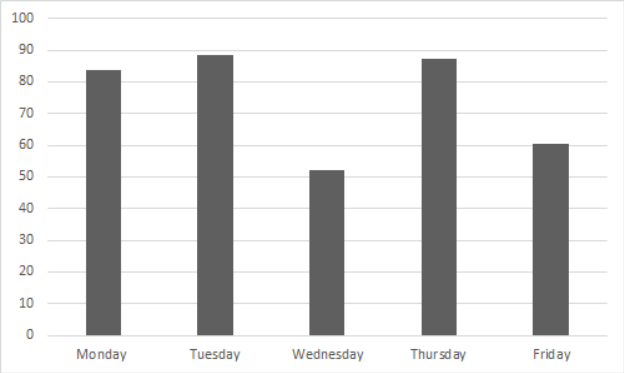
We compare the empirical performance of our algorithms against two commonly used heuristics by simulating the algorithms on appointment-scheduling data obtained from a large hospital system in New York City.

We obtain our data set from an Allergy department in the hospital system. The data set contains more than 20000 appointment entries recorded in the year 2013. Each entry in the data records information about one appointment. The entry includes the date that the patient makes the appointment, the exact time of the appointment, whether the patient eventually showed up to the original appointment, canceled the appointment some time later, or missed the appointment.

The average total number of patients who arrive to make appointments on each day is

shown in Figure 4.1. It can be readily seen that the arrival pattern is highly non-stationary, as the average total number of arrivals on Thursday is 60% more than that on Wednesday.

Figure 4.1: Average number of arrivals in a week.



We simulate a discrete horizon of 200 days. In each day, a random number of patients arrive to make appointments. Each patient needs to be assigned an appointment of 15 min, 30 min, 45 min, depending on his or her condition. By medical necessity, some patients must be assigned same-day appointments if at all. We call these patients *urgent patients*. Other patients can be assigned to any day in the future. We call these patients *regular patients*. The relative proportions of patients in each priority category are summarized in Table 4.2. We impose a requirement that regular patients must be assigned an appointment that is no more than 20-days away from the date of his or her first request for an appointment. Although this hard deadline is not strictly enforced in reality, consideration for patient satisfaction often impels the administration to limit as much as possible the number of days that each patient must be made to wait. Our deadline mimics this effect.

We assume a 5-day work week. We estimate the expected number of patients arriving

Table 4.2: Percentage of patients in different categories.

	15 min	30 min	45 min
urgent	27%	1%	0%
regular	45%	14%	9%

per day of the week as shown in Figure 4.1. We assume that each patient randomly and independently falls into one of the six categories shown in Table 4.2.

We assume that there are multiple sessions on each day. Each session corresponds to a resource in our model. We vary the session length among 1, 1.5, 2, 3, or 4 hours. We assume that a patient can be assigned to any appointment within a day, as long as there is enough service time remaining and the day falls within the deadline to serve the patient. We vary the number of sessions that are available per day.

We test the following two algorithms

- Our basic online algorithm (LS).
- Our refined algorithm (RLS).
- A greedy heuristic (GRD) that tries to assign every patient to the most recent session that is available and falls within his deadline.
- A heuristic ($RSRV$) that reserves for each category an amount of capacity that is approximately equal to the average utilization of that category. This reservation is nested in the sense that higher-priority patients have access to their reserved capacity, as well as the reserved capacity of all lower-priority categories. The heuristic then assigns patients greedily to the reserved capacity.

- The primal-dual algorithm (PD) given by Buchbinder et al. (2007).

For each algorithm and each test case, we simulate the total length of appointments made during the entire 200 periods and calculate the average total length over 1000 replicates. We report the ratio of this average number relative to the optimal objective value of the upper bound given in (4.5.1).

Table 4.3: Performance relative to the upper bound given in (4.5.1). The length of each session is 1 hour.

Number of sessions	Scale	LS	RLS	GRD	RSRV	PD
18	70.5%	68.8%	94.6%	98.4%	80.2%	97.9%
19	74.4%	68.4%	94.2%	98.4%	78.8%	97.6%
20	78.3%	68.5%	94.3%	98.1%	77.4%	97.1%
21	82.2%	68.6%	94.4%	96.2%	76.2%	96.5%
22	86.2%	69.0%	94.5%	96.1%	75.1%	95.5%
23	90.1%	69.4%	94.4%	96.0%	74.0%	94.3%
24	94.0%	69.8%	94.2%	95.5%	73.1%	92.5%
25	97.9%	70.0%	93.6%	94.5%	72.2%	90.5%
26	101.8%	70.3%	93.1%	94.3%	72.7%	90.0%
27	105.7%	70.6%	95.2%	95.6%	74.7%	91.1%
28	109.7%	70.7%	96.5%	96.4%	76.5%	92.2%
29	113.6%	70.6%	97.3%	97.1%	77.3%	93.2%
30	117.5%	70.6%	97.7%	97.6%	77.7%	94.1%
31	121.4%	70.6%	98.2%	98.0%	77.9%	94.9%
32	125.3%	70.6%	98.5%	98.3%	77.9%	96.2%
33	129.2%	70.6%	98.7%	98.5%	77.9%	97.1%

Tables 4.3 to 4.7 summarize the performance of the algorithms. The *scale* is the ratio of total capacity to total demand. We make several observations:

- The refined algorithm RLS is never more than 4% worse than the upper bound on average in each of the scenarios tested. The reservation heuristic $RSRV$ could be as much as 16% worse than the upper bound on average. The greedy heuristic GRD could be as much as 8% worse than the upper bound on average.

Table 4.4: Performance relative to the upper bound given in (4.5.1). The length of each session is 1.5 hours.

Number of sessions	Scale	LS	RLS	GRD	RSRV	PD
12	70.5%	75.6%	98.2%	98.4%	92.7%	97.9%
13	76.4%	75.5%	97.9%	98.4%	91.5%	97.4%
14	82.2%	75.9%	97.7%	96.3%	90.2%	96.7%
15	88.1%	76.3%	97.4%	96.1%	88.9%	95.7%
16	94.0%	76.8%	96.7%	95.6%	87.6%	93.9%
17	99.9%	76.9%	95.7%	93.7%	86.5%	91.5%
18	105.7%	77.4%	97.1%	95.6%	90.3%	93.6%
19	111.6%	77.4%	98.0%	96.7%	94.2%	94.9%
20	117.5%	77.4%	98.3%	97.5%	97.8%	95.7%
21	123.4%	77.5%	98.7%	98.0%	99.2%	96.4%
22	129.2%	77.5%	99.0%	98.4%	99.5%	97.0%

Table 4.5: Performance relative to the upper bound given in (4.5.1). The length of each session is 2 hours.

Number of sessions	Scale	LS	RLS	GRD	RSRV	PD
9	70.5%	78.4%	99.1%	98.5%	91.1%	97.9%
10	78.3%	78.1%	99.0%	98.4%	89.2%	97.3%
11	86.2%	78.7%	98.6%	96.2%	87.2%	96.3%
12	94.0%	79.2%	97.3%	95.6%	85.4%	94.5%
13	101.8%	79.4%	96.3%	94.4%	85.3%	92.9%
14	109.7%	79.6%	98.1%	96.5%	90.2%	95.1%
15	117.5%	79.6%	98.5%	97.5%	95.1%	96.2%
16	125.3%	79.7%	99.0%	98.2%	99.4%	97.0%

- Predictably, the refined algorithm RLS dominates the basic algorithm LS. This performance gain comes from better resource sharing.

- The greedy heuristic GRD also performs consistently better than the static reservation heuristic $RSRV$, except when the scale is high. Most likely, the greedy heuristic allows greater resource sharing among different customer types, which results in better resource utilization. However, when the scale is high, there is an abundance of capacity, so that resource sharing is less important.

Table 4.6: Performance relative to the upper bound given in (4.5.1). The length of each session is 3 hours.

Number of sessions	Scale	LS	RLS	GRD	RSRV	PD
6	70.5%	83.6%	99.3%	98.6%	93.9%	98.0%
7	82.2%	83.7%	99.0%	96.5%	91.8%	97.0%
8	94.0%	84.2%	97.4%	95.7%	89.5%	94.8%
9	105.7%	84.5%	97.5%	95.6%	92.3%	94.7%
10	117.5%	84.7%	98.7%	97.6%	99.6%	96.5%
11	129.2%	84.7%	99.3%	98.4%	99.8%	97.3%

Table 4.7: Performance relative to the upper bound given in (4.5.1). The length of each session is 4 hours.

Number of sessions	Scale	LS	RLS	GRD	RSRV	PD
5	78.3%	85.4%	99.2%	98.6%	93.7%	97.4%
6	94.0%	86.0%	97.3%	95.8%	91.1%	94.9%
7	109.7%	86.4%	98.2%	96.5%	97.0%	95.6%
8	125.3%	86.8%	99.3%	98.3%	99.8%	97.1%

- The greedy heuristic *GRD* tends to be good when the scale is either very large or very small. These are situations in which it is easier to do well. When there is little capacity, the utilization can be kept high even with a naive algorithm because there is relative very high demand. When there is an abundance of capacity, the utilization can be close to optimal because a high proportion of demand can be accommodated. Therefore, an algorithm offers the most value relative to a naive heuristic when the scale is moderate.

- Similar to *GRD*, the Primal-Dual algorithm performs well when the scale is either large or small. However, its performance is slightly worse than *GRD* in most cases. This might be because the Primal-Dual algorithm is specially designed to improve the worst-case performance, whereas we report the average-case performance.

Table 4.8: Performance relative to the upper bound given in (4.5.1), when parameters are randomly generated.

Number of sessions	LS	RLS	GRD	RSRV	PD
Worst Setting	44.3%	68.4%	66.3%	43.7%	67.6%
Average Setting	65.2%	96.3%	95.9%	85.2%	95.9%

- The refined algorithm *RLS* performs significantly better than, or is very close to, the better of the two heuristics. It performs much better than the heuristics when the scale is moderate, which is when an algorithm offers the most value relative to a naive heuristic.

We also test the algorithms under randomly generated settings. In Table 4.8, we report the worst performance and the average performance of all the algorithms over 100 random settings. The performance of algorithms in each setting is calculated by simulating 1000 replicates. Each of the 100 random settings is generated by

- uniformly generating the percentages in Table 4.2;
- uniformly picking a deadline for all regular patients between 5 and 30 days;
- uniformly setting the capacity of all resources to be between 45 and 150 minutes;
- uniformly picking a scale between 70% and 130%.

Again, our RLS algorithm consistently performs well in these test cases.

4.9 Conclusions

We study an online resource allocation problem where heterogeneous customers arrive at time-varying rates and request fractional amounts of resources. Upon an arrival of a customer, the allocation or rejection decision must be made immediately and irrevocably. The goal is to maximize the total amount of resources allocated to all customers. The model has applications in many areas, such as services, online advertising, and freight transportation.

An important realistic feature of our model is that we cannot partially fulfill the amount of resource requested by a customer. As a result, our model captures the challenging task of reserving fractional amount of a resource to high-priority customers. It is easy to see that naive heuristics such as a first-come, first-served strategy will fail because once it allocates ϵ amount of a resource, it cannot satisfy future higher-priority customers who request the entire original resource.

Our allocation algorithms directly tackle this problem based on a simple but effective idea. Specifically, our algorithms reserve each resource for a predetermined set of customer types, such that packing the requests of these customer types in any arbitrary way does not result in too much waste of the resource. We apply the definition (1.1.2) of competitive ratio and prove that our allocation algorithm is 0.321-competitive online algorithms. Further, we show that an upper bound on the competitive ratio of any algorithm is 1/2.

To test the empirical performance of our algorithms, we simulate them as well as other simple allocation heuristics in an appointment scheduling application. We use an appointment-scheduling dataset from Columbia University Medical Center. In the model used in the

simulation tests, the objective is to maximize the utilization of appointment sessions that will be assigned to patients with different appointment lengths. We find that our online algorithms have the best performance among all scheduling policies tested.

Chapter 5

Dynamic Optimization of Mobile Push Advertising Campaigns

5.1 Introduction

Recent years have seen tremendous growth in the volume of sales taking place in mobile commerce (m-commerce) markets. In these markets, customers visit online stores and purchase products via mobile platforms, such as apps for iOS and Android systems. In 2015, more than US$200 billion in sales took place via a single mobile app developed by Alibaba Group. Alibaba is an e-commerce company that provides a third-party platform for business-to-customer and customer-to-customer markets, among many other services. In China, the m-commerce market share of its mobile app, which has been installed on several hundred million devices, is rapidly displacing traditional e-commerce markets (Emarketer.com, 2016).

Given the size of its user base, the mobile app of Alibaba Group, henceforth referred to

simply as *the app*, serves as a new channel for advertising and delivering personalized recommendations. Owners of the app can choose to receive recommendations about products that are tailored to their interests. These recommendations are sent via *mobile push notifications*. When a mobile push notification is sent to a user, a short *message* describing the recommended product appears in the notification zone of the user's mobile phone. The user can either swipe the message away or click on the message to link to a new page containing full details about the recommendation. We call the latter action a *clickthrough*.

Push messages are used to achieve two objectives. First, the messages are used to make personalized recommendations about premium products selected from millions of online stores at Alibaba Group. These recommendations prompt mobile users to visit and browse products in the online markets, eventually generating more sales. Second, the messages are designed to promote products for online stores that have applied to join advertising campaigns organized by Alibaba Group. Such campaigns often aim at achieving a certain impact on the user base, such as a desired number of clickthroughs that the campaign messages need to attract. Retailers who want to advertise their products may offer to pay for clickthroughs to their products.

Push messages are sent to hundreds of millions of mobile users every day (BusinessWire, 2016). Given such a large user base, it is challenging to manage the delivery of the "right messages" to the "right users" without overwhelming each user with too many messages.

In this thesis chapter, we model the problem faced by Alibaba Group of how to manage push messages over a planning period. Without loss of generality, we take a period to be a day. The problem is a novel resource-allocation problem. In this problem, there is a set of

known users, each owning a mobile device on which the app is installed. There is also a set of distinct messages. Each message can be sent to any number of users, and yields a reward when it generates a clickthrough, subject to a budget constraint on the total reward that all the users can generate for the message. The budget represents the maximum amount that an advertiser is willing to pay for clickthroughs to his product or website on a given day. Since sending too many messages to the same user would have an adverse impact on the user's experience, each user must receive no more than one push message per day. Over the course of a day, push messages can be sent sequentially to different users. Once a message has been sent to a user, we can observe after some time whether the user clicks on the message. The observed user actions can be used to update decisions about what messages we want to send to subsequent users, so that we can make the most use of the remaining budgets. We aim to maximize the total reward earned from all messages sent, by adaptively determining the sequence of users to contact, together with their corresponding messages. We do not consider any multi-day effect.

To achieve our goal, it is necessary to learn and calibrate the clickthrough probability that we might expect from any given message-user pair. A typical method is to construct a mapping $p(x, y)$ between every vector of user features x and every vector of product features y, and then fitting this mapping to historical information, as well as to data gathered in experiments. At Alibaba, the construction and fitting of the mapping $p(\cdot, \cdot)$ is performed as a separate data-mining and calibration problem. Accordingly, in this thesis chapter, we take the view that the clickthrough probabilities will be provided as inputs to our model. Thus, we will focus on the remaining optimization problem.

The following simple example illustrates our problem.

Example 5.1.1. *There are two messages, A and B, to be sent out to two users X and Y (see Figure 5.1). Each message can be sent to any of the two users, but each user should receive exactly one message. For simplicity, we assume that each message has budget of* \$1 *and yields a reward of* \$1 *when clicked. In other words, each message will exhaust all of its budget on the first clickthrough. We assume that the clickthrough probabilities are given by Figure 5.1.*

Suppose that we send message B to both users X and Y. Then with probability $0.6 \times (1 - 0.2)$*, only X will click on message B, in which case we earn* \$1*. With probability* 0.6×0.2*, both users will click on message B, but since message B has a budget of only* \$1*, we only earn* \$1 *even though message B receives two clickthroughs.*

The optimal strategy to maximize the expected total reward is to first send message B to user Y. Then we observe whether Y clicks on B. If Y clicks on B (with probability 0.2*), thereby exhausting the budget of message B, we next send message A to X. If Y does not click on B, we next send message B to user X. The resulting total expected reward is* $0.2 \times 0.4 \times \$1 + 0.8 \times 0.6 \times \$1 = \$0.56$*. Note that the decision for user X is a dynamic one. Which message X will receive depends on whether Y clicks on message B.*

An interesting related question is how to adaptively find the optimal sequence of users to contact, together with their corresponding messages. However, in practice, we cannot implement a solution that makes an adaptive decision for every single user, simply because it would take too long to observe the actions of all users one at a time. Users might not

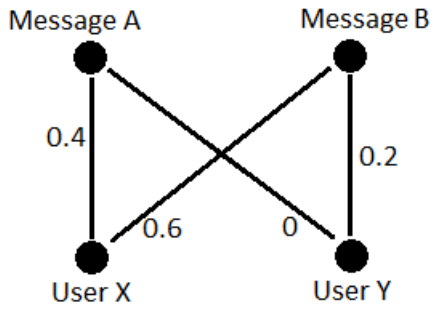


Figure 5.1: A toy example of sending push messages.

immediately notice new push messages on their mobile devices. In practice, more than half of all user actions (either clickthrough or dismissal) can be observed within 1 hour. Most user actions can be observed within 3 hours. Furthermore, the clickthrough rate of users who respond after 3 hours is negligible. Given such long response time, it is only practically possible to send out messages in a small number of batches on each day. After sending out each batch, we must wait for several hours to observe the feedback of users who were most recently contacted. In this way, we must take advantage of a very small number of recourse opportunities.

5.1.1 Overview of Algorithms and Contributions

We design an algorithm to determine how to send push messages in a small number of cycles over the course of a day. The decisions for each cycle can be based on the observed actions of all users contacted in previous cycles in the day. In the real application, it is possible to send hundreds of millions of messages in a single cycle.

Due to the large size of the real application, we analyze algorithms in an asymptotic

regime. We consider a novel scaling of the problem "size," called *big-data scaling*. In this scaling, as the problem size grows, the number of users, as well as their diversity, as characterized by the number of distinct profiles of user characteristics, both grow. We study the regret of our algorithms, which captures the revenue loss of our algorithms relative to an optimal algorithm, when the scale is large.

We analyze the performance of two algorithms for sending push messages. The first algorithm, which we call the *Static Algorithm*, does not make dynamic decisions based on user feedback, and essentially sends out all messages in one cycle. It represents a simple attempt to tackle this problem without utilizing user feedback. We will show that the Static Algorithm has asymptotic regret $O(\sqrt{t})$. Although this regret is diminishing in the size t of the system, we show that it is possible to further reduce the regret.

To this end, we analyze a second algorithm, which we call the *Reservation Algorithm*. The Reservation Algorithm sends out messages in two cycles. We prove that by making use of only one additional cycle, the Reservation Algorithm is able to reduce the regret to $O(t^{1/4} \log t)$. In other words, the regret of the Reservation Algorithm grows much more slowly than that of the Static Algorithm as we scale up the system size t. Further, we prove that the difference in regret between the Static Algorithm and the Reservation Algorithm is $\Omega(\sqrt{t})$.

Both of algorithms that we analyze make use of a solution to a linear program that matches the expected number of clickthroughs with the budgets of the messages. Linear programming has been used to design algorithms for various resource-allocation problems because optimal solutions to linear programs are often useful guides for making allocation

decisions (Wang, Truong and Bank 2015). In our model, an assignment of users to messages given by a static linear program performs the following basic tradeoff. If we send a message to too many users, we might obtain more clickthroughs than are paid for by the budget of that message. Thus, we waste the opportunity to have these users view other messages and bring in rewards from other sources. In other words, we *lose viewing opportunities*. On the other hand, if we send the message to too few viewers, we lose the opportunity of generating more clickthroughs. Thus we *lose potential rewards*.

The Reservation Algorithm performs this tradeoff by using a re-solving heuristic. To be more specific, the algorithm sends out messages in the first cycle based on an optimal solution to a linear program. Then after observing the clickthroughs that each message receives from the first batch of users contacted, the algorithm re-solves a linear program that matches the remaining budgets of the messages to the remaining users.

We test the numerical performance of the above two algorithms by simulating them on production data provided by Alibaba. The data contains three large batches of messages that were sent to several hundred million users in three separate days in March 2016, along with the clickthrough probabilities for all user-message pairs, which were estimated by Alibaba. Our computational results show that by exploiting user feedback, the Reservation Algorithm can reduce the regret of the Static Algorithm by at least 10%-50%. This result suggests that the idea of dynamically optimizing the allocation of push messages is highly promising.

5.2 Literature Review

Our problem is related to revenue-management problems, which we reviewed in Section 3.2.1, as it aims to maximize a total expected reward that can be obtained by dynamically allocating demands (users) to resources (messages). Traditional revenue-management problems focus on situations in which demands arrive randomly and exogenously over time (Talluri and van Ryzin, 2004). In contrast, in our model we can control the demands, or the users to be contacted. However, our model limits the number of opportunities at which dynamic decisions can be made. Next, we provide a more detailed review on asymptotic studies in revenue management.

Reiman and Wang (2008) first prove that for a network revenue-management problem, re-solving a linear program can help to reduce the total regret to $o(\sqrt{t})$. Their algorithm, which only re-solves the linear program once, cannot be applied to our model as they require observing the system state at every point in time. Recall that in our model, we can observe the action of users, or the consequences a decision, only a small number of times due to the long response time needed to acquire user feedback.

Jasin and Kumar (2012) also study a network revenue-management problem for which they propose an algorithm having constant regret that is independent of the system size t. However, their algorithm requires resolving LP, and thus observing the system state, infinitely many times as they scale up the system size t. As a result, their algorithm cannot be easily adapted to our setting.

Agrawal, Wang and Ye (2009) and Jasin (2015) study online resource-allocation problems

in the asymptotic regime. However, in their model, the distribution of demand arrivals is unknown and exogenous. Their algorithms either have regret that is at least $O(\sqrt{t})$, or require resolving linear programs too many times and therefore, are impractical for our setting.

We remark that our problem is different from models of online customer selection (Elmachtoub and Levi, 2016, 2015). In these models, although the decision maker has the ability to select which customers to offer resources to, the sequence of customer arrivals is still exogenous.

5.3 Model Formulation

We now formally state our model. There are m different messages and n users. Each message can be sent to any number of users. We make the following main assumptions:

Clickthrough probabilities. If message $j \in \{1, 2, ..., m\}$ is sent to user $i \in \{1, 2, ..., n\}$, the user will click the message with known probability p_{ij}. All the p_{ij}'s are given as inputs. For each user i, we call the vector $p_i = (p_{i1}, p_{i2}, ..., p_{im})$ consisting of clickthrough probabilities for user i for all messages a *user profile*. We also call the set of all user profiles $\{p_i\}_{i=1}^n$ a *user pool*.

User fatigue. To limit user fatigue, each of the n users must not receive more than one message. As mentioned earlier, we only consider a single-day problem, so this assumption is equivalent to requiring that no more than one message must be sent to each user per day.

Reward and budget. Each message $j \in \{1, 2, ..., m\}$ has budget of $c_j \cdot r_j$. We call c_j

the *capacity* of message j. If message j receives k clickthroughs in total, then we earn a reward of $r_j \min(k, c_j)$ from message j. In this way, we view messages as resources, and the capacity of a resource is the number of clickthroughs that will just exhaust the budget.

Special message 1. Among the m messages, message 1 is a personalized recommendation that recommends products selected from all online stores at Alibaba Group. We assume that message 1 has infinite capacity because there is no limit as to how many additional visits the online market can receive. The reward r_1 can be interpreted as the long-term expected marginal reward of a visit.

Other messages $2, 3, ..., m$ correspond to campaigns and commercial ads, and have finite capacity values.

In practice, most users prefer products recommended by message 1 over products in other campaign and advertising messages. The reason is that we personalize this message 1 to every user by selecting his/her favorite products from millions of online stores at Alibaba. It is unlikely that we cannot find a product from such a huge number of stores that a user prefers over products in other messages (except on the infrequent occasions that advertizers offer huge discounts for their products). We formally state these conditions in the following assumption.

Assumption 5.3.1.

$$c_1 = \infty, \;\; p_{i1} \geq p_{ij} \;\; \forall i = 1, 2, ..., n, \;\; j = 1, 2, ..., m.$$

According to this assumption, no user will ever receive message j if $r_j < r_1$, because

instead of sending message j, it is more beneficial to send message 1 that has a larger clickthrough probability and a larger reward value. Therefore, we assume that $r_j \geq r_1$ for all message $j \in \{2, 3, ..., m\}$. In other words, we assume that the marginal reward earned from campaign and advertising messages is at least r_1.

Number of cycles. The system can sequentially send messages to the n users in several cycles throughout the day. At the beginning of each cycle, the system can observe the clickthroughs in the previous cycle. Then the system sends messages to a subset of users who have not received messages yet. The decision of which messages to be sent to which users in each cycle can be adapted to the clickthroughs observed previously. An algorithm can choose how many cycles to use. However, for the algorithm to be practically useful, the number of cycles used should be small.

Objective. The objective is to maximize the expected total reward, where the expectation is taken over random user responses (i.e., clicking or not). Mathematically, under a policy Π, let I_{ij}^{Π} indicate whether user i clicks message j. The expected total reward V^{Π} of Π is

$$V^{\Pi} = \mathbf{E}[\sum_{j=1}^{m} r_j \min(\sum_{i=1}^{n} I_{ij}^{\Pi}, c_j)].$$

5.4 Performance Measure

We compare the performance of our algorithms against that of an optimal algorithm OPT by analyzing the asymptotic regret defined in (1.1.3) (with slightly different notations). As discussed earlier, an algorithm is limited to using a few cycles if it is to be practically useful.

However, we relax this requirement for OPT, and allow OPT to take an unlimited number of cycles. Then, OPT will never risk wasting opportunities by sending out too many messages in a single cycle. Therefore, we must have

$$\sum_{i=1}^{n} I_{ij}^{\mathrm{OPT}} \leq c_j, \ \forall j = 1, 2, ..., m, \ \text{w.p.1}, \tag{5.4.1}$$

And the objective value of OPT is

$$V^{\mathrm{OPT}} = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[\min(I_{ij}^{\mathrm{OPT}}, c_j)] = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[I_{ij}^{\mathrm{OPT}}]. \tag{5.4.2}$$

For a given problem instance, the regret of an algorithm Π is

$$V^{\mathrm{OPT}} - V^{\Pi} = \sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[I_{ij}^{\mathrm{OPT}}] - \mathbf{E}[\sum_{j=1}^{m} r_j \cdot \min(\sum_{i=1}^{n} I_{ij}^{\Pi}, c_j)].$$

5.5 Linear-Programming Formulation and Upper Bound on OPT

It is difficult to directly analyze the optimal policy OPT due to its complex dynamic properties. We are thus motivated to investigate an upper bound on V^{OPT} that is computationally tractable. In this section, we show that V^{OPT} can be bounded by the optimal objective value of a linear program, which allocates users to messages once in an expected sense. The

algorithms we present in subsequent sections will also be based on an optimal solution to the same linear program.

To this end, consider the following pairs of LP's that are dual to each other:

Primal:

$$\begin{aligned} V^{LP} = \max_{s \in \mathbb{R}^{n+m}} & \sum_{i=1}^{n} \sum_{j=1}^{m} r_j s_{ij} p_{ij} \\ \text{s.t.} & \sum_{i=1}^{n} s_{ij} p_{ij} \leq c_j, \ \forall j = 1, 2, ..., m \\ & \sum_{j=1}^{m} s_{ij} = 1, \ \forall i = 1, 2, ..., n \\ & s_{ij} \geq 0, \ \forall i, j. \end{aligned} \tag{5.5.1}$$

Dual:

$$\begin{aligned} \min_{\gamma \in \mathbb{R}^m, \ \eta \in \mathbb{R}^n} & \sum_{j=1}^{m} c_j \gamma_j + \sum_{i=1}^{n} \eta_i \\ \text{s.t.} & \ \eta_i \geq (r_j - \gamma_j) p_{ij}, \ \forall i = 1, 2, ..., n, j = 1, 2, ..., m \\ & \gamma_j \geq 0, \ \forall j = 1, 2, ..., m. \end{aligned} \tag{5.5.2}$$

Both the primal and dual forms of the linear program have physical interpretations. In the primal problem (5.5.1), the decision variable s_{ij} represents the fraction of user i to be allocated to message j. For each message j, the linear program matches the expected number of clickthroughs to the capacity c_j. In the dual problem (5.5.2), the dual variable γ_j corresponds to the primal constraint $\sum_{i=1}^{n} s_{ij} p_{ij} \leq c_j$, and η_i corresponds to $\sum_{j=1}^{m} s_{ij} = 1$. Let s^*, γ^*, η^* be a set of optimal dual variables. According to complementary slackness, if

$s_{ij}^* > 0$, i.e., the optimal solution allocates user i to message j with positive probability, we must have

$$j \in \text{argmax}_{k=1,2,\ldots,m}\{(r_k - \gamma_k^*)p_{ik}\}. \tag{5.5.3}$$

If we view γ_k^* as a cost incurred by every clickthrough of message k, then $(r_k - \gamma_k^*)p_{ik}$ is the expected profit of sending message k to user i. In other words, the dual problem implies that every user is allocated a message that maximizes the expected profit in this sense.

The following theorem shows that the expected total reward of OPT can be bounded by the above linear program.

Theorem 5.5.1.

$$V^{OPT} \leq V^{LP}.$$

Proof. Define

$$J_{ij}^{\Pi} \equiv \begin{cases} 1, & \text{if } \Pi \text{ sends message } j \text{ to user } i, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the objective value (5.4.2) becomes

$$\sum_{j=1}^{m} r_j \cdot \sum_{i=1}^{n} \mathbf{E}[J_{ij}^{OPT}]p_{ij}, \tag{5.5.4}$$

which is the same as the objective of the linear program (5.5.1) when $\mathbf{E}[J_{ij}^{\text{OPT}}]$ is the decision variable.

Taking expectation on both sides of (5.4.1), we get

$$\sum_{i=1}^{n} \mathbf{E}[I_{ij}^{OPT}] \leq c_j$$

$$\Longrightarrow \sum_{i \in \mathcal{N}} \mathbf{E}[J_{ij}^{OPT}] \cdot p_{ij} \leq c_j, \ \forall j = 1, 2, ..., m,$$

which is the first constraint of the linear program.

Moreover, the model requires $\sum_{j=1}^{m} J_{ij}^{OPT} = 1$, which gives the second constraint of the LP.

In sum, $\mathbf{E}[J_{ij}^{OPT}]$ satisfies all the constraints of the LP, and thus the optimal objective value of the LP is an upper bound on (5.5.4).

□

The Static Algorithm is a naive implementation of the LP solution. The algorithm sends out all messages in a single cycle based on an optimal solution to the LP (5.5.1). The name *static* comes from the fact that it does not adapt to user feedback.

Static Algorithm:

1. Solve the linear program (5.5.1). Let s^* be an optimal solution.

2. In a single cycle, send message j to user i with probability s_{ij}^*, for all $i = 1, 2, ..., n$, $j = 1, 2, ..., m$.

We will use analyze the performance of the Static Algorithm in Section 5.14 and use that as a basis for comparison when analyzing the performance of our Reservation Algorithm, which we will describe in the next section.

5.6 The Reservation Algorithm

In this section we present the Reservation Algorithm, which refines the Static Algorithm by adapting to user feedback. Specifically, the algorithm sends out messages in two cycles. In the first cycle, it partially sends out messages based on s^*. This step is similar to the Static Algorithm except that some users are reserved for the second cycle. Then based on user feedback, the algorithm re-solves a linear program for the remainder of capacity and users. In the second cycle, the algorithm sends messages according to an optimal solution to this re-optimized linear program.

Core to the algorithm is the decision of which users should be reserved (i.e., should *not* receive messages in the first cycle). Intuitively, if s^* allocates user i to message j (i.e., $s^*_{ij} = 1$) but we do not send user i any message in the first cycle, we are hoping that in case message j has its budget exhausted in the first cycle, user i can be directed to some other message having positive remaining budget. Thus, for each message j, we want to reserve a certain number of users who are allocated to message j according to s^*, who *also* have relatively high clickthrough probabilities for other messages. These users are good candidates for redirection, in case the budget of message j becomes exhausted after the first cycle.

The actual set of users reserved by the Reservation Algorithm is determined by a parameter Δ, which we call the *reservation level*. Given a problem instance, we can run the algorithm using any value of Δ. The remaining capacity of each message that is unassigned in the first cycle due to reservation will be roughly proportional to Δ. In theory, we will

show a way of choosing the parameter Δ to ensure an upper bound on the regret of the algorithm. In practice, we can tune Δ to achieve the best empirical performance.

The Reservation Algorithm works as follows:

1. Solve the linear program (5.5.1) to find an optimal solution s^* such that $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$, where $\|\cdot\|_0$ stands for the number of non-zero elements in the vector. In other words, the number of users who are simultaneously (fractionally) allocated to multiple messages should be no more than m^2. This condition is not required to implement this algorithm in practice, but only to simplify the analysis. Our Proposition 5.6.1 guarantees that one such s^* always exists and can be easily found by adding some infinitesimally small noise to the coefficients p_{ij}'s of the linear program.

2. For every pair of messages $j, k \in \{1, 2, ..., m\}$, $j \neq k$, solve the following problem to find a set R^{jk} that contains users who are intended for message j according to s^*, but who will be now reserved for message k in the second cycle.

$$\begin{aligned} \min_{R^{jk} \subseteq \{1,2,...,n\}} & \sum_{i \in R^{jk}} s_{ij}^* p_{ij} \\ \text{s.t. } & p_{ik} p_{lj} \geq p_{lk} p_{ij} \ \ \forall i \in R^{jk}, l \neq R^{jk} \\ & \sum_{i \in R^{jk}} s_{ij}^* p_{ij} \geq \min(\Delta, \sum_{i=1}^{n} s_{ij}^* p_{ij}). \end{aligned} \tag{5.6.1}$$

The first constraint of this optimization problem states that the users $i \in R^{jk}$ selected should be those having the largest ratios of p_{ik}/p_{ij} among all users. These users are

deemed the most promising to be reserved for message k. The second constraint states that either R^{jk} is the set of all users, or the expected number of clickthroughs that message j is to receive from the users in R^{jk}, according to the allocation s^*, is at least Δ. The objective function keeps the expected number of clickthroughs that are reduced due to the reservation as small as possible.

Note that for some user i and message j, it is possible to have $i \in R^{jk}$ for many different k's, meaning that user i is suitable to be redirected to multiple different messages in the second cycle. Furthermore, since the objective function keeps minimal the size of R^{jk}, we must have

$$\sum_{i \in R^{jk}} s^*_{ij} p_{ij} \leq \Delta + 1. \tag{5.6.2}$$

3. Send message j to user i in the first cycle with probability

$$x^{(1)}_{ij} \equiv s^*_{ij} \cdot \mathbf{1}(i \notin \bigcup_{k=1, k \neq j}^{m} R^{jk}). \tag{5.6.3}$$

Note that the condition $i \in R^{jk}$ only indicates that the algorithm reserves the fraction s^*_{ij} of user i that is allocated to message j; if there is another fraction of user i allocated to some other message, say j', then user i may receive message j' in the first cycle even if $i \in R^{jk}$.

Compared to the Static Algorithm, this step reduces the expected number of click-

throughs that each message j will receive in the first cycle by

$$\sum_{i=1}^{n} \mathbf{1}(i \in \bigcup_{k=1,k\neq j}^{m} R^{jk}) s_{ij}^* p_{ij} \leq \sum_{k=1,k\neq j}^{m} \sum_{i \in R^{jk}} s_{ij}^* p_{ij} \leq (m-1)(\Delta+1), \tag{5.6.4}$$

where the last inequality follows from (5.6.2).

4. Let D_j be the actual number of clickthroughs that message j receives in the first cycle. Solve the following linear program that allocates the residual demands to capacities.

$$\begin{aligned} \max_{x_{ij}, i=1,2,...,n, j=1,2,...,m} & \sum_{i=1}^{n} \sum_{j=1}^{m} r_j x_{ij} p_{ij} \\ \text{s.t.} & \sum_{i=1}^{n} x_{ij} p_{ij} \leq (c_j - D_j)^+, \ \forall j = 1,2,...,m \\ & \sum_{j=1}^{m} x_{ij} = 1 - \sum_{j=1}^{m} x_{ij}^{(1)}, \ \forall i = 1,2,...,n \\ & x_{ij} \geq 0, \ \forall i,j. \end{aligned} \tag{5.6.5}$$

5. Let $x^{(2)}$ be an optimal solution to (5.6.5). In the second cycle, for any user $i \in \{1,2,...,n\}$ who was not sent message in the first cycle, send message j to the user with probability

$$\frac{x_{ij}^{(2)}}{\sum_{k=1}^{m} x_{ik}^{(2)}}.$$

The following proposition establishes the condition that $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$ as required in the Reservation Algorithm.

Proposition 5.6.1. *Consider the non-trivial case where $r_1 > 0$ and $\|p_i\|_1 > 0$ for every*

$i \in \{1, 2, ..., n\}$. If for any two users $i, l \in \{1, 2, ..., n\}$, $i \neq l$, and any two messages $j, k \in \{1, 2, ..., m\}$, $j \neq k$, we have $p_{ik}p_{lj} \neq p_{lk}p_{ij}$, then any optimal solution s^* to the linear program (5.5.1) must satisfy $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$.

Proof. Suppose there is a solution s^* that violates the proposition, i.e., $\sum_{i=1}^{n} \|s_i^*\|_0 > n+m^2$. Then there are more than m^2 users who are simultaneously routed to more than one message according to s^*. Since there are $\binom{m}{2} < m^2$ unique ways to choose unique pairs of messages, there must exist two users $i, l \in \{1, 2, ..., n\}$, $i \neq l$, and two messages $j, k \in \{1, 2, ..., m\}$, $j \neq k$, such that $s_{ij}^* > 0$, $s_{ik}^* > 0$, $s_{lj}^* > 0$, $s_{lk}^* > 0$.

Let γ^* be the vector of optimal dual variables to (5.5.2) that correspond to the capacity constraints. According to (5.5.3), we must have

$$(r_j - \gamma_j^*)p_{ij} = (r_k - \gamma_k^*)p_{ik} \geq (r_1 - \gamma_1^*)p_{i1} = r_1 p_{i1},$$

where the last step follows from the fact that $c_1 = \infty$ and thus $\gamma_1^* = 0$ (see Lemma 5.12.1). Similarly, we have

$$(r_j - \gamma_j^*)p_{lj} = (r_k - \gamma_k^*)p_{lk} \geq (r_1 - \gamma_1^*)p_{l1} = r_1 p_{l1}.$$

Combining the above two equations, we have

$$(r_j - \gamma_j^*)p_{ij} \cdot (r_k - \gamma_k^*)p_{lk} = (r_k - \gamma_k^*)p_{ik} \cdot (r_j - \gamma_j^*)p_{lj}$$

$$\implies (r_j - \gamma_j^*)(r_k - \gamma_k^*) \cdot p_{ij}p_{lk} = (r_k - \gamma_k^*)(r_j - \gamma_j^*) \cdot p_{ik}p_{lj}.$$

Then if $p_{ik}p_{lj} \neq p_{lk}p_{ij}$, we must have $(r_j - \gamma_j^*)(r_k - \gamma_k^*) = 0$. Thus, either $r_j - \gamma_j^* = 0$ or $r_k - \gamma_k^* = 0$, which implies $r_1 p_{i1} = r_1 p_{l1} = 0$. According to Assumption 5.3.1, this implies that $p_{i1} = 0 \Rightarrow \|p_i\|_1 = 0$. Therefore, we must have either $r_1 = 0$ or $\|p_i\|_1 = 0$, which is a contradiction.

□

The regret of the Reservation Algorithm clearly depends on the value of Δ. In the special case of $\Delta = 0$, the algorithm does not reserve any users and thus reduces to the Static Algorithm. The Reservation Algorithm chooses Δ as

$$\Delta = C \cdot \sum_{j=2}^{m} \sqrt{c_j} \cdot \log \sum_{j=2}^{m} c_j, \tag{5.6.6}$$

where $C > 0$ is any small constant that we can further tune to improve the performance of the algorithm. Note that messages 2 to m have finite capacity values (see Section 5.3), so Δ is finite.

5.7 Overview of Analysis of the Reservation Algorithm

In this section, we outline a way to bound the regret of the Reservation Algorithm. We prove this result by analyzing the gap between the expected total reward of this algorithm and the upper bound on the optimal total reward given by Theorem 5.5.1.

Recall that D_j is the random number of clickthroughs that message j receives in the first

cycle, which is a Poisson binomial random variable with mean $\mathbf{E}[D_j] = \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij}$. Define δ_j to be the noise in D_j

$$\delta_j \equiv D_j - \mathbf{E}[D_j] = D_j - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij}. \tag{5.7.1}$$

Our analysis is based on the idea of viewing δ_j as a small *perturbation to the capacity* c_j. After the first cycle, the remaining capacity of message j is $(c_j - D_j)^+$, which can be written as

$$(c_j - D_j)^+ = [(c_j - \delta_j)^+ - \mathbf{E}[D_j]]^+.$$

Intuitively, we view $(c_j - \delta_j)^+$ as the perturbed capacity of message j. Then the noise δ_j arises from this random capacity $(c_j - \delta_j)^+$.

The following is the *perturbed linear program* in which the capacity of message j is perturbed by δ.

$$\begin{aligned} V^{LP}(\delta) = \max_{s_{ij}, i=1,2,...,n, j=1,2,...,m} & \sum_{i=1}^{n}\sum_{j=1}^{m} r_j s_{ij} p_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} s_{ij} p_{ij} \le (c_j - \delta_j)^+, \ \forall j = 1, 2, ..., m \\ & \sum_{j=1}^{m} s_{ij} = 1, \ \forall i = 1, 2, ..., n \\ & s_{ij} \ge 0, \ \forall i, j. \end{aligned} \tag{5.7.2}$$

Note that $V^{LP}(\delta)$ is a random variable as it is a function of the random noise δ. The

following lemma, which will later be used in the analysis, relates our Reservation Algorithm to the perturbed linear program.

Lemma 5.7.1. *If there exists an optimal solution $s(\delta)$ to the perturbed linear program (5.7.2) such that $s(\delta) \geq x^{(1)}$ component wise, then $x^{(1)} + x^{(2)}$ must be an optimal solution to (5.7.2).*

Proof. Suppose there exists an $s(\delta)$ such that $s(\delta) = x^{(1)} + y$ and $y \geq 0$. We can equivalently write the constraints of (5.7.2) as

$$\begin{aligned}
&\sum_{i=1}^{n} s_{ij}(\delta) p_{ij} \leq (c_j - \delta_j)^+ \\
\iff &\sum_{i=1}^{n} y_{ij} p_{ij} \leq (c_j - \delta_j)^+ - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} \\
\iff &\sum_{i=1}^{n} y_{ij} p_{ij} \leq (c_j - \delta_j - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij})^+ \quad \text{(because } y \geq 0\text{)} \\
\iff &\sum_{i=1}^{n} y_{ij} p_{ij} \leq (c_j - D_j)^+.
\end{aligned}$$

And

$$\sum_{j=1}^{m} s_{ij}(\delta) = 1 \iff \sum_{j=1}^{m} y_{ij} = 1 - \sum_{j=1}^{m} x_{ij}^{(1)}.$$

Therefore, knowing that $s(\delta) = x^{(1)} + y$ and $y \geq 0$, we can transform the problem of finding y into the linear program (5.6.5). Thus, $x^{(2)}$ must be at least as good as y, from which it follows that $x^{(1)} + x^{(2)}$ is optimal for (5.7.2). □

Let V^{RA} be the expected total reward of the Reservation Algorithm. We bound the gap between V^{RA} and the upper bound V^{LP} on the optimal reward by using $V^{LP}(\delta)$ as an

intermediate value between the two, and by writing the gap as the sum of two smaller gaps:

$$V^{LP} - V^{RA} = (V^{LP} - \mathbf{E}[V^{LP}(\delta)]) + (\mathbf{E}[V^{LP}(\delta)] - V^{RA}).$$

The first gap, namely $V^{LP} - \mathbf{E}[V^{LP}(\delta)]$, depends on the way in which $V^{LP}(\delta)$ behaves as a function of δ. It is easy to check that $V^{LP}(\delta)$ is concave for all $\delta \leq c$, and the first derivatives of $V^{LP}(\delta)$ with respect to δ are equal to the negative of dual variables associated with capacity constraints of (5.7.2) (Bertsimas and Tsitsiklis, 1997). Therefore, if the size of δ is much smaller than c, we have from Jensen's inequality that

$$\mathbf{E}[V^{LP}(\delta)] \leq V^{LP}(\mathbf{E}[\delta]) = V^{LP}(0) = V^{LP}.$$

The gap $V^{LP} - \mathbf{E}[V^{LP}(\delta)]$ is small if the first derivatives of $V^{LP}(\delta)$, or the dual variables of the perturbed linear program, change smoothly in δ. We will prove in Theorem 5.8.1 of Section 5.8 that if $\gamma(c)$ is a vector of optimal dual variables to the linear program (5.5.1) when the capacity vector is c and if e_k is the unit basis vector with the k-th element being 1, then provided that there exists a constant number $\bar{\gamma} > 0$ such that

$$|\gamma_j(c) - \gamma_j(c + \alpha e_k)| \leq \bar{\gamma} \tag{5.7.3}$$

for all $j, k \in \{1, 2, ..., m\}$, all $\alpha \in [0, 1]$ and for all $c \in \mathbb{R}^m_+$ with $c_1 = \infty$, we have

$$V^{LP} - \mathbf{E}[V^{LP}(\delta)] \leq m^2(n + \sqrt{n})\bar{\gamma}.$$

The second gap, namely $\mathbf{E}[V^{LP}(\delta)]-V^{RA}$, is largely determined by the difference between the assignments made by the Reservation Algorithm and the perturbed linear program. We will show that when $\|\delta\|_1$ is not too large compared to Δ, $x^{(1)}+x^{(2)}$ will be an optimal solution to the perturbed linear program. That is, the assignment made by the Reservation Algorithm will be very close to the assignment for $V^{LP}(\delta)$, and hence the total expected reward of the Reservation Algorithm will be close to $V^{LP}(\delta)$. We prove a condition by which $x^{(1)}+x^{(2)}$ is optimal for the perturbed linear program, by viewing our model as a *generalized network flow* problem and by utilizing flow properties in the generalized network. We will prove in Theorem 5.9.4 of Section 5.9 that

$$\mathbf{E}[V^{LP}(\delta)] - V^{RA} \leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})}\right],$$

where $\mathcal{O}$ denote the event that $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$.

Combining the bounds on the two gaps, we will obtain a bound on the regret of the Reservation algorithm as

$$V^{LP} - V^{RA} \leq m^2(n+\sqrt{n})\bar{\gamma} + \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})}\right]. \qquad (5.7.4)$$

Finally, in Section 5.10, we will show that this bound grows relatively slowly under an appropriate asymptotic scaling of the problem. Part of this analysis involves showing that, under reasonable assumptions, condition (5.7.3) holds when we reduce $\bar{\gamma}$ while scaling up the size of the problem.

5.8 Bound on the First Gap

In this section, we give a bound on the first of the two gaps described in Section 5.7, namely, $V^{LP} - \mathbf{E}[V^{LP}(\delta)]$.

Theorem 5.8.1. *Let $\gamma(c)$ be a vector of optimal dual variables to the linear program (5.5.1) when the capacity vector is c. Let e_k be a unit basis vector with the k-th element being 1. If there exists a constant number $\bar{\gamma} > 0$ such that*

$$|\gamma_j(c) - \gamma_j(c + \alpha e_k)| \leq \bar{\gamma} \tag{5.8.1}$$

for all $j, k \in \{1, 2, ..., m\}$, all $\alpha \in [0, 1]$ and for all $c \in \mathbb{R}^m_+$ with $c_1 = \infty$, then we have

$$V^{LP} - \mathbf{E}[V^{LP}(\delta)] \leq m^2(n + \sqrt{n})\bar{\gamma}.$$

Proof. Fix δ. We want to find the difference between $V^{LP} = V^{LP}(0)$ and $V^{LP}(\delta)$ in terms of the derivative of the function $V^{LP}(\cdot)$. It is known that the Lagrangian multiplier $\gamma_j(c)$ is the marginal benefit of increasing the capacity c_j of message j (Bertsimas and Tsitsiklis, 1997, p.155-156). Thus, for any $u \in \mathbb{R}^m$ such that $u \leq c$, the derivative of $V^{LP}(u)$ with respect to u_j is $-\gamma_j(c - u)$.

Let c be the capacity vector given in the model. Define a vector $\delta^{(j)} \in \mathbb{R}^m$ as

$$\delta_i^{(j)} = \begin{cases} \min(c_i, \delta_i), & \text{for } i \leq j, \\ 0, & \text{for } i > j. \end{cases}$$

For each component $j = 1, 2, ..., m$, we want to integrate the derivative of $V^{LP}(\cdot)$ from c_j to $(c_j - \delta_j)^+$ while keeping other components unchanged. This can be achieved by integrating from vector $c - \delta^{(j-1)}$ to $c - \delta^{(j)}$ for each component j, which can be written as

$$V^{LP}(\delta) = V^{LP} + \sum_{j=1}^{m} \int_0^{|\min(c_j, \delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j) \frac{\delta_j}{|\delta_j|} dx.$$

According to (5.7.3), we must have $\forall u \in \mathbb{R}^m$ such that $u \leq c$,

$$|\gamma_j(c - u) - \gamma_j(c)| \leq (\|u\|_1 + m)\bar{\gamma}, \ \ \forall j = 1, 2, ..., m.$$

This gives us that if $\delta_j < 0$ then

$$\begin{aligned}
\int_0^{|\min(c_j, \delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j) \frac{\delta_j}{|\delta_j|} dx &= \int_0^{-\delta_j} \gamma_j(c - \delta^{(j-1)} - xe_j) dx \\
&\geq \int_0^{-\delta_j} \left[\gamma_j(c) - (\|\delta^{(j-1)} + xe_j\|_1 + m)\bar{\gamma}\right] dx \\
&\geq \int_0^{-\delta_j} \left[\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] dx \\
&= |\delta_j| \left[\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right].
\end{aligned}$$

And if $\delta_j > 0$ then

$$\begin{aligned}
\int_0^{|\min(c_j, \delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j) \frac{\delta_j}{|\delta_j|} dx &= \int_0^{\min(c_j, \delta_j)} -\gamma_j(c - \delta^{(j-1)} - xe_j) dx \\
&\geq \int_0^{\min(c_j, \delta_j)} \left[-\gamma_j(c) - (\|\delta^{(j-1)} + xe_j\|_1 + m)\bar{\gamma}\right] dx \\
&\geq \int_0^{\min(c_j, \delta_j)} \left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] dx
\end{aligned}$$

$$\begin{aligned} &= \min(c_j, \delta_j)\left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right] \\ &\geq \delta_j\left[-\gamma_j(c) - (\|\delta\|_1 + m)\bar{\gamma}\right], \end{aligned}$$

where the last inequality follows from the fact that $\gamma_j(c)$ and $\bar{\gamma}$ are both non-negative.

In sum, we can write

$$\int_0^{|\min(c_j,\delta_j)|} -\gamma_j(c - \delta^{(j-1)} - xe_j)\frac{\delta_j}{|\delta_j|}dx \geq -\delta_j\gamma_j(c) - |\delta_j|(\|\delta\|_1 + m)\bar{\gamma}$$

$$\Longrightarrow V^{LP}(\delta) \geq V^{LP} + \sum_{j=1}^{m}[-\delta_j\gamma_j(c) - |\delta_j|(\|\delta\|_1 + m)\bar{\gamma}] = V^{LP} - \sum_{j=1}^{m}\delta_j\gamma_j(c) - (\|\delta\|_1^2 + m\|\delta\|_1)\bar{\gamma}.$$

Since $\mathbf{E}[\delta] = 0$, we then have

$$\begin{aligned} V^{LP} - \mathbf{E}[V^{LP}(\delta)] &\leq \sum_{j=1}^{m}\mathbf{E}[\delta_j]\gamma_j(c) + (\mathbf{E}[\|\delta\|_1^2] + m\mathbf{E}[\|\delta\|_1])\bar{\gamma} \\ &= (\mathbf{E}[\|\delta\|_1^2] + m\mathbf{E}[\|\delta\|_1])\bar{\gamma}. \end{aligned}$$

Using Jensen's inequality, we can obtain $\mathbf{E}[|\delta_j|] \leq \sqrt{Var(\delta_j)}$. Thus,

$$\begin{aligned} \mathbf{E}[\|\delta\|_1] &= \sum_{j=1}^{m}\mathbf{E}[|\delta_j|] \\ &\leq \sum_{j=1}^{m}\sqrt{Var(\delta_j)} \\ &= \sum_{j=1}^{m}\sqrt{\sum_{i=1}^{n} x_{ij}^{(1)}p_{ij}(1 - x_{ij}^{(1)}p_{ij})} \\ &\leq m\sqrt{n}. \end{aligned}$$

Furthermore,

$$\begin{aligned}
\mathbf{E}[\|\delta\|_1^2] &= \sum_{j=1}^{m}\sum_{k=1}^{m}\mathbf{E}[|\delta_j\delta_k|] \\
&\leq \sum_{j=1}^{m}\sum_{k=1}^{m}\frac{1}{2}\mathbf{E}[\delta_j^2+\delta_k^2] \\
&= \sum_{j=1}^{m}\sum_{k=1}^{m}\frac{1}{2}(Var(\delta_j)+Var(\delta_k)) \\
&= \sum_{j=1}^{m}\sum_{k=1}^{m}\frac{1}{2}\left(\sum_{i=1}^{n}x_{ij}^{(1)}p_{ij}(1-x_{ij}^{(1)}p_{ij})+x_{ik}^{(1)}p_{ik}(1-x_{ik}^{(1)}p_{ik})\right) \\
&\leq m^2n.
\end{aligned}$$

Combining the above inequalities, we get

$$V^{LP}-\mathbf{E}[V^{LP}(\delta)] \leq m^2(n+\sqrt{n})\bar{\gamma}.$$

□

The intuition for the above result is that if the dual vector γ changes smoothly as a function of the capacity c, then the difference in value between the two LP's can be bounded.

5.9 Bound on the Second Gap via Generalized Network Flows

In this section, we cast our static linear-programming formulation as a generalized network flow problem (Wayne, 1999). The generalized networks corresponding to the original and

the perturbed linear programs are the same, except that some of the edges have different capacity values. Based on network-flow properties, we analyze how an optimal flow, or equivalently, an optimal assignment given by the linear program (5.5.1), changes when the capacity values are perturbed by δ. This analysis enables us to bound the second of the two gaps described in Section 5.7, namely the difference $\mathbf{E}[V^{LP}(\delta)] - V^{RA}$.

As a main result of this section, we show that the resulting changes made to an optimal assignment can be bounded by the size of δ. This result will lead to a condition by which $x^{(1)} + x^{(2)}$, which is the solution used by RA (with objective value V^{RA}), is *optimal* for the perturbed linear program (with objective value $\mathbf{E}[V^{LP}(\delta)]$).

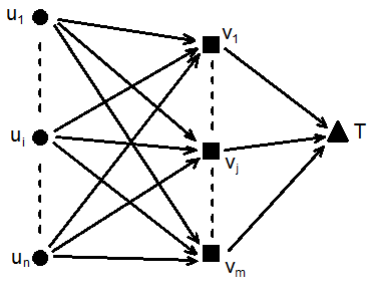


Figure 5.2: In a generalized flow network, every user i corresponds to a user node u_i, and every message j corresponds to a message node v_j. T is the sink.

The network-flow problem we describe next specializes the generalized network-flow problem (Wayne, 1999) to a specific graph. There will be a few minor changes that we will point out presently.

5.9.1 Construction of a generalized flow network

We now construct our generalized network-flow problem. In our flow network, there is a set U of user nodes and a set V of message nodes. Each user $i \in \{1, 2, ..., n\}$ corresponds to a user node $u_i \in U$, and each message $j \in \{1, 2, ..., m\}$ corresponds to a message node $v_j \in V$. There is also a sink T. Every user node u_i has an *initial excess* $e(u_i) = 1$. The initial excesses at all other nodes are 0. The unit initial excess at a user node u_i means that there is a single user i to be assigned to a message.

Figure 5.2 illustrates the set E of directed edges in the generalized network. From every user node $u_i \in U$ to every message node $v_j \in V$, there is an edge $(u_i, v_j) \in E$ having capacity $c(u_i, v_j) = \infty$. From every message node v_i to the sink, there is an edge $(v_i, T) \in E$ having capacity $c(v_j, T) = c_j$.

A *generalized pseudo-flow* is a function $f : E \to \mathbb{R}_+$ that satisfies the capacity constraints $0 \leq f(u, v) \leq c(u, v)$, $\forall (u, v) \in E$. Note that according to Assumption 5.3.1, starting from any user node $u_i \in U$, there is always a path $u_i \to v_1 \to T$ having infinite capacity. Thus, it is always possible to push all initial excesses at all user nodes to the sink.

In a generalized network-flow problem, flows might not be conserved along edges. Every edge $(u, v) \in E$ is associated with a *gain factor* $\mu(u, v)$. When a flow is sent along an edge $(u, v) \in E$ by a generalized pseudo-flow f, the size of the flow leaving from u is $f(u, v)$, while the size of the flow arriving at v is $\mu(u, v)f(u, v)$. In our problem, every edge (u_i, v_j) from a user node u_i to a message node v_j has gain factor $\mu(u_i, v_j) = p_{ij}$. In this way, the size of a flow $f(u_i, v_j)$ leaving u_i represents the allocation s_{ij} in the linear program (5.5.1), while the

size of flow $\mu(u_i, v_j)f(u_i, v_j) = p_{ij}f(u_i, v_j)$ arriving at v_j represents the expected number of clickthroughs that message j receives from user i. Every edge (v_j, T) from a message node v_j to the sink has gain factor $\mu(v_j, T) = r_j$. In this way, the size of a flow $f(v_j, T)$ leaving v_j represents the total expected number of clickthroughs assigned to message j, while the size of the flow $\mu(v_j, T)f(v_j, T) = r_j f(v_j, T)$ arriving at T represents the expected reward earned from message j.

The *excess* of a node u (as opposed to the initial excess) is the difference between the initial excess at u and the net outflow at u:

$$e(u) - \sum_{(u,v)\in E} f(u, v) + \sum_{(v,u)\in E} \mu(v, u)f(v, u).$$

In this thesis chapter, we call a generalized pseudo-flow f a *feasible flow* if f has zero excess at all nodes other than T. Consequently, every feasible flow corresponds to a feasible solution to the linear program (5.5.1) in the following way

$$f(u_i, v_j) = s_{ij},\ \forall i = 1, 2, ..., n, j = 1, 2, ..., m,$$

$$f(v_j, T) = \sum_{i=1}^{n} s_{ij}p_{ij},\ \forall j = 1, 2, ..., m.$$

The objective of our generalized network-flow problem is to find a feasible flow that maximizes the excess at the sink, which is by definition

$$e(T) - \textstyle\sum_{(T,v)\in E} f(T, v) + \sum_{(v,T)\in E} \mu(v, T)f(v, T)$$

$$
\begin{aligned}
&= && 0 - 0 + \textstyle\sum_{(v,T)\in E} \mu(v,T) f(v,T) \\
&= && \textstyle\sum_{(v,T)\in E} \mu(v,T) f(v,T) \\
&= && \textstyle\sum_{j=1}^{m} r_j f(v_j,T) \\
&= && \textstyle\sum_{j=1}^{m} r_j \sum_{i=1}^{n} s_{ij} p_{ij}. \qquad (5.9.1)
\end{aligned}
$$

Thus, every optimal solution s^* to (5.5.1) must correspond to an optimal flow f^*.

For any generalized pseudo-flow f, the *residual network* with respect to f is the network $G_f = (U, V, T, E_f, c_f, \mu, e)$, where E_f is the set of *directed* edges in the residual network defined as

$$E_f = \{(u,v) \in E : f(u,v) < c(u,v)\} \cup \{(v,u) : (u,v) \in E, f(u,v) > 0\},$$

and c_f is the residual capacity defined for every edge in E_f in the following way. If $(u,v) \in E$, then $c_f(u,v) = c(u,v) - f(u,v)$; otherwise, $c_f(u,v) = \mu(v,u) f(v,u)$.

For every edge $(u,v) \in E_f$ in the residual network, we further define its *length* as $w(u,v) = -\log \mu(u,v)$ if $(u,v) \in E$, and $w(u,v) = \log \mu(v,u)$ if $(u,v) \notin E$. By this definition, if a flow is pushed in the residual network along a path $d_1 \to d_2 \to \cdots \to d_k$ such that the excesses at all nodes are unchanged except at d_1 and d_k, then the size of the flow arriving at d_k is

$$e^{-[w(d_1,d_2)+w(d_2,d_3)+\cdots+w(d_{k-1},d_k)]} \qquad (5.9.2)$$

times the size of the flow leaving node d_1 (for a more detailed discussion, see Wayne (1999)).

Therefore, the flow arriving at the last node of the path is larger (smaller) than the flow leaving from the first node if the total length of the path $w(d_1, d_2)+w(d_2, d_3)+\cdots+w(d_{k-1}, d_k)$ is negative (positive). In particular, if the path is a cycle, i.e., d_1 is the same node as d_k, then the flow returning to d_1 is greater than the flow leaving d_1 if and only if the cycle is a negative cycle, i.e., the total length of the cycle is negative. In other words, if we push a flow along a negative cycle, we can increase the excess at one node in the cycle without changing the excesses at all other nodes.

Previous works have studied various types of *augmentations* for the generalized network flow problem. Given a generalized pseudo-flow, an augmentation modifies flow values on a certain subset of edges in the residual network, such that the resulting flow remains a generalized pseudo-flow. Most often, an augmentation keeps the excesses of most nodes unchanged. In this thesis chapter we focus on two types of augmentations: augmentation along paths and along cycles. Here a cycle can be as well seen as a path that starts and ends at the same node. Both types of augmentations aim at increasing the excess at the sink, thereby increasing the objective value of the solution corresponding to the resulting flow as we showed in (5.9.1). When an augmentation is performed along a path in the residual network from node v to the sink T, a flow is pushed along the path such that the excess at v is reduced, the excess at T is increased, and the excess at every other node is unchanged. If v is itself the sink, and the augmentation is performed along a cycle that covers $v = T$, then the excess at $v = T$ increases if and only if the cycle is a negative cycle (see (5.9.2)).

5.9.2 Properties of optimal flows in the generalized flow network

Based on special properties of our model, the following theorem strengthens a known result (for example, see Wayne (1999)) that when a generalized pseudo-flow has no negative cycle in the residual network and has zero excess at all nodes other than T, then the maximum flow is attained.

Theorem 5.9.1. *In our model, a feasible flow f is optimal if and only if there is no negative cycle in the residual network G_f.*

Proof. It has been proved that when a generalized pseudo-flow has zero excess at all nodes other than T and has no negative cycle in the residual network, the flow is optimal (for example, see Wayne (1999)). Thus, a feasible flow f with no negative cycles in G_f must be optimal.

To prove the other direction, suppose f is an optimal flow with a negative cycle in the residual network. Then we can push a flow along the cycle to create a positive excess at some user node u while keeping the excess at all other nodes unchanged. We then direct this excess created at node u through the path $u \to v_1 \to T$ that has infinite capacity. In this way, we obtain a feasible flow with a larger size of excess at the sink, which proves that the original flow f is not optimal. □

Now we consider the problem of how to modify an optimal flow when the capacity of some message $k \in \{2, ..., m\}$ is perturbed. Note that perturbing the capacity of message 1 has no impact as it has infinite capacity. We give algorithms that find an optimal flow for the perturbed network by modifying a given optimal solution to the original network.

These algorithms are designed mainly for the purpose of providing structural results for our model. The algorithms are not meant to be implemented. Rather, we use them to argue certain facts about the way in which an optimal solution changes when the capacity of some message $k \in \{2, ..., m\}$ changes.

Let $G = (U, V, T, E, c, \mu, e)$ and $\bar{G} = (U, V, T, E, \bar{c}, \mu, e)$ be two generalized networks of our model that differ by the capacity on a single edge (v_k, T) for some given $k \in \{2, ..., m\}$. Given an optimal flow f for the network G, the following algorithms find an optimal flow for $\bar{G}$ by successively performing augmentations along shortest paths or negative cycles in the residual network. We give a different algorithm depending on whether $\bar{c}(v_k, T)$ is greater than or less than $c(v_k, T)$.

Case 1. Algorithm for the case $0 \leq \bar{c}(v_k, T) < c(v_k, T)$:

1. Construct a generalized pseudo-flow $\bar{f}$ for $\bar{G}$ as follows. Let

$$\bar{f}(v_k, T) = \min(\bar{c}(v_k, T), f(v_k, T)) \tag{5.9.3}$$

for the edge (v_k, T), and let $\bar{f}(u, v) = f(u, v)$ for all other edges $(u, v) \in E$. Since f is optimal for G, there is no negative cycle in the residual network G_f according to Theorem 5.9.1. It is easy to see that there is also no negative cycle in the residual network $\bar{G}_{\bar{f}}$ because according to (5.9.3), the edge set of G_f must contain the edge set of $\bar{G}_{\bar{f}}$. Thus, if $\bar{f}$ is feasible, then $\bar{f}$ is already an optimal flow for $\bar{G}$.

Now suppose that $\bar{f}$ is not feasible. It must be that the excess at node v_k is positive.

2. Find a shortest path from v_k to T in the residual network $\bar{G}_{\bar{f}}$.

3. Perform an augmentation along the shortest path such that either the excess at node v_k reaches 0 or the residual capacity of one of the edges in the path is reduced to 0. Update the residual network $\bar{G}_{\bar{f}}$.

4. Stop if $\bar{f}$ becomes feasible. Otherwise, go to step 2.

It is easy to check that there will be no negative cycle generated in the residual network throughout the algorithm, because otherwise, the previous shortest path would have passed though the negative cycle to further reduce the total length of the path. Therefore, the algorithm outputs an optimal flow $\bar{f}$ for $\bar{G}$.

Case 2. Algorithm for the case $\bar{c}(v_k, T) > c(v_k, T)$.

1. Start with an initial flow $\bar{f} = f$. Suppose $\bar{f}$ is not optimal for $\bar{G}$. Then since $\bar{f}$ is feasible for $\bar{G}$, there must exist some negative cycle in the residual network $\bar{G}_{\bar{f}}$. Moreover, all negative cycles in $\bar{G}_{\bar{f}}$ must include the edge (v_k, T).

2. Find a simple cycle, i.e., a cycle that contains only distinct nodes, having the smallest (negative) total length. This can be solved by finding a shortest path from T to v_k.

3. Perform an augmentation along this shortest negative cycle to increase the excess at T such that the residual capacity of one of the edges in the cycle is reduced to 0. Update the residual network $\bar{G}_{\bar{f}}$. After the augmentation, all negative cycles in $\bar{G}_{\bar{f}}$, if there is any, must still pass through (v_k, T) because the augmentation is performed along the shortest cycle.

4. Stop if there is no more negative cycles in $\bar{G}_{\bar{f}}$. Otherwise, go to step 2.

This algorithm outputs an optimal flow $\bar{f}$ for the network $\bar{G}$, as $\bar{f}$ is always feasible throughout the algorithm and, by the end of the algorithm, there is no negative cycle in the residual network.

The main structural property of our model that we want to prove is given by the following theorem.

Theorem 5.9.2. *For any two message nodes $v_j, v_k \in V$ and $v_k \neq v_1$, when $c(v_k, T)$ changes to $\bar{c}(v_k, T)$ (with the capacity values of all other edges unchanged), the total size of augmenting flows leaving v_j in the above algorithm is at most $|\bar{c}(v_k, T) - c(v_k, T)|r_k/r_1$.*

Proof. Since the bound stated in the theorem is additive and linear in $|c(v_k, T) - \bar{c}(v_k, T)|$, it suffices to prove the case that the gap $|c(v_k, T) - \bar{c}(v_k, T)|$ is very small. It is easy to see that when $|c(v_k, T) - \bar{c}(v_k, T)|$ is small enough, the network flow algorithms will perform no more than one augmentation. Thus, we only focus on the case where the algorithms perform only one augmentation.

For Case 1, recall that $0 \leq \bar{c}(v_k, T) < c(v_k, T)$. Suppose initially $\bar{f}$ is not optimal for $\bar{G}$, i.e., $\bar{f}(v_k, T) < f(v_k, T)$. In other words, some users are originally assigned to message k, but now have to move to other messages due to the decrease in the capacity.

Suppose that the algorithm performs exactly one augmentation along a shortest path from v_k to T. Let θ_j be the size of the augmenting flow that leaves node v_j, $\forall v_j \in V$. We need to show that $\forall v_j \in V$,

$$\theta_j \leq \theta_k \frac{r_k}{r_1}.$$

This is clearly true when $j = k$, as our model requires $r_k \geq r_1$ (see Section 5.3. Note that the theorem holds even if we allow $r_k < r_1$, in which case changing the capacity of message k will have no affect at all).

Suppose there is some $l \neq k$ such that $\theta_l > \theta_k \frac{r_k}{r_1}$. Let u_i be the user node preceding v_l in the augmenting path. Then the size of the augmenting flow passing through u_i is θ_l / p_{il}. Let P be the part of the augmenting path from v_k to u_i. According to (5.9.2), we can obtain the total length of P from the relative size of the augmenting flow at nodes v_k and u_i. The total length of P is

$$\log \frac{\theta_k}{\theta_l / p_{il}}.$$

Consider the cycle $P \to v_1 \to T \to v_k$ that returns to v_k. This cycle must exist in the residual network because the edges (u_i, v_1) and (v_1, T) always have infinite capacity, and the edge (T, v_k) exists in G_f since $f(v_k, T) > 0$. The total length of the cycle is

$$\begin{aligned}
&\log \frac{\theta_k}{\theta_l / p_{il}} + w(u_i, v_1) + w(v_1, T) + w(T, v_k) \\
=&\log \frac{\theta_k}{\theta_l / p_{il}} - \log p_{i1} - \log r_1 + \log r_k \\
=&\log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \cdot \frac{p_{il}}{p_{i1}} \right) \\
\leq&\log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \right) \quad \text{(by Assumption 5.3.1)} \\
<&0.
\end{aligned}$$

In other words, the cycle is a negative cycle. According to Theorem 5.9.1, this contradicts the fact that f is optimal for G. Thus, we must have $\theta_j \leq \theta_k \frac{r_k}{r_1}$ for every $v_j \in V$.

For Case 2, recall that $\bar{c}(v_k, T) > c(v_k, T)$. Suppose initially $\bar{f} = f$ is not optimal for $\bar{G}$. Recall that any negative cycle in the residual network $\bar{G}_{\bar{f}}$ must pass through (v_k, T). Again suppose there is exactly one augmentation performed by the algorithm and let θ_j be the size of the augmenting flow that leaves node $v_j \in V$. Similarly, we need to prove that $\theta_j \leq \theta_k \frac{r_k}{r_1}$, $\forall j = 1, 2, ..., m$. The case of $j = k$ is again trivially true as we only consider the case $r_k \geq r_1$.

Suppose there is some $l \neq k$ such that $\theta_l > \theta_k \frac{r_k}{r_1}$. Let u_i be the user node following v_l in the augmenting cycle. Then the size of the augmenting flow that passes through u_i is θ_l / p_{il}. Furthermore, the size of the augmenting flow that leaves T is $\theta_k r_k$. Let P be the part of the augmenting cycle from T to u_i. According to (5.9.2), we can obtain the total length of P from the relative size of the augmenting flow at T and u_i. The total length of P is

$$\log \frac{\theta_k r_k}{\theta_l / p_{il}}.$$

Consider the new cycle $P \to v_1 \to T$ which returns to T. This new cycle must exist in the residual network because the edges (u_i, v_1) and (v_1, T) have infinity capacity. The total length of the new cycle is

$$\begin{aligned} &\log \frac{\theta_k r_k}{\theta_l / p_{il}} + w(u_i, v_1) + w(v_1, T) \\ =& \log \frac{\theta_k r_k}{\theta_l / p_{il}} - \log p_{i1} - \log r_1 \\ =& \log \left(\frac{\theta_k}{\theta_l} \cdot \frac{r_k}{r_1} \cdot \frac{p_{il}}{p_{i1}} \right). \end{aligned}$$

Similar to the previous argument, this total length is negative, because $\theta_l > \theta_k \frac{r_k}{r_1}$ and accord-

ing to Assumption 5.3.1, $p_{il} \leq p_{i1}$. In other words, this new cycle is also a negative cycle. However, this new cycle does not pass through edge (v_k, T), which forms a contradiction. Thus, for Case 2 we also have $\theta_j \leq \theta_k \frac{r_k}{r_1}$, $\forall j = 1, 2, ..., m$.

□

5.9.3 Bounding the second gap

Now we consider how the optimal flow for the perturbed system relates to the assignment made by the Reservation Algorithm.

When the capacity of each message k is perturbed by δ_k, we apply Theorem 5.9.2 to each message $k = 2, 3, ..., m$ by setting $\bar{c}(v_k, T) = (c_k - \delta_k)^+$ (message 1 has infinite capacity and thus perturbing it has no impact). Then the total size of augmenting flows leaving every message node v_j due to the perturbation δ, summed over all m messages, is at most

$$\sum_{k=2}^{m} |\delta_k| r_k / r_1. \tag{5.9.4}$$

Finally, we prove Theorem 5.9.3 by showing that if (5.9.4) is smaller than Δ, then the assignment $x^{(1)} + x^{(2)}$ of the Reservation Algorithm is optimal for the perturbed system. The proof relates the Reservation Algorithm to the network-flow algorithms above, by showing that the shortest augmenting paths or cycles will pass only through user nodes u_i that have the smallest ratios of p_{ij}/p_{ik} for some messages j, k. Such users are just the ones reserved by the Reservation Algorithm. To be more specific, the length of the augmenting path or cycle

that goes through nodes $v_j \to u_i \to v_k$ is

$$w(v_j, u_i) + w(u_i, v_k) = \log \frac{p_{ij}}{p_{ik}}.$$

Since the augmenting paths and cycles seek the shortest length, they will only pass through user nodes i that have the smallest values of p_{ij}/p_{ik}. We will show that such users i are reserved in the set R^{jk}. Thus augmenting flows will not affect the assignment $x^{(1)}$, which leads to the conclusion that $x^{(1)}$ is compatible with the optimal assignment for the perturbed system.

Theorem 5.9.3. *If $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$, then $x^{(1)} + x^{(2)}$ is an optimal solution to the perturbed linear program* (5.7.2).

Proof. Starting with an optimal flow f for the unperturbed network, as defined by the optimal solution s^* to the LP (5.5.1), we apply the network-flow algorithms above to obtain an optimal flow $\bar{f}$ for the perturbed network, as defined by the solution of (5.7.2). According to Lemma 5.7.1, it suffices to show that if $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$, we must have $\bar{f}(u_i, v_j) \geq x_{ij}^{(1)}$ for all $u_i \in U$ and $v_j \in V$.

By the definition (5.6.3) of $x^{(1)}$, we must have $s^* \geq x^{(1)}$ and thus initially $f(u_i, v_j) \geq x_{ij}^{(1)}$ for all $u_i \in U$ and $v_j \in V$. Note that for any edge (u_i, v_j), the only way to reduce its flow value is to send an augmenting flow that passes (v_j, u_i) in the residual graph. We claim that if $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$, no augmenting flow will ever pass through any edge (v_j, u_i) such that $x_{ij}^{(1)} > 0$. This claim implies that for any $x_{ij}^{(1)} > 0$, the flow value on (u_i, v_j) can only increase

during the network flow algorithm, which proves that eventually $\bar{f}(u_i, v_j) \geq f(u_i, v_j) \geq x_{ij}^{(1)}$ for any $x_{ij}^{(1)} > 0$.

To see the claim, suppose that at some step during augmentation, the augmenting flow passes through (v_j, u_i) for some $v_j \in V$ and $u_i \in U$. The node following u_i in the augmenting flow must be another message node $v_k \in V$. The total length of $v_j \to u_i \to v_k$ is

$$w(v_j, u_i) + w(u_i, v_k) = \log p_{ij} - \log p_{ik} = \log \frac{p_{ij}}{p_{ik}}.$$

Since the augmenting path or cycle has the *shortest length*, user i must have the smallest value of p_{lj}/p_{lk} among all users l such that (v_j, u_l) exists in the residual network right before the augmentation is performed. Recall that in the Reservation algorithm, we defined a set R^{jk} of users l who have the smallest values of the ratio p_{lj}/p_{lk}. Therefore, if any user in R^{jk} has a flow going to v_j right before the augmentation is performed, we must have $i \in R^{jk}$ and thus $x_{ij}^{(1)} = 0$. When there is a tie in determining a user having the smallest ratio, we assume without loss of generality that the network-flow algorithms first chooses for augmentation a user from the set R^{jk}.

Initially, according to Step 2 of the Reservation algorithm, the total size of flow that v_j receives from the users in R^{jk} according to the solution s^* is at least Δ (or else all users are in R^{jk}). Then if $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$, we know by Theorem 5.9.2 that the total amount of augmenting flows leaving v_j is no more than Δ. Thus, the flows that v_j receives from users in R^{jk} can be reduced by at most Δ throughout the network flow algorithms, which implies

that at any augmentation step of the network flow algorithms, at least one user in R^{jk} has a flow going to v_j. Combined with the argument above, this proves the claim. □

Next, the gap between the total expected reward V^{RA} of the Reservation Algorithm and the optimal objective value of the perturbed linear program is given by the following theorem.

Theorem 5.9.4. *Let $\mathcal{O}$ denote the event that $\sum_{j=2}^{m} |\delta_j| \frac{r_j}{r_1} \leq \Delta$. We have*

$$\mathbf{E}[V^{LP}(\delta)] - V^{RA} \leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})} \right],$$

where the expectation is taken with respect to δ, and $\bar{\mathcal{O}}$ is the complement of $\mathcal{O}$.

Proof. Let $D_j^{(2)}$ be the number of clickthroughs that message j receives in the second cycle. The total expected reward of the Reservation Algorithm is

$$\begin{aligned}
V^{RA} &= \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j, D_j + D_j^{(2)})] \\
&= \sum_{j=1}^{m} r_j \mathbf{E}[\delta_j + \min(c_j - \delta_j, D_j - \delta_j + D_j^{(2)})] \\
&= \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, D_j - \delta_j + D_j^{(2)})] \\
&= \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, \mathbf{E}[D_j] + D_j^{(2)})] \\
&= \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)})]. \qquad (5.9.5)
\end{aligned}$$

Here we interpret $c_j - \delta_j$ as the perturbed capacity of message j, and $\sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)}$ as

the 'actual' number of clickthroughs that message j receives in both two cycles. In other words, we view δ as an exogenous noise that arises in the capacity values.

According to Theorem 5.9.3, $x^{(1)}+x^{(2)}$ is an optimal solution to the perturbed linear program conditional on $\mathcal{O}$. Then the expected objective value of the perturbed linear program can be written as

$$\begin{aligned}\mathbf{E}[V^{LP}(\delta)] &= \mathbf{E}[V^{LP}(\delta)|\mathcal{O}]P(\mathcal{O}) + \mathbf{E}[V^{LP}(\delta)|\bar{\mathcal{O}}]P(\bar{\mathcal{O}}) \\ &= \mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + \mathbf{E}[V^{LP}(\delta)|\bar{\mathcal{O}}]P(\bar{\mathcal{O}}) \\ &\leq \mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij}|\mathcal{O}]P(\mathcal{O}) + n \cdot \max_{j=1,2,...,m} r_j \cdot P(\bar{\mathcal{O}}). \qquad (5.9.6)\end{aligned}$$

Here the last inequality follows from the fact that $n \cdot \max_{j=1,2,...,m} r_j$ is an upper bound on the total reward value of any allocation.

Conditional on $\mathcal{O}$, the constraint of the perturbed linear program gives

$$\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} \leq (c_j - \delta_j)^+, \forall j = 1, 2, ..., m.$$

This implies that, if $\sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} > 0$, we must have $c_j > \delta_j$. On the other hand, if $\sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} = 0$, we must have $D_j = 0 \Longrightarrow \delta_j = 0$. In sum, conditional on the event $\mathcal{O}$, we always have $c_j \geq \delta_j$, and thus

$$\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} \leq c_j - \delta_j, \forall j = 1, 2, ..., m. \qquad (5.9.7)$$

Define

$$\delta_j^{(2)} \equiv D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}.$$

Combining (5.9.5) and (5.9.6), we get

$$\begin{aligned}
&\mathbf{E}[V^{LP}(\delta)] - V^{RA} \\
\leq &\mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} | \mathcal{O}] P(\mathcal{O}) + n \cdot \max_{j=1,2,\ldots,m} r_j \cdot P(\bar{\mathcal{O}}) \\
&- \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)})] \\
\leq &\mathbf{E}[\sum_{j=1}^{m} r_j \sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} | \mathcal{O}] P(\mathcal{O}) + n \cdot \max_{j=1,2,\ldots,m} r_j \cdot P(\bar{\mathcal{O}}) \\
&- \sum_{j=1}^{m} r_j \mathbf{E}[\min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)}) | \mathcal{O}] P(\mathcal{O}) \\
= &\sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij} + D_j^{(2)}) | \mathcal{O}] P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}) \\
= &\sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} + \delta_j^{(2)}) | \mathcal{O}] P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}),
\end{aligned}$$

where $\delta_j^{(2)} = D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}$. We apply the identity $a - \min(b, a + c) \leq \max(0, -c)$ for any $a \leq b$ to the above equation and continue to deduce that

$$\begin{aligned}
&\sum_{j=1}^{m} r_j \mathbf{E}[\sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} - \min(c_j - \delta_j, \sum_{i=1}^{n} (x_{ij}^{(1)} + x_{ij}^{(2)}) p_{ij} + \delta_j^{(2)}) | \mathcal{O}] P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}) \\
\leq &\sum_{j=1}^{m} r_j \mathbf{E}[\max(0, -\delta_j^{(2)}) | \mathcal{O}] P(\mathcal{O}) + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}) \quad \text{because of (5.9.7)} \\
\leq &\sum_{j=1}^{m} r_j \mathbf{E}[\max(0, -\delta_j^{(2)})] + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}})
\end{aligned}$$

$$\leq \sum_{j=1}^{m} r_j \mathbf{E}[|\delta_j^{(2)}|] + n \max_{j=1,2,\ldots,m} r_j P(\bar{\mathcal{O}}). \tag{5.9.8}$$

For any user i who satisfies $\|s_i^*\|_0 = 1$, where s^* is an optimal solution to the LP (5.5.1), we must have either $\|x_i^{(1)}\|_1 = 1$ or $\|x_i^{(2)}\|_1 = 1$. That is, we know for sure in which cycle this user will receive a message. Then according to the condition $\sum_{i=1}^{n} \|s_i^*\|_0 \leq n + m^2$ required in Step 1 of the Reservation Algorithm, there are at most m^2 users i that satisfy $\|s_i^*\|_0 > 1$. This implies that at most m^2 users i have $0 < \|x_i^{(2)}\|_1 < 1$ (i.e., each of these users has positive probabilities to both receive and not receive a message in the first cycle).

Let $U^{(2)}$ be the set of users who are sent messages in the second cycle. Then there are at most m^2 users who have positive probabilities to be either in or not in $U^{(2)}$. Thus,

$$\begin{aligned}
\mathbf{E}[|\delta_j^{(2)}|] =& \mathbf{E}[|D_j^{(2)} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|] \\
\leq& \mathbf{E}[|D_j^{(2)} - \sum_{i \in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}|] + \mathbf{E}[|\sum_{i \in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij} - \sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|] \text{ by the triangle inequality} \\
\leq& \mathbf{E}[|D_j^{(2)} - \sum_{i \in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}|] + m^2 \quad \text{since at most } m^2 \text{ users } i \text{ have } 0 < \|x_i^{(2)}\|_1 < 1 \\
\leq& \sqrt{\mathbf{E}\left[\left(D_j^{(2)} - \sum_{i \in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}\right)^2\right]} + m^2 \quad \text{by Jensen's inequality} \\
=& \sqrt{\mathbf{E}\left[\sum_{i \in U^{(2)}} \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij} \left(1 - \frac{x_{ij}^{(2)}}{\|x_i^{(2)}\|_1} p_{ij}\right)\right]} + m^2 \text{ by the variance of binary variables} \\
\leq& \sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij} \left(1 - x_{ij}^{(2)} p_{ij}\right)\right]} + 2m^2 \quad \text{since at most } m^2 \text{ users } i \text{ have } 0 < \|x_i^{(2)}\|_1 < 1
\end{aligned}$$

$$\leq \sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}\right]} + 2m^2. \tag{5.9.9}$$

By Theorem 5.9.3, conditional on $\mathcal{O}$, $\sum_{i=1}^{n}(x_{ij}^{(1)}+x_{ij}^{(2)})p_{ij}$ and $\sum_{i=1}^{n} s_{ij}^{*} p_{ij}$ stand for the amount of the capacity of message j allocated to users in the perturbed and original linear programs, respectively. In the generalized residual network, an increment in the expected number of clickthroughs that message j receives corresponds to an augmenting flow that passes through the edge (v_j, T). According to Theorem 5.9.2, the total increase in the expected number of clickthroughs that message j receives due to the perturbation δ must be no more than $\sum_{k=2}^{m} |\delta_k| \frac{r_k}{r_1}$. In other words, conditional on $\mathcal{O}$,

$$\mathbf{E}[\sum_{i=1}^{n}(x_{ij}^{(1)} + x_{ij}^{(2)})p_{ij} - \sum_{i=1}^{n} s_{ij}^{*} p_{ij}|\mathcal{O}] \leq \sum_{k=2}^{m} |\delta_k| \frac{r_k}{r_1} \leq \Delta, \ \forall j = 1, 2, ..., m.$$

$$\begin{aligned}
\Longrightarrow \mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|\mathcal{O}] &\leq \Delta + \mathbf{E}[\sum_{i=1}^{n} s_{ij}^{*} p_{ij} - \sum_{i=1}^{n} x_{ij}^{(1)} p_{ij}|\mathcal{O}] \\
&= \Delta + \mathbf{E}[\sum_{i=1}^{n} s_{ij}^{*} p_{ij} \mathbf{1}(i \in \bigcup_{k=1, k\neq j}^{m} R^{jk})|\mathcal{O}] \\
&\leq \Delta + (m-1)(\Delta + 1) \\
&= m\Delta + m - 1,
\end{aligned}$$

where the last inequality follows from (5.6.4). From this, we can deduce that

$$\mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}] \ \leq \ \mathbf{E}[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}|\mathcal{O}]P(\mathcal{O}) + nP(\bar{\mathcal{O}})$$

$$\leq \quad m\Delta + m - 1 + nP(\bar{\mathcal{O}}).$$

Combining this result with (5.9.8) and (5.9.9), we can deduce that

$$\begin{aligned}
\mathbf{E}[V^{LP}(\delta)] - V^{RA} &\leq \sum_{j=1}^{m} r_j \left[\sqrt{\mathbf{E}\left[\sum_{i=1}^{n} x_{ij}^{(2)} p_{ij}\right]} + 2m^2 \right] + n \max_{j=1,2,...,m} r_j P(\bar{\mathcal{O}}) \\
&\leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta + m - 1 + nP(\bar{\mathcal{O}})} + 2m^2 \right] + n \max_{j=1,2,...,m} r_j P(\bar{\mathcal{O}}) \\
&\leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta} + 3m^2 + nP(\bar{\mathcal{O}}) + \sqrt{nP(\bar{\mathcal{O}})} \right].
\end{aligned}$$

□

5.10 Performance Analysis in an Asymptotic Regime

Due to the huge number of users and large capacity values involved in the real problem, we are interested in studying the regret in the asymptotic regime. That is, we will quantify the rate at which the regret grows as we scale up the size of the system.

5.11 Big-Data scaling

Define a series of problems with increasing sizes as follows. In the problem of size t, for $t = 1, 2, ..., \infty$, the number of users is $n = t \cdot \bar{n}$, and the capacity of message j is $c_j = t \cdot \bar{c}_j, \forall j = 1, 2, ..., m$, for some fixed $\bar{c}$ and $\bar{n}$. The number m of distinct messages and the

reward r_j of each message j are fixed for all t. We require that Assumption 5.3.1 always holds.

A naive way of scaling n users is to replicate the same set of users t times. However, as we increase the size of the user pool, we also increase its diversity. In reality, it is extremely rare that two active users have identical profiles because the profiles are learned based on users' personal history. A user's history includes, for example, the types of products viewed, purchased and put in the shopping cart over the past 3, 7, 30 days, the operating system of his or her mobile phone, and the type of products sent to the user via push notification over the past several days. Given such high granularity of user histories, it is unlikely that any two active users would have identical profiles. Thus, we can think of the profiles of all $n = t\bar{n}$ users as samples drawn from some continuous distribution. As we scale up t, we are drawing more sample points from this continuous distribution. We called this type of scaling *big-data scaling*.

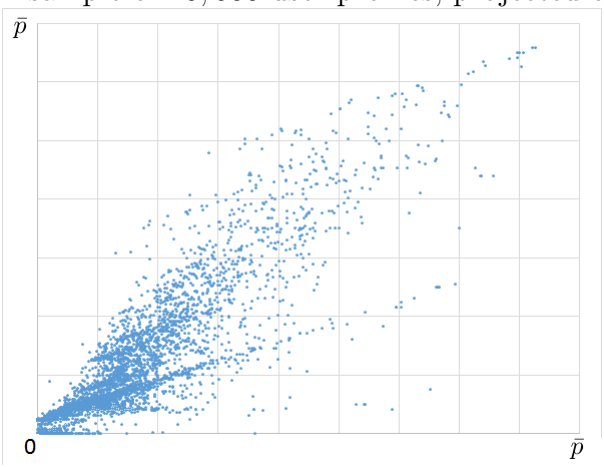
Hitherto, we have viewed the user pools as fixed. From this point onwards, we will begin to view user pools as random. Formally, let $p_i^t = (p_{i1}^t, p_{i2}^t, ..., p_{im}^t)$ denote the (random) profile of user i in the problem of size t. According to Assumption 5.3.1, p_i^t takes values in the region

$$\mathcal{P} = \{y \in [0, \bar{p}]^m : y_1 \geq y_j, \forall j = 2, 3, ..., m\} \tag{5.11.1}$$

for some constant $\bar{p}$. By default, we take $\bar{p} = 1$. All of our proofs extend to the case that $0 < \bar{p} < 1$.

Assumption 5.11.1 (Big-Data Scaling)**.** *Every p_i^t is independently drawn from a continuous*

Figure 5.3: A random sample of 10,000 user profiles, projected onto two coordinates.



distribution with density $f(\cdot)$ having domain $\mathcal{P}$. Furthermore, the density $f(\cdot)$ has upper and lower bounds

$$0 < \underline{f} \leq f(y) \leq \bar{f}, \ \forall y \in \mathcal{P}.$$

This assumption can be justified by empirical evidence. We verified that in the real dataset, less than 2% of *active users* have profiles that collide with those of other users. In Figure 5.3, we illustrate the density $f(\cdot)$ by plotting the estimated click-through probabilities for more than 10,000 users randomly drawn from the real dataset.

Based on big-data scaling, we will analyze the rate at which the regret $V_t^{\mathrm{OPT}} - V_t^{\Pi}$ grows as a function of the scaling parameter t. Here, V_t^{Π} and V_t^{OPT} denote the expected total reward of the t-th problem under algorithms Π and OPT, respectively, conditional upon the random user pools for problem t.

Note that as functions of the random user pools for the respective problems, $\{V_t^{\Pi}\}$ and $\{V_t^{\mathrm{OPT}}\}$ are sequences of random variables. Thus, our bound on the regret will be a prob-

abilistic bound. More precisely, in Theorem 5.13.1 of Section 5.13, we will show that with probability 1, the regret $V_t^{\mathrm{OPT}} - V_t^{RA}$ of the Reservation Algorithm is $O(t^{1/4}\log t)$.

5.12 Implication on Smoothness of Big-Data Scaling

In this section, we show how big-data scaling leads to a smoothness property for the optimal dual variables to (5.5.2). We will use this property to establish condition (5.7.3). This condition will help us to prove our asymptotic bound on the regret of the Reservation Algorithm.

Let γ be a vector of dual variables for problem (5.5.2). This vector γ must satisfy several conditions for it to be dual-optimal, as stated in the following lemma.

Lemma 5.12.1. *If γ^* is an optimal dual solution to (5.5.2), we must have*

$$\gamma^* \in \mathcal{G} \equiv \{\gamma \in \mathbb{R}^m : 0 \leq \gamma_j \leq r_j - r_1,\ \forall j = 1, 2, ..., m\}.$$

Proof. First, the assumption $c_1 = \infty$ implies that $\gamma_1 = 0$.

Second, the assumption $p_{ij} \leq p_{i1}, \forall j = 2, ..., m$, implies that we always have $\gamma_j \leq r_j - r_1$. This is because, according to (5.5.3), as long as γ_j is greater than $r_j - r_1$, no user will ever be assigned to message j because it is always better to assign users to message 1:

$$\gamma_j \geq r_j - r_1 \Longrightarrow r_j - \gamma_j \leq r_1 \Longrightarrow p_{ij}(r_j - \gamma_j) \leq p_{i1}r_1, \quad \forall i = 1, 2, ..., n.$$

In sum, the set of valid vectors of optimal dual variables γ is included in the region $\mathcal{G}$.

□

Now for each problem of size t, let $c^t(\gamma)$ be a vector of capacity values for which γ is a vector of optimal dual variables to the problem (5.5.2) (with c being replaced by $c^t(\gamma)$). It is easy to check that for any $\gamma \in \mathcal{G}$, one such vector $c^t(\gamma)$ can always be found by allocating users to messages $j = 2, 3, ..., m$ according to the rule (5.5.3), i.e.,

$$\begin{aligned} c_j^t(\gamma) &= \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[j = \arg\max_k (r_k - \gamma_k) p_{ik}^t] \\ &= \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[(r_j - \gamma_j) p_{ij}^t - (r_k - \gamma_k) p_{ik}^t \geq 0 \ \forall \ k = 1, \ldots, m], \ \forall j = 2, 3, ..., m. \end{aligned}$$

Note that we have ignored the events that $(r_j - \gamma_j)p_{ij}^t - (r_k - \gamma_k)p_{ik}^t = 0$, i.e., a user can be assigned to different messages without affecting dual feasibility, as these events occur with probability 0 for a given γ.

Furthermore, we should set $c_1^t(\gamma) = \infty$ according to Assumption 5.3.1. Constructed in this way, $c^t(\gamma)$ is a random vector, as it is a function of $p_1^t, \ldots, p_n^t$.

As a main result in this section, we will prove that with probability 1, for all sufficiently large t's, the total expected number of clickthroughs, conditional on the random pool of users, moved from one message to another is at least 1 when γ changes to β or vice versa, whenever $\gamma, \beta \in \mathcal{G}$ with $\|\gamma - \beta\|_\infty \geq t^{-3/4}$.

Theorem 5.12.2. *There exists a finite positive random integer t_0 such that, with probability*

1, *for all* $t > t_0$ *and for any vectors of dual variables* $\gamma, \beta \in \mathcal{G}$ *with* $\|\gamma - \beta\|_\infty \geq t^{-3/4}$,

$$\sum_{j=2}^{m} |c_j^t(\gamma) - c_j^t(\beta)| > 1.$$

The proof of Theorem 5.12.2 relies on a geometric analysis of the space $\mathcal{P}$ of user profiles. Let us define in $\mathcal{P}$ the sub-polytope

$$A_j(\gamma) = \{p \in \mathcal{P} \mid (r_j - \gamma_j)p_j - (r_k - \gamma_k)p_k \geq 0 \ \forall k = 1, \ldots, m\}.$$

Then for $j = 2, 3, ..., m$, $c_j^t(\gamma)$ can be expressed equivalently as

$$c_j^t(\gamma) \quad = \quad \sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma)].$$

In other words, the capacity $c_j^t(\gamma)$ of message j is the sum of clickthrough probabilities contributed by users whose profiles fall inside the polytope $A_j(\gamma)$.

Consider two different vectors of dual variables $\gamma, \beta \in \mathcal{G}$. If we change the dual variables from γ to β, the users i who were assigned to message j under γ, but who are now moved to other messages under β, must satisfy

$$p_i^t \in A_j(\gamma) \setminus A_j(\beta).$$

For problem t, the total expected number of clickthroughs, conditional on the random pool

of users, moved away from message j is

$$\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]. \tag{5.12.1}$$

It is easy to check that the expectation (over all randomly drawn user profiles) of (5.12.1) is

$$\mathbf{E}\left[\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]\right] = n\mathbf{E}\left[p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)]\right] = n \int_{A_j(\gamma)\setminus A_j(\beta)} x_j \vec{dx}.$$

The following lemma bounds the expected number of clickthroughs moved from one message to another when γ changes to β.

Lemma 5.12.3. *Given any two different vectors of dual variables $\gamma, \beta \in \mathcal{G}$ such that $\beta_l < \gamma_l$ for some index l, we must have*

$$\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 \vec{dx} \geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j}.$$

Proof. We must have $l \neq 1$ because $\beta_1 = \gamma_1 = 0$.

Define a vector $\theta \in \mathcal{G}$ to be $\theta_j = \gamma_j$, $\forall j \neq l$, and $\theta_l = \beta_l$. Next, we will prove the lemma by showing

$$\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 \vec{dx} \geq \int_{A_1(\gamma)\setminus A_1(\theta)} x_1 \vec{dx} \geq \frac{1}{m+1}\left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l}\right) \prod_{j=2, j\neq l}^{m} \frac{r_1}{r_j - \gamma_j}.$$

Any $x \in A_1(\theta)$ must satisfy

$$x_1 r_1 \geq x_j(r_j - \theta_j),\ \forall j = 2, 3, ..., m$$

$$\implies x_1 r_1 \geq x_j(r_j - \gamma_j),\ \forall j = 2, 3, ..., m,$$

which implies that

$$A_1(\theta) \subseteq A_1(\gamma). \tag{5.12.2}$$

Furthermore, since γ and θ only differ by the l-th component, for any $x \in A_1(\gamma) \setminus A_1(\theta)$ we must have

$$x_1 r_1 < x_l(r_l - \theta_l) \quad \implies \quad x_1 r_1 < x_l(r_l - \beta_l),$$

which implies that

$$A_1(\gamma) \setminus A_1(\theta) \subseteq A_1(\gamma) \setminus A_1(\beta). \tag{5.12.3}$$

Then (5.12.2) and (5.12.3) give us

$$\begin{aligned}
\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 \vec{dx} &\geq \int_{A_1(\gamma)\setminus A_1(\theta)} x_1 \vec{dx} \\
&= \int_{A_1(\gamma)} x_1 \vec{dx} - \int_{A_1(\theta)\cap A_1(\gamma)} x_1 \vec{dx} \\
&= \int_{A_1(\gamma)} x_1 \vec{dx} - \int_{A_1(\theta)} x_1 \vec{dx}.
\end{aligned}$$

Recall that

$$A_j(\gamma) = \{p \in \mathcal{P} \mid (r_j - \gamma_j)p_j - (r_k - \gamma_k)p_k \geq 0 \ \forall k = 1, \ldots, m\}$$

and $\gamma_1 = 0$. Therefore, the above integrals can be expressed in closed form as

$$\begin{aligned}
\int_{A_1(\gamma)} x_1 \vec{dx} &= \int_0^1 x_1 \left(\int_0^{\frac{x_1 r_1}{r_2 - \gamma_2}} dx_2 \right) \left(\int_0^{\frac{x_1 r_1}{r_3 - \gamma_3}} dx_3 \right) \cdots \left(\int_0^{\frac{x_1 r_1}{r_m - \gamma_m}} dx_m \right) dx_1 \\
&= \int_0^1 x_1 \prod_{j=2}^{m} \frac{x_1 r_1}{r_j - \gamma_j} dx_1 \\
&= \int_0^1 x_1^m dx_1 \cdot \prod_{j=2}^{m} \frac{r_1}{r_j - \gamma_j} \\
&= \frac{1}{m+1} \prod_{j=2}^{m} \frac{r_1}{r_j - \gamma_j}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\int_{A_1(\gamma) \setminus A_1(\beta)} x_1 \vec{dx} &\geq \frac{1}{m+1} \prod_{j=2}^{m} \frac{r_1}{r_j - \gamma_j} - \frac{1}{m+1} \prod_{j=2}^{m} \frac{r_1}{r_j - \theta_j} \\
&= \frac{1}{m+1} \left(\frac{r_1}{r_l - \gamma_l} - \frac{r_1}{r_l - \beta_l} \right) \prod_{j=2, j \neq l}^{m} \frac{r_1}{r_j - \gamma_j}.
\end{aligned}$$

□

For each message $j = 2, 3, ..., m$, Define a number of increments n_j^t as

$$n_j^t \equiv \lfloor \frac{r_j}{0.5 t^{-3/4}} \rfloor.$$

Define a discretization of the real line in n_j^t increments as follows. Let $g_{j0}^t = 0$, and let $g_{j1}^t, ..., g_{jn_j}^t$ be such that $g_{jk}^t = g_{j,k-1}^t + 0.5t^{-3/4}$ for $k = 1, 2, ..., n_j^t$.

Let e_k denote the unit vector with the k-th element being one. Define regions $S_{jk}^t \subseteq \mathcal{P}, j = 2, 3, ..., m, k = 1, 2, ..., n_j$, in the user profile space as the difference between successive polytopes $A_1(g_{jk-1}^t e_j)$ and $A_1(g_{jk}^t e_j)$.

$$S_{jk}^t \equiv A_1(g_{jk}^t e_j) \setminus A_1(g_{jk-1}^t e_j).$$

The sets S_{jk}^t, which we call *cells*, $j = 1, \ldots, m$, $k = 1, \ldots, n_j^t$, $t = 1, 2, \ldots$ are a way to divide up the space of user profiles into countably many fixed regions.

The following lemma establishes that there is at least one cell S_{lk}^t in the difference of polytopes $A_1(\gamma) \setminus A_1(\beta)$, for any pair of vectors γ and β in $\mathcal{G}$ such that $\gamma_l \geq \beta_l + t^{-3/4}$.

Lemma 5.12.4. *Fix t. For any two dual vectors $\gamma, \beta \in \mathcal{G}$ such that*

$$\gamma_l \geq \beta_l + t^{-3/4}$$

for some $l \in \{2, 3, ..., m\}$, there must exist some $k \in \{1, 2, ..., n_l^t\}$ such that

$$S_{lk}^t \subseteq A_1(\gamma) \setminus A_1(\beta).$$

Proof. Since $\gamma_l \geq \beta_l + t^{-3/4}$ and $n_l^t \leq \frac{r_l}{0.5t^{-3/4}}$, there must exist a k such that $\beta_l \leq g_{l,k-1}^t < g_{lk}^t \leq \gamma_l$.

For any $x \in A_1(g_{lk}^t e_l)$, by definition we have $x_1 r_1 \geq x_l(r_l - g_{lk}^t)$ and $x_1 r_1 \geq x_j r_j$ for all $j \neq l, j = 2, 3, ..., m$, which gives

$$x_1 r_1 \geq x_l(r_l - g_{lk}^t) \geq x_l(r_l - \gamma_l)$$

and

$$x_1 r_1 \geq x_j r_j \geq x_j(r_j - \gamma_j) \ \ \forall j \neq l, j = 2, 3, ..., m.$$

This implies that $x \in A_1(\gamma)$. It follows that $A_1(g_{lk}^t e_l) \subseteq A_1(\gamma)$.

Next, for any $x \in A_1(g_{lk}^t e_l) \cap A_1(\beta)$, we must have $x_1 r_1 \geq x_l(r_l - \beta_l) \geq x_l(r_l - g_{l,k-1}^t)$ and $x_1 r_1 \geq x_j r_j$ for all $j \neq l$, $j = 2, 3, ..., m$. This implies $x \in A_1(g_{l,k-1}^t e_l)$. It follows that $A_1(g_{lk}^t e_l) \cap A_1(\beta) \subseteq A_1(g_{l,k-1}^t e_l)$.

In sum, we conclude that $A_1(g_{lk}^t e_l) \setminus A_1(g_{l,k-1}^t e_l) \subseteq A_1(\gamma) \setminus A_1(\beta)$.

□

Lemma 5.12.5. *Let $\mathcal{E}_{lk}^t$ denote the event that*

$$\frac{r_1}{\max_{j=1,2,...,m} r_j} \cdot \sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in S_{lk}^t] > 1.$$

There exist constants C_1 and t^* such that

$$P(\overline{\mathcal{E}_{lk}^t}) \leq e^{\frac{\max_{j=1,2,...,m} r_j}{r_1}} \cdot \left(1 - \frac{C_1}{t^{3/4}}\right)^{t\bar{n}}$$

for all $l = 2, \ldots, m$, $t \geq t^*$, and $k = 1, \ldots, n_l^t$.

Proof. Recall that

$$S^t_{lk} \equiv A_1(g^t_{lk}e_l) \setminus A_1(g^t_{lk-1}e_l).$$

Let $\gamma = g^t_{lk}e_l$ and $\beta = g^t_{lk-1}e_l$. Note that

$$\begin{aligned}
\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x} &\geq \frac{1}{m+1}\left(\frac{r_1}{r_l-\gamma_l} - \frac{r_1}{r_l-\beta_l}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j} \\
&\geq \frac{1}{m+1}\left(\frac{r_1}{r_l-\gamma_l} - \frac{r_1}{r_l-(\gamma_l-0.5t^{-3/4})}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j} \\
&= \frac{1}{m+1}\left(\frac{r_1}{r_l-\gamma_l} - \frac{r_1}{r_l-\gamma_l+0.5t^{-3/4}}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j} \\
&= \frac{1}{m+1}\left(\frac{0.5r_1t^{-3/4}}{(r_l-\gamma_l)(r_l-\gamma_l+0.5t^{-3/4})}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j} \\
&\geq \frac{1}{m+1}\left(\frac{0.5t^{-3/4}}{r_1+0.5t^{-3/4}}\right)\prod_{j=2,j\neq l}^{m}\frac{r_1}{r_j-\gamma_j} \\
&\geq \frac{1}{m+1}\left(\frac{0.5t^{-3/4}}{r_1+0.5t^{-3/4}}\right) \\
&= \frac{1}{m+1}\left(\frac{1}{2t^{3/4}r_1+1}\right) \\
&\geq Ct^{-3/4}
\end{aligned}$$

for all $t \geq t^*$, for some constant C and t^*. Above, the first inequality follows from Lemma 5.12.3; the second inequality from $\beta_l = \gamma_l - 0.5t^{-3/4}$; the third inequality from $\gamma_l \geq r_l - r_1$; and the fourth inequality from $\gamma_j \geq r_j - r_1$ for all $j > 1$.

Next, using Chernoff's bound, we have for all $t \geq t^*$,

$$P(\frac{r_1}{\max_{j=1,2,\ldots,m} r_j} \cdot \sum_{i=1}^{n} p^t_{i1}\mathbf{1}[p^t_i \in A_1(\gamma) \setminus A_1(\beta)] \leq 1)$$

$$\leq e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(\mathbf{E}[e^{-p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)]}]\right)^n.$$

Further, it is easy to check that for any $x \in [0,1]$, we have $e^{-x} \leq 1-(1-e^{-1})x$. Thus,

$$\begin{aligned}
\mathbf{E}[e^{-p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)]}] \leq & \mathbf{E}[1-(1-e^{-1})p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)]] \\
= & 1-(1-e^{-1})\mathbf{E}[p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)]] \\
= & 1-(1-e^{-1})\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 f(\vec{x}) d\vec{x} \\
\leq & 1-(1-e^{-1})\underline{f}\int_{A_1(\gamma)\setminus A_1(\beta)} x_1 d\vec{x} \\
\leq & 1-(1-e^{-1})\underline{f}Ct^{-3/4}.
\end{aligned}$$

It follows that

$$\begin{aligned}
& P\left(\frac{r_1}{\max_{j=1,2,\ldots,m} r_j} \cdot \sum_{i=1}^n p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma)\setminus A_1(\beta)] \leq 1\right) \\
\leq & e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1-(1-e^{-1})\underline{f}Ct^{-3/4}\right)^n \\
= & e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1-(1-e^{-1})\underline{f}Ct^{-3/4}\right)^{t\bar{n}} \\
= & e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1-\frac{C_1}{t^{3/4}}\right)^{t\bar{n}}
\end{aligned}$$

for $C_1 = (1-e^{-1})\underline{f}C$ and for $t \geq t^*$.

□

Proof of Theorem 5.12.2. We have shown in Theorem 5.9.2 that when γ changes to β, the expected number of clickthroughs (5.12.1) (conditional on the random pool of users) leaving

message j can be bounded using the change in the capacity values as follows:

$$\sum_{i=1}^{n} p_{ij}^t \mathbf{1}[p_i^t \in A_j(\gamma) \setminus A_j(\beta)] \leq \sum_{k=2}^{m} |c_k^t(\gamma) - c_k^t(\beta)| r_k / r_1.$$

Note that in the above bound we do not consider the change made to the capacity of message 1, because message 1 has infinite capacity and thus any finite change made to this capacity value has no impact on the assignment of users. The above bound leads to

$$\sum_{j=2}^{m} |c_j^t(\gamma) - c_j^t(\beta)| \geq \frac{r_1}{\max_{j=1,2,...,m} r_j} \cdot \sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma) \setminus A_1(\beta)]. \tag{5.12.4}$$

Therefore, it suffices to show that with probability one,

$$\frac{r_1}{\max_{j=1,2,...,m} r_j} \cdot \sum_{i=1}^{n} p_{i1}^t \mathbf{1}[p_i^t \in A_1(\gamma) \setminus A_1(\beta)] > 1$$

for all sufficiently large t and all $\gamma, \beta \in \mathcal{G}$ with $\gamma_l \geq \beta_l + t^{-3/4}$ for some index l.

Based on Lemma 5.12.4, to prove the theorem, we just need to prove that there exists a finite random number t_0 such that $\mathcal{E}_{jk}^t$ holds for all $t \geq t_0$, $j = 2, 3, ..., m$, $k = 1, 2, ..., n_j^t$. By the Borel-Cantelli Lemma, it suffices to show that

$$\sum_{t=1}^{\infty} P\left(\overline{\cap_{j=2}^{m} \cap_{k=1}^{n_j^t} \mathcal{E}_{jk}^t}\right) < \infty.$$

Indeed,

$$
\begin{aligned}
P\left(\overline{\cap_{j=2}^{m}\cap_{k=1}^{n_j^t}\mathcal{E}_{jk}^t}\right) &= 1 - P\left(\cap_{j=2}^{m}\cap_{k=1}^{n_j^t}\mathcal{E}_{jk}^t\right) \\
&\leq 1 - \prod_{j=2}^{m}\prod_{k=1}^{n_j^t} P\left(\mathcal{E}_{jk}^t\right) \quad \text{(because the events } \mathcal{E}_{jk}^t\text{'s are positively correlated)} \\
&= 1 - \prod_{j=2}^{m}\prod_{k=1}^{n_j^t}(1 - P(\overline{\mathcal{E}_{jk}^t})) \\
&\leq 1 - \prod_{j=2}^{m}\prod_{k=1}^{n_j^t}\left[1 - e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1 - C_1 t^{-3/4}\right)^{t\bar{n}}\right] \quad \text{(by Lemma 5.12.5)} \\
&\leq 1 - \left[1 - e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1 - C_1 t^{-3/4}\right)^{t\bar{n}}\right]^{C_2 t^{3/4}}
\end{aligned}
$$

for an appropriately defined constant C_2. That the events $\mathcal{E}_{jk}^t$'s are positively correlated can be explained by the fact that $\mathcal{E}_{jk}^t$'s are independent if the corresponding sets S_{jk}^t do not overlap. On the other hand, they contain a common set of profiles if they do overlap.

We can check that when $t \to \infty$,

$$
\left(1 - C_1 t^{-3/4}\right)^{t\bar{n}} = e^{-C_1 \bar{n} t^{1/4} + o(1)}
$$

$$
\begin{aligned}
\Longrightarrow & \left[1 - e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot \left(1 - C_1 t^{-3/4}\right)^{t\bar{n}}\right]^{C_2 t^{3/4}} \\
&= \left[1 - \frac{e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}}{e^{C_1 \bar{n} t^{1/4} + o(1)}}\right]^{C_2 t^{3/4}} \\
&= \exp\left(-C_2 t^{3/4} \cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot e^{-C_1 \bar{n} t^{1/4}} + o(e^{-C_1 \bar{n} t^{1/4}})\right) \\
&= 1 - C_2 t^{3/4} \cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}} \cdot e^{-C_1 \bar{n} t^{1/4}} + o(e^{-C_1 \bar{n} t^{1/4}})
\end{aligned}
$$

$$\begin{aligned}\Longrightarrow & 1-\left[1-e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot\left(1-C_1 t^{-3/4}\right)^{t\bar{n}}\right]^{C_2 t^{3/4}} \\ = & C_2 t^{3/4}\cdot e^{\frac{\max_{j=1,2,\ldots,m} r_j}{r_1}}\cdot e^{-C_1\bar{n}t^{1/4}}+o(e^{-C_1\bar{n}t^{1/4}}).\end{aligned}$$

This proves that

$$\sum_{t=1}^{\infty} P\left(\overline{\cap_{j=2}^{m}\cap_{k=1}^{n_j^t}\mathcal{E}_{jk}^t}\right)<\infty.$$

□

5.13 Asymptotic Regret of the Reservation Algorithm

In this section we will bound the regret $V_t^{LP}-V_t^{RA}$ in a probabilistic sense, in the big-data scaling regime.

Recall that we choose the reservation level for a system of size t as (see (5.6.6))

$$\Delta(t)\equiv C\cdot\sum_{j=2}^{m}\sqrt{c_j}\cdot\log\sum_{j=2}^{m}c_j=C\cdot\sum_{j=2}^{m}\sqrt{t\bar{c}_j}\cdot\log\sum_{j=2}^{m}t\bar{c}_j. \tag{5.13.1}$$

Here, $C>0$ is fixed for all t. Recall that m is constant for all system size t, but $n=t\cdot\bar{n}$, and $c_j=t\cdot\bar{c}_j$, $\forall j=1,2,...,m$. For the instance of size t, let V_t^{LP} denote the optimal objective value of the linear program (5.5.1), and let $\gamma^t(c)$ be a vector of its optimal dual variables when the capacity vector is c; let V_t^{RA} denote the expected total reward of the Reservation Algorithm for size t (conditional on the random pool of users); let δ^t be the noise defined as in (5.7.1), and let $\mathcal{O}_t$ denote the event that $\sum_{j=2}^{m}|\delta_j^t|\frac{r_j}{r_1}\leq\Delta(t)$.

Theorem 5.13.1. *Suppose that for each problem t, the reservation level used in RA is $\Delta(t)$. Then the regret of RA is $O(t^{1/4}\log t)$ with probability 1.*

Proof. First, we claim that with probability 1, there exists a finite random integer t_0 such that for all $t > t_0$, condition (5.7.3) holds when $\bar{\gamma}$ is set to be $\bar{\gamma}_t \equiv t^{-3/4}$ in the problem of size t.

Let t_0 be the random variable in Theorem 5.12.2. Suppose (5.7.3) does not hold for some $t > t_0$, i.e., with positive probability, for some $t > t_0$, we can find c, α, j, and k, such that

$$|\gamma_j^t(c) - \gamma_j^t(c + \alpha e_k)| > t^{-3/4}.$$

Then Theorem 5.12.2 implies that $\|c - (c + \alpha e_k)\|_1 > 1 \Longrightarrow \|\alpha e_k\|_1 > 1$, which is a contradiction.

Therefore, (5.7.3) holds for $t > t_0$ when $\bar{\gamma}_t = t^{-3/4}$. It follows that Theorem 5.8.1 holds with probability 1 for $t > t_0$. Let $U_t \equiv \sigma(p_1^t, p_2^t, ..., p_n^t)$ denote the random user pool for problem t. Then the regret (5.7.4) of the algorithm becomes, for $t > t_0$,

$$V_t^{LP} - V_t^{RA} \leq \sum_{j=1}^{m} r_j \left[\sqrt{m\Delta(t)} + 3m^2 + nP(\bar{\mathcal{O}}_t|U_t) + \sqrt{nP(\bar{\mathcal{O}}_t|U_t)}\right] + m^2(n + \sqrt{n})t^{-3/4}.$$

Since

$$\Delta(t) = C \cdot \sum_{j=2}^{m} \sqrt{t\bar{c}_j} \cdot \log \sum_{j=2}^{m} t\bar{c}_j = O(\sqrt{t}\log t),$$

we have

$$\sum_{j=1}^{m} r_j \sqrt{m\Delta(t)} = O(t^{1/4} \log t).$$

Furthermore,

$$m^2(n + \sqrt{n})t^{-3/4} = O(t^{1/4}).$$

Define

$$Var(\delta_j^t | U_t) \equiv \mathbf{E}[(\delta_j^t)^2 | U_t] - (\mathbf{E}[\delta_j^t | U_t])^2$$

as the variance of δ_j^t given U_t as input. According to the central limit theorem, when the standard deviation of δ_j^t becomes large, $\delta_j^t / \sqrt{Var(\delta_j^t | U_t)}$ approaches a standard normal distribution. Furthermore, since

$$Var(\delta_j^t | U_t) = \sum_{i=1}^{n} p_{ij} x_{ij}^{(1)} (1 - p_{ij} x_{ij}^{(1)}) \leq \sum_{i=1}^{n} p_{ij} x_{ij}^{(1)} \leq c_j = \bar{c}_j t,$$

we must have, according to the central limit theorem,

$$P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} > \bar{C} \log t \middle| U_t \right) = O(P(|Z_t| > \bar{C} \log t))$$

for any constant $\bar{C} > 0$ and for a sequence $Z_1, Z_2, Z_3, ...$ of i.i.d. standard normal random variables. It is easy to check that

$$\begin{aligned} P(|Z_t| > \bar{C} \log t) &= 2 \cdot \frac{1}{\sqrt{2\pi}} \int_{\bar{C} \log t}^{\infty} e^{-x^2/2} dx \\ &< \frac{2}{\sqrt{2\pi}} \int_{\bar{C} \log t}^{\infty} \frac{x}{\bar{C} \log t} e^{-x^2/2} dx \end{aligned}$$

$$= \frac{2}{\sqrt{2\pi}} \frac{1}{\bar{C}\log t} e^{-(\bar{C}\log t)^2/2}$$

$$= o(1/t),$$

from which we can deduce that

$$P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} > \bar{C}\log t \middle| U_t\right) = o(1/t)$$

$$\Longrightarrow P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} \cdot \frac{r_j}{r_1} > \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) = o(1/t)$$

$$\Longrightarrow P\left(\sum_{j=2}^{m} \frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} \frac{r_j}{r_1} > \sum_{j=2}^{m} \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) \leq \sum_{j=2}^{m} P\left(\frac{|\delta_j^t|}{\sqrt{\bar{c}_j t}} \cdot \frac{r_j}{r_1} > \frac{r_j}{r_1}\bar{C}\log t \middle| U_t\right) = o(1/t)$$

$$\Longrightarrow P\left(\sum_{j=2}^{m} |\delta_j^t| \frac{r_j}{r_1} > \left(\max_{j=2,\ldots,m} \sqrt{\bar{c}_j}\right) \sum_{j=2}^{m} \frac{r_j}{r_1}\bar{C}\sqrt{t}\log t \middle| U_t\right) = o(1/t).$$

Then by choosing an appropriate value of $\bar{C}$, we can easily obtain

$$\left(\max_{j=2,\ldots,m} \sqrt{\bar{c}_j}\right) \sum_{j=2}^{m} \frac{r_j}{r_1}\bar{C}\sqrt{t}\log t = \Delta(t),$$

and hence,

$$P(\bar{\mathcal{O}}_t|U_t) = P(\sum_{j=2}^{m} |\delta_j^t| \frac{r_j}{r_1} > \Delta(t)|U_t) = o(1/t).$$

Thus, $nP(\bar{\mathcal{O}}_t|U_t) = t\bar{n}P(\bar{\mathcal{O}}_t|U_t) = o(1)$.

In sum, $V_t^{LP} - V_t^{RA} = O(t^{1/4}\log t)$ with probability 1. This proves the theorem because V_t^{LP} is an upper bound on V_t^{OPT}. □

5.14 Regret of the Static Algorithm

In this section, we analyze the regret of the Static Algorithm and compare it to that of the Reservation Algorithm.

Under the Static Algorithm, the random number of clickthroughs that message $j \in \{1, 2, ..., m\}$ receives has mean

$$b_j \equiv \sum_{i=1}^{n} s_{ij}^* p_{ij} \leq c_j \tag{5.14.1}$$

and standard deviation

$$\sigma_j \equiv \sqrt{\sum_{i=1}^{n} s_{ij}^* p_{ij}(1 - s_{ij}^* p_{ij})} \leq \sqrt{b_j} \leq \sqrt{c_j}. \tag{5.14.2}$$

Since σ_j is the size of noise in the random number of clickthroughs that message j receives under this algorithm, we can expect that the regret of this algorithm grows as $O(\sigma_j) = O(\sqrt{c_j}) = O(\sqrt{t})$. The following theorem shows that $O(\sqrt{t})$ is an upper bound on the regret of the Static Algorithm. Later, we will further show that $O(\sqrt{t})$ is a tight bound.

Theorem 5.14.1. *The regret of the Static Algorithm is $O(\sqrt{t})$ almost surely.*

Proof. We first focus on the analysis of a single problem of size t and suppress t in notation. Fix the set of user profiles. Let V^{Static} be the expected total reward of the static algorithm. We have

$$V^{OPT} - V^{Static} \leq V^{LP} - V^{Static}$$

$$
\begin{aligned}
&= \sum_{j=1}^{m} r_j [b_j - \mathbf{E}[\min(c_j, \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j [b_j - \mathbf{E}[\min(c_j, \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j \mathbf{E}[\max(b_j - c_j, b_j - \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&\leq \sum_{j=2}^{m} r_j \mathbf{E}[\max(0, b_j - \sum_{i=1}^{n} I_{ij}^{Static})]] \\
&= \sum_{j=2}^{m} r_j \mathbf{E}[|b_j - \sum_{i=1}^{n} I_{ij}^{Static}|] \\
&\leq \sum_{j=2}^{m} r_j \sigma_j \\
&\leq \sum_{j=2}^{m} r_j \sqrt{c_j}.
\end{aligned}
$$

In the asymptotic regime, since $c_j = t \cdot \bar{c}_j$, we have almost surely,

$$V_t^{OPT} - V_t^{Static} \leq \sum_{j=2}^{m} r_j \sqrt{c_j} = O(\sqrt{t}).$$

□

Finally, we prove a lower bound on the performance of the Static Algorithm with respect to V_t^{LP}. This result provides a worst-case lower bound on $V_t^{RA} - V_t^{Static}$.

Theorem 5.14.2. *There exist problem instances in which almost surely,*

$$V_t^{RA} - V_t^{Static} = \Omega(\sqrt{t}).$$

Proof. Again let U_t denote the random user pool for problem t. Fix some message $j \in \{2, 3, ..., m\}$. Conditional on U_t, let K_t be the number of clickthroughs that message j receives under the Static Algorithm.

Assume that $\bar{p} = 0.5$ in (5.11.1), i.e., all clickthrough probabilities are less than 0.5. Also assume that the capacity of each message $j \geq 2$ is fully allocated to users by the linear program (5.5.1) almost surely for all t. This condition can be ensured by having small capacities for all messages $j \geq 2$. Then we have for all large t, $\mathbf{E}[K_t|U_t] = \bar{c}_j t$. It is easy to see that we must have, conditional upon the random user pools,

$$V_t^{LP} - V_t^{Static} \geq r_j \mathbf{E}[\max(0, \bar{c}_j t - K_t)|U_t] = r_j \mathbf{E}[\max(0, \mathbf{E}[K_t] - K_t)|U_t]$$

with probability 1 for each t. Furthermore, let $Var(K_t|U_t) = \mathbf{E}[K_t^2|U_t] - (\mathbf{E}[K_t|U_t])^2$ denote the variance of K_t given input U_t. We must have

$$\sqrt{Var(K_t|U_t)} = \sqrt{\sum_{i=1}^{n} s_{ij}^* p_{ij}(1 - s_{ij}^* p_{ij})} \geq \sqrt{\sum_{i=1}^{n} s_{ij}^* p_{ij} 0.5} = \sqrt{\mathbf{E}[K_t|U_t]0.5} = \sqrt{\bar{c}_j t 0.5} = \Omega(\sqrt{t}).$$

When t is large, $K_t/\sqrt{Var(K_t|U_t)}$ approaches a normal random variable with standard deviation 1 almost surely. Thus, we must have almost surely

$$\mathbf{E}[\max(0, \mathbf{E}[K_t] - K_t)|U_t] = \Omega(\sqrt{Var(K_t|U_t)}) = \Omega(\sqrt{t})$$

$$\Longrightarrow V_t^{LP} - V_t^{Static} = \Omega(\sqrt{t}).$$

In the proof of Theorem 5.13.1, we have proved $V_t^{LP} - V_t^{RA} = o(\sqrt{t})$. This leads to $V_t^{RA} - V_t^{Static} = \Omega(\sqrt{t})$.

□

5.15 Numerical Studies

We test the performance of the Reservation Algorithm and the Static Algorithm by simulating their total reward values using real data of clickthrough probabilities.

We use three different data-sets. Each data set contains a set of messages that were sent to several hundred million users on a certain day in March 2016, and the estimated clickthrough probabilities between all the users and messages for that day.

We implement the algorithms on a distributed computing system using the MapReduce framework. The linear program used in the algorithms is solved by smoothing the dual problem (5.5.2) and then using a descent method. We refer the reader to Zhong et al. (2015) for a similar distributed algorithm for solving an LP. Problem (5.6.1) in the second step of the Reservation Algorithm is solved by using a binary search to find an appropriate ratio ω such that $p_{ik}/p_{ij} \geq \omega$ for every $i \in R^{jk}$.

The expected total reward for both algorithms are simulated based on clickthrough probabilities provided by Alibaba. We vary the reservation level Δ used in the Reservation Algorithm over a range. For each test case, we report the relative regret of the two algorithms as

$$\alpha \equiv \frac{V^{LP} - V^{RA}}{V^{LP} - V^{Static}}.$$

Note that since

$$\frac{V^{LP} - V^{RA}}{V^{LP} - V^{Static}} \geq \frac{V^{\mathrm{OPT}} - V^{RA}}{V^{\mathrm{OPT}} - V^{Static}},$$

the ratios we report overestimate the actual regret of the Reservation Algorithm compared to that of the Static Algorithm. That is, if compared against V^{OPT} instead of V^{LP}, the actual performance of the Reservation Algorithm will be better than the results reported. Figure 5.4 summarizes our test results. The actual performance of the Reservation Algorithm highly depends on the types of products sent out on each day. Among the 3 test cases, our Reservation Algorithm improves the total regret of the Static Algorithm by at least 10% and as much as 50%. Thus, implementing the Reservation Algorithm is very promising in improving the overall benefit earned from this mobile-based recommendation system.

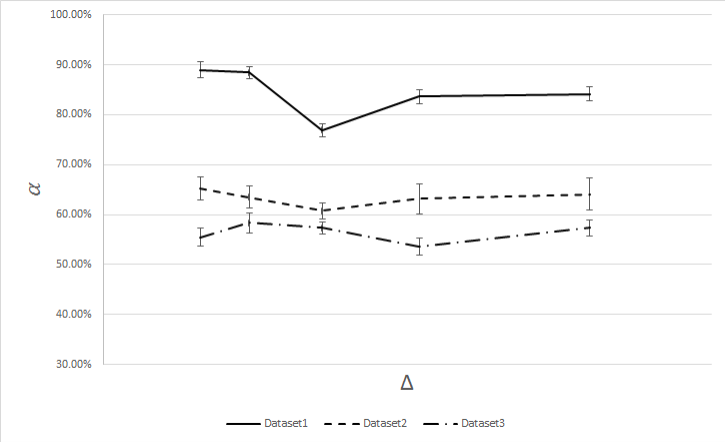


Figure 5.4: Regret of the Reservation Algorithm relative to the Static Algorithm under different values of Δ.

5.16 Conclusions

We study a budget allocation problem faced by Alibaba Group. Unlike traditional e-commerce problems where ads can only be displayed when customers open up websites, the system in this problem can proactively send out ads or campaign information to users of mobile devices.

This resource allocation problem involves hundreds of millions of users. As a result, useful allocation algorithms must be distributed and written in the MapReduce framework. The allocation algorithms we propose rely on existing methods for solving large-scale linear programming problems. But more importantly, our algorithms are novel examples of implementing Operations Research techniques, such as resource reservation and re-solving heuristics, on a cloud computing platform.

In our analysis, we first prove that a conventional LP-based algorithm has asymptotic regret $O(\sqrt{t})$, where t is the size of the system. We improve upon this regret by proposing a new algorithm, called the Reservation Algorithm, that sends out ads and campaign information in two cycles over a day, and updates decisions for the second cycle based on the information observed in the first cycle.

We test the numerical performance of the algorithms by simulating them on terabytes of production data provided by Alibaba. We find that our Reservation algorithm improves the regret of the standard LP-based algorithm by 10%-50%.

Bibliography

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

Alaei, Saeed, Mohammad T Hajiaghayi, Vahid Liaghat. 2012. Online prophet-inequality matching with applications to ad allocation. *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, 18–35.

Albers, Susanne, Hiroshi Fujiwara. 2007. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)* **3**(4) 49.

Ayvaz, Nur, Woonghee Tim Huh. 2010. Allocation of hospital capacity to multiple types of patients. *Journal of Revenue & Pricing Management* **9**(5) 386–398.

Babaioff, Moshe, Nicole Immorlica, David Kempe, Robert Kleinberg. 2008. Online auctions and generalized secretary problems. *SIGecom Exch.* **7**(2) 7:1–7:11.

Bahmani, Bahman, Michael Kapralov. 2010. Improved bounds for online stochastic matching. *Proceedings of the 18th Annual European Conference on Algorithms: Part I*. ESA'10, Springer-Verlag, Berlin, Heidelberg, 170–181.

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

Bansal, Nikhil, DavidP. Bunde, Ho-Leung Chan, Kirk Pruhs. 2011. Average rate speed scaling. *Algorithmica* **60**(4) 877–889.

Bansal, Nikhil, Ho-Leung Chan, Kirk Pruhs. 2009a. Speed scaling with an arbitrary power function. *Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms*. Society for Industrial and Applied Mathematics, 693–701.

Bansal, Nikhil, Ho-Leung Chan, Kirk Pruhs, Dmitriy Katz. 2009b. Improved bounds for speed scaling in devices obeying the cube-root rule. *Automata, Languages and Programming*. Springer, 144–155.

Bansal, Nikhil, Tracy Kimbrel, Kirk Pruhs. 2007a. Speed scaling to manage energy and temperature. *J. ACM* **54**(1) 3:1–3:39.

Bansal, Nikhil, Kirk Pruhs, Cliff Stein. 2007b. Speed scaling for weighted flow time. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 805–813.

Bertsimas, Dimitris, John Tsitsiklis. 1997. *Introduction to Linear Optimization*. 1st ed. Athena Scientific.

Bienkowski, M. 2008. Ski rental problem with dynamic pricing. *Institute Of Computer Science, University Of Wroclaw, Report* **3**(08).

Blackburn, Joseph D. 1972. Optimal control of a single-server queue with balking and reneging. *Management Science* **19**(3) 297–313.

Borodin, Allan, Ran El-Yaniv. 2005. *Online computation and competitive analysis*. Cambridge University Press.

Buchbinder, Niv, Kamal Jain, Joseph Seffi Naor. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. *Algorithms–ESA 2007*. Springer, 253–264.

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.

BusinessWire. 2016. Alibaba group announces june quarter 2016 results. `http://www.businesswire.com/news/home/20160811005428/en`.

Cardoen, Brecht, Erik Demeulemeester, Jeroen Beliën. 2010. Operating room planning and scheduling: A literature review. *European Journal of Operational Research* **201**(3) 921–932.

Carr, Scott, Izak Duenyas. 2000. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Operations Research* **48**(5) 709–720.

Chakrabarty, Deeparnab, Yunhong Zhou, Rajan Lukose. 2013. Online knapsack problems. *Workshop on internet and network economics (WINE)*.

Chan, Ho-Leung, Wun-Tat Chan, Tak-Wah Lam, Lap-Kei Lee, Kin-Sum Mak, Prudence WH Wong. 2007. Energy efficient online deadline scheduling. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 795–804.

Chen, B., Chris N. Potts, Gerhard J. Woeginger. 1998. *A Review of Machine Scheduling: Complexity, Algorithms and Approximability*. Kluwer Academic Publishers.

Chen, Kai, Sheldon M Ross. 2014. An adaptive stochastic knapsack problem. *European Journal of Operational Research* **239**(3) 625–635.

Chen, Shaoxiang, Guillermo Gallego, Michael Z.F. Li, Bing Lin. 2010. Optimal seat allocation for two-flight problems with a flexible demand segment. *European Journal of Operational Research* **201**(3) 897 – 908.

Ciocan, Dragos Florin, Vivek Farias. 2012. Model predictive control for dynamic resource allocation. *Mathematics of Operations Research* **37**(3) 501–525.

Ciocan, Dragos Florin, Vivek Farias. 2014. Fast demand learning for ad-display allocation.

Dellaert, N.P., M.T. Melo. 1998. Make-to-order policies for a stochastic lot-sizing problem using overtime. *International Journal of Production Economics* **5657**(0) 79 – 97. Production Economics: The Link Between Technology And Management.

Denton, Brian T., Andrew J. Miller, Hari J. Balasubramanian, Todd R. Huschka. 2010. Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research* **58**(4-part-1) 802–816.

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

Devanur, Nikhil R, Kamal Jain, Balasubramanian Sivan, Christopher A Wilkens. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *Proceedings of the 12th ACM conference on Electronic commerce*. ACM, 29–38.

Devanur, Nikhil R, Balasubramanian Sivan, Yossi Azar. 2012. Asymptotically optimal algorithm for stochastic adwords. *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, 388–404.

Elmachtoub, Adam, Yehua Wei. 2013. Retailing with opaque products. Unpublished manuscript.

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.

Emarketer.com. 2016. E-commerce turns into m-commerce in china. `http://www.emarketer.com/Article/Ecommerce-Turns-Mcommerce-China/1013736`.

Fay, Scott, Jinhong Xie. 2008. Probabilistic goods: A creative way of selling products and services. *Marketing Science* **27**(4) 674–690.

Fay, Scott, Jinhong Xie. 2015. Timing of product allocation: Using probabilistic selling to enhance inventory management. *Management Science* **61**(2) 474–484.

Federgruen, Awi, Kut C So. 1990. Optimal maintenance policies for single-server queueing systems subject to breakdowns. *Operations Research* **38**(2) 330–343.

Feldman, Jacob, Nan Liu, Huseyin Topaloglu, Serhan Ziya. 2014. Appointment scheduling under patient preference and no-show behavior. *Operations Research* **62**(4) 794–811.

Feldman, Jon, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, Cliff Stein. 2010. Online stochastic packing applied to display ad allocation. *Algorithms–ESA 2010*. Springer, 182–194.

Feldman, Jon, Aranyak Mehta, Vahab Mirrokni, S Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*. IEEE, 117–126.

Fujiwara, Hiroshi, Takuma Kitano, Toshihiro Fujito. 2011. On the best possible competitive ratio for multislope ski rental. *Algorithms and Computation*. Springer, 544–553.

Gallego, G., G. Van Ryzin. 1997. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research* **45**(1) 24–41.

Gallego, Guillermo, Garud Iyengar, R Phillips, Abha Dubey. 2004. Managing flexible products on a network. Unpublished.

Gallego, Guillermo, Robert Phillips. 2004. Revenue management of flexible products. *Manufacturing & Service Operations Management* **6**(4) 321–337.

Gallego, Guillermo, Richard Ratliff, Sergey Shebalov. 2015. A general attraction model and sales-based linear program for network revenue management under customer choice. *Operations Research* **63**(1) 212–232.

Gerchak, Yigal, Diwakar Gupta, Mordechai Henig. 1996. Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science* **42**(3) pp. 321–334.

Gocgun, Yasin, Archis Ghate. 2012. Lagrangian relaxation and constraint generation for allocation and advanced scheduling. *Computers & Operations Research* **39**(10) 2323–2336.

Goel, Gagan, Aranyak Mehta. 2008. Online budgeted matching in random input models with applications to adwords. *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 982–991.

Gönsch, Jochen, Sebastian Koch, Claudius Steinhardt. 2014. Revenue management with flexible products: The value of flexibility and its incorporation into dlp-based approaches. *International Journal of Production Economics* **153**(0) 280 – 294.

Greenhouse, Steven. 2012. A part-time life, as hours shrink and shift. *The New York Times* .

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, Anupam, Amit Kumar, Tim Roughgarden, et al. 2007. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *Journal of the ACM (JACM)* **54**(3) 11.

Gupta, D. 2007. Surgical suites' operations management. *Production and Operations Management* **16**(6) 689–700.

Gupta, Diwakar, Lei Wang. 2008. Revenue management for a primary-care clinic in the presence of patient choice. *Operations Research* **56**(3) 576–592.

Haeupler, Bernhard, Vahab S Mirrokni, Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. *Internet and Network Economics*. Springer, 170–181.

Huh, Woonghee Tim, Nan Liu, Van-Anh Truong. 2013. Multiresource allocation scheduling in dynamic environments. *Manufacturing & Service Operations Management* **15**(2) 280–291.

Husain, Iltifat. 2014. Epic updates mychart app to sync with apple health, huge for mobile health. `http://www.imedicalapps.com/2014/10/epic-updates-mychart-app-apple-health/`.

Jaillet, Patrick, Xin Lu. 2013. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* **39**(3) 624–646.

Jasin, Stefanus. 2015. Performance of an lp-based control for revenue management with unknown demand parameters. *Operations Research* **63**(4) 909–915.

Jasin, Stefanus, Sunil Kumar. 2012. A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research* **37**(2) 313–345.

Karande, Chinmay, Aranyak Mehta, Pushkar Tripathi. 2011. Online bipartite matching with unknown distributions. *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 587–596.

Karlin, Anna R., Mark S. Manasse, Lyle A. McGeoch, Susan Owicki. 1990. Competitive randomized algorithms for non-uniform problems. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '90, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 301–309.

Karlin, Anna R, Mark S Manasse, Larry Rudolph, Daniel D Sleator. 1988. Competitive snoopy caching. *Algorithmica* **3**(1-4) 79–119.

Karp, R. M., U. V. Vazirani, V. V. Vazirani. 1990. An optimal algorithm for on-line bipartite matching. *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*. STOC '90, 352–358.

Keskinocak, Pinar, R. Ravi, Sridhar Tayur. 2001. Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science* **47**(2) 264–279.

Kesselheim, Thomas, Klaus Radke, Andreas Tönnis, Berthold Vöcking. 2013. *Algorithms – ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, chap. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. Springer Berlin Heidelberg, Berlin, Heidelberg, 589–600.

Kim, Song-Hee, Ward Whitt. 2014. Are call center and hospital arrivals well modeled by nonhomogeneous poisson processes? *Manufacturing & Service Operations Management* **16**(3) 464–480.

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.

Lautenbacher, Conrad J., Shaler Stidham, Jr. 1999. The underlying markov decision process in the single-leg airline yield-management problem. *Transportation Science* **33**(2) 136–146.

Lee, Misuk, Alexandre Khelifa, Laurie A. Garrow, Michel Bierlaire, David Post. 2012. An analysis of destination choice for opaque airline products using multidimensional binary logit models. *Transportation Research Part A: Policy and Practice* **46**(10) 1641 – 1653.

Lee, Tak C., Marvin Hersh. 1993. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science* **27**(3) 252–265.

Levi, R., R. O. Roundy, D. B. Shmoys, V. A. Truong. 2008a. Approximation algorithms for capacitated stochastic inventory control models. *Operations Research* **56**(5) 1184–1199.

Levi, Retsef, Ganesh Janakiraman, Mahesh Nagarajan. 2008b. A 2-approximation algorithm for stochastic inventory control models with lost sales. *Mathematics of Operations Research* **33**(2) 351–374.

Levi, Retsef, Martin Pál, Robin Roundy, David B. Shmoys. 2007. Approximation algorithms for stochastic inventory control models. *Mathematics of Operations Research* **32**(2) 284–302.

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.

Littlewood, Ken. 1972. Forecasting and control of passenger bookings. *Proceedings of the 12th AGIFORS Symposium, October*.

Liu, Q., G. van Ryzin. 2008. On the choice-based linear programming model for network revenue management. *Manufacturing & Service Operations Management* **10**(2) 288.

Lotker, Zvi, Boaz Patt-Shamir, Dror Rawitz. 2012. Rent, lease, or buy: Randomized algorithms for multislope ski rental. *SIAM Journal on Discrete Mathematics* **26**(2) 718–736.

Lueker, George S. 1998. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms* **29**(2) 277–305.

Mahdian, Mohammad, Qiqi Yan. 2011. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 597–606.

Manshadi, Vahideh H, Shayan Oveis Gharan, Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research* **37**(4) 559–573.

Marchetti-Spaccamela, Alberto, Carlo Vercellis. 1995. Stochastic on-line knapsack problems. *Mathematical Programming* **68**(1-3) 73–104.

Martin, G. E., Joyce L. Grahn, Lyn D. Pankoff, Laurence A. Madeo. 1992. A mechanism for reducing small-business customer waiting-line dissatisfaction. *Managerial and Decision Economics* **13**(4) 353–361.

May, Jerrold H, William E Spangler, David P Strum, Luis G Vargas. 2011. The surgical scheduling problem: Current research and future opportunities. *Production and Operations Management* **20**(3) 392–405.

Megow, Nicole, Marc Uetz, Tjark Vredeveld. 2006. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research* **31**(3) 513–525.

Mehta, Aranyak. 2012. Online matching and ad allocation. *Theoretical Computer Science* **8**(4) 265–368.

Min, Daiki, Yuehwern Yih. 2010. An elective surgery scheduling problem considering patient priority. *Computers & Operations Research* **37**(6) 1091 – 1099.

Mirrokni, Vahab S, Shayan Oveis Gharan, Morteza Zadimoghaddam. 2012. Simultaneous approximations for adversarial and stochastic online budgeted allocation. *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1690–1701.

Molinaro, Marco, R. Ravi. 2014. The geometry of online packing linear programs. *Mathematics of Operations Research* **39**(1) 46–59.

Noga, John, Steven S. Seiden. 2001. An optimal online algorithm for scheduling two machines with release times. *Theoretical Computer Science* **268**(1) 133 – 143. On-line Algorithms '98.

Özdamar, Linet, Tülin Yazgaç. 1997. Capacity driven due date settings in make-to-order production systems. *International Journal of Production Economics* **49**(1) 29 – 44.

Papastavrou, Jason D, Srikanth Rajagopalan, Anton J Kleywegt. 1996. The dynamic and stochastic knapsack problem with deadlines. *Management Science* **42**(12) 1706–1718.

Patrick, Jonathan, Martin L. Puterman, Maurice Queyranne. 2008. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations Research* **56**(6) 1507–1525.

Petrick, Anita, Jochen Gnsch, Claudius Steinhardt, Robert Klein. 2010. Dynamic control mechanisms for revenue management with flexible products. *Computers & Operations Research* **37**(11) 2027 – 2039. Metaheuristics for Logistics and Vehicle Routing.

Publication, TechnologyAdvice Research. 2015. What digital services to patients want the most. `http://research.technologyadvice.com/trends-in-patient-engagement`.

Reiman, Martin I., Qiong Wang. 2008. An asymptotically optimal policy for a quantity-based network revenue management problem. *Mathematics of Operations Research* **33**(2) 257–282.

Rubino, Melanie, Bar Ata. 2009. Dynamic control of a make-to-order, parallel-server system with cancellations. *Operations Research* **57**(1) 94–108.

Society of Thoracic Surgeons. 2016. Explanation of sts chsd mortality risk model. `http://www.sts.org/quality-research-patient-safety/sts-public-reporting-online/explanation-of-sts-chsd-mortality-risk-model`.

Spivey, Michael Z, Warren B Powell. 2004. The dynamic assignment problem. *Transportation Science* **38**(4) 399–419.

Stein, Clifford, Van-Anh Truong, Xinshang Wang. 2017. Advance service reservations with heterogeneous customers. Working paper.

Stidham, Jr., S. 1985. Optimal control of admission to a queueing system. *Automatic Control, IEEE Transactions on* **30**(8) 705–713.

Talluri, K., G. Van Ryzin. 1998. An analysis of bid-price controls for network revenue management. *Management Science* **44**(11) 1577–1593.

Talluri, K., G. Van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* **50**(1) 15–33.

Talluri, Kalyan T., Garrett J. van Ryzin. 2004. *The Theory and Practice of Revenue Management*. Springer.

Truong, Van-Anh. 2014. Approximation algorithm for the stochastic multiperiod inventory problem via a look-ahead optimization approach. *Mathematics of Operations Research* .

Truong, Van-Anh. 2015. Optimal advance scheduling. *Management Science* **61**(7) 1584–1597.

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

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

Wang, Xinshang, Van-Anh Truong. 2017. Multi-priority online scheduling with cancellations. *Operations Research* . Forthcoming.

Wang, Xinshang, Van-Anh Truong, David Bank. 2015. Online advance admission scheduling for services, with customer preferences. Working paper.

Wang, Xinshang, Van-Anh Truong, Shenghuo Zhu, Qiong Zhang. 2016. Dynamic optimization of mobile push campaigns. Working paper.

Wayne, Kevin Daniel. 1999. Generalized maximum flow algorithms. Ph.D. thesis, Cornell University, Ithaca, NY.

Yao, Frances, Alan Demers, Scott Shenker. 1995. A scheduling model for reduced cpu energy. *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*. IEEE, 374–382.

Zhang, Dan, William L. Cooper. 2005. Revenue management for parallel flights with customer-choice behavior. *Operations Research* **53**(3) 415–431.

Zhang, Liqi, Lingfa Lu, Jinjiang Yuan. 2009. Single machine scheduling with release dates and rejection. *European Journal of Operational Research* **198**(3) 975 – 978.

Zhong, Wenliang, Rong Jin, Cheng Yang, Xiaowei Yan, Qi Zhang, Qiang Li. 2015. Stock constrained recommendation in tmall. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15, ACM, New York, NY, USA, 2287–2296.