Online Optimization Problems

by

Xin Lu

B.S., Peking University (2008)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author . . . .
Sloan School of Management
May 17th, 2013

Certified by . . . .
Patrick Jaillet
Dugald C. Jackson Professor
Thesis Supervisor

Accepted by . . . .
Dimitris Bertsimas
Boeing Leaders for Global Operations Professor
Codirector, Operations Research Center

Online Optimization Problems

by

Xin Lu

Submitted to the Sloan School of Management
on May 17th, 2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

In this thesis, we study online optimization problems in routing and allocation applications. Online problems are problems where information is revealed incrementally, and decisions must be made before all information is available. We design and analyze algorithms for a variety of online problems, including traveling salesman problems with rejection options, generalized assignment problems, stochastic matching problems, and resource allocation problems. We use worst case competitive ratios to analyze the performance of proposed algorithms.

We begin our study with online traveling salesman problems with rejection options where acceptance/rejection decisions are not required to be explicitly made. We propose an online algorithm in arbitrary metric spaces, and show that it is the best possible. We then consider problems where acceptance/rejection decisions must be made at the time when requests arrive. For different metric spaces, we propose different online algorithms, some of which are asymptotically optimal.

We then consider generalized online assignment problems with budget constraints and resource constraints. We first prove that all online algorithms are arbitrarily bad for general cases. Then, under some assumptions, we propose, analyze, and empirically compare two online algorithms, a greedy algorithm and a primal dual algorithm.

We study online stochastic matching problems. Instances with a fixed number of arrivals are studied first. A novel algorithm based on discretization is proposed and analyzed for unweighted problems. The same algorithm is modified to accommodate vertex-weighted cases. Finally, we consider cases where arrivals follow a Poisson Process.

Finally, we consider online resource allocation problems. We first consider the problems with free but fixed inventory under certain assumptions, and present near optimal algorithms. We then relax some unrealistic assumptions. Finally, we generalize the technique to problems with flexible inventory with non-decreasing marginal costs.

Thesis Supervisor: Patrick Jaillet
Title: Dugald C. Jackson Professor

Acknowledgments

First, I would like to thank my thesis advisor Professor Patrick Jaillet, for his support, guidance, and encouragement through my graduate years. He taught me innumerable lessons in research as well as in life. I could not wish for a better advisor. I am also grateful to the other members of my thesis committee, Professor Vivek Farias and Professor Michel Goemans. Their technical and editorial advice improves the quality of this thesis.

I would like to thank my friends and colleagues, Andrew Mastin, Maokai Lin, Dawson Hwang, Swati Gupta, and Yehua Wei. Discussion with them provides me countless insights on my research.

I gratefully acknowledge the funding sources that supported my Ph.D. study. I was funded by Chyn Duog Shiah Memorial Fellowship for my first year. My work was then supported by the National Science Foundation, the Office of Naval Research, and the Air Force Office of Scientific Research.

Finally, I must thank my mother. This thesis would not be possible with her unconditional love and support. I am also deeply grateful to my father, who always believed I would succeed. He is long gone but will never be forgotten.

Contents

1 Introduction **15**

1.1 Examples and Applications 16

1.2 Competitive Analysis 16

1.2.1 Online versus offline 16

1.2.2 Deterministic algorithms 17

1.2.3 Randomized online algorithms 18

1.2.4 An alternative definition of competitive ratios 19

1.3 Literature Review 20

1.3.1 Online optimization 20

1.3.2 Online TSP with rejection options 20

1.3.3 Generalized online assignment problems 22

1.3.4 Online stochastic matching problems 23

1.3.5 Online resource allocation problems 24

1.4 Thesis Outline and Contributions 25

1.4.1 Chapter 2, online TSP with rejection options 25

1.4.2 Chapter 3, generalized online assignment problems 26

1.4.3 Chapter 4, online stochastic matching problems 26

1.4.4 Chapter 5, online resource allocation problems 27

2 Online TSP with Rejection Options **29**

2.1 Introduction 29

2.1.1 Formal definitions of the problems 31

2.2 Notations 31

2.3 The Basic Version 32

2.3.1 Best possible 2-competitive online algorithm 32

2.3.2 Polynomial-time algorithms 36

2.3.3 Prize-collecting generalization 39

2.3.4 Multi-server generalization 45

2.4 The Real-time Version 46

2.4.1 The case of the non-negative real-line $\mathbb{R}_+$ 46

2.4.2 The case of the real-line $\mathbb{R}$ 49

2.4.3 The case of general metric spaces 53

3 Generalized Online Assignment Problems **63**

3.1 Introduction 63

3.1.1 Formal problem definitions 64

3.1.2 Mathematical programming formulations 65

3.2 General Cases 66

3.2.1 Inhomogeneous cases 66

3.2.2 Large bids 67

3.2.3 Assumptions on the model 67

3.3 Special Cases 68

3.3.1 Upper bound 68

3.3.2 Greedy algorithm 69

3.3.3 Primal-dual algorithm 71

3.4 Numerical Results 75

3.4.1 Underlying structures 75

3.4.2 Evolutionary performance 76

3.4.3 Updating rules 76

3.5 Randomized Algorithms 77

3.6 Extensions 80

3.6.1 Bounded cases 80

3.6.2 Fractional bids 81

4 Online Stochastic Matching Problems **83**

4.1 Introduction 83

4.1.1 Our results and techniques 84

4.2 Preliminary on Competitive Ratios 87
4.3 Stochastic Matching with Integral Arrival Rates 88
4.3.1 Online algorithm 89
4.3.2 Upper bound on the offline algorithm 91
4.3.3 Certificate events 91
4.3.4 Lower bound on the online algorithm 94
4.3.5 Tight example 102
4.4 Extension to Vertex-Weighted Stochastic Matching 103
4.4.1 Original algorithm 103
4.4.2 Modification 104
4.5 Unweighted Stochastic Matching Problems Revisit 111
4.6 Stochastic Matching with General Arrival Rates 115
4.6.1 Offline solution 116
4.6.2 Upper bound on the optimal offline solution 116
4.6.3 Randomized algorithm 117
4.6.4 Lower bound on the online algorithm 117
4.7 Poisson Arrivals 121
4.7.1 Algorithms 121

5 Online Resource Allocation Problems 125
5.1 Introduction 125
5.1.1 Our techniques and contributions 128
5.2 One-Time Learning 129
5.3 Dynamic Pricing 134
5.4 Learning From the Past 136
5.5 Heterogeneous Customers 140
5.6 Flexible Inventory 143

6 Conclusion and Future Work 147
6.1 Summary of Thesis 147
6.2 Future Research 148

List of Figures

2-1 WOGI Shortcuts 33

2-2 Splitting Operation 53

2-3 An illustration of $\mathcal{M}_2$ 54

3-1 Performance v.s. underling graph 76

3-2 Performance v.s. time 77

3-3 Comparison between different updating rules 78

4-1 Cycles of length 4. Thin edges carry 1/3 flow; and thick edges carry 2/3 flow. 90

4-2 Possible configurations of i's neighborhood in the graph. All edges carry 1/3 flow. The number next to advertiser a indicates $f_a = \sum_{i\sim a} f_{a,i}$. 91

4-3 Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates $f_a = \sum_{i\sim a} f_{a,i}$. 94

4-4 Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a. 95

4-5 Cycle of length 4. Thin edges carry 1/3 flow; and thick edges carry 2/3 flow. 97

4-6 Modification of $\mathbf{f}$. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a. 104

4-7 Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a. 105

4-8 Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a. 109

4-9 Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a. 110

List of Tables

Chapter 1

Introduction

Traditionally, optimization tools and methodologies within the operations research community have mainly focused on deterministic planning for various problems such as routing problems and resource allocation problems. However, when reality differs from anticipation, such solutions may perform poorly. For example, unexpected events such as bad weather could lead to large losses in the transportation industry. Furthermore, information about practical problems to solve is rarely completely known a priori. For example, requests for taxi services are generally revealed over time without advance notice. Waiting for all necessary information is costly if not impossible for many applications. Thus, for such problems, it is necessary to develop approaches that make decisions online.

The focus of this thesis is on designing and studying algorithms for various online optimization problems. For the traveling salesman problem (TSP), we design online algorithms for various scenarios. We also study generalized online assignment problems, and propose two best possible algorithms. Then we focus on two special cases of the assignment problems: online stochastic matching problems and online resource allocation problems.

Outline: The rest of the chapter is organized as follows: In Section 1.1, we further motivate our study by presenting some examples and applications. In Section 1.2, we introduce competitive analysis, the performance measurement used in online optimization. In Section 1.3, we give a literature review of problems related to the work in this thesis. Finally, in Section 1.4, we summarize the contributions of the thesis.

1.1 Examples and Applications

In this section, we discuss some examples to illustrate the importance of online optimization. Courier companies such as UPS and FedEx offer expedited services like same day or overnight deliveries. A priori planning may not be possible because forecast may not be accurate enough. In such cases, making good real time decisions as service requests arrive is essential.

Another example is about sponsored ads on search engines. Google and other search engine companies make billions of dollars by displaying ads to users via pay-per-click or pay-per-impression models. In both cases, ads should be displayed to users only if they are related to users' keywords. Although existing historical data helps forecast future keywords, heavy-tail phenomena and unexpected rare events make the forecast less than ideal. Hence, making real time decisions based on historical data as well as incoming information could help improve the performance of ads placement.

For some NP-hard problems, although they do not have to be solved online, online algorithms provide good approximation solutions. For example, for bin packing problems, simple online algorithms (e.g., first-fit) always yield 1.7-approximation solutions.

Other applications can be also found in paging problems, inventory control, and airline ticket sales problems.

1.2 Competitive Analysis

Competitive analysis is widely used for evaluating the quality of online algorithms. The key concept in analyzing an online algorithm is the notion of the competitive ratio, which compares a solution produced by the online algorithm with the best possible solution.

1.2.1 Online versus offline

As mentioned earlier, for many applications, problem inputs are revealed in an online fashion. There are two main models that describe how problem inputs are revealed: sequential and real time. In the sequential model, requests are ordered and presented one by one. Decisions regarding the currently revealed requests must be made before the next request. Generalized online assignment problems studied in Chapter 3 and online stochastic matching problems studied in Chapter 4 mainly use this model. In the real time model, all requests are associated with released times, the time they are revealed to the online algorithm. The TSP with rejection options in Chapter 2 and online resource allocation problems in Chapter 5 use the real time model.

An algorithm is called an online algorithm if its decision at time t (or step i as in sequential models) only depends on information that is revealed up to time t (or step i), not afterwards. Note that, depending on the applications, some decisions are irrevocable while others are not. For the purpose of evaluating online algorithms, let us introduce offline algorithms. An offline algorithm is to solve the same problem as an online algorithm, except that all information of problem instances is revealed to an offline algorithm at the beginning. Therefore, a decision that an offline solution makes at time t (or step i) could depend on information revealed before and after t (or step i). However, an offline solution must satisfy all constraints of the problem. For example, in the online TSP, the release time of a request is the earliest time that the request can be served. Thus, although an offline algorithm knows the existence of a request, it cannot serve the request before its release time.

1.2.2 Deterministic algorithms

For minimization problems, an online algorithm is called c-competitive ($c \geq 1$), if c is the smallest number such that for any instance of the problem, the cost of the solution given by the online algorithm is at most c times the cost of an optimal offline solution for the instance:

$$\text{Cost}_{\text{online}}(I) \leq c \cdot \text{Cost}_{\text{optimal}}(I), \forall \text{ instance } I.$$

Equivalently, the competitive ratio equals to

$$c = \sup_{I} \frac{\text{Cost}_{\text{online}}(I)}{\text{Cost}_{\text{optimal}}(I)}.$$

An online algorithm is said to be best possible if no other algorithm has a strictly smaller competitive ratio.

Similarly, for maximization problems, an online algorithm is called c-competitive ($c \leq 1$), if c is the largest number such that for any instance of the problem, the cost of the solution given by the online algorithm is at least c times the cost of an optimal offline solution for the instance:

$$\text{Cost}_{\text{online}}(I) \geq c \cdot \text{Cost}_{\text{optimal}}(I), \forall \text{ instance } I.$$

Or equivalently, it equals to

$$c = \inf_I \frac{\text{Cost}_{\text{online}}(I)}{\text{Cost}_{\text{optimal}}(I)}.$$

An online algorithm is said to be the best possible if no algorithm has a strictly larger competitive ratio.

1.2.3 Randomized online algorithms

For randomized online algorithms, competitive ratios could be define in a way similar to the one mentioned above. However, depending on how the offline version of the problem is defined, there are three different types of adversaries: oblivious, adaptive online, and adaptive offline. The following are their definitions for sequential models:

Oblivious adversary: The oblivious adversary knows the randomized algorithm, but does not know the realization of the algorithm. The adversary constructs the request sequence in advance before observing the outcome of the randomized algorithm. Hence, the offline optimal solution in this case is a deterministic solution. For maximization problems, the competitive ratio is defined as

$$c_{\text{oblivious}} = \inf_I \frac{\mathbb{E}[\text{Cost}_{\text{online}}(I)]}{\text{Cost}_{\text{optimal}}(I)}.$$

Adaptive online adversary: The adaptive online adversary knows the randomized algorithm and the realization of the algorithm up to the current time. The adversary makes the next request based on the online algorithm's answers to previous ones, and has to serve it immediately as well. Therefore, both the online and offline solution are random. The competitive ratio is defined as

$$c_{\text{adaptive online}} = \inf_I \frac{\mathbb{E}[\text{Cost}_{\text{online}}(I)]}{\mathbb{E}[\text{Cost}_{\text{adversary}}(I)]}.$$

Adaptive offline adversary: The adaptive offline adversary knows the randomized algorithm and the realization of the algorithm up to the current time. The adversary makes the next request based on the online algorithm's answers to previous ones, but serves all requests optimally at the end. The competitive ratio is defined as:

$$c_{\text{adaptive offline}} = \inf_I \frac{\mathbb{E}[\text{Cost}_{\text{online}}(I)]}{\mathbb{E}[\text{Cost}_{\text{optimal}}(I)]}.$$

The three types of adversaries and competitive ratios can be defined similarly for real time models, by discretizing continuous time into small time intervals.

In this thesis, we only design and analyze online algorithm against oblivious adversaries, because as the following results state, randomness against the other two types of adversaries does not help much:

Theorem 1 ([17]). $c_{\text{oblivious}} \leq c_{\text{adaptive online}} \leq c_{\text{adaptive offline}}$.

Theorem 2 ([17]). *If there exists a c-competitive randomized online algorithm against an adaptive offline adversary, then there exists a c-competitive deterministic online algorithm.*

Theorem 3 ([17]). *If there exists a c_1-competitive online algorithm against an oblivious adversary, and a c_2-competitive online algorithm against an adaptive online adversary, then there exists a c_1c_2-competitive online algorithm against an adaptive offline adversary.*

Generally speaking, randomized online algorithms are difficult to analyze. However, Yao's principle [69] provide an approach to establish lower bounds on randomized algorithms against oblivious adversaries. It transform analysis of randomized online algorithms to analysis of deterministic online algorithms against random instances, which is easier.

Theorem 4 ([69]). *Let p be a probability distribution over the algorithms $\mathcal{A}$, and let A denote a random algorithm chosen according to p. Let q be a probability distribution over the inputs $\mathcal{X}$, and let X denote a random input chosen according to q. Then,*

$$\max_{x \in \mathcal{X}} \mathbb{E}[c(A,x)] \geq \min_{a \in \mathcal{A}} \mathbb{E}[c(a,X)].$$

1.2.4 An alternative definition of competitive ratios

When both online algorithms and problem instances are random, competitive ratios could be defined in an alternative way, as we do in Section 4.2. Instead of comparing expected costs, we compare the realized costs of online solutions and optimal offline solutions:

$$\Pr\Big(\text{Cost}_{\text{online}}(I) \leq c \cdot \text{Cost}_{\text{optimal}}(I)\Big) \geq 1-\epsilon, \forall \text{ instance } I,$$

where ϵ goes to 0 as the instance size goes to infinite. As we show in Section 4.2, the two competitive ratios are closely related for the class of algorithms proposed in Chapter 4. However, this is not true for algorithms in general.

Throughout the thesis, we use competitive analysis for evaluating online algorithms.

1.3 Literature Review

1.3.1 Online optimization

The first systematic study of online algorithms is given by Sleator and Tarjan [65], who suggest comparing an online algorithm with an optimal offline algorithm. Karlin et al. [49] introduce the notion of a *competitive ratio*. Online algorithms have been used to analyze paging in computer memory systems, distributed data management, navigation problems in robotics, multiprocessor scheduling, etc. (e.g. see the survey paper of Albers [6] and the books of Borodin and El-Yaniv [22] and Fiat and Woeginger [32] for more details and references.)

1.3.2 Online TSP with rejection options

The literature for the TSP is vast. The interested reader is referred to the books by Lawler et al. [54] and Korte and Vygen [52] for comprehensive coverage of results concerning the classical TSP.

Research concerning online versions of the TSP is more recent but has been growing steadily. Kalyanasundaram and Pruhs [46] examine a unique version where new cities are revealed locally during the traversal of a tour (i.e., an arrival at a city reveals any adjacent cities that must also be visited). Angelelli et al. [8, 9] study related online routing problems in a multi-period setting. Bent and Van Hentenryck [18] have looked at online stochastic optimization techniques (e.g. scenario-based approaches) for addressing dynamic online routing problems; see their book [38] for more references and applications of these approaches.

More closely related to the results in the thesis is the stream of works which started with the paper by Ausiello et al. [13]. In this paper, the authors study the online version of the TSP with release dates (but with no service flexibility); they analyze the problem on the real line and on general metric spaces, developing online algorithms for both cases and achieving an optimal online algorithm for general metric spaces, with a competitive ratio of 2. They also provide a polynomial-time online algorithm, for general metric spaces, which is 3-competitive. Subsequently, the paper by Ascheuer et al. [10] implies the existence of a polynomial-time algorithm, for general metric spaces, which is 2.65-competitive as well as a $(2 + \epsilon)$-competitive $(\epsilon > 0)$ algorithm for Euclidean spaces. Lipmann [55] develops an optimal online algorithm for the real line, with a competitive ratio of 1.64. Blom et al. [21] give an optimal online algorithm for the non-negative real line, with a competitive ratio of $\frac{3}{2}$, and also consider different adversarial algorithms in the definition of the competitive ratio. Jaillet and Wagner [44] introduce the notion of a disclosure date, which is a form of advanced notice for the online salesman, and quantify the improvement in competitive ratios as a function of the advanced notice. A similar approach was taken by Allulli et al. [7] in the form of a *lookahead*.

There has also been work on generalizing the basic online TSP framework. The paper by Feuerstein and Stougie [31] considers the online Dial-a-Ride problem, where each city is replaced by an origin-destination pair. The authors consider both the uncapacitated case, giving a best-possible 2-competitive algorithm, and the capacitated case, giving a 2.5-competitive algorithm. The previously cited paper by Ascheuer et al. [10] also gives a 2-competitive online algorithm and a $(1 + \sqrt{1 + 8\rho})/2$-competitive polynomial-time online algorithm for the uncapacitated online Dial-a-Ride problem (ρ being the approximation ratio of a simpler but related offline problem). Their algorithm is generalizable to the case where there are multiple servers with capacities; this generalization is also 2-competitive. Jaillet and Wagner [45] consider the (1) online TSP with precedence and capacity constraints and the (2) online TSP with m salesmen. For both problems they propose 2-competitive online algorithms (optimal in case of the m-salesmen problem), consider polynomial-time online algorithms, and then consider resource augmentation, where the online servers are given additional resources to offset the powerful offline adversary advantage. Finally, they study online algorithms from an asymptotic point of view, and show that, under general stochastic structures for the problem data, *unknown and unused by the online player*, the online algorithms are almost surely asymptotically optimal.

There also has been other recent work dealing with online routing problems which do not require the server to visit every revealed request. Ausiello et al. [12] analyze the online Quota TSP, where each city to be visited has a weight associated with it and the server is given the task to find the shortest sub-tour through cities in such a way to collect a given quota of weights by visiting the chosen cities. They present an optimal 2-competitive algorithm for a general metric space. In Ausiello et al. [11] provide a competitive analysis of the "prize collecting TSP", a generalization of the quota problem where penalties for not visiting cities are also included, beyond meeting a given quota. They provide a 7/3-competitive algorithm and a lower bound on any competitive ratios of 2 for a general metric space, and refer to a 2-competitive algorithm and a lower bound of 1.89 on the non-negative real line. More generally, assuming a ρ-approximation algorithm for the offline problem, they show that their online algorithm is a $(2\rho + \frac{\rho}{1+2/\rho})$-competitive polynomial time algorithm. In Jaillet and Lu [43], we provide a competitive analysis of the "TSP with flexible service", a special case of the online prize-collecting TSP with no quotas. On the half-line, we provide and prove the optimality of a 2-competitive polynomial time online algorithm based on reoptimization subroutines, and extend it to an optimal 2-competitive online algorithm on the real line. Finally we consider the case of a general metric space and propose an original c-competitive online algorithm, where $c = \frac{\sqrt{17}+5}{4} \approx 2.28$. We also give a polynomial-time $(1.5\rho + 1)$-competitive online algorithm which uses a polynomial-time ρ-approximation for the offline problem.

1.3.3 Generalized online assignment problems

Many research efforts have been made on generalized assignment problems (GAP) in an offline fashion. Shmoys and Tardos [64] present an LP-rounding 2-approximation algorithm for a special case of GAP. Chekuri and Khanna [25] develop PTAS for a special case of GAP, multiple knapsack problems. They also classify the APX-hard special cases of GAP. Given an approximation algorithm for a single-bin problem, Fleischer et al. [34] give an LP-rounding algorithm and a simple local search algorithm for SAP.

Matching problems and generalized assignment problems in an online setting have also been extensively studied in the last two decades. Karp, Vazirani and Vazirani [50] study the online bipartite matching problem. They propose a randomized algorithm RANKING with competitive ratio of $1 - 1/e$, and prove its optimality. Kalyanasundaram and Pruhs [47] study online b-matching problem, where each bidder has a budget of b and each item has unit price. They propose a deterministic algorithm BALANCE with competitive ratio of $1-\frac{1}{(1+\frac{1}{b})^b}$. They also show that it is asymptotically optimal for deterministic algorithms. For the same problem, Goel and Mehta [36] analyzes the performance of the greedy algorithm with queries arriving in a random permutation. They prove a tight competitive ratio of $1 - 1/e$. Mehta et al. [59] and Lahaie et al. [53] consider the adwords problem, where each bidder has a budget and each item has a price. By introducing a potential function to find the right tradeoff between acting greedily and keeping bidders able to bid in the future, they propose an optimal deterministic algorithm with competitive ratio of $1 - 1/e$. For the same problem, Buchbinder et al. [24] gives a simple primal-dual algorithm by using weak duality, which also is $(1 - 1/e)$-competitive, and thus optimal. The same technique is also applied to other problems [23].

Several different aspects of the adwords problem have also been studied. On the game theoretic aspect, Aggarwal et al. [2], Edelman et al. [27] and Varian [68] study properties of different auction mechanism. There are also some papers focus on the optimization problems from advertisers' prospective. Rusmevichientong and Williamson [63] consider how to change the set of interested keywords dynamically to maximize the revenue. Feldman et al. [30] propose an algorithm to maximize the number of bids won by the advertiser within his budget.

Depending on the size of budgets, some special cases of GAP have also been studied extensively. When budgets are all 1, it is known as online matching problems; when budgets are all large, it is known as online resource allocation problems. We review the two areas in details in the following two sections.

1.3.4 Online stochastic matching problems

Bipartite matching problems and related advertisement allocation problems have been studied extensively in the operations research and computer science literature.

As stated in last section, adversarial models where no information is known about requests have been studied. Karp et al. [50] look at the bipartite matching problem and give a best possible randomized algorithm (RANKING) with competitive ratio $1 - 1/e$. Kalyanasundaram and Pruhs [47] give a $1 - 1/e$-competitive algorithm for b-matching problems. Mehta et al. [59, 60] and Buchbinder et al. [24] propose two different $1 - 1/e$ competitive algorithms for the AdWords problem. More recently, Aggarwal et al. [3] give a $1 - 1/e$-competitive algorithm for the vertex-weighted bipartite matching problem.

However, adversarial models may be too conservative for some applications where worst-case scenarios are unlikely to happen. Less conservative models have been proposed. In the random permutation model, when the set of requests is unknown, but the order of the sequence is random, Goel and Mehta [36] show that a greedy algorithm is $1-1/e$ competitive. Devanur and Hayes [26] propose a near optimal algorithm for AdWords under some mild assumptions. Agrawal et al. [4] further propose a near optimal algorithm for general online linear programming problems using similar techniques. Mahdian and Yan [56] and Karande et al. [48] simultaneously show RANKING algorithm is 0.696-competitive for matching problem. Mirrokni et al. [61] propose an algorithm works well under both adversarial and random arrival model for Adwords.

The random permutation model may still be too conservative in practice, when statistics about requests may be available. In the stochastic i.i.d. model, when requests are drawn repeatedly and independently from a known probability distribution over the different impression types, Feldman et al. [29] prove that one can do better than $1 - 1/e$. Under the restriction that the expected number of request of each impression type is an integer, they provide a 0.670-competitive algorithm. They also show that no algorithm can achieve a competitive ratio of 0.989. Bahmani and Kapralov [15] modify the algorithm and give a competitive ratio of 0.699 under the same assumption. They also improved the upper bound to 0.902. More recently, Manshadi et al. [57] removed the assumption that the expected number of arrivals is integral, and present a 0.702-competitive algorithm (the same algorithm achieves a competitive ratio of 0.705 under the integral assumption). They also improve the upper bound to 0.86 with the integral assumption and 0.823 without the integral assumption. Finally Haeupler et al. [37] recently proposed a 0.667-competitive algorithm for the edge-weighted problem under the stochastic i.i.d. model.

1.3.5 Online resource allocation problems

Online resource allocation problems have attracted wide interests in the operations research, computer science, and management science communities. Various special cases of the problems have been studied extensively. Kleinberg [51] presents a $1-O(\sqrt{k})$-competitive algorithm for k secretary problem under random permutation model. This is the first online algorithm with a competitive ratio approaching 1 as the input parameters become large. Devanue and Hayes [26] study the adwords problem under random permutation model. They propose an online algorithm with a competitive ratio $1-O(\sqrt[3]{m^2 \log(n)/OPT})$. Feldman et al. [33] study an online packing problem. They also propose an online algorithm with a competitive ratio approaching 1 as the input parameters go to infinite. Both papers assume that the total number of requests is given explicitly in advance, and in order to achieve $1-\epsilon$-competitive ratio, the number of requests $n \geq O(1/\epsilon^3)$. More recently, Agrawal et al. [4] study an online linear programming problem under random permutation model. They present an online algorithm that updates price threshold dynamically. As a result of the dynamic updating, they reduce the lower bound on the number of requests from $O(1/\epsilon^3)$ to $O(1/\epsilon^2)$.

Approximate dynamic programming approaches are also used to study resource allocation problems. For example, Bertsimas and Demir [20], Adelman [1], and Zhang and Adelman [70] designed approximate dynamic programming heuristics for multidimensional knapsack problems and network revenue management problems. Although these heuristics have excellent practical performance, theoretical guarantees are very difficult to obtain.

Fluid models have been considered as well, where stochastic discrete arrival processes are replaced by fluid arrival processes with deterministic arrival rates. Gallego and van Ryzin [35] and Akan and Ata [5] studied network revenue management problems under such models.

1.4 Thesis Outline and Contributions

1.4.1 Chapter 2, online TSP with rejection options

In this chapter, we study two versions of online traveling salesman problems with rejection options; we propose and evaluate algorithms for a variety of cases. We first consider the problems where acceptance/rejection decisions need not to be made explicitly, and provide a best possible online algorithm, with a competitive ratio of 2. We also show that the same algorithm can be applied for the Prize Collecting TSP as well as some other problems, and remain 2-competitive. Then, a polynomial-time version of the algorithm that is 2ρ-competitive is presented, where ρ is the approximation ratio for the offline version of the problems.

We then consider the problems where acceptance/rejection decisions must be made at arrivals of requests. We design algorithms for the problems in a variety of metric spaces, including non-negative real line, real line, and arbitrary metric spaces. In the non-negative real line, a best possible 2.5-competitive polynomial time algorithm is presented. In the real line, we prove a lower bound of 2.64 on any competitive ratios, and propose a 3-competitive online algorithm. In general metric spaces, we prove that $\Omega(\sqrt{\ln n})$ is a lower bound on any competitive ratios. Finally, among the restricted class of online algorithms with prior knowledge about the total number of requests n, we also provide an asymptotically best possible $O(\sqrt{\ln n})$-competitive algorithm.

This chapter is also available in [42].

1.4.2 Chapter 3, generalized online assignment problems

In this chapter, we consider generalized online assignment problems, where buyers with limited budgets are interested in purchasing items coming one at a time from a set of distinct object types. Each type having a limited resource. Assignment of an item of a given type to a buyer consumes a given amount of that limited resource as well as the buyer's budget. We propose, analyze, and empirically compare two online algorithms, a greedy algorithm and a primal dual algorithm. Our main result is that both algorithms are 1/2-competitive under two key assumptions: (i) each buyer's budget and each object type's allocated resource are large compared to both the prices buyers are willing to pay for a given item, and the amount of resources consumed by the assignment of an item to a buyer; (ii) prices are proportional to amount of resources consumed. We also show that there are no other online algorithms, either deterministic or randomized, that achieve a strictly better competitive ratio. We also prove that without the two assumptions stated above, no non-trivial algorithm exists.

This chapter is also available in [39].

1.4.3 Chapter 4, online stochastic matching problems

In this chapter, we first consider unweighted stochastic matching problems. A general class of online algorithms are proposed. Algorithms in this class are robust, and use computationally efficiently offline procedures. Under the integrality restriction on the expected number of impressions of each types, a $(1 - 2e^{-2})$-competitive algorithm is presented. Without the restriction, we provide a 0.706-competitive algorithm.

Our techniques can be applied to other related problems such as online stochastic b-matching problems (quite trivially) and vertex-weighted version of online stochastic problems. For the latter one, we obtain a 0.725-competitive algorithm under the integrality restriction.

Finally, we show the validity of all our results under a Poisson arrival model, removing the need to assume that the total number of arrivals is fixed and know in advance, as is required for the analysis of the stochastic models.

This chapter is also available in [41].

1.4.4 Chapter 5, online resource allocation problems

In this chapter, we first consider online resource allocation problems, where inventory level is fixed and given in advance, and an unknown number of customers arrive according to a random process. We present a learning-based online algorithm that updates dual price for resource based on observation from arrived customers. The algorithm is proven to be near optimal under certain assumptions. We then relax the assumptions and show similar results. Our results significantly improve previous ones by removing the need to know a priori the number of customers.

We then consider a similar problem with flexible inventory level. Instead of fixed inventory level, inventory can be replenished at non-decreasing marginal costs. Under certain assumptions, we present and analysis a near optimal online algorithm.

This chapter is also available in [40].

Chapter 2

Online TSP with Rejection Options

2.1 Introduction

In the classical Traveling Salesman Problem (TSP) in a metric space, we are given an origin, a set of points in that space, and the task is to find a tour of minimum total length, beginning and ending at the origin, that visits each point at least once. If one introduces a “time” aspect to the problem by considering a server visiting these points with a given constant speed, the objective can equivalently be stated as to minimize the time required to complete a tour. When requests to visit points include release dates, i.e., when a point can only be visited on or after its release date, we obtain the so-called “TSP with release dates”. Removing the need to visit all requests, one can associate a penalty with each request to visit a point. The server can then decide which points to serve and the objective is to minimize a linear combination of the time to go through all accepted requests plus the penalties of the rejected ones. In this chapter, we consider online versions of the TSP with release dates and rejection penalty. When the decisions to accept or reject requests can be done any time after their release dates, the online version of the problem will be called the *basic* version, and when the decisions must be made immediately at the release dates, the corresponding online version will be called the *real-time* version.

There are many motivations for looking at such problems. Some come from large scale fleet management problems associated with the pick-up and delivery of packages in a dynamic environment, where a sizable fraction of the requests come in a real-time fashion, requiring both flexibility and fast response in an ever changing environment, and profitability over a longer horizon.

Other examples come from the emerging field of autonomous spatial exploration and information harvesting problems, there are many potential applications (some futuristic, yet within near-term technological capabilities), where micro (unmanned) autonomous vehicles (MAVs) will be launched with the mission of “finding out” and “communicating back” key information (such as casualties and immediate rescue needs in case of a natural disaster). Overall, the algorithmic challenges for such complex missions to be possible are formidable and diverse. In particular, it involves the development of fully autonomous algorithmic capabilities which would allow an MAV to “function” without external help, and in some cases, without the availability of an existing external infrastructure (for example a malfunctioning or non-existing GPS infrastructure). Our proposed online versions of the TSP variant we discussed above provide some of the fundamental building blocks needed to meet such algorithmic challenges. Requests to visit points could be various signals received from the multi-sensing capabilities of the MAV and indicating specific location to explore, rejecting such a request could incur a “penalty” (opportunity cost for missing out on key information). During the exploration the MAV collects information associated with accepted and visited requests. At the final state, the collected information can be communicated to a dispatch center.

2.1.1 Formal definitions of the problems

Online TSP with Rejection Options

Instance: A metric space $\mathcal{M}$ with a given origin o and a distance metric $d(\cdot,\cdot)$. A series of n requests represented by triples $(l_i, r_i, p_i)_{1 \le i \le n}$, where $l_i \in \mathcal{M}$ is the location (point in metric space) of request i, $r_i \in \mathbb{R}_+$ is its release date (first time after which it can be served), and $p_i \in \mathbb{R}_+$ is its penalty (for not being served). The problem begins at time 0; the server is initially idle at the origin (initial state), can travel at unit speed (when not idle), and eventually must be back and idle at the origin (final state). The earliest time the server reaches this final state is called the makespan.

Offline context: The number of requests n is known to the offline server. All requests are revealed to the offline server at time 0.

Online context: The number of requests n is not known to the online server. Requests are revealed to the online server at their release dates $r_i \ge 0$; assume $r_1 \le r_2 \cdots \le r_n$. There are two online versions:

> *Basic*: The online server can accept or reject a request any time after the request's release date.
>
> *Real-time*: The online server must accept or reject a request immediately at the time of the request's release date. Decisions are then final.

Objective: In all cases, minimize {the makespan to serve all accepted requests plus the total penalties of all rejected requests} among all feasible solutions.

The offline problem is thus a TSP with release dates and penalty, and the two online versions of the problem differ as to when decisions to accept or reject a request can be done.

2.2 Notations

The formal definitions of the problems considered in this chapter have been given in Section 2.1.1. We assume that we have at our disposal an exact algorithm (a black box) that solves any instance of the corresponding offline problems.

An instance I consisting of n requests is gradually revealed. The online server observes a series of partial instances $\{I_k\}_{1 \le k \le n}$, where I_k is the instance consisting of the first k

requests. For the instance I_k, let $C_{\text{opt}}(k)$ be the objective value of an optimal offline solution, τ_k be the corresponding optimal route (tour or path), T_k be the corresponding makespan, and S_k be the set of accepted requests by the black box. Given a route τ, we also use τ as a function $\tau(t) : \mathbb{R}^+ \to \mathcal{M}$, where $\tau(t)$ is the position of the server who follows τ at t. $L(\tau)$ represents the length of the route, i.e. the shortest time to travel through it, ignoring release dates of requests.

For a given online algorithm A, we let $C_A(k)$ be the total cost incurred by the online server on the instance I_k. We measure the quality of this online algorithm via its competitive ratio, i.e., the smallest upper bound on $C_A(n)/C_{\text{opt}}(n)$ for any instances of any size n. If there exists such a finite upper-bound c, then we say that A is c-competitive, and, in case no other online algorithms have smaller competitive ratios, we say that A is best possible. If a finite upper bound does not exist, then one can characterize the asymptotic behavior of the competitive ratios as a function of n (n representing the problem size) by providing functions f and g such that an $\Omega(f(n))$ and $O(g(n))$ are asymptotic lower and upper bounds on the competitive ratios for the problem.

2.3 The Basic Version

We focus here on the basic version of the problem. In Jaillet and Lu [43], we show a lower bound of 2 on the competitive ratio of any online algorithms for this problem on metric space:

Theorem 5 ([43]). *Any c-competitive online algorithm on $\mathbb{R}_+$ must have $c \geq 2$.*

In the remainder of the section, we first propose an online algorithm whose competitive ratio matches this lower bound in any general metric spaces. We then consider the design of a polynomial time online algorithm for this problem and, we finally address a slight generalization of the problem, involving both penalty for rejection of a request and prize for collection, if a request is accepted and served.

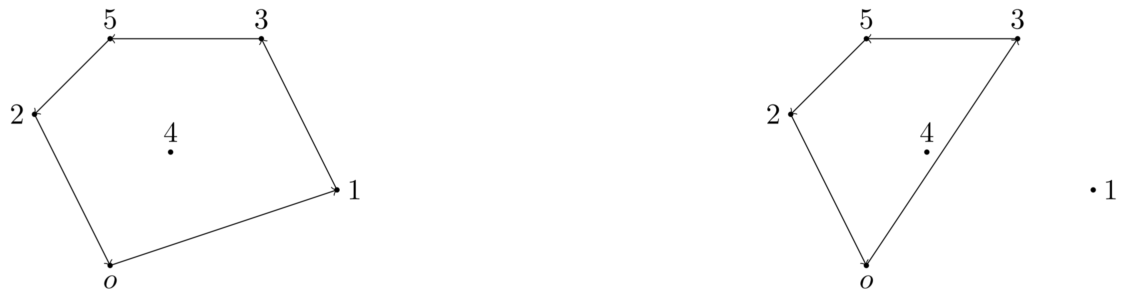
2.3.1 Best possible 2-competitive online algorithm

In the proposed algorithm, the online server makes use of the offline black box only when at the origin, and, any time at the origin, waits an appropriate amount of time (to be determined) before it starts on a new route. While engaged on a route, the server ignores all new requests. We label this algorithm WOGI for "Wait, Optimize, Go, and Ignore".

Before presenting the algorithm in details, let us first define some notations. We use i as a counter indicating how many times the online server has left the origin. We let u_i be the number of requests that have been released so far when the online server leaves the origin for the i^{th} time. We let P_i be the set of all requests among the first u_i requests that have not been served by the online server when it returns to the origin for the i^{th} time. We let s_i be the first request, not among the first u_{i-1} requests, which the online server visits on route τ_{u_i} (so $s_i > u_{i-1}$).

Our algorithm is designed in such a way that when the online server leaves the origin for the i^{th} time, it has two candidate routes to follow. It either follows the route τ_{u_i} exactly, or it uses a WOGI shortcut $\tau_{u_i,u_{i-1}}$, defined as follows: it skips the first requests on τ_{u_i} whose indices are no greater than u_{i-1}, goes directly to request s_i, and then follows the remaining part of τ_{u_i}.

Figure 2-1 below provides an illustration of a WOGI shortcut. In this example, five requests are released sequentially. The route τ_5 on the left, computed by the black box, passes through requests $1, 5, 3$, and 2. According to the definitions above, $s_5 = 3$. As showed on the right, the WOGI shortcut $\tau_{5,2}$ skips request 1 and goes directly to request 3. Note that no request is skipped afterwards. Assume that the server have returned to the origin once and request 1 has not been served, then $P_2 = \{1, 4\}$. Even if some requests are released shortly after the server leaves the origin to follow $\tau_{5,2}$, they are not included in P_2.



route τ_5 by blackbox

WOGI shortcut $\tau_{5,2}$

Figure 2-1: WOGI Shortcuts

Now, we are ready to provide a full description of Algorithm WOGI:

Algorithm 1 (WOGI).

0. Initialization: counter $i = 0$, $u_0 = 0$, and $P_0 = \emptyset$.

1. Assume k requests have been released. If $S_k \subset \{1, 2, ..., u_i\}$, wait for the next released request, and go to Step 1; otherwise, go to Step 2.

2. Assume k requests have been released, the WOGI server waits until $\max\{C_{\text{opt}}(k), t_{k,u_i}\}$, where $t_{k,u_i} \doteq 2C_{\text{opt}}(k) - L(\tau_{k,u_i}) - \sum_{j>u_i, j\notin S_k} p_j - \sum_{j\in P_i} p_j$ is the latest time to leave the origin to follow τ_{k,u_i} and maintain a competitive ratio of 2. If a new request is released during the waiting time, go to Step 1; otherwise, update $u_{i+1} = k, i = i + 1$, and go to Step 3.

3. The WOGI server takes one of the two routes and ignores all new requests before reaching back the origin:

3a. If $t_{u_i,u_{i-1}} \geq C_{\text{opt}}(u_i)$, he follows the WOGI shortcut $\tau_{u_i,u_{i-1}}$. After finishing the route, go to Step 4.

3b. If $t_{u_i,u_{i-1}} < C_{\text{opt}}(u_i)$, he follows τ_{u_i}. After finishing the route, go to Step 4.

4. Update P_i. Go to Step 1.

Let us first look at some properties of WOGI:

Lemma 1. $r_{s_{i+1}} \geq C_{\text{opt}}(u_i)$.

Proof. If $i = 0$, $C_{\text{opt}}(0) = 0$. Thus, the lemma is trivially true. If $i > 0$, let t be the time when the WOGI server leaves the origin for the i^{th} time. According to Step 2, $t \geq C_{\text{opt}}(u_i)$. On the other hand, at time t, only u_i requests are released. Thus, $\forall j > u_i$, $r_j \geq t \geq C_{\text{opt}}(u_i)$. In particular, it is true for $j = s_{i+1} > u_i$. □

Lemma 2. $t_{u_{i+1},u_i} \geq 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$.

Proof. If $i = 0$, $C_{\text{opt}}(0) = 0$. The lemma is trivially true. If $i > 0$, the offline server cannot visit request s_{i+1} before its release time $r_{s_{i+1}}$. Thus, $T_{u_{i+1}} \geq r_{s_{i+1}} + L(\tau_{u_{i+1},u_i}) - |l_{s_{i+1}}|$. Therefore,

$$\begin{aligned}
t_{u_{i+1},u_i} &= 2C_{\text{opt}}(u_{i+1}) - L(\tau_{u_{i+1},u_i}) - \textstyle\sum_{j>u_i,j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&= 2T_{u_{i+1}} + 2\textstyle\sum_{1\le j\le u_{i+1},j\notin S_{u_{i+1}}} p_j - L(\tau_{u_{i+1},u_i}) - \sum_{j>u_i,j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + 2L(\tau_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| + 2\textstyle\sum_{1\le j\le u_{i+1},j\notin S_{u_{i+1}}} p_j - L(\tau_{u_{i+1},u_i}) \\
&\quad - \textstyle\sum_{j>u_i,j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + L(\tau_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| - \textstyle\sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} - \textstyle\sum_{j\in P_i} p_j.
\end{aligned}$$

According to Lemma 1, $r_{s_{i+1}} \ge C_{\text{opt}}(u_i)$, we conclude $t_{u_{i+1},u_i} \ge 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. □

Lemma 3. *The WOGI server returns to the origin for the* i^{th} *time before* $2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$.

Proof. We use induction on i to prove this lemma. It is trivially true for $i = 0$.

Consider i. By induction, the server finishes his $(i-1)^{\text{th}}$ trip before $2C_{\text{opt}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j$. According to Lemma 2, $2C_{\text{opt}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j \le t_{u_i,u_{i-1}}$. Thus, the WOGI is at the origin at $\max\{t_{u_i,u_{i-1}}, C_{\text{opt}}(u_i)\}$. According to Step 2, he leaves the origin for the i^{th} time at exactly $\max\{t_{u_i,u_{i-1}}, C_{\text{opt}}(u_i)\}$. Based on which of the two is larger, we have two cases:

1. If $t_{u_i,u_{i-1}} \ge C_{\text{opt}}(u_i)$, then the WOGI server will take the shortcut τ_{u_{i+1},u_i}, and arrive at the origin at time $t_{u_i,u_{i-1}} + L(\tau_{u_{i+1},u_i}) = 2C_{\text{opt}}(u_i) - \sum_{u_{i-1}<j\le u_i,j\notin S_{u_i}} p_j - \sum_{j\in P_{i-1}} p_j \le 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset P_{i-1} \cup \{j : u_{i-1} < j \le u_i, j \notin S_{u_i}\}$.

2. If $t_{u_i,u_{i-1}} < C_{\text{opt}}(u_i)$, then the server will follow τ_{u_i}, and arrive at the origin at time $C_{\text{opt}}(u_i) + L(\tau_{u_i}) \le C_{\text{opt}}(u_i) + T_{u_i} = 2C_{\text{opt}}(u_i) - \sum_{j\notin S_{u_i}} p_j \le 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset S_{u_i}^c$, because τ_{u_i} pass through all requests in S_{u_i}.

□

We now can prove our main result:

Theorem 6. *Algorithm WOGI is 2-competitive and best possible.*

Proof. Assume there are a total of m requests and the WOGI server leaves and returns to the origin i times. According to Lemma 3, after the i^{th} trip, the server returns to the origin before time $2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$ and never leaves again. Therefore, total cost $C_{WOGI} \le 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j + \sum_{j\in P_i} p_j + \sum_{j=u_i+1}^{m} p_j = 2C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Since the WOGI server does not leaves the origin afterwards, $\forall u_i + 1 \le j \le m, j \notin S_m$. Thus, $C_{WOGI} \le 2C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j = 2C_{\text{opt}}(m) - \sum_{j=u_i+1}^{m} p_j \le 2C_{\text{opt}}(m)$. $\square$

2.3.2 Polynomial-time algorithms

WOGI repeatedly calls a black box that provides optimal solutions to corresponding offline problems. However, because these offline problems are NP-hard, WOGI can't be considered to be a polynomial-time algorithm, and this makes WOGI impractical for very large size problems. To address the complexity issue, we propose here a polynomial time algorithm, WOGI-apx, at the expense of an increase in the competitive ratio. WOGI-apx is simply the analog of WOGI with an approximation black box. Instead of solving offline TSPs optimally, WOGI-apx uses a ρ-approximation black box algorithm. Noting that, other than optimality, few properties of offline solutions are used in proving 2-competitiveness of WOGI, we expect the analysis in Section 2.3.1 to carry through for WOGI-apx.

Before presenting and analyzing WOGI-apx, let us first define some notations, which are analogs of the ones used for WOGI. Given the instance I_k, the offline approximation algorithm provides a solution that has cost $\tilde{C}_{\text{apx}}(k) \le \rho C_{\text{opt}}(k)$. Let $\tilde{T}_k$ be the makespan of the approximation solution, $\tilde{\tau}_k$ be the corresponding route, and $\tilde{S}_k$ be the set of requests served by the approximation solution. u_i is the number of released requests when the online server leaves the origin for the i^{th} time. The rejection set P_i is the set of requests that have not been served when the online server returns to the origin for the i^{th} time. Request s_i is the first request on the route $\tilde{\tau}_{u_i}$ whose index is strictly greater than u_{i-1}. When the online server leaves the origin for the i^{th} time, it has two candidate routes to follow. It either follows $\tilde{\tau}_{u_i}$, computed by the approximation solver, or a WOGI shortcut $\tilde{\tau}_{u_i,u_{i-1}}$. The WOGI shortcut $\tilde{\tau}_{u_i,u_{i-1}}$ skips the first few requests on $\tilde{\tau}_{u_i}$ whose indices are no greater than u_{i-1}, goes directly to request s_{i-1}, and then follows the remaining fraction of $\tilde{\tau}_{u_i}$.

Now we can present WOGI-apx:

Algorithm 2 (WOGI-apx)**.**

0. Initialization: counter $i = 0$, $u_0 = 0$, and $P_0 = \emptyset$.

1. Assume k requests have been released. If $\tilde{S}_k \subset \{1, .., u_i\}$, wait for the next released request, and go to Step 1; otherwise, go to Step 2.

2. Assume k requests have been released, the WOGI-apx server waits until $\max\{\tilde{C}_{\text{apx}}(k), t_{k,u_i}\}$, where $t_{k,u_i} \doteq 2\tilde{C}_{\text{apx}}(k) - L(\tilde{\tau}_{k,u_i}) - \sum_{j>u_i, j\notin\tilde{S}_k} p_j - \sum_{j\in P_i} p_j$. If a new request is released during the waiting time, go to Step 1; otherwise, update $u_{i+1} = k, i = i + 1$, and go to Step 3.

3. The WOGI-apx server takes one of the two routes and ignores all new requests before reaching back the origin:

3a. If $t_{u_i,u_{i-1}} \geq \tilde{C}_{\text{apx}}(u_i)$, he follows the WOGI-apx shortcut $\tilde{\tau}_{u_i,u_{i-1}}$. After finishing the route, go to Step 4.

3b. If $t_{u_i,u_{i-1}} < \tilde{C}_{\text{apx}}(u_i)$, he follows $\tilde{\tau}_{u_i}$. After finishing the route, go to Step 4.

4. Update P_{i+1}. $i = i + 1$. Go to Step 1.

As Lemma 1 through 3 use no property of the offline solutions from the black box, their analogs for the approximated version are also valid:

Lemma 4. $r_{s_{i+1}} \geq \tilde{C}_{\text{apx}}(u_i)$.

Proof. If $i = 0$, $\tilde{C}_{\text{apx}}(0) = 0$. Thus, the lemma is trivially true. If $i > 0$, let t be the time when the WOGI-apx server leaves the origin for the i^{th} time. According to Step 2, $t \geq \tilde{C}_{\text{apx}}(u_i)$. On the other hand, at time t, only u_i requests are released. Thus, $\forall j > u_i$, $r_j \geq t \geq \tilde{C}_{\text{apx}}(u_i)$. In particular, it is true for $j = s_{i+1} > u_i$. □

Lemma 5. $t_{u_{i+1},u_i} \geq 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j\in P_i} p_j$.

Proof. If $i = 0$, $\tilde{C}_{\text{apx}}(0) = 0$. The lemma is trivially true. If $i > 0$, the offline server cannot visit request s_{i+1} before its release time $r_{s_{i+1}}$. Thus, $\tilde{T}_{u_{i+1}} \geq r_{s_{i+1}} + L(\tilde{\tau}_{u_{i+1},u_i}) - |l_{s_{i+1}}|$. Therefore,

$$\begin{aligned}
t_{u_{i+1},u_i} &= 2\tilde{C}_{\text{apx}}(u_{i+1}) - L(\tilde{\tau}_{u_{i+1},u_i}) - \textstyle\sum_{j>u_i, j\notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&= 2\tilde{T}_{u_{i+1}} + 2\textstyle\sum_{1\le j\le u_{i+1}, j\notin \tilde{S}_{u_{i+1}}} p_j - L(\tilde{\tau}_{u_{i+1},u_i}) - \sum_{j>u_i, j\notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + 2L(\tilde{\tau}_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| + 2\textstyle\sum_{1\le j\le u_{i+1}, j\notin \tilde{S}_{u_{i+1}}} p_j - L(\tilde{\tau}_{u_{i+1},u_i}) \\
&\quad - \textstyle\sum_{j>u_i, j\notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + L(\tilde{\tau}_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| - \textstyle\sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} - \textstyle\sum_{j\in P_i} p_j.
\end{aligned}$$

According to Lemma 5, $r_{s_{i+1}} \ge \tilde{C}_{apx}(u_i)$, we conclude $t_{u_{i+1},u_i} \ge 2\tilde{C}_{apx}(u_i) - \sum_{j\in P_i} p_j$. □

Lemma 6. *The WOGI-apx server finishes his i^{th} trip before $2\tilde{C}_{\text{apx}}(u_i) - \sum_{j\in P_i} p_j$.*

Proof. We use induction on i to prove this lemma. It is trivially true for $i = 0$.

Consider i. By induction, the server finishes his $(i-1)^{\text{th}}$ trip before $2\tilde{C}_{\text{apx}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j$. According to Lemma 5, $2\tilde{C}_{\text{apx}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j \le t_{u_i,u_{i-1}}$. Thus, the WOGI-apx server is at the origin at $\max\{t_{u_i,u_{i-1}}, \tilde{C}_{\text{apx}}(u_i)\}$. According to Step 2, he leaves the origin for the i^{th} time at exactly $\max\{t_{u_i,u_{i-1}}, \tilde{C}_{\text{apx}}(u_i)\}$. Based on which of the two is larger, we have two cases:

1. If $t_{u_i,u_{i-1}} \ge \tilde{C}_{\text{apx}}(u_i)$, then the WOGI-apx server will take the shortcut $\tilde{\tau}_{u_{i+1},u_i}$, and arrive at the origin at time $t_{u_i,u_{i-1}} + L(\tilde{\tau}_{u_{i+1},u_i}) = 2\tilde{C}_{\text{apx}}(u_i) - \sum_{u_{i-1}<j\le u_i, j\notin \tilde{S}_{u_i}} p_j - \sum_{j\in P_{i-1}} p_j \le 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset P_{i-1} \cup \{j : u_{i-1} < j \le u_i, j \notin \tilde{S}_{u_i}\}$.

2. If $t_{u_i,u_{i-1}} < \tilde{C}_{\text{apx}}(u_i)$, then the server will follow $\tilde{\tau}_{u_i}$, and arrive at the origin at time $\tilde{C}_{\text{apx}} + L(\tilde{\tau}_{u_i}) \le \tilde{C}_{\text{apx}}(u_i) + \tilde{T}_{u_i} = 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j\notin \tilde{S}_{u_i}} p_j \le 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset \tilde{S}^c_{u_i}$, because τ_{u_i} pass through all requests in $\tilde{S}_{u_i}$.

□

We are now ready to prove our main result. The proof is different from the one of Theorem 6, because unlike $C_{\text{opt}}(k)$, $\tilde{C}_{\text{apx}}(k)$ is not necessarily non-decreasing in k.

Theorem 7. *Algorithm WOGI-apx is 2ρ-competitive.*

Proof. Assume there are a total of m requests, and the WOGI-apx server leaves and returns to the origin i times. According to Lemma 6, after the i^{th} trip, the server returns to the origin before time $2\tilde{C}_{\text{apx}}(u_i) - \sum_{l \in P_i} p_l$ and never leaves again. Therefore, total cost

$$\begin{aligned} C_{\text{WOGI-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) - \textstyle\sum_{j \in P_i} p_j + \sum_{j \in P_i} p_j + \sum_{j=u_i+1}^{m} p_j \\ &= 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j. \end{aligned}$$

If $\forall u_i + 1 \leq j \leq m, j \notin S_m$, i.e. none of these requests are served in the optimal offline solution, then $C_{\text{opt}}(m) = C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Therefore,

$$\begin{aligned} C_{\text{WOGI-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq 2\rho C_{\text{opt}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \leq 2\rho C_{\text{opt}}(m). \end{aligned}$$

If $\exists u_i + 1 \leq j \leq m$, such that $j \in S_m$, then $C_{\text{opt}}(m) \geq r_j \geq \tilde{C}_{\text{apx}}(u_i)$, where the last inequality is due to Step 2 in the WOGI-apx. Because of Step 1 of the algorithm, $\tilde{C}_{\text{apx}}(m) = \tilde{C}_{\text{apx}}(u_i) + \sum_{j=u_i+1}^{m} p_j \geq C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$, we have $\rho C_{\text{opt}}(m) \geq \tilde{C}_{\text{apx}}(m) \geq C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Therefore,

$$\begin{aligned} C_{\text{WOGI-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq (2 - 1/\rho)\tilde{C}_{\text{apx}}(u_i) + C_{\text{opt}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq (2 - 1/\rho)C_{\text{opt}}(m) + \rho C_{\text{opt}}(m) \\ &\leq \rho C_{\text{opt}}(m) + \rho C_{\text{opt}}(m) = 2\rho C_{\text{opt}}(m). \end{aligned}$$

From the discussion above, we can conclude that WOGI-apx is 2ρ-competitive.

□

2.3.3 Prize-collecting generalization

The prize collecting TSP (PCTSP) is a generalization of the TSP (see [16, 28]), where associated with each request is a penalty (if rejected) and a prize (if accepted and served). The server must collect enough prizes exceeding a given quota while minimizing the makespan needed to collect the prizes plus the total penalty of rejected requests. We consider the online version of this problem.

Online PCTSP

Instance: A metric space $\mathcal{M}$ with a given origin o and a distance metric $d(\cdot,\cdot)$. A series of n requests represented by quadruples $(l_i, r_i, p_i, w_i)_{1\le i\le n}$, where $l_i \in \mathcal{M}$ is the location (point in metric space) of request i, $r_i \in \mathbb{R}_+$ its release date (first time after which it can be served), $p_i \in \mathbb{R}_+$ its penalty (for not being served), and $w_i \in \mathbb{R}_+$ its prize (collected if served). A parameter $W_{min} \in \mathbb{R}_+$ (a quota for prizes to be collected). The problem begins at time 0; the server is initially idle at the origin (initial state), can travel at unit speed (when not idle), and eventually must be back and idle at the origin (final state). The earliest time the server reaches this final state is called the makespan.

Feasible solution: Any subset $\mathcal{S} \subset \{1,\dots,n\}$ of requests to be served and a feasible TSP tour with release dates $\tau(\mathcal{S})$ through $\mathcal{S}$ so that $\sum_{i\in\mathcal{S}} w_i \ge W_{min}$.

Offline context: The number of requests n is known to the offline server. All requests are revealed to the offline server at time 0.

Online context: The number of requests n is not known to the online server. Requests are revealed to the online server at their release dates $r_i \ge 0$; assume $r_1 \le r_2 \cdots \le r_n$. The online server can accept or reject a request any time after the request's release date.

Objective: In all cases, minimize {the makespan to serve all accepted requests plus the total penalties of all rejected requests} among all feasible solutions.

Assume there is a blackbox that provides the optimal offline solution for PCTSP. For the simplicity of the algorithm, let us assume that if there is no feasible solution for the offline problem, no request will be accepted. Let us replace the blackbox in WOGI by the blackbox for PCTSP; and the resulting algorithm is WOGI-PC.

Algorithm 3 (WOGI-PC)**.**

0. *Initialization: counter* $i = 0$*,* $u_0 = 0$*, and* $P_0 = \emptyset$*.*

1. *Assume k requests have been released. If* $\sum_{i=1}^{k} w_i \ge W_{min}$ *and* $S_k \subset \{1, 2, ..., u_i\}$*, wait for the next released request, and go to Step 1; otherwise, go to Step 2.*

2. *Assume k requests have been released, the WOGI server waits until* $\max\{C_{\text{opt}}(k), t_{k,u_i}\}$*, where* $t_{k,u_i} \doteq 2C_{\text{opt}}(k) - L(\tau_{k,u_i}) - \sum_{j>u_i, j\notin S_k} p_j - \sum_{j\in P_i} p_j$ *is the latest time to leave*

the origin to follow τ_{k,u_i} and maintain a competitive ratio of 2. If a new request is released during the waiting time, go to Step 1; otherwise, update $u_{i+1} = k, i = i + 1$, and go to Step 3.

3. *The WOGI server takes one of the two routes and ignores all new requests before reaching back the origin:*

   3a. *If $t_{u_i,u_{i-1}} \geq C_{\text{opt}}(u_i)$, he follows the WOGI shortcut $\tau_{u_i,u_{i-1}}$. After finishing the route, go to Step 4.*

   3b. *If $t_{u_i,u_{i-1}} < C_{\text{opt}}(u_i)$, he follows τ_{u_i}. After finishing the route, go to Step 4.*

4. *Update P_i. Go to Step 1.*

For simplicity, let us assume that there exists an feasible offline solution, i.e. $\sum_{i=1}^{n} w_i \geq W_{min}$. According to Step 2 and 3, it is easy to see that the WOGI-PC server leaves the origin at least once. During his first trip, the WOGI-PC server follows τ_{u_1} exactly. Since τ_{u_1} is a feasible solution, $\sum_{i \in S_{u_1}} w_i \geq W_{min}$, i.e. the online server collects enough prizes during his first trip. Therefore, the online solution is also a feasible solution.

Since in the proofs of Lemma 1, 2, 3 and Theorem 6, no property other than the optimality of the solution provided by the blackbox is used, analogs of these results are also for WOGI-PC. Please note that in the following statements and proofs, $C_{opt}(\cdot)$ and other notations correspond to the optimal solution obtained by the blackbox for offline PCTSP:

Lemma 7. *$\forall i \geq 1, r_{s_{i+1}} \geq C_{\text{opt}}(u_i)$.*

Proof. Let t be the time when the WOGI-PC server leaves the origin for the i^{th} time. According to Step 2, $t \geq C_{\text{opt}}(u_i)$. On the other hand, at time t, only u_i requests are released. Thus, $\forall j > u_i$, $r_j \geq t \geq C_{\text{opt}}(u_i)$. In particular, it is true for $j = s_{i+1} > u_i$. □

Lemma 8. *$\forall i \geq 1, t_{u_{i+1},u_i} \geq 2C_{\text{opt}}(u_i) - \sum_{j \in P_i} p_j$.*

Proof. The offline server cannot visit request s_{i+1} before its release time $r_{s_{i+1}}$. Thus, $T_{u_{i+1}} \geq r_{s_{i+1}} + L(\tau_{u_{i+1},u_i}) - |l_{s_{i+1}}|$. Therefore,

$$\begin{aligned}
t_{u_{i+1},u_i} &= 2C_{\text{opt}}(u_{i+1}) - L(\tau_{u_{i+1},u_i}) - \textstyle\sum_{j>u_i, j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&= 2T_{u_{i+1}} + 2\textstyle\sum_{1\le j\le u_{i+1}, j\notin S_{u_{i+1}}} p_j - L(\tau_{u_{i+1},u_i}) - \sum_{j>u_i, j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + 2L(\tau_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| + 2\textstyle\sum_{1\le j\le u_{i+1}, j\notin S_{u_{i+1}}} p_j - L(\tau_{u_{i+1},u_i}) \\
&\quad - \textstyle\sum_{j>u_i, j\notin S_{u_{i+1}}} p_j - \sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} + L(\tau_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| - \textstyle\sum_{j\in P_i} p_j \\
&\ge 2r_{s_{i+1}} - \textstyle\sum_{j\in P_i} p_j.
\end{aligned}$$

According to Lemma 7, $r_{s_{i+1}} \ge C_{\text{opt}}(u_i)$, we conclude $t_{u_{i+1},u_i} \ge 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. □

Lemma 9. *The WOGI-PC server returns to the origin for the i^{th} time before $2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$.*

Proof. We use induction on i to prove this lemma. It is trivially true for $i = 0$.

Consider i. By induction, the server finishes his $(i-1)^{\text{th}}$ trip before $2C_{\text{opt}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j$. According to Lemma 8, $2C_{\text{opt}}(u_{i-1}) - \sum_{j\in P_{i-1}} p_j \le t_{u_i,u_{i-1}}$. Thus, the WOGI-PC server is at the origin at $\max\{t_{u_i,u_{i-1}}, C_{\text{opt}}(u_i)\}$. According to Step 2, he leaves the origin for the i^{th} time at exactly $\max\{t_{u_i,u_{i-1}}, C_{\text{opt}}(u_i)\}$. Based on which of the two is larger, we have two cases:

1. If $t_{u_i,u_{i-1}} \ge C_{\text{opt}}(u_i)$, then the WOGI-PC server will take the shortcut τ_{u_{i+1},u_i}, and arrive at the origin at time $t_{u_i,u_{i-1}} + L(\tau_{u_{i+1},u_i}) = 2C_{\text{opt}}(u_i) - \sum_{u_{i-1}<j\le u_i, j\notin S_{u_i}} p_j - \sum_{j\in P_{i-1}} p_j \le 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset P_{i-1} \cup \{j : u_{i-1} < j \le u_i, j \notin S_{u_i}\}$.

2. If $t_{u_i,u_{i-1}} < C_{\text{opt}}(u_i)$, then the server will follow τ_{u_i}, and arrive at the origin at time $C_{\text{opt}}(u_i) + L(\tau_{u_i}) \le C_{\text{opt}}(u_i) + T_{u_i} = 2C_{\text{opt}}(u_i) - \sum_{j\notin S_{u_i}} p_j \le 2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$. The last inequality is due to $P_i \subset S^c_{u_i}$, because τ_{u_i} pass through all requests in S_{u_i}.

□

Theorem 8. *Algorithm WOGI-PC is 2-competitive for PCTSP.*

Proof. Assume there are a total of m requests and the WOGI-PC server leaves and returns to the origin i times. According to Lemma 9, after the i^{th} trip, the server returns to the origin before time $2C_{\text{opt}}(u_i) - \sum_{j\in P_i} p_j$ and never leaves again. Therefore, total cost

$C_{WOGI-PC} \le 2C_{\text{opt}}(u_i) - \sum_{j \in P_i} p_j + \sum_{j \in P_i} p_j + \sum_{j=u_i+1}^{m} p_j = 2C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Since the WOGI-PC server does not leaves the origin afterwards, $\forall u_i + 1 \le j \le m, j \notin S_m$. Thus, $C_{WOGI-PC} \le 2C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j = 2C_{\text{opt}}(m) - \sum_{j=u_i+1}^{m} p_j \le 2C_{\text{opt}}(m)$. □

Similarly, assume there is a blackbox that provides ρ-approximation offline solution for PCTSP. Replacing the blackbox in WOGI-apx by an approximation blackbox for PCTSP results in algorithm WOGI-PC-apx.

Algorithm 4 (WOGI-PC-apx)**.**

0. Initialization: counter $i = 0$, $u_0 = 0$, and $P_0 = \emptyset$.

1. Assume k requests have been released. If $\sum_{i=1}^{k} w_i \ge W_{min}$ and $\tilde{S}_k \subset \{1, .., u_i\}$, wait for the next released request, and go to Step 1; otherwise, go to Step 2.

2. Assume k requests have been released, the WOGI-PC-apx server waits until $\max\{\tilde{C}_{\text{apx}}(k), t_{k,u_i}\}$, *where $t_{k,u_i} \doteq 2\tilde{C}_{\text{apx}}(k) - L(\tilde{\tau}_{k,u_i}) - \sum_{j > u_i, j \notin \tilde{S}_k} p_j - \sum_{j \in P_i} p_j$. If a new request is released during the waiting time, go to Step 1; otherwise, update $u_{i+1} = k, i = i + 1$, and go to Step 3.*

3. The WOGI-PC-apx server takes one of the two routes and ignores all new requests before reaching back the origin:

   3a. If $t_{u_i,u_{i-1}} \ge \tilde{C}_{\text{apx}}(u_i)$, he follows the WOGI-PC-apx shortcut $\tilde{\tau}_{u_i,u_{i-1}}$. After finishing the route, go to Step 4.

   3b. If $t_{u_i,u_{i-1}} < \tilde{C}_{\text{apx}}(u_i)$, he follows $\tilde{\tau}_{u_i}$. After finishing the route, go to Step 4.

4. Update P_{i+1}. $i = i + 1$. Go to Step 1.

For simplicity, let us assume that there exists at least a feasible offline solution. Similar to the argument for the WOGI-PC algorithm, the WOGI-PC-apx algorithm also provides a feasible online solution.

Lemma 10. $\forall i \ge 1, r_{s_{i+1}} \ge \tilde{C}_{\text{apx}}(u_i)$.

Proof. Let t be the time when the WOGI-PC-apx server leaves the origin for the i^{th} time. According to Step 2, $t \ge \tilde{C}_{\text{apx}}(u_i)$. On the other hand, at time t, only u_i requests are released. Thus, $\forall j > u_i$, $r_j \ge t \ge \tilde{C}_{\text{apx}}(u_i)$. In particular, it is true for $j = s_{i+1} > u_i$. □

Lemma 11. $\forall i \geq 1, t_{u_{i+1},u_i} \geq 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j \in P_i} p_j$.

Proof. The offline server cannot visit request s_{i+1} before its release time $r_{s_{i+1}}$. Thus, $\tilde{T}_{u_{i+1}} \geq r_{s_{i+1}} + L(\tilde{\tau}_{u_{i+1},u_i}) - |l_{s_{i+1}}|$. Therefore,

$$\begin{aligned}
t_{u_{i+1},u_i} &= 2\tilde{C}_{\text{apx}}(u_{i+1}) - L(\tilde{\tau}_{u_{i+1},u_i}) - \textstyle\sum_{j>u_i, j \notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j \in P_i} p_j \\
&= 2\tilde{T}_{u_{i+1}} + 2\textstyle\sum_{1 \leq j \leq u_{i+1}, j \notin \tilde{S}_{u_{i+1}}} p_j - L(\tilde{\tau}_{u_{i+1},u_i}) - \sum_{j>u_i, j \notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j \in P_i} p_j \\
&\geq 2r_{s_{i+1}} + 2L(\tilde{\tau}_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| + 2\textstyle\sum_{1 \leq j \leq u_{i+1}, j \notin \tilde{S}_{u_{i+1}}} p_j - L(\tilde{\tau}_{u_{i+1},u_i}) \\
&\quad - \textstyle\sum_{j>u_i, j \notin \tilde{S}_{u_{i+1}}} p_j - \sum_{j \in P_i} p_j \\
&\geq 2r_{s_{i+1}} + L(\tilde{\tau}_{u_{i+1},u_i}) - 2|l_{s_{i+1}}| - \textstyle\sum_{j \in P_i} p_j \\
&\geq 2r_{s_{i+1}} - \textstyle\sum_{j \in P_i} p_j.
\end{aligned}$$

According to Lemma 11, $r_{s_{i+1}} \geq \tilde{C}_{apx}(u_i)$, we conclude $t_{u_{i+1},u_i} \geq 2\tilde{C}_{apx}(u_i) - \sum_{j \in P_i} p_j$. $\square$

Lemma 12. *The WOGI-PC-apx server finishes his* i^{th} *trip before* $2\tilde{C}_{\text{apx}}(u_i) - \sum_{j \in P_i} p_j$.

Proof. We use induction on i to prove this lemma. It is trivially true for $i = 0$.

Consider i. By induction, the server finishes his $(i-1)^{\text{th}}$ trip before $2\tilde{C}_{\text{apx}}(u_{i-1}) - \sum_{j \in P_{i-1}} p_j$. According to Lemma 11, $2\tilde{C}_{\text{apx}}(u_{i-1}) - \sum_{j \in P_{i-1}} p_j \leq t_{u_i,u_{i-1}}$. Thus, the WOGI-PC-apx server is at the origin at $\max\{t_{u_i,u_{i-1}}, \tilde{C}_{\text{apx}}(u_i)\}$. According to Step 2, he leaves the origin for the i^{th} time at exactly $\max\{t_{u_i,u_{i-1}}, \tilde{C}_{\text{apx}}(u_i)\}$. Based on which of the two is larger, we have two cases:

1. If $t_{u_i,u_{i-1}} \geq \tilde{C}_{\text{apx}}(u_i)$, then the WOGI-PC-apx server will take the shortcut $\tilde{\tau}_{u_{i+1},u_i}$, and arrive at the origin at time $t_{u_i,u_{i-1}} + L(\tilde{\tau}_{u_{i+1},u_i}) = 2\tilde{C}_{\text{apx}}(u_i) - \sum_{u_{i-1}<j \leq u_i, j \notin \tilde{S}_{u_i}} p_j - \sum_{j \in P_{i-1}} p_j \leq 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j \in P_i} p_j$. The last inequality is due to $P_i \subset P_{i-1} \cup \{j : u_{i-1} < j \leq u_i, j \notin \tilde{S}_{u_i}\}$.

2. If $t_{u_i,u_{i-1}} < \tilde{C}_{\text{apx}}(u_i)$, then the server will follow $\tilde{\tau}_{u_i}$, and arrive at the origin at time $\tilde{C}_{\text{apx}} + L(\tilde{\tau}_{u_i}) \leq \tilde{C}_{\text{apx}}(u_i) + \tilde{T}_{u_i} = 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j \notin \tilde{S}_{u_i}} p_j \leq 2\tilde{C}_{\text{apx}}(u_i) - \sum_{j \in P_i} p_j$. The last inequality is due to $P_i \subset \tilde{S}^c_{u_i}$, because τ_{u_i} pass through all requests in $\tilde{S}_{u_i}$.

$\square$

Theorem 9. *Algorithm WOGI-PC-apx is* 2ρ*-competitive for PCTSP.*

Proof. Assume there are a total of m requests, and the WOGI-PC-apx server leaves and returns to the origin i times. According to Lemma 12, after the i^{th} trip, the server returns to the origin before time $2\tilde{C}_{\text{apx}}(u_i) - \sum_{l \in P_i} p_l$ and never leaves again. Therefore, total cost

$$\begin{aligned} C_{\text{WOGI-PC-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) - \textstyle\sum_{j \in P_i} p_j + \sum_{j \in P_i} p_j + \sum_{j=u_i+1}^{m} p_j \\ &= 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j. \end{aligned}$$

If $\forall u_i + 1 \leq j \leq m, j \notin S_m$, i.e. none of these requests are served in the optimal offline solution, then $C_{\text{opt}}(m) = C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Therefore,

$$\begin{aligned} C_{\text{WOGI-PC-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq 2\rho C_{\text{opt}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \leq 2\rho C_{\text{opt}}(m). \end{aligned}$$

If $\exists u_i + 1 \leq j \leq m$, such that $j \in S_m$, then $C_{\text{opt}}(m) \geq r_j \geq \tilde{C}_{\text{apx}}(u_i)$, where the last inequality is due to Step 2 in the WOGI-PC-apx. Because of Step 1 of the algorithm, $\tilde{C}_{\text{apx}}(m) = \tilde{C}_{\text{apx}}(u_i) + \sum_{j=u_i+1}^{m} p_j \geq C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$, we have $\rho C_{\text{opt}}(m) \geq \tilde{C}_{\text{apx}}(m) \geq C_{\text{opt}}(u_i) + \sum_{j=u_i+1}^{m} p_j$. Therefore,

$$\begin{aligned} C_{\text{WOGI-PC-apx}} &\leq 2\tilde{C}_{\text{apx}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq (2 - 1/\rho)\tilde{C}_{\text{apx}}(u_i) + C_{\text{opt}}(u_i) + \textstyle\sum_{j=u_i+1}^{m} p_j \\ &\leq (2 - 1/\rho)C_{\text{opt}}(m) + \rho C_{\text{opt}}(m) \\ &\leq \rho C_{\text{opt}}(m) + \rho C_{\text{opt}}(m) = 2\rho C_{\text{opt}}(m). \end{aligned}$$

From the discussion above, we can conclude that WOGI-PC-apx is 2ρ-competitive.

□

2.3.4 Multi-server generalization

The multi-server version of the problem is almost the same, except that there are multiple servers and the final state is when all servers are idle at the origin:

Online multi-server TSP with Rejection Options

Instance: A metric space $\mathcal{M}$ with a given origin o and a distance metric $d(\cdot,\cdot)$. A series of n requests represented by triples $(l_i, r_i, p_i)_{1\leq i\leq n}$, where $l_i \in \mathcal{M}$ is the location (point in metric space) of request i, $r_i \in \mathbb{R}_+$ is its release date (first time after which it can be served), and $p_i \in \mathbb{R}_+$ is its penalty (for not being served). The problem begins at time 0; k servers are initially idle at the origin (initial state), can travel at unit speed (when not idle), and eventually must be back and idle at the origin (final state). The earliest time the server reaches this final state is called the makespan.

Offline context: The number of requests n is known to the offline server. All requests are revealed to the offline servers at time 0.

Online context: The number of requests n is not known to the online server. Requests are revealed to the online servers at their release dates $r_i \geq 0$; assume $r_1 \leq r_2 \cdots \leq r_n$. There are two online versions:
Objective: In all cases, minimize {the makespan to serve all accepted requests plus the total penalties of all rejected requests} among all feasible solutions.

Therkelsen [67] shows that multi-WOGI, a multi-server version of the WOGI algorithm is 2-competitive.

2.4 The Real-time Version

In this section, we consider the real-time version of our online problem. The decision to accept or reject a given request must be made immediately upon its arrival.

2.4.1 The case of the non-negative real-line $\mathbb{R}_+$

We first study the problem when the locations of the requests are all on the non-negative real line, equipped with the traditional Euclidean distance. In that case, the notation for the location of a request i, l_i, will also represent the distance from the origin to the point.

Lower bound on competitive ratios

Theorem 10. *Any c-competitive online algorithm on $\mathbb{R}_+$ has $c \geq 2.5$.*

Before proving the theorem, let us first present how we construct instances for Theorem 10. The same idea is used for Theorem 12 later. A series of requests with small penalties with the same location and almost the same release date are released until the online algorithm accept one such request. The online algorithm then faces a dilemma on whether to accept or reject such requests. On the one hand, accepting one such request too early may not be beneficial, because the offline solution only pays a small amount of penalties while the online solution must spend some time to visit the accepted request; on the other hand, accepting one too late may not be beneficial either, because the offline solution can simply accept and visit all these requests while the online solution must visit the accepted request and pay a large amount of penalties for rejected ones. Such a dilemma leads to large competitive ratios. In the proofs of Theorem 10 and Theorem 12, penalties, locations and release dates are chosen carefully to take advantage of the dilemma.

Proof of Theorem 10. Assume that an online server follows a c-competitive online algorithm, where c is a finite constant. Let $c_0 = 2.5$ and ϵ be a small positive number. For an arbitrary integer n, consider a series of up to $n+1$ requests as follows: $(l_i, r_i, p_i) = (1, 1+i\epsilon, 3/c_0^i)$ for $1 \le i \le n$, and $(l_{n+1}, r_{n+1}, p_{n+1}) = (1, 1+(n+1)\epsilon, \infty)$.

Let t_0 be the time when the online server begins to move away from the origin for the first time. If $0 \le t_0 < r_1$, no request would be presented. As a result, $C_A > 0$ and $C_{\text{opt}} = 0$, which contradicts with the assumption that algorithm A is finite competitive. Thus, $t_0 \ge r_1$. Since the last possible request has an infinite penalty, any finite competitive online algorithm cannot reject all requests. Let request $m (1 \le m \le n+1)$ be the first request that is accepted by A. After the online server accepts request m, no more request is presented. We now consider two cases to compute the competitive ratio:

1. If $m \le n$, then the optimal solution is to reject all m requests. In this case, $C_{\text{opt}}(m) = \sum_{i=1}^{m} p_i = \frac{3(c_0^m - 1)}{(c_0-1)c_0^m}$, and $C_A(m) \ge r_1 + 2l_m + \sum_{i=1}^{m-1} p_i \ge \frac{3(c_0^m-1)}{(c_0-1)c_0^{m-1}}$. Thus, $c \ge \frac{C_A(m)}{C_{\text{opt}}(m)} = c_0 = 2.5$.

2. If $m = n+1$, then the optimal solution is to accept all $n+1$ requests. In this case, $C_{\text{opt}}(n+1) = r_{n+1} + l_{n+1} = 2 + (n+1)\epsilon$, and $C_A(n+1) \ge r_1 + 2l_{n+1} + \sum_{i=1}^{n} p_i = \epsilon + \frac{3(c_0^{n+1}-1)}{(c_0-1)c_0^n}$. Using the fact that $c_0 = 2.5$, $\epsilon + \frac{3(c_0^{n+1}-1)}{(c_0-1)c_0^n} = \epsilon + 5 - 2/c_0^n$. Thus, $c \ge \frac{C_A(n+1)}{C_{\text{opt}}(n+1)} \ge \frac{5-2/c_0^n+\epsilon}{2+(n+1)\epsilon}$. By letting $\epsilon = 1/(n+1)^2$, and $n \to +\infty$, we have $\frac{5-2/c_0^n+\epsilon}{2+(n+1)\epsilon} \to 2.5$. Thus, $c \ge 2.5$.

□

An optimal 2.5-competitive online algorithm

The algorithm we propose here is an extension of the "move right if necessary" (MRIN) algorithm introduced in Blom et al. [21] for the online TSP without rejection options. The acceptance/rejection decisions are based on the offline optimal solutions. Let us call this algorithm the "estimate and move right if necessary" (EMRIN) algorithm:

Algorithm 5 (EMRIN)**.**

1. *Whenever a new request m comes, if $m \in S_m$, accept it; otherwise reject it.*

2. *If there is an accepted and unserved request on the right side of the server, move toward it.*

3. *If there are no accepted and unserved requests on the right side of the server, move back toward the origin. Upon reaching the origin, become idle.*

First, let us show that the total penalty of rejected requests is not large:

Lemma 13. *$\forall m$, $C_{\text{opt}}(m) \geq \sum_{i \leq m, i \notin S_i} p_i$.*

Proof. We use induction on m. When $m = 1$, if $1 \in S_1$, $C_{\text{opt}}(1) \geq 0 = \sum_{i \leq 1, i \notin S_i} p_i$; otherwise, $C_{\text{opt}}(1) = p_1 = \sum_{i \leq 1, i \notin S_i} p_i$. Therefore, the assertion is true. Assume now that the assertion holds for $m-1$, and let us consider m. If $m \in S_m$, $C_{\text{opt}}(m) \geq C_{\text{opt}}(m-1) \geq \sum_{i \leq m-1, i \notin S_i} p_i = \sum_{i \leq m, i \notin S_i} p_i$; otherwise, $C_{\text{opt}}(m) = C_{\text{opt}}(m-1) + p_n \geq \sum_{i \leq m-1, i \notin S_i} p_i + p_m = \sum_{i \leq m, i \notin S_i} p_i$. □

Then, let us prove our main result:

Theorem 11. *Algorithm 5 (EMRIN) is 2.5-competitive, and thus best possible.*

Proof. We will use induction on the number of released requests m. When $m = 1$, if $1 \notin S_1$, $C_A(1) = C_{\text{opt}}(1)$; otherwise, $C_A(1) = r_1 + 2l_1$, $C_{\text{opt}}(1) = \max\{r_1 + l_1, 2l_1\}$, and thus $\frac{C_A(1)}{C_{\text{opt}}(1)} \leq 1.5$. Assume now that the assertion holds for $m-1$, and let us consider m:

1. If $m \notin S_m$, $C_A(m) = C_A(m-1) + p_m \leq 2.5C_{\text{opt}}(m-1) + p_m \leq 2.5(C_{\text{opt}}(m-1) + p_m) = 2.5C_{\text{opt}}(m)$. Thus, $c \leq 2.5$.

2. If $m \in S_m$, assume request k is the rightmost request that is accepted but not be served at time r_n; assume x is the position of the online server at r_n.

   2a. If $x \leq l_k$, because request k is an accepted and unserved request, the online server has been moving right since time r_k. Hence, the online server will return to the origin at later than $r_k + 2l_k$. Therefore, $C_A(m) \leq r_k + 2l_k + \sum_{i \leq m, i \notin S_i} p_i$. Because $k \in S_k$, $C_{\text{opt}}(m) \geq C_{\text{opt}}(k) \geq \max\{r_k + l_k, 2l_k\}$. As a result, we conclude $\frac{C_A(m)}{C_{\text{opt}}(m)} \leq \frac{r_k + l_k}{C_{\text{opt}}(k)} + \frac{l_k}{C_{\text{opt}}(k)} + \frac{\sum_{i \leq m, i \notin S_i} p_i}{C_{\text{opt}}(m)} \leq 1 + 0.5 + 1 = 2.5$.

   2b. If $x \geq l_k$, no extra time is needed to serve the last request, because it can be served on the online's way back to the origin. Thus, $C_A(m) = C_A(m-1) \leq 2.5C_{\text{opt}}(m-1) \leq 2.5C_{\text{opt}}(m)$.

□

2.4.2 The case of the real-line $\mathbb{R}$

In this section, we study the problem when the locations of the requests are on the real line, equipped with the traditional Euclidean distance. On the positive ("right") side of the line, the notation for the location of a request i, l_i, will also represent the distance from the origin to the point. On the negative ("left") side of the line, the location of a request i, l_i will be given by a negative number, and the distance from the origin to the point will be its absolute value $|l_i|$.

Lower bounds on competitive ratios

General lower bound

Theorem 12. *Any c-competitive online algorithm on $\mathbb{R}$ has $c \geq \frac{17+\sqrt{17}}{8} \approx 2.64$.*

Proof. Assume that an online server A follows a given c-competitive online algorithm. For any given $\varepsilon > 0$, there exists $N \in \mathbb{N}$, such that $N\varepsilon > c(4 + 2\varepsilon)$, and there also exists $M \in \mathbb{N}$, such that $15c - 8 < \varepsilon c^{M-1}$.

Again we use the idea mentioned in Section 2.4.1 to construct an example. In this example, three series of requests are presented.

First, consider a series of up to N requests as follows: $(l_i, r_i, p_i) = (1, 1 + \frac{i\varepsilon}{N}, \varepsilon)$ for $1 \leq i \leq N$. Note that A cannot reject them all. Otherwise, the cost will be $N\varepsilon > c(4 + 2\varepsilon)$, while the optimal cost is at most $2+\varepsilon$, which is a contradiction. Assume the first request accepted by A is n_1. Truncate the first series: only the first n_1 requests are presented.

Consider a second series of up to N requests as follows: $(l_{n_1+i}, r_{n_1+i}, p_{n_1+i}) = (-1, 1+\varepsilon+\frac{i\varepsilon}{N}, \varepsilon)$, for $1 \le i \le N$. Note that A cannot reject them all. Otherwise, the cost will be at least $N\varepsilon > c(4+2\varepsilon)$, while the optimal cost is at most $4+2\varepsilon$, which is a contradiction. Assume the first request in this second series accepted by A is n_1+n_2. Truncate the second series: only the first n_2 requests in the second series are presented. Let $P_1 = (n_1-1)\varepsilon$, $P_2 = (n_2-1)\varepsilon$, $a_1 = \min\{2+2\varepsilon, P_1+\varepsilon\}$, and $a_2 = \min\{2+2\varepsilon, P_2+\varepsilon\}$.

At time $1+\varepsilon+\frac{n_2\varepsilon}{N}$, A has two requests at ± 1 to visit. Without loss of generality, assume A is visiting 1 before visiting -1. Assume t_0 is the first time when A is at the origin after visiting 1. Note that the optimal cost is at most a_1+a_2. In order to be c-competitive, we have $3 \le t_0 \le (c-1)(a_1+a_2)-2 \le 5c-2$.

Consider a third series of up to M requests as follows: $l_{n_1+n_2+i} = t_0-2$, $r_{n_1+n_2+i} = t_0+\frac{i\varepsilon}{M+1}$, $p_{n_1+n_2+i} = \max\{0, \frac{3t_0-2-(c-1)(P_1+P_2)-(2c+1)\varepsilon}{c^i}\}$, for $1 \le i \le M$. We claim that A has to reject all these M requests. Otherwise, assume the first one accepted is n_1+n_2+m; then truncate the third series so that only the first m requests in the third series are presented. The online server's cost is at least $3t_0-2+P_1+P_2+\sum_{i=1}^{m-1} p_{n_1+n_2+i} \ge c(a_1+a_2+\sum_{i=1}^{m} p_{n_1+n_2+i})+\varepsilon$, while optimal cost is at most $a_1+a_2+\sum_{i=1}^{m} p_{n_1+n_2+i}$, which is a contradiction.

Then, we present the last request $(l_{n_1+n_2+M+1}, r_{M+1+n_1+n_2}, p_{M+1+n_1+n_2}) = (t_0-2, t_0+\varepsilon, \infty)$. Because of its infinite penalty, A has to accept this request. Thus, online server's cost is at least $3t_0-2+P_1+P_2+\sum_{i=1}^{M} p_{i+n_1+n_2}$. Noting that $\sum_{i=1}^{M} p_{i+n_1+n_2} = p_{1+n_1+n_2}\sum_{i=1}^{M}\frac{1}{c^{i-1}} = \frac{c}{c-1}p_{1+n_1+n_2} - \frac{p_{1+n_1+n_2}}{c^M-c^{M-1}}$, $p_{1+n_1+n_2} \le 3t_0-2 \le 15c-8$, $c^M-c^{M-1} > c^{M-1}$, and $15c-8 < c^{M-1}\varepsilon$, we then have $\ge 3t_0-2+P_1+P_2+\frac{3t_0-2-(2c+1)\varepsilon}{c-1}-(P_1+P_2)-\varepsilon = \frac{c(3t_0-2)}{c-1}-\frac{3c\varepsilon}{c-1}$. The optimal cost is at most $2t_0-2+\varepsilon$. Consequently, $c\cdot(2t_0-2+\varepsilon) \ge \frac{c(3t_0-2)}{c-1}-\frac{3c\varepsilon}{c-1} \Rightarrow 2c-4-(c+2)\varepsilon \le (2c-5)t_0 \le (2c-5)((c-1)(a_1+a_2)-2) \le (2c-5)(4c+(4c_4)\varepsilon)-6$. By letting $\varepsilon \to 0$, we conclude $2c-4 \le (2c-5)(4c-6) \Rightarrow c \ge \frac{17+\sqrt{17}}{8}$. □

Lower bound on a restricted family of online algorithms

Theorem 13. *Any c-competitive online algorithm on $\mathbb{R}$ that accepts request i if and only if $i \in S_i$ has $c \ge \frac{\sqrt{33}+27}{12} \approx 2.73$.*

Proof. Assume that an online server follows a c-competitive online algorithm A that accepts request i if and only if $i \in S_i$. Let ε be an arbitrary small positive number. Consider a series of up to 6 requests as follows: $(l, r, p) = (0, 1, 1-\varepsilon), (0, 1+\varepsilon, \infty), (-1, 1+2\varepsilon, 1), (1, 1+3\varepsilon, 1), (-1, 1+4\varepsilon, 4\varepsilon), (1, 1+5\varepsilon, 6\varepsilon)$. The online server will accept the 2nd, 5th and 6th request, and he has to visit ± 1. Without loss of generality, assume the server visits 1 before visiting -1, and time $3+a$ is the first time that he is at the origin after visiting 1. It is easy to see $a \geq 0$. If no more request is presented, the online cost is at least $8+a$ and the optimal cost is 3. Then, if necessary, present two more requests at $1+a$, $(l, r, p) = (1+a, 3+a, 1+a-100\varepsilon), (1+a, 3+a+\varepsilon, 1000\varepsilon)$. The online server will accept the 8th and reject the 7th request. So, the online cost will be at least $11+5a$, while the optimal cost will be $4+2a$. Consequently, competitive ratio $c \geq \max\{\frac{11+5a}{4+2a}, \frac{8+a}{3}\} \geq \frac{\sqrt{33}+27}{12} \approx 2.73$. □

A 3-competitive online algorithm

This algorithm uses a different offline subroutine, hereafter called black box 2, that solves a variant of the offline problem where the server starts initially at a point x that may be different from the origin.

Algorithm 6 (ReOpt)**.**

1. *Whenever a new request m comes, if $m \in S_m$, then accept it; otherwise, reject it.*
2. *At any time when a new request is accepted, reoptimize (using black box 2) and follow the corresponding new optimal route to serve all accepted and unserved requests.*

First, let us show that the total penalty of rejected requests is not very large. Consider if only the first k requests are released, let $L_k = \min_{1\leq i\leq k, i\in S_k}\{l_i, 0\}$ be the leftmost accepted request and $R_k = \max_{1\leq i\leq k, i\in S_k}\{l_i, 0\}$ be the rightmost accepted request. Then,

Lemma 14. *If $k \in S_k$ and $l_k < 0$, then $\forall l \in (l_k, 0]$, $\sum_{i:l_i<l} p_i \geq l - l_k$.*

Proof. Consider the route τ_k that the offline server follows if only the first k requests are released. Let $\tau_k(t)$ be the server's position at time t. Assume that request k is served at time $t_0 (\geq r_k)$. Let $t_1 = \max\{t : \tau_k(t) = l, t < t_0\}$ and $t_2 = \min\{t : \tau_k(t) = l, t > t_0\}$. Consider another feasible solution:

$$\tau'_k(t) = \begin{cases} \tau_k(t), & t \leq t_1 \\ l, & t_1 < t \leq t_0 \\ \tau_k(t-(t_2-t_0)), & t \geq t_0 \end{cases} .$$

Since both solutions have the same motion before t_1, all requests served by τ_k before t_1 are also served by the new solution. Furthermore, because the interval covered by τ_k after t_2 is also covered by τ'_k after t_0 and all the first k requests are released before t_0, every request that is served by τ_k after t_2 will be served by τ'_k. Therefore, all requests served by τ_k but not by τ'_k are served by τ_k between t_1 and t_2. Because of the definition of t_1 and t_2, all those requests are located beyond l. Thus, τ_k saves at most $\sum_{i:l_i<l} p_i$ on penalties and spends $t_2 - t_0$ more units of time on traveling. From τ_k, we have $\sum_{i:l_i<l} p_i \geq t_2 - t_0 \geq l - l_k$, where the last inequality is due to the unit speed of the offline server. $\square$

By symmetry, we also have:

Lemma 15. *If $k \in S_k$ and $l_k > 0$, then $\forall l \in [0.l_k)$, $\sum_{i:l_i>l} p_i \geq l_k - l$.*

Then we are ready to prove the competitive ratio of ReOpt:

Theorem 14. *Algorithm 6 (ReOpt) is 3-competitive.*

Proof. Assume n requests are released:

1. If $n \in S_n$. Both the online and offline servers accept request n. Let $L_{\text{on}} = \min\{0, l_i : i \in S_i\}$ be the leftmost request accepted by the online server, $R_{\text{on}} = \max\{0, l_i : i \in S_i\}$ be the rightmost request accepted by the online server, $L_{\text{off}} = \min\{0, l_i : i \in S_n\}$ be the leftmost request accepted by the offline server, and $R_{\text{off}} = \max\{0, l_i : i \in S_n\}$ be the rightmost request accepted by the offline server. From the description of ReOpt, the online server never moves beyond interval $[L_{on}, R_{on}]$. Therefore, the online server serves all accepted requests and returns to the origin no later than $r_n - 2L_{\text{on}} + 2R_{\text{on}}$. Thus,

$$\begin{aligned}
C_{\text{ReOpt}}(n) &\leq r_n - 2L_{\text{on}} + 2R_{\text{on}} + \textstyle\sum_{i\leq n, i\notin S_i} p_i \\
&\leq r_n + (-2L_{\text{off}} + 2\textstyle\sum_{i:l_i<L_{\text{off}}} p_i) + (2R_{\text{off}} + 2\textstyle\sum_{i:l_i>R_{\text{off}}}) + \textstyle\sum_{i\leq n, i\notin S_i} p_i \\
&\leq (r_n + \textstyle\sum_{i\notin S_n} p_i) + (2R_{\text{off}} - 2L_{\text{off}} + \textstyle\sum_{i\notin S_n} p_i) + \textstyle\sum_{i\leq n, i\notin S_i} p_i \\
&\leq 3C_{\text{opt}}(n).
\end{aligned}$$

2. If $\forall i \in \{1, 2, \cdots, n\}, i \notin S_i$, then $C_{\text{ReOpt}}(n) = \sum_{i=1}^{n} p_i = C_{\text{opt}}(n)$.

3. Assume m is the last request such that $m \in S_m$. According to Case 1, $C_{\text{ReOpt}}(m) \leq 3C_{\text{opt}}(m)$. So, $C_{ReOpt}(n) = C_{ReOpt}(m) + \sum_{i=m+1}^{n} p_i \leq 3C_{\text{opt}}(m) + \sum_{i=m+1}^{n} p_i \leq 3(C_{\text{opt}}(m) + \sum_{i=m+1}^{n} p_i) = 3C_{\text{opt}}(n)$.

□

Note that 3 is a tight competitive ratio for ReOpt as the following example illustrates. Let ϵ be an arbitrarily small positive number, k be an arbitrary large integer, and let the instance consist of the following $2k+3$ requests: $(l_1, r_1, p_1) = (1, 0, 2 - 1/k)$, $(l_2, r_2, p_2) = (-2/k, \epsilon/k, \infty)$, $(l_3, r_3, p_3) = (1, 1/k, \infty)$, $(l_i, r_i, p_i) = (-2/k, (i-1)/k, \infty)$, $4 \leq i \leq 2k+3$. It is easy to check that $C_{ReOpt} = 6 + 4/k$ and $C_{\text{opt}} = 2 + 4/k$. By letting $k \to \infty$, we have $c \geq 3$.

2.4.3 The case of general metric spaces

In this subsection, we first construct a series of special metric spaces, for which we prove that there are no online algorithms with a constant competitive ratio, and show a $\Omega(\sqrt{\ln n})$ lower bound on any competitive ratios (where n is the number of requests in the given instance of the problem). Then, among the restricted class of online algorithms with prior knowledge about the total number of requests n, we propose one which is $O(\sqrt{\ln n})$-competitive; hence, asymptotically best possible among that class.

Lower bound on competitive ratios

Splitting operation: Such an operation on an edge AB of length l consists in splitting it into countably infinite many copies, each represented by a middle points $\{C_i\}_{i \in \mathbb{N}}$ such that: each C_i satisfies: $AC_i = BC_i = l/2$; and any path from one middle point C_{i_1} to another middle point C_{i_2} must pass through either A or B.

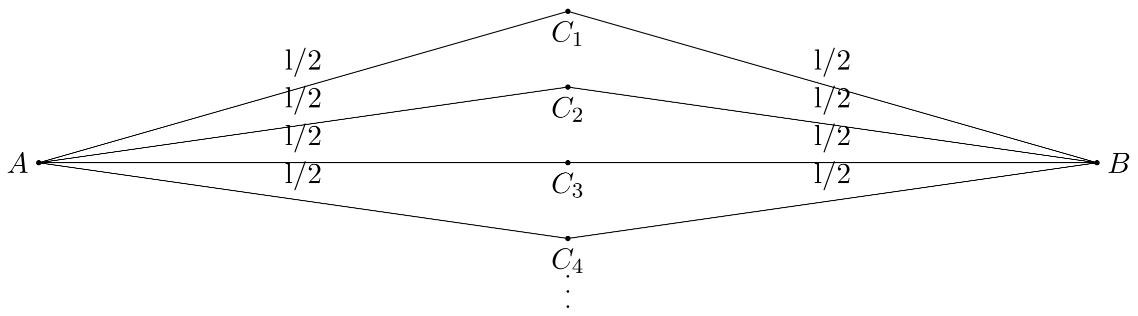


Figure 2-2: Splitting Operation

The metric spaces $\{\mathcal{M}_j\}_{j \in \mathbb{N}}$: The spaces $\{\mathcal{M}_j\}_{j \in \mathbb{N}}$ are created iteratively by the splitting operation described above. Given one space $\mathcal{M}_j$, we split each of its edge into countably infinite many copies to create $\mathcal{M}_{j+1}$:

- $\mathcal{M}_0$: A line segment A_0A_1 of length 1.
- $\mathcal{M}_1$: Split A_0A_1 into copies with middle points $A_{1/2,i_1}$.
- $\mathcal{M}_2$:
  - For every i_1, split $A_0A_{1/2,i_1}$ into copies with middle points $A_{1/4,i_1i_2}$ (Here the part before the comma indicates the point's distance from A_0. The part after the comma indicates its location, e.g. $A_{1/4,11}$ is a middle point of $A_0A_{1/2,1}$, but $d(A_{1/4,11}, A_{1/2,2}) \neq 1/4$.);
  - For every i_1, split $A_{1/2,i_1}A_1$ into copies with middle points $A_{3/4,i_1i_2}$;
- etc, ...

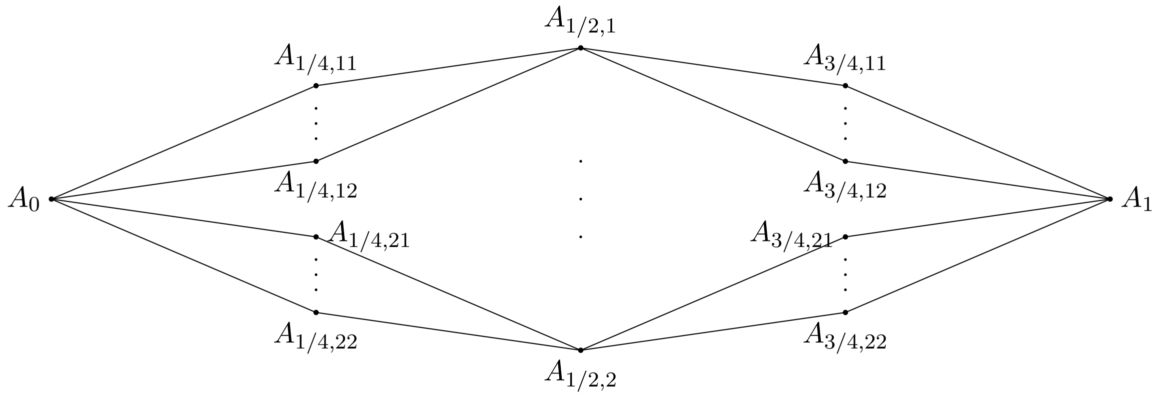


Figure 2-3: An illustration of $\mathcal{M}_2$

A point is called α-point if its distance from A_0 is α. For instance, $A_{1/2,i_1}$ are all 1/2-points and $A_{3/4,i_1i_2}$ are all 3/4-points. Let $V_0 = \{A_0, A_1\}$ and V_j be the set of middle points created when creating the space $\mathcal{M}_j$. For example, $V_2 = \{A_{1/4,i_1i_2}, A_{3/4,i_1i_2} | \forall i_1, i_2\}$. By the construction of the spaces and the splitting operations, the distance between two nearest points in V_j is $1/2^{j-1}$, and the distance between V_j and V_{j-1} is $1/2^j$.

Proof of unbounded competitive ratio: In order to present requests one by one to the online server, the release dates of all successive requests should be different. However, for simplicity in the exposition of our proofs, all release dates are set to be 0. Simply assume that the online server is given each request one by one, and has to make an acceptance/rejection decision before the next one is revealed. (Formally one could instead assume that for all $i \geq 1$, $r_i = i\epsilon$, where ϵ is an arbitrarily small positive number.)

Theorem 15. *For any $m \in \mathcal{N}$, there is no algorithm with a competitive ratio less than $m-1$.*

Proof. Consider the following instance defined on the metric space $\mathcal{M}_{2m^2}$. The online server is at A_0 initially. First, two requests with infinite penalties are released at A_0 and A_1. Then requests with penalties 1/m are released one by one at 1/2-points $\{A_{1/2,i_1}\}$ until the online server rejects one of them. We will later show that this is well defined, i.e. the online server must reject one such request. Assume he accepts $A_{1/2,1}, \cdots, A_{1/2,a_1-1}$ and rejects $A_{1/2,a_1}$. Then, release requests with penalties $1/(2m)$ at 1/4-th points $A_{1/4,a_1 i_2}$ until the online server rejects one (again our formal proof shows that this is a well defined stopping criteria), and at 3/4-th point $A_{3/4,a_1 i_2}$ until the online server rejects one. Repeat this procedure at $1/8, 3/8, 5/8, 7/8, ..., 1/2^{2m^2}, ..., (2^{2m^2}-1)/2^{2m^2}$-points. The penalty of a request at a point in V_j is $1/(2^{j-1}m)$. First, let us show that requests accepted by the online server are not close to each other:

Lemma 16. *Consider any two requests that are accepted by the online server. Assume that one request is at a point in V_{j_1} and the other is at a point to V_{j_2} (j_1 and j_2 may or may not be different). Then, the distance between those two requests is at least $1/2^{j_1} + 1/2^{j_2}$.*

Proof. Because of the way the instance and the metric spaces are constructed, any path from one request to the other must pass through a point in set $V_{\min\{j_1,j_2\}-1}$. Since the distance between V_{j_1} and $V_{\min\{j_1,j_2\}-1}$ is $1/2^{j_1}$ and the distance between V_{j_2} and $V_{\min\{j_1,j_2\}-1}$ is $1/2^{j_2}$, the distance between these two requests is at least $1/2^{j_1} + 1/2^{j_2}$. □

Lemma 16, combined with the fact that any request at a point in V_j has a penalty $1/(2^{j-1}m)$, indicates that the distance between two requests accepted by the online server is at least $m/2$ times the sum of the penalties of the two requests. Therefore, if the total penalty of requests accepted by the online server is P, the online server must travel at least mP units of time to serve all of them. Then, let us show that the instance is well defined, i.e. the online server cannot accept all requests:

Lemma 17. *Assume the online algorithm is $(m-1)$-competitive. For any $1 \leq j \leq 2m^2$, let k_j be the total number of requests at points in V_j that are accepted by the online server. Then, $k_j \leq m^2 2^j$.*

Proof. Assume there exists j such that $k_j > m^2 2^j$. Consider the instance in which no request at points in $\bigcup_{i=j+1}^{2m^2} V_i$ is presented. We will show that the algorithm is worse than $(m-1)$-competitive for the instance.

First, let us consider the cost of the online server. Since the total penalty of accepted requests is $\sum_{i=1}^{j} k_i/(2^{i-1}m)$, from the discussion above, the online server must spend at least $\sum_{i=1}^{j} k_i/2^i$ units of time serving these requests. Because for each $1 \leq i \leq j$ the total penalty of rejected requests at points in V_i is $1/m$, the total penalty of rejected requests is j/m. Thus, the online cost is $\sum_{i=1}^{j} k_i/2^{i-1} + j/m$.

Then, let us consider a feasible solution B and its cost. Requests rejected by the online server are accepted and requests accepted by the online server are rejected. Noting a carefully chosen shortest path from A_0 to A_1 passes through all requests rejected by the online server, the new feasible solution spends 2 units of time to visit these rejected requests. Thus, $C_B = \sum_{i=1}^{j} k_i/(2^{i-1}m) + 2$.

Since the online algorithm is $(m-1)$-competitive, $C_{online} \leq (m-1)C_{OPT} \leq (m-1)C_B$. Thus, $k_j/(2^{j-1}m) \leq \sum_{i=1}^{j} k_i/(2^{i-1}m) + j/m \leq 2(m-1) < 2m$, which implies $k_j \leq m^2 2^j$. Contradict with our assumption. $\square$

We are now ready to finish the prove Theorem 15. According to the proof of Lemma 17, the online cost is at least $\sum_{i=1}^{2m^2} k_j/2^{i-1} + 2m$; there exists a feasible solution B whose cost C_B is $\sum_{i=1}^{2m^2} k_i/(2^{i-1}m) + 2$. However, $C_{online} \geq mC_B \geq mC_{OPT}$. Thus, the online algorithm is not $(m-1)$-competitive. $\square$

Asymptotic lower bound: From the detailed proof of Theorem 15, at most $m^2 2^j$ requests are presented at points in V_j. At most $n \leq m^2 2^{2m^2}$ requests make any online algorithm worse than $(m-1)$-competitive. In other words, we have showed:

Theorem 16. *Any c-competitive online algorithm on an instance with n requests must have $c \geq \Omega(\sqrt{\ln n})$, even assuming n is given in advance.*

Best possible $O(\sqrt{\ln n})$-competitive algorithm

The algorithm proposed in this section requires a priori knowledge on the total number of requests. It is not clear that there exists an algorithm that achieves the same competitive ratio without such a knowledge.

For the simplicity of the algorithm and its analysis, we would like to consider problems without release dates. All requests arrive in sequence. The online algorithm must accept or reject a request before seeing the next. After making all the decisions, the online algorithm then decides how to serve all accepted requests. We argue that removing release dates (but preserving the order) only changes competitive ratios by at most 2:

Lemma 18. *Given an online algorithm A that is designed to solve the instances with all release dates 0(but still have to make decisions sequentially before knowing future requests), it can be transformed to another online algorithm A', such that for any instance I, $\dfrac{C_{A'}(I)}{C_{OPT}(I)} \leq \dfrac{C_A(I')}{C_{OPT}(I')} + 2$, where I' is almost the same instance as I, the only difference is all release dates are 0 in I', but the order of requests remains.*

Proof. Consider the following algorithm A':

> Whenever a new request m comes, the server applies algorithm A on the instance consisting of first m requests with release dates all zeros, to make an accept/reject decision for the new request. If he accepts the new one, he goes back to the origin and then follows the newly computed route; otherwise, he just continues his current route.

Assume for instance I, $\rho = \frac{C_{A'}(I)}{C_{OPT}(I)}$ and the last request accepted by offline server is request m. According to the construction, the online server will go back to the origin at time r_m and then follows his last route. Note that at time r_m, the server is at most r_m units of distance away from the origin, thus, $C_{A'}(I) \leq r_m + r_m + C_A(I') \leq 2C_{OPT}(I) + \rho C_{OPT}(I') \leq (2+\rho)C_{OPT}(I)$. □

As mentioned in Section 2.4.1, accepting a request at a faraway location with a small penalty may not be beneficial; however, if many requests with small penalties are close to each other, it may be beneficial to accept them. Let us first define the concept of distance for a group of requests:

Definition 1. *Given a set U of requests and a route τ, define $T(U, \tau)$ as the shortest time to visit all requests in U along τ if τ passes all requests in U; otherwise, define as ∞.*

According to the definition, a server can begin at any request in U, visit all the other requests in U, and then go back to its initial position within $2T(U, \tau)$ units of time. A group of requests that are close and have a large overall penalty may be beneficial to visit. More formally,

Definition 2. *If τ passes all requests in U, given any nonempty subset of U: $\{i_1, i_2, ..., i_k\}$, there exists a division of $U = \bigcup_{l=1}^{k} I_l$, such that $i_{k_l} \in I_l, \forall l = 1, ..., k$, and $\sum_{l=1}^{k} T(I_l, \tau) \leq \sqrt{\ln n}(\sum_{i \in U} p_i - \sum_{l=1}^{k} p_{i_l})$, then we call τ a good route to visit U.*

We are now ready to present our algorithm. It consists of two stages: decision making and route traversing. In the decision making stage, decisions to accept or reject requests are made one by one. At the end of step k, $P_k \subset \{1, ..., k\}$ is the set of requests that have not been selected for a visit so far, and $V_k = \{1, ..., k\} \backslash Q$ is the set of requests to be visited (i.e. those who have been accepted, and those who have been rejected but will be served anyway). Every request i receives a label b_i during the decision making stage, which will be used for construct a route later. $i > b_i$ indicates request i is accepted by the online algorithm; $b_i < i < \infty$ indicates request i is rejected but visited; $b_i = \infty$ indicates request i is rejected and not visited.

```
DECISION MAKING:

0. Initialization. $P_0 = \emptyset$, $V_0 = \{0\}$, and $k = 1$.

1. Request $k$ is accepted if and only if there exists a subset $Q_k \subset P_{k-1}$ and a good route
   $\mu_k$ to visit $Q_k \cup \{k\}$, such that $T(Q_k \cup \{k\}, \mu_k) + d(Q_k \cup \{k\}, V_{k-1}) \leq (\sum_{i \in Q_k} p_i + p_k)\sqrt{\ln n}$.

2. If request $k$ is accepted, assume $j \in V_{k-1}$ satisfies $d(Q_k \cup \{k\}, j) = d(Q_k \cup \{k\}, V_{k-1})$.
   Update $P_k = P_{k-1} \backslash Q_k$, $V_k = V_{k-1} \cup \{k\} \cup Q_k$ update labels: $b_k = j$ and $b_i = k$ for
   all $i \in Q_k$.

3. If request $k$ is rejected, update $P_k = P_{k-1} \cup \{k\}, V_k = V_{k-1}$, and $b_k = \infty$.

4. $k = k + 1$. Go to Step 1.
```

Let S_a be the subset of requests accepted by the online algorithm. After making accept/reject decisions, a route is constructed iteratively, based on the information gathered in the DECISION MAKING stage. Traveling along the route, the online server visits requests in S_a and in V_n:

ROUTE CONSTRUCTION:

0. Initialization. $\tilde{\tau}_0 = \tau_n$, and $k = 1$.

1. If $k \in S_a$ and $\tilde{\tau}_{k-1}$ covers at least one request in $Q_k \cup \{k\}$, let requests $c_{i_1}, c_{i_2}, \cdots, c_{i_w}$ be all the requests in $Q_k \cup \{k\}$ that are covered by $\tilde{\tau}_{k-1}$. Find the division of $Q_k \cup \{k\} = \bigcup_{t=1}^{w} I_t$ that satisfies the condition in the definition of good routes. Construct $\tilde{\tau}_k$ as follows, it is the same as $\tilde{\tau}_{k-1}$ except for w detours: when arriving at request $c_{i_t}, (1 \le t \le w)$, follow the shortest route to visit I_t, go back to c_{i_t}, and then continue to follow $\tilde{\tau}_{k-1}$.

2. If $k \in S_a$ and τ does not cover any request in $Q_k \cup \{k\}$, construct $\tilde{\tau}_k$ as follows: it is the same as $\tilde{\tau}_{k-1}$ except for a detour: when arriving at request b_k, follow the shortest route to visit all the requests in Q_k, go back to b_k, and then continue to follow $\tilde{\tau}_{k-1}$.

3. If $k \notin S_a$, $\tilde{\tau}_k = \tilde{\tau}_{k-1}$.

4. $k = k + 1$. Go to Step 1.

The cost of the resulting solution comes from two parts: the penalties of rejected requests and the time to visit requests. We will provide upper bounds for both parts. First, let us consider the penalty part. Let W be the set of requests that are accepted by the optimal offline solution, but rejected by the online solution. Following is a upper bound for penalties of requests in W:

Lemma 19. $\sum_{k \in W} p_k \le (2\sqrt{\ln n} + 1) C_{\text{opt}}(n)$.

Proof. We divide requests in W into some subsets by the following algorithm:

```
0. Initialization: $W_1 = W, k = 1$.

1. Divide all requests in $W_k$ into $m_k$ subsets, such that each subset consists of one or
   several successive requests, in every subset $A_k^j (1 \leq j \leq m_k)$, the largest index is
   smaller than the smallest label, and $\tau_n$ is a good route to visit $A_k^j$. The division
   has the fewest subsets among all possible divisions.

2. If $m_k \leq 1$, terminate; otherwise, go to Step 3.

3. Let $B_k = \{j | \sum_{i \in A_k^j} p_i \geq \max\{\sum_{i \in A_k^{j-1}} p_i, \sum_{i \in A_k^{j+1}} p_i\}\}$. Let $W_{k+1} = \bigcup_{j \in B_k} A_k^j$.

4. Update $k = k + 1$. Go to step 1.
```

We first show the existence of divisions that satisfy the requirements in Step 1: a trivial division consists of $|W_k|$ singletons. Thus, this algorithm is well defined. Noting that $\bigcup_{j \in B_k} A_k^j$ is also a feasible division of W_{k+1}, from the minimum number of subsets, we have $m_{k+1} \leq |B_k| \leq m_k/2$. Furthermore, because $m_1 \leq |W| \leq n$, the algorithm terminates after at most $\ln n$ iterations. Let K be the number of iterations before termination.

We then consider any two adjacent subset A_k^j and A_k^{j+1} in iteration $k < K$. We will show that $T(A_k^j \cup A_k^{j+1}, \tau_n) \geq \sqrt{\ln n} \min\{\sum_{i \in A_k^j} p_i, \sum_{i \in A_k^{j+1}} p_i\}$. Without loss of generality, let us assume $\max_{i \in A_k^j} i < \max_{i \in A_k^{j+1}} i$. Consider $\min_{i \in A_k^j} b_i$:

1. If $\min_{i \in A_k^j} b_i < \max_{i \in A_k^{j+1}} i$, let $z = \arg\min_{i \in A_k^j} b_i$. When request $\max_{i \in A_k^{j+1}} i$ is released, all requests in A_k^{j+1} are released and not covered by τ, and z is covered by τ. Noting that τ_n is a good route to visit A_k^{j+1}, we have $\sum_{i \in A_k^{j+1}} p_i \sqrt{\ln n} < T(A_k^{j+1}, \tau_n) + d(A_k^{j+1}, z)$; or request $\max_{i \in A_k^{j+1}} i$ will be accepted. Therefore, $T(A_k^j \cup A_k^{j+1}, \tau_n) \geq T(A_k^{j+1}, \tau_n) + d(A_k^{j+1}, z) > \sum_{i \in A_k^{j+1}} p_i \sqrt{\ln n}$.

2. If $\min_{i \in A_k^j} b_i < \max_{i \in A_k^{j+1}} i$, then τ is not a good route to visit $A_k^j \cup A_k^{j+1}$; otherwise, the two subsets can be merged to one. According to the definition of good routes ,we can show that $T(A_k^j \cup A_k^{j+1}, \tau_n) \geq \min\{\sum_{i \in A_k^j} p_i \sqrt{\ln n}, \sum_{i \in A_k^{j+1}} p_i \sqrt{\ln n}\}$.

From the above inequality, we have

$$\begin{aligned} 2C_{\text{opt}}(n) &\geq \textstyle\sum_{j=1}^{m_k} T(A_k^j \cup A_k^{j+1}, \tau_n) \\ &\geq \textstyle\sum_{j=1}^{m_k} \min\{\sum_{i\in A_k^j} p_i\sqrt{\ln n}, \sum_{i\in A_k^{j+1}} p_i\sqrt{\ln n}\} \\ &\geq \textstyle\sum_{j\notin B_k} \sum_{i\in A_k^j} p_i\sqrt{\ln n} \\ &= \textstyle\sum_{i\in W_k\setminus W_{k+1}} p_i\sqrt{\ln n} \end{aligned}$$

In the last iteration, from the requirement of divisions, $\min_{i\in W_K} b_i \geq \max_{i\in W_K} i$. Hence when request $\max_{i\in W_K} i$ is revealed, all requests in W_K are still in $P_{\max_{i\in W_K} i-1}$. On the other hand, request $\max_{i\in W_K} i$ is not accepted. Combined with the fact that τ_n is a good route to visit W_K, we have $C_{\text{opt}}(n) \geq \sum_{i\in W_K} p_i\sqrt{\ln n}$.

By summing these inequalities up, we conclude $\sum_{k\in W} p_k \leq (2\sqrt{\ln n}+1)C_{\text{opt}}(n)$. □

Consider now the route $\tilde{\tau}_n$ that visits all requests in V_n. We have a corresponding lemma:

Lemma 20. $L(\tilde{\tau}_n) \leq (2\sqrt{\ln n}+1)C_{\text{opt}}(n)$.

Proof. Let us establish the upper bound $L(\tilde{\tau}_n)$ by considering $L(\tilde{\tau}_k) - Ł(\tilde{\tau}_{k-1})$ for all $1 \leq k \leq n$:

1. If $k \in S_a$ and $\tilde{\tau}_{k-1}$ passes at least one request in $Q_k \cup \{k\}$, then according to the definition of good route, $L(\tilde{\tau}_k) - L(\tilde{\tau}_{k-1}) \leq 2\sqrt{\ln n}\sum_{t\in Q_k\cup\{k\}\setminus S_n} p_t$.
2. If $k \in S_a$ and $\tilde{\tau}_{k-1}$ does not cover any request in $Q_k \cup \{k\}$, then $L(\tilde{\tau}_k) - L(\tilde{\tau}_{k-1}) \leq 2T(Q_k\cup\{k\}, \mu_k) + 2d(Q_k\cup\{k\}, R_{k-1}) \leq 2\sqrt{\ln n}\sum_{t\in Q_k\cup\{k\}} p_t \leq 2\sqrt{\ln n}\sum_{t\in Q_k\cup\{k\}\setminus S_n} p_t$.
3. If $k \notin S_a$, then $L(\tilde{\tau}_k) = L(\tilde{\tau}_{k-1})$.

Since every request in R_n belongs to at most one of Q_i, by summing these inequalities up, we have $L(\tilde{\tau}_n) \leq L(\tau_n) + 2\sqrt{\ln n}\sum_{t\in R_n} p_t \leq (2\sqrt{\ln n}+1)C_{\text{opt}}(n)$. □

After providing upper bounds on both parts, we can now conclude:

Theorem 17. *The algorithm is $O(\sqrt{\ln n})$-competitive.*

Proof. Since the requests rejected by the online server are in $R_n \cup W$,

$$\begin{aligned} C_{on}(n) &\leq \textstyle\sum_{i\in R_n} p_i + \sum_{i\in W} p_i + L(\tilde{\tau}_n) \\ &\leq C_{\text{opt}}(n) + (2\sqrt{\ln n}+1)C_{\text{opt}}(n) + (2\sqrt{\ln n}+1)C_{\text{opt}}(n) \\ &= (4\sqrt{\ln n}+3)C_{\text{opt}}(n). \end{aligned}$$

Therefore, the algorithm is $O(\sqrt{\ln n})$-competitive. $\square$

Chapter 3

Generalized Online Assignment Problems

3.1 Introduction

In this chapter, we consider a generalized online assignment problem,whose introduction is motivated in part by the consideration of several applications arising within the internet (e.g., sponsored search auctions, load balancing and distributed caching in content-delivery networks, on demand video/movie requests, etc.) with the following inherent basic characteristics:

- a set of N buyers each interested in purchasing items from a set of K object types;
- items arrive one by one (as "requests") and their types become known upon their arrivals only;
- an "operating" entity is managing one key resource (e.g., bandwidth, cpu, web page size, etc.) essential for enabling the desired transactions (both matching a request to a buyer and allowing its "consumption");
- buyer i is willing to pay (the operating entity) an amount f_{ik} for getting a request of type k;
- buyer i has a total available budget a_i;
- all items of object type k can use up to a total of b_k amount of dedicated resources from the operating entity;
- when one unit of object type k is assigned to buyer i, it consumes s_{ik} amount of resources away from b_k;

- upon the arrival of any request, the operating entity must either turn down the request (due to lack of feasibility or possibly lack of economic attractiveness), or allocate it to one buyer. In the latter case, it receives a revenue corresponding to the fee that this buyer was willing to pay;
- taking the perspective of the operating entity, the goal is to assign requests to buyers so as to maximize the total revenue of the operating entity, while respecting the various constraints.

These characteristics are relevant for modeling many other applications in various settings such as in finance (e.g., execution over time of buying or selling very large orders in a volatile and unpredictable real-time market), yield management (e.g., room allocation in hotels, seat allocation in airlines, trains), online auctions (e.g., many key processes underlying operations such as those from eBay), to name just a few examples.

3.1.1 Formal problem definitions

Instance: N buyers, each buyer i with a budget a_i, and K distinct request types, each type k with a capacity b_k. Each buyer i specifies a price f_{ik} for a request of type k, and each request of type k leads to a resource consumption of s_{ik} for buyer i. A sequence of requests (index by j) arrive online, and the type of a request is revealed at its arrival. Every request can be assigned to at most one buyer.

Offline context: All the information (number and types of requests) is known before assigning requests.

Online context: No information on the total number of requests M and on the types of future requests $j+1, j+2, ..., M$ is known when assigning request j.

Objective: To maximize the total revenue while respecting the budget and capacity constraints.

Special cases:

- If $a_i = b_k = f_{ik} = s_{ik} = 1$, the problem reduces to an online bipartite matching problem as defined in [50]. We will consider the problem in Chapter 4.

- If there are (i) budget constraints, (ii) no resource capacity constraints (i.e., either $s_{ik} = 0$ for all i, k, or $b_k = \infty$ for all k), the problem reduces to the initial basic adwords problem as defined in [59, 30]. We will consider the problem in Chapter 5.

- If $f_{ik} = s_{ik}, \forall(i,k)$, we will say that we have an homogeneous case of the generalized online assignment problem. In this chapter, we will mainly focus on this homogeneous case.

We recall that for a maximization problem, an online algorithm is said to be r-competitive $(0 \leq r \leq 1)$ if, given any instance I of the problem, the cost of the solution given by the online algorithm is no less than r multiplied by that of an optimal offline algorithm:

$$\text{Cost}_{\text{online}}(I) \geq r\text{Cost}_{\text{optimal}}(I), \ \forall \text{ problem instances } I.$$

The supremum over all r such that an online algorithm is r-competitive is called the competitive ratio of the online maximization algorithm. An online algorithm is said to be best possible if there does not exist another online algorithm with a strictly larger competitive ratio.

3.1.2 Mathematical programming formulations

Note that for the offline version of the problem, it is the number of requests rather than the order of requests that matters. Thus, the offline problem can be formulated as an integer programming in a concise way:

$$\begin{array}{lll} \text{Maximize:} & \sum\limits_{i,k} f_{ik}x_{ik} & \\ \text{Subject to:} & \sum\limits_{k} x_{ik} \leq m_k & \forall j \\ & \sum\limits_{k} f_{ik}x_{ik} \leq a_i & \forall i \\ & \sum\limits_{i} s_{ik}x_{ik} \leq b_k & \forall k \\ & x_{ik} \in \mathcal{N} & \forall i,k \end{array} \tag{3.1}$$

where m_k is the number of requests of type k, x_{ik} are decision variables that denote the number of requests of type k assigned to bidder i.

However for the online version, the order of requests does matter, and thus an alternative mathematical formulation of the offline problem, which keeps track of each request identity, is more useful:

$$
\begin{aligned}
\text{Maximize:} \quad & \sum_{i,j,k} f_{ik} x_{ijk} & & \\
\text{Subject to:} \quad & \sum_{i,k} x_{ijk} \le 1 & & \forall j \\
& \sum_{j,k} f_{ik} x_{ijk} \le a_i & & \forall i \\
& \sum_{i,j} s_{ik} x_{ijk} \le b_k & & \forall k \\
& x_{ijk} = 0 & & \forall \text{ request } j \text{ not of type } k \\
& x_{ijk} \in \{0,1\} & & \forall i,j
\end{aligned}
\tag{3.2}
$$

where x_{ijk} is 1 if request j is of type k and is assigned to buyer i, 0 otherwise. It is worth noting that the offline problem is NP-hard, since it is no easier than the knapsack problem.

3.2 General Cases

3.2.1 Inhomogeneous cases

Intuitively, in order to maximize the revenue, we tend to assign a request to the bidder who is willing to pay the most so that we get more revenue. On the other side, if there are far more requests than the capacities, because of the capacity constraints, we would like to use capacity more efficiently, i.e. make more money per unit of capacity. In short, a good assignment will have both f_{ik} and f_{ik}/s_{ik} large. However, for problems in general where there is no consistency on the relation between f_{ik} and s_{ik}, we may not able to make both both f_{ik} and f_{ik}/s_{ik} large at the same time. In fact, we can show that any deterministic algorithm can be arbitrarily bad for problems in general:

Theorem 18. *$\forall m \in \mathbb{N}$, no deterministic algorithm has a ratio $\rho \ge 2/m$.*

Proof. Assume Algorithm A is a $2/m$-competitive deterministic algorithm.

Consider the following instance: there are $m+3$ bidders and only 1 request type. $\forall i \in \{1, 2, \cdots, m+3\}, a_i = \infty, f_{i1} = 1/m^{i-1}, s_{i1} = 1/m^{2i-2}$, $b_1 = m$. There are m^{2J-1} requests, $J \in \{1, \cdots, m+2\}$ to be determined by the adversary later depending on how the online algorithm behaves.

$\forall j \in \{1, 2, \cdots, m+2\}$, let x_i^j be the numbers of requests assigned to bidder i after first m^{2j-1} requests if applicable. Noting that, after first m^{2j-1} requests, the optimal revenue is m^j (all requests are assigned to bidder j). Thus, to keep the algorithm $2/m$-competitive, revenue generated by Algorithm A should be at least $2m^{j-1}$, i.e. $\sum_{i=1}^{j} \frac{x_i^j}{m^{i-1}} + m^{2j-1} \cdot \frac{1}{m^j} \geq 2m^{j-1}$, otherwise, we can find a contradiction by letting $J = j$. Because $\forall i, x_i^1 \leq x_i^2 \leq \cdots \leq x_i^{m+2} \doteq x_i$, $\sum_{i=1}^{j} \frac{x_i}{m^{i-1}} \geq m^{j-1}$, we can conclude $\sum_{i=1}^{m+2} \frac{x_i}{m^{2i-2}} \geq m + (m-1)/m$. Contradiction with capacity constraint. □

One solution to this issue to make the revenue and the capacity usage consistent, namely $f_{ik} = s_{ik}, \forall i, k$.

3.2.2 Large bids

If the revenue or the capacity usage is large compared to the budgets and the capacities, because of knapsack-like constraints, we can show that there is no non-trivial performance guarantee for any deterministic algorithm:

Theorem 19. *In general, any deterministic algorithm can be arbitrarily bad.*

Proof. Consider an instance with only 1 bidder and 2 request types, with $a_1 = b_1 = b_2 = n$, $f_{11} = s_{11} = 1, f_{12} = s_{12} = n$. A request of type 1 comes first. If it is assigned (to bidder 1), present another request of types. Note that at that time, the remaining budget is $n-1$, not enough to pay for the second one. Therefore, online revenue is 1, while optimal revenue is n. If the first request is not assigned, then no more request comes. In this case, online revenue is 0 while optimal revenue is 1. From the discussion above, we can conclude $\rho \leq 1/n$. Note that n is a arbitrary integer, the algorithm can be arbitrarily bad. □

As Theorem 19 indicates, there is no hope in finding an algorithm with a non-trivial competitive ratio for those instances where bid prices are big compared to budgets and capacities. Thus, we need to assume that the relative size of bid $c = \max_{i,k}\{\max\{\frac{f_{ik}}{a_i}, \frac{f_{ik}}{b_k}\}\}$ is small.

3.2.3 Assumptions on the model

As Theorem 18 and Theorem 19 show, in general, there are no algorithm with a non-trivial competitive ratio. Thus, we must make some assumptions on the model to have meaningful results. As discussed in Section 3.2.1 and 3.2.2, homogeneity assumption and small bid assumption are necessary:

Homogeneity Assumption: $f_{ik} = s_{ik}, \forall i, k$.

Small Bid Assumption: the relative size of bid $c = \max_{i,k}\{\max\{\frac{f_{ik}}{a_i}, \frac{f_{ik}}{b_k}\}\}$ is small.

One thing worth noting is that those two assumptions are independent: the instance constructed in the proof of Theorem 18 satisfies small bid assumption, and the one in the proof of Theorem 19 satisfies homogeneity assumption. Therefore, we do need both assumptions.

3.3 Special Cases

In this section, we will only consider the problems with homogeneity assumption and small bid assumption. As we will show, under those two assumptions, there are deterministic algorithms with non-trivial competitive ratios.

3.3.1 Upper bound

Theorem 20. *No deterministic algorithm has a competitive ratio $\rho > 1/2$.*

We construct an instance such that different bidders have very different valuation on requests. The number of requests is determined by the adversary. The online algorithm has to make a tradeoff between the short term revenue and the long term revenue. In the short term, the algorithm tends to act greedily to keep its revenue not too little compared to the optimal revenue. Thus, requests tends to be assigned to those bidders who is willing to pay more. On the other hand, for the sake of long term, the algorithm tends to assign requests evenly to leave enough flexibility for the future. As a result of this dilemma, any deterministic algorithm can not do better than 1/2-competitive. The formal proof is given as follows:

Proof. Consider the following instance: there are $2m$ bidders and $2m$ request types. $\forall i, j, a_i = b_j = n, f_{ij} = \epsilon^{i-1}$, where ϵ is a very small number such that $n\epsilon << 1$. There are M groups of requests sequentially (within a group, the order of requests doesn't matter), $M \in \{1, \cdots, 2m\}$ to be determined by the adversary depending on how the online algorithm behaves. In group k, there are requests of type $1, 2, \cdots, k$, and the number of type j requests is $n\epsilon^{j-k}$.

Assume there is a ρ-competitive algorithm. Assume in group k, the algorithm assigns $nx^j_{i,k}\epsilon^{1-i}$ requests of type j to bidder i ($i \leq k+1-j$), i.e. it gets $nx^j_{i,k}$ revenue from those requests. We assume we get nothing from assigning requests of type j to bidder i, where $i > k+1-j$, since we can get at most $n\epsilon^{k-j}\epsilon^{i-1} \leq n\epsilon$ which is neglectable by our assumption.

We define following notations: $\forall j \leq k, \bar{x}_j^k = \sum_{l=k}^{2m} x_{k+1-j,l}^j$, $\bar{a}_i = \sum_{j=1}^{2m-i+1} \bar{x}_j^{i+j-1}$, $\bar{b}_j = \sum_{i=j}^{2m} \bar{x}_j^i$, $\bar{r}_l = \sum_{k=1}^{l}\sum_{j=1}^{k} \bar{x}_j^k$. Here $n\bar{x}_j^k$ means the revenue gained from assigning requests of type j to bidder $k+1-j$, $n\bar{a}_i$ means the revenue gained from bidder i, $n\bar{b}_j$ means the space used of type j, $n\bar{r}_l$ is a upper bound of the revenue gained from the first l groups. To ease the analysis, we define two more notations: $\forall 2m \geq k \geq 2i+1, \sigma_i^k = \bar{x}_1^k + \cdots + \bar{x}_i^k + \bar{x}_{k-i+1}^k + \cdots + \bar{x}_k^k$, $\sigma_i = \sum_{k=2i+1}^{2m} \sigma_i^k$.

Because of budget and capacity constraints, we have $\bar{a}_i \leq 1$ and $\bar{b}_j \leq 1$. It is easy to see, if there are only first l groups presented, the optimal revenue is nl, while the online revenue is at most $n\bar{r}_l$. Since the algorithm is ρ-competitive, we have $\bar{r}_l \geq l\rho$.

$\forall l \leq m-1, \sigma_l \leq \sum_{i=1}^{l}(\bar{a}_i + \bar{b}_i) + \sum_{i=1}^{l-1}\sigma_i^{i+l} - \bar{r}_{2l-1} - \bar{r}_{2l} \leq 2l - (4l-1)\rho + \sum_{i=1}^{l-1}\sigma_i^{i+l}$. Adding all these $m-1$ inequalities together, we then have $\sum_{i=1}^{m-1}\sigma_i \leq (m-1)m - (m-1)(2m-1)\rho + \sum_{i=1}^{m-2}\sum_{k=2i+1}^{m-1+i}\sigma_i^k \leq (m-1)m - (m-1)(2m-1)\rho + \sum_{i=1}^{m-2}\sigma_i$, which implies $\sigma_{m-1} \leq (m-1)m - (m-1)(2m-1)\rho$. Obviously $\sigma_{m-1} \geq 0$, which implies $\rho \leq m/(2m-1)$. By letting $m \to \infty$, we have $\rho \leq 1/2$. □

3.3.2 Greedy algorithm

One natural algorithm is the greedy algorithm, namely, it always assigns a request to the bidder who can pay the most. Formally:

The Greedy Algorithm

```
0. Initialization, j=1.

1. Let $k$ be the type of request $j$.

    1a. Find argmax$_i\{f_{ik} > 0 | f_{ik} \leq \min\{$ remaining budget of $i$, remaining capacity
        of $k\}\}$. If there is no such $i$, then $j$ is not assigned to anyone.

    1b. Charge $i$, $k$ $f_{ik}$ units of money and capacity.

2. j=j+1. Go to 1.
```

One good thing about homogeneous cases is that, for every request, the money earned equals to the capacity used. Thus, we can transform the revenue in terms of one to another. The idea to prove the competitive ratio of greedy algorithm is similar to the analysis of applying the greedy algorithm to the simple bipartite matching problem. All bidders are divided into two groups: A_1, all those bidders who have little money available at the end; and A_2, all the other bidders. All request types are divided in a similar way: B_1, all those request types who have few capacity available at the end; and B_2, all the other request types. (We will give a more precise definition of those sets in the formal proof.) For a request type in B_2, given that the algorithm always does the assignment in a greedy way, either all requests of that type are won by bidders in A_2, or bidders in A_2 are not willing to pay a high price on that request type. In either way, capacity used of that request type in B_2 can be lower bounded in terms of money spend by bidders in A_2 of the optimal solution. On the other hand, capacity used of request type in B_1 is close to the total capacity of that type, which can easily be lower bounded by capacity used of that type of the optimal solution. By doing some careful algebraic manipulation, we can conclude:

Theorem 21. *Greedy algorithm is* $(1-c)/2$*-competitive, where* c *is the relative bid size.*

To prove this theorem, we need some notations and some easy observations. Let $f_{i.} = \max_k f_{ik}, f_{.k} = \max_i f_{ik}, c = \max\{\max_i \frac{f_{i.}}{a_i}, \max_k \frac{f_{.k}}{b_k}\}$. $A_1 = \{i \in A |$remaining budget of i is less than $f_{i.}\}$, $B_1 = \{k \in B|$remaining capacity of k is less than $f_{.k}\}$. $A_2 = A \backslash A_1, B_2 = B \backslash B_1$. Let $a_i^g, b_k^g, a_i^{\text{opt}}, b_k^{\text{opt}}$ be money/capacity used by the greedy solution/optimal solution. Let $a_{ik}^{\text{opt}} = f_{ik} \cdot x_{ik}^{\text{opt}}$ be the revenue earned by assigning requests of type k to bidder i in the optimal solution.

Claim 1. $\forall i \in A_1, a_i^g \geq a_i^{\text{opt}} - f_{i.}$.

Proof. Obviously from definition of A_1. □

Claim 2. $\forall k \in B_1, b_k^g \geq \sum_{i \in A_2} a_{ik}^{\text{opt}} - f_{.k}$.

Proof. $b_k^g \geq b_k - f_{.k} \geq b_k^{\text{opt}} - f_{.k} \geq \sum_{i \in A_2} a_{ik}^{\text{opt}} - f_{.k}$ □

Lemma 21. $\forall k \in B_2, b_k^g \geq \sum_{i \in A_2} a_{ik}^{\text{opt}}$.

Proof. There are two different cases,

1. If every request in type k is assigned to someone, then anyone who wins a bid in type k pay more than anyone in A_2. Because request type k in greedy algorithm is assigned more times than in optimal solution, so, $b_k^g \geq \sum_{i \in A_2} a_{ik}^{\text{opt}}$.
2. If some requests in type k is not assigned, then that means no one in A_2 is interested in that type. So, $b_k^g \geq 0 = \sum_{i \in A_2} a_{ik}^{\text{opt}}$.

□

Claim 3. $\forall i \in A_1, f_{i\cdot} \leq \frac{c}{1-c} a_i^g$.

Proof. $f_{i\cdot} \leq c \cdot a_i \leq c(a_i^g + f_{i\cdot})$, which implies $f_{i\cdot} \leq \frac{c}{1-c} a_i^g$. □

Claim 4. $\forall k \in B_1, f_{\cdot k} \leq \frac{c}{1-c} b_k^g$.

Proof. $f_{\cdot k} \leq c \cdot b_k \leq c(b_k^g + f_{\cdot k})$, which implies $f_{\cdot k} \leq \frac{c}{1-c} b_k^g$. □

Proof. After having all those lemmas, we can easily conclude Theorem 21. $\frac{2}{1-c} R^g \geq \sum_{i \in A_1} a_i^g + \frac{c}{1-c} \sum_{i \in A_1} a_i^g + \sum_{k \in B_2} b_k + \sum_{k \in B_1} b_k + \frac{c}{1-c} \sum_{k \in B_1} b_k \geq \sum_{i \in A_1} (a_i^{\text{opt}} - f_{i\cdot}) + \sum_{i \in A_1} f_{i\cdot} + \sum_{i \in A_2, k \in B_2} a_{ik}^{\text{opt}} + \sum_{i \in A_2, k \in B_1} (a_{ik}^{\text{opt}} - f_{\cdot k}) + \sum_{k \in B_1} f_{\cdot k} = \sum_{i \in A_1} a_i^{opt} + \sum_{i \in A_2, k \in B} a_{ij}^{\text{opt}} = R^{\text{opt}}$, which implies $R^g \geq (1-c)/2 \cdot R^{\text{opt}}$. □

Under the small bid assumption, as c goes to 0, the competitive ratio goes to 1/2. Thus, the greedy algorithm is an asymptotic optimal algorithm.

3.3.3 Primal-dual algorithm

Primal-Dual algorithms are proven to be useful in many online optimization problems. For example, the classic adwords problem, where there is no capacity constraint, can be solved by a primal-dual algorithm [24] with the best possible competitive ratio of that problem. The basic idea of primal-dual algorithms is to use the weak duality to bound the optimal solution. The key thing in the primal dual algorithm is to construct a dual feasible solution which is close to the primal feasible solution.

However, in our problem, it is hard to construct a good dual feasible solution because of the extra capacity constraints. If we use the same technique used in [24], we will get a pair of primal and dual solutions with a duality gap of $2 + \frac{1}{e-1}$, worse than the greedy algorithm. Thus, to develop an algorithm with a better competitive ratio, we need to introduce some new techniques. One way is to project the problem into a smaller space that is similar to the one in [24], generate a pair of primal and dual solutions in that space, and link back to the original space. What we are going to do next is to view the capacity constraints as side constraints, and consider LP relaxation of the remaining problem and its dual problem:

	Primal Problem P		Dual Problem D
Maximize:	$\sum_{i,j,k} f_{ik}x_{ijk}$	Minimize:	$\sum_j z_j + \sum_i a_i r_i$
Subject to:		Subject to:	
$\forall j$:	$\sum_{i,k} x_{ijk} \le 1$	$\forall (i,j,k)$:	$f_{ik}r_i + z_j \ge f_{ik}$
$\forall i$:	$\sum_{j,k} f_{ik}x_{ijk} \le a_i$	$\forall i,j$:	$r_i, z_j \ge 0$
$\forall j$ not of type k:	$x_{ijk} = 0$		
$\forall (i,j,k)$:	$x_{ijk} \ge 0$		

Based on this pair of LPs, we can develop a primal dual algorithm:

Primal-Dual Algorithm

```
0. Initially, all $r, x, z = 0$. Let $j = 1$.
1. Let $k$ be the type of request $j$.
    1a. Find $i$ that maximizes $f_{ik}(1 - r_i)$. If $r_i \ge 1$ or there is no more capacity for
        type $k$, then reject that request and go to 2.
    1b. Charge $i, k$ $f_{ik}$ units of money and capacity.
    1c. $x_{ijk} = 1, z_j = f_{ik}(1 - r_i), r_i = r_i + \frac{f_{ik}}{a_i}$
2. $j = j + 1$, go to 1.
```

Claim 5. *x is almost feasible to the original problem (2), namely, every budget constrain and capacity constraint can be violated at most once.*

Proof. In the Primal-Dual Algorithm, the dual variable r_i is the proportion of remaining budget of bidder i. If bidder i runs out of money, then $r_i \ge 1$, and no more requests will be assigned to him ever again. Thus, bidder i will be overcharged at most once. Similarly, noting Step 1 of the algorithm, every capacity constraint can be violated at most once. Therefore, the primal solution generated by this algorithm is almost feasible to the original problem. □

Given the almost feasible solution, we can construct a feasible solution by removing those bid assignments which violate budget constraints or capacity constrains. Let R be revenue actually got, R_p be the primal revenue from the algorithm. Because, the almost feasible solution violates every budget constraint and capacity constraint at most once, transforming it to a feasible solution will only lose a small amount of revenue. To be more precise, let A be the set of all bidders, let A' be the set of bidders who violate budget constraints; let B be the set of all request types, let B' be the set of those types which violate capacity constraints. The lost revenue will be $\sum_{i \in A'} \max_k\{f_{ik}\} + \sum_{k \in B'} \max_i\{f_{ik}\} \le \sum_{i \in A'} ca_i + \sum_{k \in B'} cb_k \le 2cR_p$. Thus $R_p - 2cR_p \le R$, which implies,

Lemma 22. $R \ge R_p(1-2c)$.

However, the dual solution may not be feasible. When request j of type k comes, if there are some capacities available at that time, because i is the bidder that maximize $f_{ik}(1-r_i)$, all dual constraints involve request j are valid, even if $r_i \ge 1$. Nevertheless, if there are no more capacity for type k at that time, those dual constraints involve request j may not hold. Thus, to have dual feasibility, we have to remove some dual constraints. We will only keep request types in $B \backslash B'$. For ease of notations, we will index the requests types in $B \backslash B'$ by k', and the requests of types in $B \backslash B'$ by j'.

When request j' comes, if it is rejected, according to the definition of B', k' has some capacity left. Therefore, for all i, either $f_{ik'} = 0$ or $r_i \ge 1$. In either case, $f_{ik'}r_i + z_{j'} \ge f_{ik'}$ is satisfied, i.e.

Claim 6. *$\forall (i, j', k')$, r, z satisfies dual feasibility $f_{ik'}r_i + z_{j'} \ge f_{ik'}$.*

Let's remove all those request types in B': further relax the primal problem P by removing all requests of types in B', and consider its dual problem.

	Primal Problem P'		Dual Problem D'
Maximize:	$\sum_{i,j',k'} f_{ik'}x'_{ij'k'}$	Minimize:	$\sum_{j'} z_{j'} + \sum_i a_i r'_i$
Subject to:		Subject to:	
$\forall j'$:	$\sum_{i,k'} x'_{ij'k'} \le 1$	$\forall (i,j',k')$:	$f_{ik'}r'_i + z'_{j'} \ge f_{ik'}$
$\forall i$:	$\sum_{j',k'} f_{ik'}x'_{ij'k'} \le a_i$	$\forall i, j'$:	$r'_i, z'_{j'} \ge 0$
$\forall j'$ not of type k':	$x_{ij'k'} = 0$		
$\forall (i,j',k')$:	$x'_{ij'k'} \ge 0$		

Noting those two pairs of primal dual problems are very similar except for that the second pair are of a lower dimension, we will construct (x', z', r') by projecting (x, z, r) into the smaller space. Mathematically, $x'_{k'} = x_{k'}, r'_{i'} = r_{i'}, z'_{j'} = z_{j'}$. (x, z, r) is updated when a new request comes in the original problem; as a projection of (x, z, r), (x', z', r') will also be updated. To more precise, the updating for (x', z', r') will be:

```
Update for$(x', z', r')$

0. Initially, let all $r', x', z' = 0$. Let $j = 1$.

1. Let $k$ be the type of request $j$.

    1a. If it is not assigned to anyone, then nothing changes. Go to 2.
    1b. If it is assigned to bidder $i$, and $j$ is of type $k$ in $B\backslash B'$, $x'_{ij} = x_{ij}, z'_j = f_{ij}(1 - r'_i), r'_i = r'_i + \frac{f_{ij}}{a_i}$. Go to 2.
    1c. If it is assigned to bidder $i$, and $j$ is of type $k$ in $B'$, $r'_i = r'_i + \frac{f_{ij}}{a_i}$ (note $x'_{ij}$ and $z'_j$ doesn't exist in $P'$ and $D'$). Go to 2.

2. $j = j + 1$. Go to 1.
```

Given that (z', r') is just a projection of (z, r), Claim 6 implies,

Claim 7. *z', r' are dual feasible to dual problem D'.*

Let $R_{B'} = \sum_{k \in B'} b_k$, let R_{opt} be the optimal value of the original problem (with capacity constraints), let R'_{opt} be the optimal value of P'. Let $R'_p = \sum_{i,j',k'} x'_{ik'} f_{ij'k'}$ be the primal revenue of P'. Let $R'_d = \sum_{j'} z'_{j'} + \sum_i a_i r'_i$ be the dual cost of D'. Because the primal problems P and P' are closely related, their optimal solutions are too different. Neither are the solutions constructed by the algorithm.

Lemma 23. $R_{\text{opt}} \leq R'_{\text{opt}} + R_{B'}$.

Proof. Projecting the optimal assignment to $B\backslash B'$ (i.e. withdraw those bids for requests in B', and keep the rest) is still a feasible assignment, and will lose at most $R_{B'}$. □

Lemma 24. $R_p \geq R'_p + R_{B'}$

Proof. $R_p = \sum_{i,j,k} x_{ijk} f_{ik} = \sum_{i,j',k'} x'_{ij'k'} f_{ik'} + \sum_{i,j,k \in B'} x_{ijk} f_{ik} = R'_p + \sum_{k \in B'} \sum_{i,j,k} x_{ijk} f_{ik} \geq R'_p + \sum_{k \in B'} b_k = R'_p + R_{B'}$. □

After establishing the relation between P and P', the only thing left is to bound the primal solution to P' in terms of the optimal solution to P'. One natural way is to find the gap between the primal solution and the dual feasible solution.

Lemma 25. *$R'_d \le 2R'_p + R_{B'} + cR$.*

Proof. When request j comes:

1. If j belongs to types in B', then R'_p remains the same and R'_d increases.
2. If j belongs to types not in B', and assigned to bidder i, then R'_p increases by f'_{ik} and R'_d increases by $z'_j + a_i \Delta r'_i \le 2f'_{ik}$.
3. If j belongs to types not in B', and not assigned to anyone, then both R'_p and R'_d remain the same.

Note that the sum of increase of R'_d in the first case is at most $R_{B'} + cR$, and increase of R'_d is no more than twice of R'_p in the other two cases. Thus, $R'_d \le 2R'_p + R_{B'} + cR$. □

Theorem 22. *$R \ge \frac{1-2c}{2+c-2c^2} R_{\text{opt}}$*

Proof. $\frac{2+c-2c^2}{1-2c} R = \frac{2}{1-2c} R + cR \ge 2R_p + cR = 2R'_p + 2R_{B'} + cR \ge R'_d + R_{B'} \ge R'_{\text{opt}} + R_{B'} \ge R_{\text{opt}}$, thus, $R \ge \frac{1-2c}{2+c-2c^2} R_{\text{opt}}$ □

Theorem 22 shows that the Primal-Dual Algorithm is asymptotically 1/2-competitive as c goes to 0.

3.4 Numerical Results

3.4.1 Underlying structures

Here we use the case with 10 bidders and 10 types, all budgets and capacities are equal, i.e. $a_i = b_j = 20$. $f_{ij} = \begin{cases} 1 & \text{w.p. } 0.5 \\ x & \text{w.p. } 0.5 \end{cases}$. In each instance, there are 400 requests uniformly i.i.d. over all 10 types. We test 100 i.i.d. instances and find the average revenue.

We can see that the primal dual algorithm is better when the difference between bids are larger and there are sufficient requests. This result is quite intuitive. Since there are sufficient requests, the short term revenue doesn't matter that much. Thus, the primal dual algorithm, which takes the long term revenue into account when making an assignment, works better than the greedy algorithm.

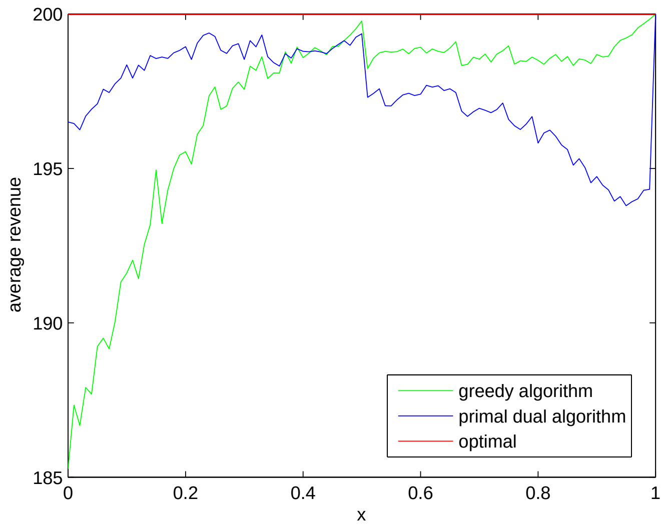


Figure 3-1: Performance v.s. underling graph

3.4.2 Evolutionary performance

Here we use the case with 10 bidders and 10 types, all budgets and capacities are equal, i.e. $a_i = b_j = 50$. $f_{ij} = \begin{cases} U_{ij} & \text{w.p. } 0.5 \\ 0 & \text{w.p. } 0.5 \end{cases}$, where U_{ij} is uniformly i.i.d. over $[0, 2]$ for all i, j. In each instance, there are 500 requests uniformly i.i.d. over all 10 types. We test 300 i.i.d. instances and find the average revenue.

From Figure 3-2, we can see that the greedy algorithm works better when there are fewer requests while the primal dual algorithm is better with more requests. It is easy to understand because the primal dual algorithm tends to assign requests evenly.

3.4.3 Updating rules

It is hard to tell what is a better updating rule for the primal dual algorithm. For example, in the adwords problem, a nonlinear updating will lead to an optimal algorithm, meanwhile, in this problem, a linear one will do the trick. We will compare these two different updating to find out which one works better empirically.

Here we use the case with 10 bidders and 10 types, all budgets and capacities are equal, i.e. $a_i = b_j = 20$. $f_{ij} = \begin{cases} U_{ij} & \text{w.p. } 0.5 \\ 0 & \text{w.p. } 0.5 \end{cases}$, where U_{ij} is uniformly i.i.d. over $[0, 2]$ for all i, j. In each instance, there are 400 requests uniformly i.i.d over all 10 types. We test 100 i.i.d. instances and find the average revenue.

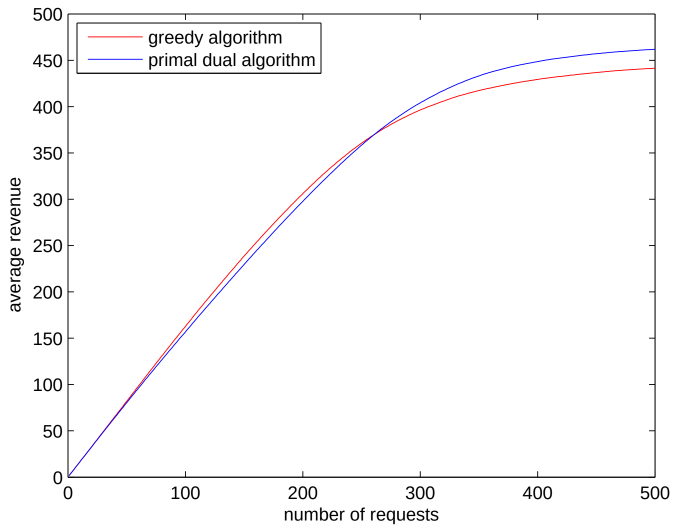


Figure 3-2: Performance v.s. time

From Figure 3-3, we can see that the linear updating rules outperform nonlinear updating rules, unlike in the Adwords problem [23]. It is not very clearly to us, why this is the case here. It would be interesting to know the set of problems for which linear/nonlinear updating rules are better.

3.5 Randomized Algorithms

In this section, we will examine how randomized algorithms do for this problem. In general, randomized algorithms work better than deterministic algorithms for online optimization problems, because the randomness increases offline cost a lot, while it does not affect online cost that much. However, as we will show, for this problem, randomized algorithms do not do any better.

Randomized algorithms are usually much more difficult to characterize than deterministic algorithms. Thus, instead of applying randomized algorithms on deterministic instances, applying deterministic algorithms on randomized instances will be much easier. It is a general practice to use the Yao principle [69] to derive bounds on competitive ratios of randomized algorithms, by transforming randomized algorithms applying on deterministic instances to deterministic algorithms applying on randomized algorithms. However, in many online problems, some subtle care should be taken, which leads to annoying technical

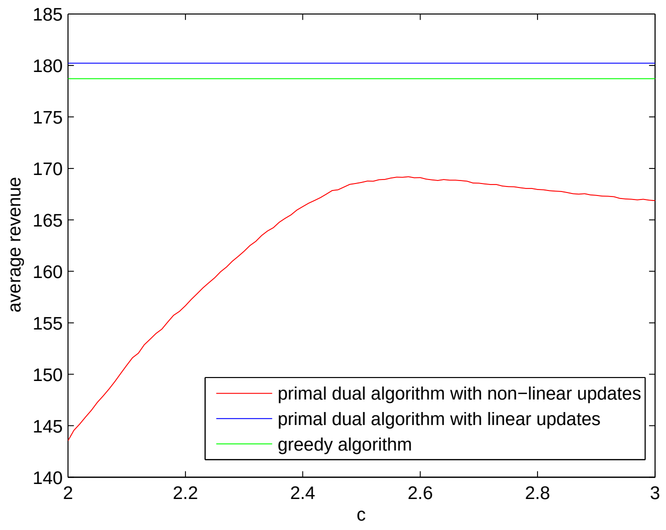


Figure 3-3: Comparison between different updating rules

details. Instead, we will use a technique, introduced in [66], to derive a upper bound in a more straightforward way.

First, let's introduce some notations. For an online maximization problem, let $\mathcal{I}$ be a set of instances, and $\mathcal{A}$ be a set of deterministic algorithms. A randomized algorithm A_q can be defined as a probability distribution over $\mathcal{A}$ and a randomized instance I_q can be defined as a probability distribution over $\mathcal{I}$. Let $Z^A(I)$ and $Z^{\text{OPT}}(I)$ be the objective value produced by algorithm A and optimal objective value on instance I. The competitive ratio of a randomized algorithm A_q is defined as $\min\limits_{I \in \mathcal{I}} \frac{E_{A_q}[Z^{A_q}(I)]}{Z^{\text{OPT}}(I)}$. Then, we have the following lemma on the upper bound of the competitive ratio of randomized algorithms:

Lemma 26. *[66] Given an online maximization problem, with possible input sequences $\mathcal{I}$, and possible deterministic algorithms $\mathcal{A}$, both possibly infinity. Then, for any random sequence I_p and any randomized algorithm A_q, we have*

$$\max_{A \in \mathcal{A}} \frac{E_{I_p}[Z^A(I_p)]}{E_{I_p}[Z^{\text{OPT}}(I_p)]} \geq \min_{I \in \mathcal{I}} \frac{E_{A_q}[Z^{A_q}(I)]}{Z^{\text{OPT}}(I)}.$$

Proof. Suppose there is a randomized algorithm A_q with competitive ratio ρ. Then $\forall I \in \mathcal{I}$, $E_{A_q}[Z^{A_q}(I)] \geq \rho Z^{\text{OPT}}(I)$. Hence, $\rho E_{I_p}[Z^{\text{OPT}}(I_p)] \leq E_{I_p}[E_{A_q}[Z^{A_q}(I_p)]] = E_{A_q}[E_{I_p}[Z^{A_q}(I_p)]] \leq \max\limits_{A \in \mathcal{A}} E_{I_p}[Z^{A_q}(I_p)]$. □

To derive the upper bound on randomized algorithms, we only need to find a upper bound on applying deterministic algorithms on randomized algorithms:

Lemma 27. *There exists such I_p that* $\max_{A\in\mathcal{A}} \frac{E_{I_p}[Z^A(I_p)]}{E_{I_p}[Z^{\mathrm{OPT}}(I_p)]} \leq 1/2$.

The instance here is the same one used in Theorem 20 except for the number of requests. The number of requests is a random variable here other than some number determined by the adversary in Theorem 20. The online algorithm still has to make a tradeoff between the short term revenue and the long revenue. The proof is given as follows:

Proof. Consider the following instance: there are $2m$ bidders and $2m$ request types. $\forall i, j, a_i = b_j = n, f_{ij} = \epsilon^{i-1}$, where ϵ is a very small number such that $n\epsilon << 1$. There are $2M$ groups of requests sequentially: in group k, there are requests of type $1, 2, \cdots, k$, and the number of type j requests is $n\epsilon^{j-k}$. Here, M is a random variable, uniformly distributed over $\{1, 2, \cdots, 2m\}$

For any deterministic algorithm, if all $2m$ groups of requests are presented, assume in group k, the algorithm assigns $nx^j_{i,k}\epsilon^{1-i}$ requests of type j to bidder i ($i \leq k+1-j$), i.e. it gets $nx^j_{i,k}$ revenue from those requests. We assume we get nothing from assigning requests of type j to bidder i, where $i > k+1-j$, since we can get at most $n\epsilon^{k-j}\epsilon^{i-1} \leq n\epsilon$ which is neglectable by our assumption.

We define following notations: $\forall j \leq k, \bar{x}^k_j = \sum_{l=k}^{2m} x^j_{k+1-j,l}$, $\bar{a}_i = \sum_{j=1}^{2m-i+1} \bar{x}^{i+j-1}_j$, $\bar{b}_j = \sum_{i=j}^{2m} \bar{x}^i_j$, $\bar{r}_l = \sum_{k=1}^{l}\sum_{j=1}^{k} \bar{x}^k_j$. Here $n\bar{x}^k_j$ means the revenue gained from assigning requests of type j to bidder $k+1-j$, $n\bar{a}_i$ means the revenue gained from bidder i, $n\bar{b}_j$ means the space used of type j, $n\bar{r}_l$ is a upper bound of the revenue gained from the first l groups. To ease the analysis, we define two more notations: $\forall 2m \geq k \geq 2i+1, \sigma^k_i = \bar{x}^k_1 + \cdots + \bar{x}^k_i + \bar{x}^k_{k-i+1} + \cdots + \bar{x}^k_k$, $\sigma_i = \sum_{k=2i+1}^{2m} \sigma^k_i$.

Because of budget and capacity constraints, we have $\bar{a}_i \leq 1$ and $\bar{b}_j \leq 1$. It is easy to see, if there are only first l groups presented, the optimal revenue is nl, while the online revenue is at most $n\bar{r}_l$.

$\forall l \leq m-1, \sigma_l \leq \sum_{i=1}^{l}(\bar{a}_i + \bar{b}_i) + \sum_{i=1}^{l-1}\sigma^{i+l}_i - \bar{r}_{2l-1} - \bar{r}_{2l} \leq 2l - \bar{r}_{2l-1} - \bar{r}_{2l} + \sum_{i=1}^{l-1}\sigma^{i+l}_i$. Adding all these $m-1$ inequalities together, we then have $\sum_{i=1}^{m-1}\sigma_i \leq (m-1)m - \sum_{i=1}^{2m-2}\bar{r}_i + \sum_{i=1}^{m-2}\sum_{k=2i+1}^{m-1+i}\sigma^k_i \leq (m-1)m - \sum_{i=1}^{2m-2}\bar{r}_i + \sum_{i=1}^{m-2}\sigma_i$, which implies $\sigma_{m-1} + \sum_{i=1}^{2m-2}\bar{r}_i \leq (m-1)m$. Obviously $\sigma_{m-1} \geq 0$, which implies $\sum_{i=1}^{2m-2}\bar{r}_i \leq (m-1)m$.

It is easy to see, the optimal revenue $R_{\text{opt}} = M$. Therefore, the expected optimal revenue $E[R_{\text{opt}}] = E[M] = m + 1/2$. Meanwhile, the expected online revenue $E[R] = \frac{1}{2m}\sum_{i=1}^{2m} \bar{r}_i \leq \frac{1}{2m}(2m + 2m + \sum_{i=1}^{2m-2} \bar{r}_i) \leq \frac{(m+3)m}{2m} = (m+3)/2$. By letting m goes to infinity, we can conclude the lemma. □

These two lemmas above conclude our main result here:

Theorem 23. *No randomized algorithm has a competitive ratio of $\rho \leq 1/2$.*

Theorem 23 shows that randomized algorithms don't help in this problem. The greedy algorithm and the primal dual algorithm are the best algorithms among all possible algorithms, either deterministic or randomized.

3.6 Extensions

In this section, we will discuss how to solve the problem without those two assumptions we made previously, and still come up with something that makes sense.

3.6.1 Bounded cases

Although, as Theorem 18 shows, any algorithm can be arbitrarily bad if the instance doesn't satisfies homogeneity assumption, its proof also indicates that the worst case happens in extreme situations where s_{ij}/f_{ij} varies widely. Actually, as it turns out, if s_{ij}/f_{ij} is bounded on both sides away from 0, there exist algorithms with positive competitive ratio.

If we are given a ρ-competitive algorithm for the homogeneous version (eg. greedy algorithm and primal-dual algorithm), then we can come up with a $\frac{u\rho}{v}$-competitive algorithm, where $0 < u \leq \frac{s_{ij}}{f_{ij}} \leq v, \forall i, j$.

Modified Algorithm

1. Construct a new instance P2 based on the original one P1. The numbers of bidders and request types are exactly the same. $b_j^2 = b_j^1, s_{ij}^2 = s_{ij}^1, f_{ij}^2 = s_{ij}^1/v, a_i^2 = u/v \cdot a_i^1$.

2. Whenever a new request comes in P1, present a request of the same type in P2. If the original algorithm assigns it to bidder i in P2, assign it to bidder i in P1. If the original one rejects it, reject it.

Theorem 24. *The modified algorithm is $\frac{u\rho}{v}$-competitive.*

Proof. Because $\frac{f^2_{ij}}{f^1_{ij}} = \frac{s^1_{ij}}{vf^1_{ij}} \geq \frac{u}{v}$ and $\frac{a^2_i}{a^1_i} = \frac{u}{v}$, the modified algorithm always gives a feasible solution. Also notice that $f^2_{ij} = s^1_{ij}/v \leq f^1_{ij}$, we have $R^1 \geq R^2 \geq \rho R^2_{\text{opt}}$.

To establish the relation between R^1_{opt} and R^2_{opt}, we consider another problem P3, where the numbers of bidders and request types remain the same, $b^3_j = b^1_j, s^3_{ij} = s^1_{ij}, f^3_{ij} = s^1_{ij}/u = v/u \cdot f^2_{ij}, a^3_i = a^1_i = v/u \cdot a^2_i$. It is easy to see $R^1_{\text{opt}} \leq R^3_{\text{opt}} = v/u \cdot R^2_{\text{opt}}$.

Therefore, $R^1 \geq R^2 \geq \rho R^2_{\text{opt}} \geq \frac{u\rho}{v} R^1_{\text{opt}}$. □

3.6.2 Fractional bids

Indeed, as Theorem 19 and its proof indicate, if the relative bid size is large, there is nothing we can do in the original settings. However, it will be different if fractional bids are allowed, where a request can split can be assigned to different bidders. Formally, in terms of mathematical programming, we do not have the integrality constraints any more:

$$\begin{array}{lll} \text{Maximize:} & \sum\limits_{i,j,k} f(i,j,k)x(i,j,k) & \\ \text{Subject to:} & \sum\limits_{i,k} x(i,j,k) \leq 1 & \forall j \\ & \sum\limits_{j,k} f(i,j,k)x(i,j,k) \leq a_i & \forall i \\ & \sum\limits_{i,j} s(i,j,k)x(i,j,k) \leq b_k & \forall k \\ & 0 \leq x(i,j,k) \leq 1 & \forall i,j,k \end{array} , \qquad (3.3)$$

Given that fractional bids are allowed, we do not lose the revenue of the very last bids that bidders cannot afford previously, which counts for c in the competitive ratios of both greedy algorithm and primal-dual algorithm. In fact, both greedy algorithm and primal-dual algorithm are 1/2-competitive, no matter how large the relative bid size is. We will propose the algorithms and show the results on the competitive ratios without proof:

```
Greedy Algorithm with Fractional Bids

0. Initialization, j=1.

1. Let $k$ be the type of request $j$, let $n_j = 1$ be the number of request to be assigned.

    1a. Find $i$=argmax$_i\{f_{ik} > 0|\min\{$ remaining budget of $i$, remaining capacity of $k\} \geq 0\}$. If there is no such $i$, Go to 2.

    1b. $n_{ij} = \min\{$ remaining budget of $i$, remaining capacity of $k\}/f_{ik}$. Assign $n_{ij}$ unit of request $j$ to $i$. Charge $i, k$ $n_{ij} f_{ik}$ units of money and capacity. $n_j = n_j - n_{jk}$. If $n_j = 0$, go to 2; otherwise, go to 1a.

2. j=j+1. Go to 1.
```

Theorem 25. *Greedy algorithm with fractional bids is* 1/2*-competitive.*

```
Primal-Dual Algorithm with Fractional Bids

0. Initially, all $r, x, z = 0$. Let $j = 1$.

1. Let $k$ be the type of request $j$. Let $n_j = 1$ be the number of request to be assigned.

    1a. Find $i$ that maximizes $f_{ij}(1 - r_i)$. If $r_i \geq 1$ or there is no more capacity for type $k$, then reject that request and go to 2.

    1b. $n_{ij} = \min\{$ remaining budget of $i$, remaining capacity of $k\}/f_{ij}$. Assign $n_{ij}$ unit of request $j$ to $i$. Charge $i, k$ $n_{ij} f_{ij}$ units of money and capacity. $n_j = n_j - n_{jk}$.

    1c. $x_{ij} = n_{ij}, z(j) = n_{ij} f_{ij}(1 - r_i), r_i = r_i + \frac{n_{ij} f_{ij}}{a_i}$. Update $n_j = n_j - n_{ij}$. If $n_j = 0$, go to 2; otherwise, go to 1a.

2. $j = j + 1$, go to 1.
```

Theorem 26. *Primal-dual algorithm with fractional bids is* 1/2*-competitive.*

Chapter 4

Online Stochastic Matching Problems

4.1 Introduction

Bipartite matching problems have been studied extensively in the operations research and computer science literature. We consider in this chapter variants of the online stochastic bipartite matching problem motivated by Internet advertising display applications, as introduced in Feldman et al. [29].

We are given a bipartite graph $G = \{A \cup I, E\}$, where A is a set of advertisers and I is a set of impression types. An edge $(a, i) \in E$ if and only if advertiser $a \in A$ is interested in impressions of type $i \in I$. The set of advertisers and their preferences are fixed and known in advance. Requests for impression come online one at a time at periods $t = 1, 2, \cdots, n$ (n being fixed and known in advance), and the impression type of each request is chosen randomly and independently from a given probability distribution over the set I.

Upon the arrival of a request, an online algorithm must irrevocably assign it to one of the interested advertisers or drop it. Overall, every request cannot be assigned to more than one advertiser, and every advertiser can be assigned at most one request. The goal is to maximize the expected number of assigned requests over the random sequence of impressions.

Given that there is a total of n requests, the probability that a request is for an impression of type i can be written as r_i/n, where r_i is the expected number of requests of type i

among the random sequence. Without loss of generality, we assume that $r_i \leq 1$ for all type i; note that if a type i were to be such that $r_i > 1$, we could duplicate node in $i \in I$ into a set of identical nodes, each with the same adjacent edge structure as the original node, and each with expected number of arrival no more than one.

In this chapter, we consider two variants of this online stochastic i.i.d. model: a special case for which $r_i = 1$ for all i, which we refer to as the case with integral arrival rates; and the unrestricted case with general arrival rates.

We also consider a Poisson arrival model, removing the need to assume that the total number of arrivals is fixed and known in advance, as is required for the analysis of the stochastic i.i.d. models.

4.1.1 Our results and techniques

In Feldman et al. [29], the authors provide a 0.670-competitive algorithm for the online stochastic bipartite matching problem with integral arrival rates, the first result to show that stochastic information on request arrivals could provably improve upon the $1 - 1/e$ competitive ratio of Karp et. al. [50]. Removing this integrality restriction, Manshadi et al. [57] show it is still possible to do better than $1 - 1/e$ and propose a 0.702-competitive algorithm, using offline statistics drawn from Monte Carlo sampling. The authors further prove that the algorithm has a better competitive ratio of 0.705 when the arrival rates are integral. More recently, Mahdian and Yan [56] and Karande et al. [48] study a much less restrictive version of the problem where not only the arrival rates are arbitrary, they are not known to the algorithm a priori.

In this chapter we consider a general class of online algorithms for the i.i.d. model which improve on [29][57] and which use computationally efficient offline procedures (based on the solution of simple linear programs of maximum flow types). Under the integrality restriction on the expected number of impressions of each types, we get a $(1 - 2e^{-2})$-competitive algorithm. Without this restriction, we get a 0.706-competitive algorithm. Although the model we consider is more restrictive than the one in [56][48], we obtain better competitive ratio.

Our techniques can be applied to other related problems such as the online stochastic b-matching problem (quite trivially) and for the online stochastic vertex-weighted bipartite matching problem as defined in Aggarwal et al. [3]. For that problem, we obtain a 0.725-

competitive algorithm under the stochastic i.i.d. model with integral arrival rates. Our vertex-weighted model is a special case of the edge-weighted model considered by Haeupler et al. [37], who propose a 0.667-competitive algorithm for the edge-weighted case. Finally we show the validity of all our results under a Poisson arrival model, removing the need to assume that the total number of arrivals is fixed and known in advance, as is required for the analysis of the stochastic i.i.d. models.

In order to introduce the main general ideas behind our techniques, let us first define some basic concepts about optimal offline solutions. From the available information about the problem (the initial graph $G = \{A \cup I, E\}$, the probability distribution over the set of impression types I, and the number n of i.i.d. draws from that distribution), one can solve an optimal maximum cardinality matching for each possible realization of the n i.i.d. draws. Let OPT be the random variable corresponding to the values obtained by these offline matchings. The expected value $\mathbb{E}[\text{OPT}]$ can be written as $\sum_{e \in E} f_e^*$, where f_e^* is the probability that edge e is part of an optimal solution in a given realization. Note that, as in [57], $\mathbf{f}^* = (f_e^*)_{e \in E}$ can also be defined as a so-called optimal offline fractional matching.

Instead of computing $\mathbf{f}^*$ (or estimating it as in [57]) and using the information for guiding online strategies, our strategy is to formulate special maximum flow problems whose optimal solutions provide the input for the design of good online algorithms. Moreover these maximum flow problems are defined in such a way that $\mathbf{f}^*$ corresponds to feasible flows, allowing us to derive upper bounds on $\sum_{e \in E} f_e^*$, and get valid bounds for the competitive ratios of the related online algorithms.

We now provide more details. Consider an instance of a single-source single-destination node-capacitated maximum flow problem on G, with a source s connected to all elements of A, a destination t connected to all elements of I, a unit capacity on all $a \in A$, and a capacity r_i (expected number of arrivals) on each $i \in I$. Define $f_e = f_{a,i}$ to be the flow on $e = (a, i)$ for all $e \in E$. This problem can equivalently be formulated as a linear program (LP):

$$\begin{aligned} \max \quad & \sum_{e} f_e \\ & \sum_{i \sim a} f_{a,i} \leq 1 \quad && \forall a \in A \\ & \sum_{a \sim i} f_{a,i} \leq r_i \quad && \forall i \in I \\ & f_e \geq 0 \quad && \forall e \in E \end{aligned} \tag{4.1}$$

where $i \sim a$ and $a \sim i$ are shortcuts for $i : (a, i) \in E$ and $a : (a, i) \in E$, respectively.

One of the key steps behind our approach is to find appropriate additional constraints on the flows (to add to (4.1)) so that the resulting optimal solutions of the constrained LP lead to improved guidance for online strategies, while keeping the optimal offline fractional matching $\mathbf{f}^*$ feasible with respect to the constrained LP.

Let us now formally introduce the concept of a "list of interested advertisers for an impression of type i". Consider the set $A_i = \{a \in A : (a, i) \in E\}$ and let Ω_i be the set of all possible non-empty ordered subsets of elements of A_i. An element of Ω_i will be called a list of interested advertisers for impression of type i. We are ready to describe our class of online algorithms:

Random Lists Algorithms (RLA)

1. Add appropriate constraints to (4.1) to get a new constrained LP. Let $\mathbf{f}$ be an optimal solution to this LP.
2. Using $\mathbf{f}$, construct a probability distribution $\mathcal{D}_i$ over the set Ω_i for each impression type i.
3. When a request for an impression of type i arrives, select a list from Ω_i using the probability distribution $\mathcal{D}_i$:
   - if all the advertisers in the list are already matched, then drop the request;
   - otherwise, assign the request to the first unmatched advertiser in the list.

Steps 1 and 2 are problem-specific. Different solutions $\mathbf{f}$ and different construction of distributions $\mathcal{D}_i$ will lead to online algorithms that may have different properties and competitive ratios. However, these algorithms all share one common and important property: with high probability, they are robust with respect to different realizations of the n i.i.d. sequence of impression types. This property will be useful for the rigorous analysis of competitive ratios. Random lists used in this chapter are extensions of ideas given in [57], but unlike [57], where the length of lists is at most 2, we consider lists of length 3 in this chapter.

4.2 Preliminary on Competitive Ratios

As a measure of performance, online algorithms are typically compared to optimum offline solutions using ratios. In this chapter, an algorithm is called α-competitive if $\frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]} \geq \alpha$ for any given probability distributions. The goal is to find algorithms with large competitive ratios. One could use a stronger notion of competitive ratio that $\frac{\text{ALG}}{\text{OPT}} \geq \alpha$ would hold for most realizations as used in [29]. In this section we show that for the algorithms in our proposed class, the two concepts are in fact closely related and lead to competitive ratios that are valid under either of these measures.

Let $L_1, L_2, \cdots, L_n$ be the sequence of random lists for the n successive requests. Every list only contains interested advertisers, and the assignment of requests only depends on the order of advertisers in the list and their current status. Thus, from a given realization of this sequence of random lists, we can construct the corresponding matching and find its cardinality. We can show that the cardinality is stable with respect to the realization in the following sense:

Claim 8. *If two realizations $(l_1, \cdots, l_t, \cdots, l_n)$ and $(l_1, \cdots, l'_t, \cdots, l_n)$ only differ by one list, then the cardinality of their resulting matchings differs at most by one.*

Proof. Let W_j and W'_j be the set of matched advertisers right after the j^{th} arrival corresponding to the two realizations above, respectively. We will show by induction that $\forall j, |W_j \backslash W'_j| \leq 1$ and $|W'_j \backslash W_j| \leq 1$. For all $j \leq t-1$, since the two realizations are identical for the first j lists, $W_j = W'_j$. Since in every period, at most one advertiser becomes matched, the claim is also true for $j = t$. Let us consider $j \geq t+1$. If $W_j \backslash W_{j-1} \subset W'_{j-1}$, then by induction, $|W_j \backslash W'_j| \leq |W_{j-1} \backslash W'_{j-1}| \leq 1$. Otherwise, let $\{k\} = W_j \backslash W_{j-1}$. Then, in the list l_j, all advertisers in front of k are in W_{j-1}. Noting that k is unmatched for ALG$'$ before the j^{th} period, we have $W'_j \backslash W'_{j-1} \subset W_{j-1} \cup \{k\}$. Therefore, $|W_j \backslash W'_j| = |W_{j-1} \backslash W'_{j-1}| \leq 1$. Similarly, we can show $|W'_j \backslash W_j| \leq 1$. Hence, $|\text{ALG} - \text{ALG}'| \leq ||W_n| - |W'_n|| \leq \max\{|W_n \backslash W'_n|, |W'_n \backslash W_n|\} \leq 1$. □

Note that L_j only depends on the impression type of the j^{th} request, and does not depend on types and assignments of earlier impressions. Thus, $L_1, \cdots, L_n$ are independently and identically distributed. We can then apply McDiarmid's Inequality which we recall here for convenience:

McDiarmid's Inequality [58]: Let $X_1, X_2, \ldots, X_n$ be independent random variables all taking values in the set $\mathcal{X}$. Let $f : \mathcal{X}^n \mapsto \mathbb{R}$ be a function of $X_1, X_2, \ldots, X_n$ that satisfies $\forall i$, $\forall x_1, \ldots, x_n, x'_i \in \mathcal{X}$, $|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x'_i, \ldots, x_n)| \leq c_i$. Then $\forall \epsilon > 0$,

$$\Pr(f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)] > \epsilon) \leq \exp(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2})$$

and

$$\Pr(f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)] < -\epsilon) \leq \exp(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}).$$

Combining McDiarmid's Inequality with Claim 8 we obtain:

Lemma 28. $\Pr(\mathrm{ALG} - \mathbb{E}[\mathrm{ALG}] < -n\epsilon) \leq \exp(-2n\epsilon^2)$.

Similarly, note that the offline solution only depends on the realization of the impression types. So we can show a similar result:

Lemma 29. $\Pr(\mathrm{OPT} - \mathbb{E}[\mathrm{OPT}] > n\epsilon) \leq \exp(-2n\epsilon^2)$.

From the two lemmas above, we can conclude that

$$\Pr(\frac{\mathrm{ALG}}{\mathrm{OPT}} \geq \frac{\mathbb{E}[\mathrm{ALG}]}{\mathbb{E}[\mathrm{OPT}]} - \frac{2\epsilon}{c+\epsilon}) \geq 1 - 2\exp(-2n\epsilon^2),$$

where $c = \mathbb{E}[\mathrm{OPT}]/n$. If $\mathbb{E}[\mathrm{OPT}] = \Theta(n)$, the inequality above indicates that the two notions of competitive ratios are closely related and essentially equivalent as far as our results are concerned.

Throughout the chapter we will assume that n is large enough so that a factor of $1 + O(1/n)$ is negligible when analyzing the performance of online algorithms.

4.3 Stochastic Matching with Integral Arrival Rates

In this section and the next two, we assume that $r_i = 1$ for all i.

4.3.1 Online algorithm

As we mentioned, two steps in RLA are problem-specific: finding offline solutions and constructing random lists. In this subsection, we propose methods for these two steps.

An offline solution

Let us consider the following maximum flow problem on the graph G:

$$\begin{aligned} \max \quad & \sum_{a,i} f_{a,i} \\ s.t. \quad & \sum_{i \sim a} f_{a,i} \le 1 \quad \forall a \in A \\ & \sum_{a \sim i} f_{a,i} \le 1 \quad \forall i \in I \\ & f_e \in [0, 2/3] \quad \forall e \in E \end{aligned} \tag{4.2}$$

Note that compared to (4.1) as introduced in 4.1.1, the additional constraints on the flows are very simple. Since the set of vertices of the feasible polytope of (4.2) is a subset of $\{0, 1/3, 2/3\}^E$, there exists an optimal solution to (4.2) in $\{0, 1/3, 2/3\}^E$.

To ease the construction of random lists and the analysis, based on the optimal solution $\mathbf{f}$, we first construct a resulting graph $G_{\mathbf{f}} = \{A \cup I, E_{\mathbf{f}}\}$, where $E_{\mathbf{f}} = \{e \in E : f_e > 0\}$. For simplicity, we try to make $E_{\mathbf{f}}$ as sparse as we can by doing the following two types of transformations. As argued below, there exists an optimal solution $\mathbf{f}$ such that its resulting graph $G_{\mathbf{f}}$ does not contain cycles of length 4, unless the four nodes in such a cycle do not have any other neighbors; and such a solution can be found in polynomial time. Cycles in Figure 4-1 are the only three possible cycles of length 4. The four nodes in the left cycle cannot have any other neighbor outside the cycle; the middle and right cycle can be transformed into a non-cycle with the same objective value. Furthermore, if there exists impression i that has two neighbors a_1 and a_2 with $f_{i,a_1} = f_{i,a_2} = 1/3$ and $f_{a_1} + f_{a_2} < 2$, without loss of generality, we assume $f_{a_2} < 1$. Another solution $\mathbf{f}'$ with $f'_{i,a_1} = 0$, $f'_{i,a_2} = 2/3$, and everything else unchanged has the same objective value, and less edges in its resulting graph. We transform $\mathbf{f}$ to $\mathbf{f}'$. Note that each time, transformations mentioned above remove one or two edges and does not introduce any new edge. Thus, given any initial optimal solution, after at most $|E|$ transformations, the optimal solution cannot be transformed further in the above two ways.

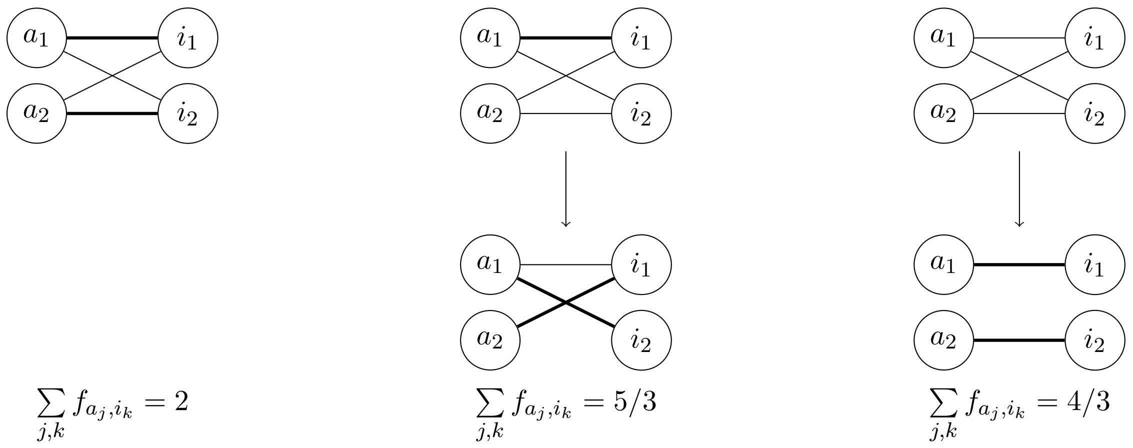

a_1
a_2
i_1
i_2
$\sum_{j,k} f_{a_j,i_k} = 2$
$\sum_{j,k} f_{a_j,i_k} = 5/3$
$\sum_{j,k} f_{a_j,i_k} = 4/3$

Figure 4-1: Cycles of length 4. Thin edges carry 1/3 flow; and thick edges carry 2/3 flow.

The extra constraint $f_e \leq 2/3$ is added for two reasons: LP (4.2) provides a upper bound on the offline solution; the resulting graph is sparse. In fact, as showed in Section 4.3.2, any constraint $f_e \leq c$ with $c \geq 1 - e^{-1}$ provides an upper bound on the offline solution; however, only $c = 2/3$ makes the resulting graph sparse. The sparsity not only helps the construction of random lists as described in Section 4.3.1, but also eases the analysis of the algorithm.

Generation of the random lists

In order to simplify the description of the specific probability distribution used to generate the random lists, and the analysis of the corresponding online algorithm, let us first add dummy advertisers a_d^i and dummy edges (a_d^i, i) with $f_{a_d^i,i} = r_i - \sum_{a \in A} f_{a,i}$ for all $i \in I$ with $\sum_{a \sim i} f_{a,i} < 1$. Dummy advertisers are flagged as matched from the start, so no impression are ever assigned to them. Since every edge in the graph has value 1/3 or 2/3, every node has two or three neighbors.

The construction of the random lists goes as follows. Given an impression type i, if it has two neighbors a_1 and a_2 in the resulting graph, the list is $\langle a_1, a_2 \rangle$ with probability $f_{a_1,i}$; the list is $\langle a_2, a_1 \rangle$ with probability $f_{a_2,i}$. Otherwise, if i has three neighbors a_1, a_2, and a_3 (in this case, $f_{a_1,i} = f_{a_2,i} = f_{a_3,i} = 1/3$), the list is a uniformly random permutation of $\langle a_1, a_2, a_3 \rangle$.

4.3.2 Upper bound on the offline algorithm

In order to show that LP (4.2) provides a upper bound on the offline solution, we prove that the offline solution is feasible to the LP. The feasibility of first two constraints is obvious. The feasibility of the last constraint is implied by a simply lemma:

Lemma 30 (Manshadi et al.[57])**.** $\forall e \in E, f_e^* \leq 1 - e^{-1} < 2/3$.

From Lemma 30, the expected optimal offline solution is feasible to LP (4.2). Therefore, the optimal solution $\mathbf{f}^T \cdot \mathbf{1}$ is a upper bound on the offline optimal $\mathbf{f}^{*T} \cdot \mathbf{1}$. From now on, we will compare the online algorithm with the optimal solution of LP (4.2) instead of the offline solution, because the former one is much easier to find than the latter one.

4.3.3 Certificate events

One difficulty encountered in previous papers is that an advertiser being matched is highly dependent on other advertisers. The strong dependence is difficult to deal with, and difficult to be decoupled. In this chapter, we use a local approach to avoid this issue. To be more specific, we compute a lower bound on p_a, the probability that advertiser a is matched, using only knowledge of a's neighborhood.

To ease the analysis, we consider lists associated with all arriving impressions rather than the types of impressions, because online matching results are a deterministic function of the former one. As mentioned in Section 4.2, all lists are i.i.d. distributed. It is not difficult to see that the distribution can be easily inferred from the resulting graph $G_{\mathbf{f}}$. For example, a local structure as showed in Figure 4-2 implies that with probability $1/n$, a list starts with $\langle a_1, ...\rangle$; and with probability $1/6n$, a list is $\langle a_1, a, a_2\rangle$.

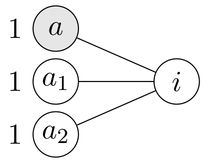


Figure 4-2: Possible configurations of i's neighborhood in the graph. All edges carry 1/3 flow. The number next to advertiser a indicates $f_a = \sum_{i \sim a} f_{a,i}$.

Assume a is the advertiser we are considering, and i is a neighbor of a. Let us consider the following two types of events: $B_a = \{$among the n lists, there exists a list starting with

$\langle a, ...\rangle\}$ and $G_a^i = \{$among the n lists, there exist successive lists starting with advertisers different from a but which are neighbors of i, and ensuring that a is matched$\}$. For example, in a local structure as showed in Figure 4-2, if three lists appear in order: $\langle a_1, ...\rangle$, $\langle a_2, ...\rangle$, and $\langle a_1, a_2, a\rangle$, then advertiser a is matched; and hence G_a^i happens. B_a and G_a^i (for any i) will be called "certificate events", in the sense that if any of these events happen, they provide a certificate that advertiser a is matched.

We now show that these certificate events have some good properties and their probabilities are easy to find. In this section and the next, we will use these certificate events to lower bound the probability that an advertiser is matched; and further lower bound the competitive ratios of our algorithms.

Asymptotic Independence

For notation simplicity, we define supporting set $S(G_a^i)$ as the set of lists that start with advertisers that are neighbors of i but not a; $S(B_a)$ as the set of lists that start with a. The supporting set of the intersection of two certificate events is defined as the union of the supporting sets of the two certificate events.

Lemma 31. *Let E_1 and E_2 be certificate events or intersections of two certificate events. If their supporting sets $S(E_1) \cap S(E_2) = \emptyset$, then E_1 and E_2 are asymptotically independent, i.e. $|\Pr(E_1 \cap E_2) - \Pr(E_1)\Pr(E_2)| < O(1/n)$.*

Proof. Let M_1 be the number of lists among all n lists in $S(E_1)$; M_2 be the number of lists among all n lists in $S(E_2)$. The proof consists of three key parts: with high probability M_1 and M_2 are small; when M_1 and M_2 are small, they are asymptotically independent; given M_1 and M_2, E_1 and E_2 are independent.

According to the construction of our algorithm, we can show that a given list belongs to $S(E_1)$ (or $S(E_2)$) with probability less than $6/n$. From the Chernoff bound, with high probability M_1 and M_2 are close to their mean: $\Pr(M_1 \geq 6\mu) \leq \exp(-\frac{6\mu^2}{2+\mu}) \leq O(1/n)$ and $\Pr(M_2 \geq 6\mu) \leq \exp(-\frac{6\mu^2}{2+\mu}) \leq O(1/n)$, where $\mu = n^{1/3}$.

Assuming $\mathbb{E}[M_1] = n_1$ and $\mathbb{E}[M_2] = n_2$, for all $m_1 < 6\mu$ and $m_2 < 6\mu$, we have

$$\frac{\Pr(M_1 = m_1, M_2 = m_2)}{\Pr(M_1 = m_1)\Pr(M_2 = m_2)} = \frac{(n-m_1)!(n-m_2)!}{n!(n-m_1-m_2)!}\frac{(1-(n_1+n_2)/n)^{n-m_1-m_2}}{(1-n_1/n)^{n-m_1}(1-n_2/n)^{n-m_2}} = 1+O(1/n),$$

where the last inequality is due to $m_1 m_2 = o(n)$.

Since all advertisers other than neighbors of i are assumed to have infinite capacities, all the lists that are not in $S(E_1)$ do not affect E_1. Thus, given $M_1 = m_1$, E_1 is independent of $n - m_1$ lists that are not in $S(E_1)$. Because of the assumption $S(E_1) \cap S(E_2) = \emptyset$, E_1 is independent of E_2 given M_1 and M_2.

From the three facts above, we have

$$\begin{aligned}
\Pr(E_1 \cap E_2) &= \sum_{m_1,m_2} \Pr(M_1 = m_1, M_2 = m_2) \Pr(E_1, E_2 | M_1 = m_1, M_2 = m_2) \\
&= \sum_{m_1,m_2<6\mu} \Pr(M_1 = m_1, M_2 = m_2) \Pr(E_1, E_2 | M_1 = m_1, M_2 = m_2) + O(1/n) \\
&= \sum_{m_1,m_2<6\mu} \Pr(M_1 = m_1, M_2 = m_2) \Pr(E_1 | M_1 = m_1) \Pr(E_2 | M_2 = m_2) + O(1/n) \\
&= \sum_{m_1,m_2<6\mu} \Pr(M_1 = m_1) \Pr(M_2 = m_2) \Pr(E_1 | M_1 = m_1) \Pr(E_2 | M_2 = m_2) + O(1/n) \\
&= \sum_{m_1,m_2} \Pr(M_1 = m_1) \Pr(M_2 = m_2) \Pr(E_1 | M_1 = m_1) \Pr(E_2 | M_2 = m_2) + O(1/n) \\
&= \Pr(E_1) \Pr(E_2) + O(1/n)
\end{aligned}$$

□

By applying Lemma 31 twice, we can show that any four (or less than four) certificate events are asymptotic independent, as long as their supporting sets do not intersect:

Corollary 1. *Consider a set of at most four certificate events* $\{C_j\}_{j\in J}$ *(*$|J| \le 4$*). If* $\cap_{j\in J} S(C_j) = \emptyset$*, then* $\Pr(\cap_{j\in J} C_j) = \prod_{j\in J} \Pr(C_j) + o(1/n)$*.*

Computing Probabilities

As in this section and the next, supporting sets of certificate events are of small sizes because of the construction of the distribution. In such cases, the probabilities of certificate events can be calculated via double summation, which is doable even by hand, though time-consuming.

On the other hand, it also can be done via a dynamic programming approach. Given n, the probability of an advertiser being matched at the end given the current state can be computed backward. As we can easily check, the probability converges to the limit with an error term of $O(1/n)$. In fact, when $n = 10^4$, the computed probability is within 10^{-5} accuracy.

In this chapter, we will only leave the probabilities of certificate events and omit the process of finding them due to the following reasons. First, the computation of probabilities is not the key to our approach, though the actual number matters. Second, it is just simple algebra and too long to present in the chapter.

4.3.4 Lower bound on the online algorithm

For notational simplicity, define $f_a \triangleq \sum_{i\sim a} f_{a,i}$, and let p_a be the probability that an advertiser a is matched by the online algorithm. Since every edge in the graph G with a non-zero flow will carry a flow of 1/3 or 2/3, there are very few different local configurations in the graph $G_f = \{A \cup I, E_f\}$, where $E_f = \{e \in E | f_e > 0\}$. For example, for an edge $e = (a, i)$ such that $f_a = 1$ and $f_{a,i} = 2/3$, the only four possibilities for i's neighborhood are $\alpha1$, $\alpha2$, $\alpha3$, and $\alpha4$ in Figure 4-3; for an edge $e = (a, i)$ such that $f_a = 1$ and $f_{a,i} = 2/3$, the only five possibilities for i's neighborhood are $\beta1$, $\beta2$, $\beta3$, $\beta4$, and $\beta5$ in Figure 4-3.

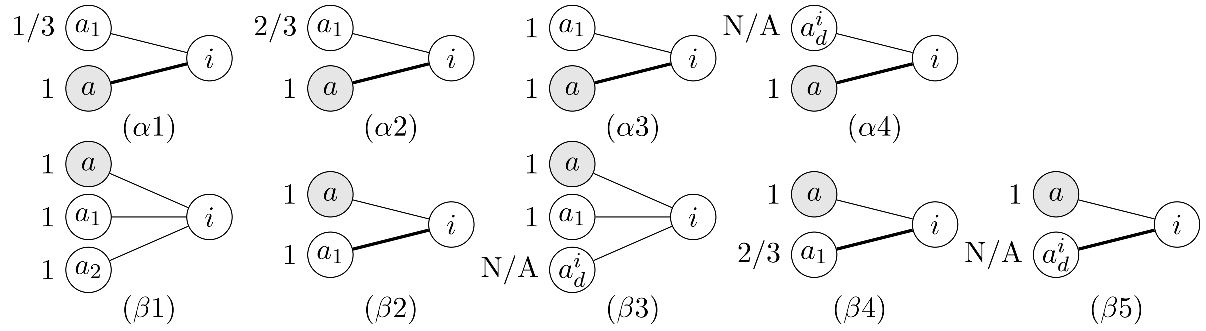


Figure 4-3: Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates $f_a = \sum_{i\sim a} f_{a,i}$.

For each configuration, because a has at most three neighbors, we can easily compute a lower bound on the probability of its being matched. For example, assume a has two neighbors i_1 and i_2, and they are not part of a cycle of length 4. a is matched if one of the three certificate events happens: B_a, $G_a^{i_1}$ or $G_a^{i_2}$. Since those three events are asymptotically independent, $p_a \geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1-p)(1-p_1)(1-p_2)$, where $p = \Pr(B_a)$, $p_1 = \Pr(G_a^{i_1})$, and $p_2 = \Pr(G_a^{i_2})$ are easy to find. Using such an idea, we can show case by case that:

Lemma 32. *$\forall a \in A$, let N_a be the set of advertisers who are at an edge-distance no more than 4 from a in G_f. Then, there exists $\mu_{a,a'} \in [0,1]$ for all $a' \in N_a$, such that*

$$\sum_{a' \in N_a} \mu_{a,a'} p_{a'} \geq (1 - 2e^{-2}) \sum_{a' \in N_a} \mu_{a,a'} f_{a'}.$$

Proof. For advertiser a with $f_a = 1/3$, $p_a \geq \Pr(B_a) = 1 - e^{-1/3} \geq 0.850 f_a$. For advertiser a with $f_a = 2/3$, $p_a \geq \Pr(B_a) = 1 - e^{-2/3} \geq (1 - 2e^{-2}) f_a$. Thus, we only need to prove the lemma for a with $f_a = 1$.

Before doing so, let us first find probabilities of events B_a and G_a^i exa.

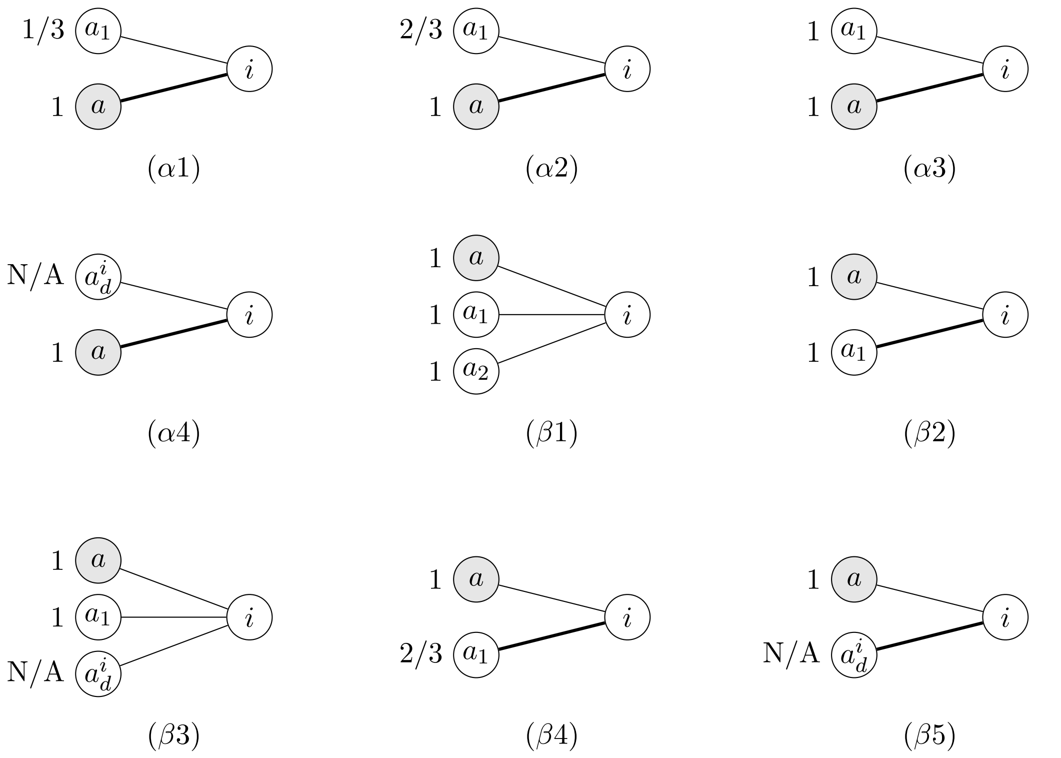


Figure 4-4: Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a.

α) $f_{a,i} = 2/3$.

If i has 2 neighbors a and a_1, then $f_{a_1,i} = 1/3$:

α1. $f_{a_1} = 1/3$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{j=1}^{n} \tfrac{1}{3n} e^{-\frac{j}{3n}} \left(1 - e^{-\frac{n-j}{3n}}\right) \\ &\approx 1 - \tfrac{4}{3} e^{-\frac{1}{3}} \triangleq p_1 (\geq 0.044).\end{aligned}$$

α2. $f_{a_1} = 2/3$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{i=1}^{n} \tfrac{2}{3n} e^{-\frac{2j}{3n}} \left(1 - e^{-\frac{n-j}{3n}}\right) \\ &= 1 - e^{-\frac{2}{3}} - e^{-\frac{1}{3}} \cdot 2(1 - e^{-\frac{1}{3}}) \triangleq p_2 (\geq 0.080).\end{aligned}$$

$\alpha 3$. $f_{a_1} = 1$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{j=1}^{n} \tfrac{1}{n} e^{-\frac{j}{n}} \bigl(1 - e^{-\frac{n-j}{3n}}\bigr) \\ &= 1 - e^{-1} - e^{-\frac{1}{3}} \cdot \tfrac{3}{2}(1 - e^{-\frac{2}{3}}) \triangleq p_3 (\geq 0.109).\end{aligned}$$

If i has only one neighbor:

$\alpha 4$.

$$\Pr(G_a^i) \geq 1 - e^{-\frac{1}{3}} \triangleq p_4 (\geq 0.283).$$

β) $f_{a,i} = 1/3$.

If i_1 has 3 neighbors a, a_1, and a_2:

$\beta 1$. We have $f_{a_1} = f_{a_2} = 1$; otherwise, we can find another optimal solution to LP (4.2) with less non-zero flow edges. Therefore,

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{k>j} \tfrac{2}{n} e^{-\frac{2j)}{n}} \cdot \tfrac{4}{3n} e^{-\frac{4(k-j)}{3n}} \cdot (1 - \tfrac{7}{8} e^{-\frac{2(n-k)}{3n}}) \\ &= 1 - e^{-2} - \tfrac{21}{8} e^{-\frac{2}{3}} (1 - e^{-\frac{4}{3}}) + \tfrac{9}{4} e^{-\frac{4}{3}} (1 - e^{-\frac{2}{3}}) \triangleq p_5 (\geq 0.160).\end{aligned}$$

If i has 2 neighbors a and a_1. Note that $f_{a_1,i} = 1/3$ and $f_{a_1} < 1$ cannot happen together; otherwise, $\mathbf{f}$ cannot be a maximum flow:

$\beta 2$. $f_{a_1,i} = 2/3$ and $f_{a_1} = 1$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{j=1}^{n} \tfrac{1}{n} e^{-\frac{j}{n}} \bigl(1 - e^{-\frac{2(n-j)}{3n}}\bigr) \\ &= 1 - e^{-1} - e^{-\frac{2}{3}} \cdot 3(1 - e^{-\frac{1}{3}}) \triangleq p_6 (\geq 0.195).\end{aligned}$$

$\beta 3$. $f_{a_1,i} = 1/3$ and $f_{a_1} = 1$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{j=1}^{n} \tfrac{4}{3n} e^{-\frac{4j}{3n}} (1 - \tfrac{7}{8} e^{-\frac{2(n-j)}{3n}}) \\ &= 1 - e^{-\frac{4}{3}} - \tfrac{7}{6} e^{-\frac{2}{3}} (1 - e^{-\frac{2}{3}}) \triangleq p_7 (\geq 0.299).\end{aligned}$$

$\beta 4$. $f_{a_1,i} = 2/3$ and $f_{a_1} = 2/3$.

$$\begin{aligned}\Pr(G_a^i) &\geq \sum_{j=1}^{n} \tfrac{2}{3n} e^{-\frac{2j}{3n}} \bigl(1 - e^{-\frac{2(n-j)}{3n}}\bigr) \\ &= 1 - \tfrac{5}{3} e^{-\frac{2}{3}} \triangleq p_8 (\geq 0.144).\end{aligned}$$

If i has only one neighbor:

$\beta 5$.

$$\Pr(G_a^i) \quad \geq \quad 1 - e^{-\frac{2}{3}} \triangleq p_9 (\geq 0.486).$$

We say that "i is in $\alpha 1$ with respect to a" if (a, i) has the neighborhood structure shown in $\alpha 1$ in Figure 4-4. The same for $\alpha 2, \alpha 3, \alpha 4, \beta 1, \ldots, \beta 5$. "With respect to a" will be omitted unless otherwise specified. We are now ready to compute lower bounds on p_a when $f_a = 1$. We have two cases:

Case 1: a is contained in a cycle of length 4 in G_f. Let $[a_1(= a), i_1, a_2, i_2]$ be the cycle. According to the choice of the offline solution (see Section 4.3.1), $\sum_{j,k} f_{a_j,i_k} = 2$ as showed in Figure 4-5. Let N be the number of impressions of type i_1 or i_2. Then,

$$p_{a_1} + p_{a_2} = \Pr(N = 1) + 2\Pr(N \geq 2) = 2 - 4e^{-2} = (1 - 2e^{-2})(f_{a_1} + f_{a_2}) \approx 0729(f_{a_1} + f_{a_2}).$$

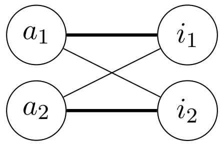


Figure 4-5: Cycle of length 4. Thin edges carry 1/3 flow; and thick edges carry 2/3 flow.

Case 2: a is not contained in a cycle of length 4. Then it has either three or two neighbors:

1) a has three neighbors i_1, i_2, and i_3, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2} \cup G_a^{i_3}) \\ &= 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2}))(1 - \Pr(G_a^{i_3})) \\ &\geq 1 - e^{-1}(1 - p_8)^3 \geq 0.769 f_a. \end{aligned}$$

Please note that the second equality is due to Corollary 1, which says that four or less certificate events are asymptotically independent if their supporting sets do not intersect. We will also use this asymptotic independence property repeatedly in the rest of the proof.

2) a has two neighbors i_1 and i_2. Without loss of generality, let us assume that $f_{a,i_1} = 1/3$ and $f_{a,i_2} = 2/3$:

2a. i_1 is in case $\beta3$ or $\beta5$.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1}) = 1 - (1-\Pr(B_a))(1-\Pr(G_a^{i_1})) \\ &\geq 1 - e^{-1}(1-p_7) \geq 0.742 f_a. \end{aligned}$$

2b. i_1 is in case $\beta4$. Let a_1 be the other neighbor of i_1.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1}) = 1 - (1-\Pr(B_a))(1-\Pr(G_a^{i_1})) \\ &\geq 1 - e^{-1}(1-p_8). \end{aligned}$$

Similarly, we can compute

$$\begin{aligned} p_{a_1} &\geq \Pr(B_{a_1} \cup G_{a_1}^{i_1}) \\ &\geq 1 - e^{-\frac{2}{3}} + e^{-\frac{2}{3}} \sum_j \tfrac{1}{n} e^{-\frac{j}{n}} \left(1 - e^{-\frac{n-j}{3n}}\right) \\ &= 1 - e^{-\frac{2}{3}} + e^{-\frac{2}{3}} \left(1 - e^{-1} - e^{-\frac{1}{3}} \cdot \tfrac{3}{2}(1 - e^{-\frac{2}{3}})\right) \triangleq p_{10} (\geq 0.542). \end{aligned}$$

Since $f_a = 1, f_{a_1} = 2/3$, we have

$$p_a + p_{a_1} \geq 1 - e^{-1}(1-p_8) + p_{10} \geq 0.736(f_a + f_{a_1}).$$

2c. i_1 is in case $\beta2$.

i. i_2 is in case $\alpha1$. Let a_1 be the other neighbor of i_2. Since

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1-\Pr(B_a))(1-\Pr(G_a^{i_1}))(1-\Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1-p_1)(1-p_6) \end{aligned}$$

and

$$p_{a_1} \geq \Pr(B_{a_1}) = 1 - e^{-\frac{1}{3}} = p_4,$$

Since $f_a = 1, f_{a_1} = 1/3$, we have

$$p_a + p_{a_1} \geq 1 - e^{-1}(1-p_1)(1-p_6) + p_4 \geq 0.750(f_a + f_{a_1}).$$

ii. i_2 is in case $\alpha 2$. Let a_1 be the other neighbor of i_2.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1 - p_2)(1 - p_6). \end{aligned}$$

Similarly, we can compute

$$\begin{aligned} p_{a_1} &\geq \Pr(B_{a_1} \cup G_{a_1}^{i_1}) \\ &\geq 1 - e^{-\frac{2}{3}} + e^{-\frac{2}{3}} \sum_j \frac{1}{n} e^{-\frac{j}{n}} \left(1 - e^{-\frac{n-j}{3n}}\right) \\ &= 1 - e^{-\frac{2}{3}} + e^{-\frac{2}{3}} \left(1 - e^{-1} - e^{-\frac{1}{3}} \cdot \frac{3}{2}(1 - e^{-\frac{2}{3}})\right) = p_{10}. \end{aligned}$$

Since $f_a = 1, f_{a_1} = 2/3$, we have

$$p_a + 0.5 p_{a_1} \geq 1 - e^{-1}(1 - p_2)(1 - p_6) + 0.5 p_{10} \geq 0.749(f_a + 0.5 f_{a_1}).$$

iii. i_2 is in case $\alpha 3$ or $\alpha 4$.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1 - p_3)(1 - p_6) \geq 0.736 f_a. \end{aligned}$$

2d. i_1 is in case $\beta 1$ and i_2 is not in case $\alpha 3$.

i. i_2 is in case $\alpha 1$. Let a_1 be the other neighbor of i_2. Since

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1 - p_1)(1 - p_5) \end{aligned}$$

and

$$p_{a_1} \geq \Pr(B_{a_1}) = 1 - e^{-\frac{1}{3}} = p_4,$$

Since $f_a = 1, f_{a_1} = 1/3$, we have

$$p_a + p_{a_1} \geq 1 - e^{-1}(1 - p_1)(1 - p_5) + p_4 \geq 0.741(f_a + f_{a_1}).$$

ii. i_2 is in case $\alpha 2$. Let a_1 be the other neighbor of i_2.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1-(1-\Pr(B_a))(1-\Pr(G_a^{i_1}))(1-\Pr(G_a^{i_2})) \\ &\geq 1-e^{-1}(1-p_2)(1-p_5). \end{aligned}$$

Similarly, we can compute

$$\begin{aligned} p_{a_1} &\geq \Pr(B_{a_1} \cup G_{a_1}^{i_1}) \\ &\geq 1-e^{-\frac{2}{3}}+e^{-\frac{2}{3}}\sum_j \tfrac{1}{n}e^{-\frac{j}{n}}\left(1-e^{-\frac{n-j}{3n}}\right) \\ &= 1-e^{-\frac{2}{3}}+e^{-\frac{2}{3}}\left(1-e^{-1}-e^{-\frac{1}{3}}\cdot\tfrac{3}{2}(1-e^{-\frac{2}{3}})\right) = p_{10}. \end{aligned}$$

Since $f_a = 1, f_{a_1} = 2/3$, we have

$$p_a + 0.5p_{a_1} \geq 1-e^{-1}(1-p_2)(1-p_5) + 0.5p_{10} \geq 0.740(f_a + 0.5f_{a_1}).$$

iii. i_2 is in case $\alpha 4$,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1-(1-\Pr(B_a))(1-\Pr(G_a^{i_1}))(1-\Pr(G_a^{i_2})) \\ &\geq 1-e^{-1}(1-p_4)(1-p_5) \geq 0.778f_a. \end{aligned}$$

2e. i_1 is in case $\beta 1$ and i_2 is in case $\alpha 3$, then i_2 has two neighbors. Let a_1 and a_2 be the other two neighbors of i_1, and a_3 be the other neighbor of i_2.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1-(1-\Pr(B_a))(1-\Pr(G_a^{i_1}))(1-\Pr(G_a^{i_2})) \\ &\geq 1-e^{-1}(1-p_3)(1-p_5). \end{aligned}$$

i. a_3 has three neighbors. According to the discussion in (1),

$$p_{a_3} \geq 1-e^{-1}(1-p_8)^3.$$

Since $f_a = f_{a_1} = 1$, we have

$$p_a + \frac{1}{3}p_{a_3} \geq 0.736(f_a + \frac{1}{3}f_{a_3}).$$

If a_3 has two neighbors. Let the other neighbor of a_3 is i_3.

ii. i_3 is in $\alpha 1$ with respect to a_3. Let the other neighbor of i_3 be a_4. According to the discussion in (2c-i),

$$p_{a_3} + p_{a_4} \geq 1 - e^{-1}(1-p_1)(1-p_6) + p_4.$$

Since $f_a = f_{a_3} = 1, f_{a_4} = 1/3$, we have

$$p_a + p_{a_3} + p_{a_4} \geq 0.739(f_a + f_{a_3} + f_{a_4}).$$

iii. i_3 is in $\alpha 2$ with respect to a_3. Let the other neighbor of i_3 be a_4. According to the discussion in (2c-ii),

$$p_{a_3} + 0.5p_{a_4} \geq 1 - e^{-1}(1-p_2)(1-p_6) + 0.5p_{10}.$$

Since $f_a = f_{a_3} = 1, f_{a_4} = 2/3$, we have

$$p_a + p_{a_3} + 0.5p_{a_4} \geq 0.738(f_a + f_{a_3} + 0.5f_{a_4}).$$

iv. i_3 is in $\alpha 3$ or $\alpha 4$ with respect to a_3. According to the discussion in (2c-iii),

$$p_{a_3} \geq 1 - e^{-1}(1-p_3)(1-p_6).$$

Since $f_a = f_{a_3} = 1$, we have

$$p_a + p_{a_3} \geq 0.730(f_a + f_{a_3}).$$

□

Lemma 33. *$\exists\{\lambda_a \geq 0\}_{a\in A}$ such that $\sum_a \lambda_a \mu_{a,a'} = 1, \forall a'$.*

Proof. Let us first obtain expression of λ_a for various types of advertisers. Consider an advertiser a such that $f_a = 1$: if a is in case (1a) in the proof of Lemma 32, $\lambda_a = 1 - \#(\text{nodes in (2e) that are of distance 2 from } a)/3$; if a is in (2c) and there exists a node in (2e) that is of distance 2 from a, then $\lambda_a = 0$; otherwise, $\lambda_a = 1$. For all the other advertisers a such that $f_a < 1$, we have $\lambda_a = 1 - \sum_{a':f_{a'}=1} \mu_{a',a}\lambda_{a'}$. We can now verify that for all a, $\lambda_a \geq 0$ and $\sum_{a'} \lambda_{a'}\mu_{a',a} = 1$:

- If a is in case (1a), let $N'_a = \{a' : a' \text{ is a 2-neighbor of } a \text{ and } a' \text{ is in (2e)}\}$. Because $|N'_a| \leq 3$, we have $\lambda_a \geq 0$. On the other hand, from the proof of Lemma 32, $\mu_{a',a} = 1/3$ if $a' \in N'_a$, and 0, otherwise. From the construction of λ above, $\lambda_{a'} = 1$ for $a' \in N'_a$ and $\lambda_a = 1 - |N'_a|/3$. Therefore, $\sum_{a'} \lambda_{a'} \mu_{a',a} = 1$.

- If a is in (2c) and it has a 2-neighbor a_1 who is in (2e), then from the proof of Lemma 32, $\mu_{a_1,a} = 1$ and $\mu_{a',a} = 0$ for all $a' \neq a$ or a_1. From the construction of λ above, $\lambda_a = 0$ and $\lambda_{a_1} = 1$. Therefore, $\sum_{a'} \lambda_{a'} \mu_{a',a} = 1$.

- For all a with $f_a = 1$ and not in the two cases above, $\mu_{a',a} = 0$ for all $a' \neq a$. Since $\lambda_a = 1$, we have $\sum_{a'} \lambda_{a'} \mu_{a',a} = 1$.

- For all a with $f_a < 1$, $\mu_{a',a} = 0$ for all $a' \neq a$ with $f_{a'} < 1$. Because of the construction of λ, $\sum_{a'} \lambda_{a'} \mu_{a',a} = 1$ is trivially true. We will show that $\lambda_a \geq 0$.

  - $f_a = 1/3$. We can show that there is at most one advertiser a' such that $\lambda_{a'} \mu_{a',a} > 0$. Therefore, $\lambda_a \geq 0$.
  - $f_a = 2/3$, and a has only 1 neighbor. We can show that there is at most one advertiser a' such that $\lambda_{a'} \mu_{a',a} > 0$. Therefore, $\lambda_a \geq 0$.
  - $f_a = 2/3$, and a has 2 neighbors. We can show that there is at most two advertisers a' such that $\lambda_{a'} \mu_{a',a} > 0$. Furthermore, for all a', $\mu_{a',a} \leq 1/2$. Therefore, $\lambda_a \geq 0$.

□

Combining the two lemmas above, a conical combination of inequalities leads to our main result:

Theorem 27. $\mathbb{E}[ALG] = \sum_{a \in A} p_a \geq (1 - 2e^{-2}) \sum_{a \in A} f_a \geq (1 - 2e^{-2}) \mathbb{E}[OPT]$.

4.3.5 Tight example

It is worth mentioning that the ratio of $1 - 2e^{-2}$ is tight for the algorithm. The ratio can be achieve in the following example: the underlying graph is a complete bipartite graph $K_{n,n}$ with n even. One optimal solution to LP (4.2) consists of a disjoint unions of $n/2$ cycles of length 4; within each cycle, two of edges carry 1/3 flow, and two carry 2/3 flow. Since the underlying graph is $K_{n,n}$, the optimal offline solution is n. On the other hand, for any cycle in the offline optimal solution, the expected number of matches is $2(1-e^{-2})$. Therefore, the competitive ratio in this instance is $1-2e^{-2} \approx 0.729$.

4.4 Extension to Vertex-Weighted Stochastic Matching

In this section, we consider the online stochastic vertex-weighted matching problem as defined in Aggarwal et al. [3]. The problem is exactly the same as the online stochastic matching problem introduced in Section 4.1 except for the objective function. In the weighted problem, every advertiser a has a nonnegative weight w_a, indicating his importance or value. The objective is to maximize the sum of weights of matched advertisers rather than the number of matched advertisers as in the unweighted problem.

The techniques used in Section 4.3.4 are based on local properties of graphs and thus also work for the vertex-weighted case.

4.4.1 Original algorithm

Let us consider the maximum flow problem on the graph G:

$$\begin{aligned} \max \quad & \sum_{a,i} w_a f_{a,i} & \\ s.t. \quad & \sum_i f_{a,i} \le 1 & \forall a \in A \\ & \sum_a f_{a,i} \le 1 & \forall i \in I \\ & f_e \in [0, 2/3] & \forall e \in E \end{aligned} \tag{4.3}$$

Again, since the set of vertices of the feasible polytope of (4.3) is a subset of $\{0, 1/3, 2/3\}^E$, there exists an optimal solution to (4.3) in $\{0, 1/3, 2/3\}^E$, and let $\mathbf{f}$ be such an optimal solution that satisfies requirements in Section 4.3.2. To ease the analysis, we try to make $E_{\mathbf{f}}$ as sparse as we can by doing the following two types of transformations as we did in Section 4.3.2. As argued before, there exists an optimal solution $\mathbf{f}$ such that its resulting graph $G_{\mathbf{f}}$ does not contain cycles of length 4, unless the four nodes in such a cycle do not have any other neighbors. Furthermore, if there exists impression i that has two neighbors a_1 and a_2 with $f_{i,a_1} = f_{i,a_2} = 1/3$, $f_{a_1} < 1$, and $f_{a_2} < 2$, without loss of generality, we assume $w_{a_1} < w_{a_2}$. Another solution $\mathbf{f}'$ with $f'_{i,a_1} = 0$, $f'_{i,a_2} = 2/3$, and everything else unchanged has a larger or equal objective value, and less edges in its resulting graph. We transform $\mathbf{f}$ to $\mathbf{f}'$. Note that each time, transformations mentioned above remove one or two edges and does not introduce any new edge. Thus, given any initial optimal solution, after at most $|E|$ transformations, the optimal solution cannot be transformed further in the above two ways.

Based on $\mathbf{f}$, the probability distributions over lists can be constructed as in 4.3.1, and the same idea as in 4.3.4 leads to the proof that $p_a \geq 0.682 f_a$ for all $a \in A$. Summing up these inequalities, we have $\sum_{a\in A} w_a p_a \geq 0.682 \sum_{a \in A} w_a f_a$, which implies that the algorithm is 0.682-competitive.

Worth noting that although Lemma 32 and Lemma 33 still hold, they are of little value to weighted problems because of different weights associated with different advertisers. Out of the same reason, results and techniques proposed in previous papers dealing with unweighted stochastic matching problems are unlikely to be adapted for weighted problems.

4.4.2 Modification

However, modifying $\mathbf{f}$ and construction of random lists leads to a better algorithm. If i has neighbors with $f = 1$ and $f < 1$, as showed in Figure 4-6, $\mathbf{f}$ will be modified as follows: in (1), $\tilde{f}_{a_1,i} = 0.1$ and $\tilde{f}_{a_2,i} = 0.9$; in (2), $\tilde{f}_{a_1,i} = 0.15$ and $\tilde{f}_{a_2,i} = 0.85$; in (3), $\tilde{f}_{a_1,i} = 0.6$ and $\tilde{f}_{a_2,i} = 0.4$; in (4), $\tilde{f}_{a_1,i} = 0.1$, $\tilde{f}_{a_2,i} = 0.45$ and $\tilde{f}_{a_3,i} = 0.45$; in (5), $\tilde{f}_{a_1,i} = 0.15$, $\tilde{f}_{a_2,i} = 0.425$ and $\tilde{f}_{a_3,i} = 0.425$. For all the other edges e, $\tilde{f}_e = f_e$.

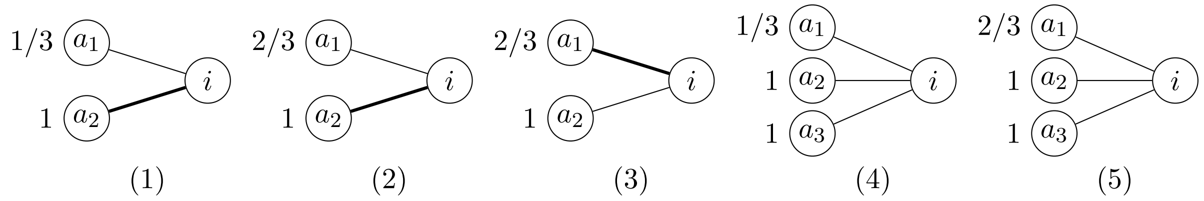


Figure 4-6: Modification of $\mathbf{f}$. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a.

Now use $\tilde{\mathbf{f}}$ instead of $\mathbf{f}$ for the construction of the probability distributions over lists in a way similar to the one described in Section 4.3.1 as follows. Given an impression type i, if it has two neighbors a_1 and a_2 in the resulting graph, the list is $\langle a_1, a_2 \rangle$ with probability $\tilde{f}_{a_1,i}$; the list is $\langle a_2, a_1 \rangle$ with probability $\tilde{f}_{a_1,i}$. If i has three neighbors a_1, a_2, and a_3; the list is $\langle a_j, a_k, a_l \rangle$ with probability $\tilde{f}_{a_j,i}\tilde{f}_{a_k,i}/(1 - \tilde{f}_{a_j,i})$.

Let $\tilde{p}_a$ be the probability that advertiser a is matched in the modified algorithm. Using the same idea as in Section 4.3.4, we can then show that:

Lemma 34. *$\tilde{p}_a \geq 0.725 f_a, \forall a \in A$.*

Proof. As discussed in Section 4.3.1, the left case in Figure 4-1 is the only possible cycle in the resulting graph. Let N be the number of impressions of type i_1 or i_2. Then, $\tilde{p}_{a_1} + \tilde{p}_{a_2} = \Pr(N = 1) + 2\Pr(N \geq 2) = 2 - 4e^{-2}$. Because of the symmetry between a_1 and a_2, $\tilde{p}_{a_1} = \tilde{p}_{a_2} = 1 - 2e^{-2} = 0.729$.

From now on, we only need to consider advertisers a who are not part of cycles of length 4. Therefore, the supporting sets of their certificate events do not intersect, thus are asymptotically independent.

We first consider the case $f_a = 1$. We can show case by case that:

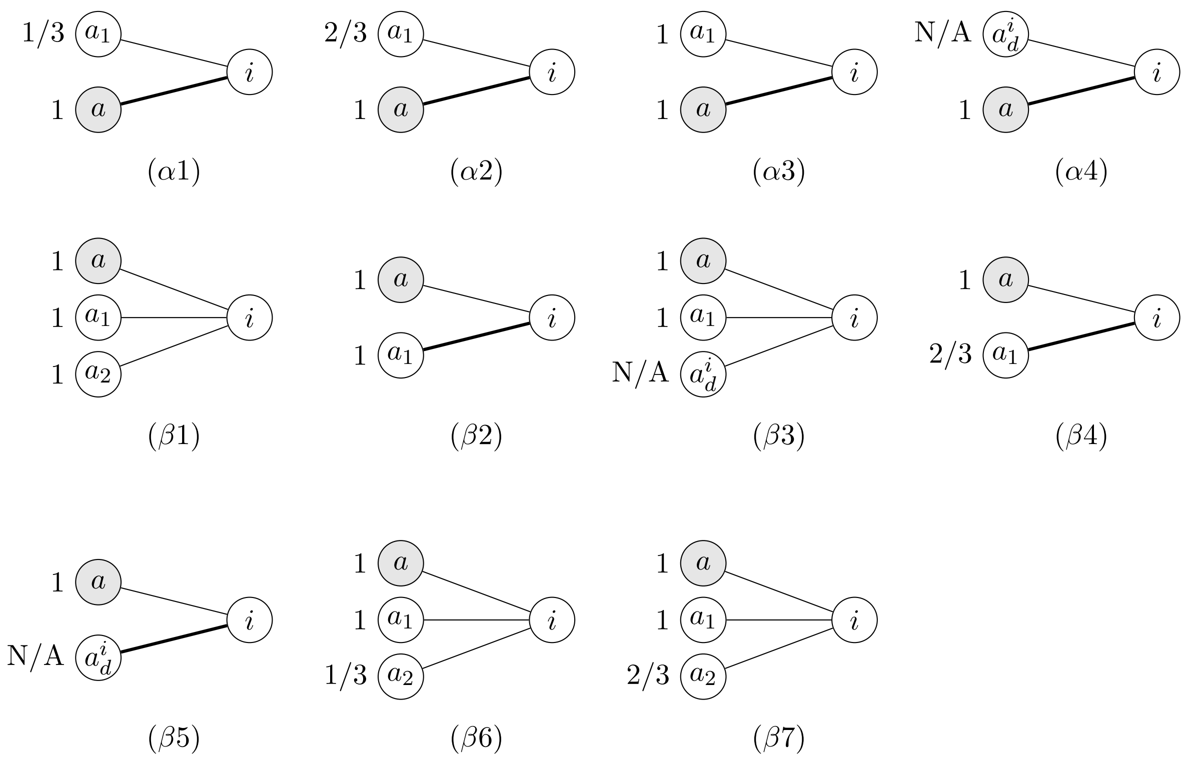


Figure 4-7: Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a.

Claim 9. *$\forall a$ with $f_a = 1$, $p_a \geq 0.7250 f_a$.*

Proof. Let us first compute probabilities of certificate events:

α) $f_{a,i} = 2/3$.

If i has 2 neighbors a and a_1, then $f_{a_1,i} = 1/3$.

$\alpha 1$. $f_{a_1} = 1/3$. We use a Markov Chain approach to approximate $\Pr(G_a^i)$. The state space consists of three states: "a is full" (state 1), "a is empty and a_1 is full" (state 2), and "a is empty and a_1 is empty" (state 3). The transition probabilities are $p_{32} = 0.1/n, p_{33} = 1 - 0.1/n, p_{22} = 0.1/n, p_{21} = 1 - 0.1/n$, and $p_{11} = 1$. The initial probability distribution is (0,0,1), i.e. both a and a_1 are empty. $\Pr(G_a^i)$ is the probability of state 1 after n time step. We use $n = 10^6$ here and for all other cases:

$$\Pr(G_a^i) \quad \geq \quad 0.0047 (\triangleq p_1)$$

The same idea can be used to compute the probability for all cases. The only difference is the size of state space, and the transition probability. Please note that we can also calculate $\Pr(G_a^i)$ exactly, as we did in the proof of Lemma 32.

$\alpha 2$. $f_{a_1} = 2/3$.

$$\Pr(G_a^i) \quad \geq \quad 0.0194 (\triangleq p_2)$$

$\alpha 3$. $f_{a_1} = 1$.

$$\Pr(G_a^i) \quad \geq \quad 0.1091 (\triangleq p_3)$$

If i has only one neighbor:

$\alpha 4$.

$$\Pr(G_a^i) \quad \geq \quad 0.2835 (\triangleq p_4)$$

β) $f_{a,i_1} = 1/3$.

If i has 3 neighbors a, a_1 and a_2. Then at least one of f_{a_1} or f_{a_2} is 1; otherwise, we can find another solution that has less non-zero flow edges and a better objective value.

$\beta 1$. $f_{a_1} = f_{a_2} = 1$.

$$\Pr(G_a^i) \quad \geq \quad 0.1608 (\triangleq p_5)$$

$\beta 6$. $f_{a_1} = 1$ and $f_{a_2} = 1/3$.

$$\Pr(G_a^i) \quad \geq \quad 0.1396 (\triangleq p_6)$$

$\beta 7$. $f_{a_1} = 1$ and $f_{a_2} = 2/3$.

$$\Pr(G_a^i) \geq 0.1304(\triangleq p_7)$$

If i has 2 neighbors a and a_1. Note that $f_{a_1,i} = 1/3$ and $f_{a_1} < 1$ cannot happen together; otherwise, $\mathbf{f}$ cannot be a maximum flow.

$\beta 2$. $f_{a_1,i} = 2/3$ and $f_{a_1} = 1$.

$$\Pr(G_a^i) \geq 0.1955(\triangleq p_8)$$

$\beta 3$. $f_{a_1,i} = 1/3$ and $f_{a_1} = 1$.

$$\Pr(G_a^i) \geq 0.2992(\triangleq p_9)$$

$\beta 4$. $f_{a_1,i} = 2/3$ and $f_{a_1} = 2/3$.

$$\Pr(G_a^i) \geq 0.1219(\triangleq p_{10})$$

If i_1 has only one neighbor:

$\beta 5$.

$$\Pr(G_a^i) \geq 0.4866(\triangleq p_{11})$$

We are now ready to compute lower bounds on p_a when $f_a = 1$:

1) a has 3 neighbors i_1, i_2, and i_3.

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2} \cup G_a^{i_3}) \\ &= 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2}))(1 - \Pr(G_a^{i_3})) \\ &\geq 1 - e^{-1}(1 - p_{10})^3 = 0.7509 f_a. \end{aligned}$$

2) a has 2 neighbors i_1 and i_2. Without loss of generality, let us assume that $f_{a,i_1} = 1/3$ and $f_{a,i_2} = 2/3$.

2a. i_2 is in case $\alpha 4$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1 - p_4) = 0.7364 f_a. \end{aligned}$$

2b. i_2 is in case $\alpha 1$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-0.9-1/3}(1 - p_1)(1 - p_{10}) = 0.7449 f_a. \end{aligned}$$

2c. i_2 is in case $\alpha 2$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-0.85-1/3}(1 - p_2)(1 - p_{10}) = 0.7360 f_a. \end{aligned}$$

2d. i_2 is in case $\alpha 3$,

i. i_1 is in case $\beta 1, 2, 3$, or 5, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-1}(1 - p_3)(1 - p_5) = 0.7250 f_a. \end{aligned}$$

ii. i_1 is in case $\beta 4$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-2/3-0.4}(1 - p_3)(1 - p_{10}) = 0.7308 f_a. \end{aligned}$$

iii. i_1 is in case $\beta 6$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-2/3-0.45}(1 - p_3)(1 - p_6) = 0.7491 f_a. \end{aligned}$$

iv. i_1 is in case $\beta 7$, then

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-2/3-0.425}(1 - p_3)(1 - p_7) = 0.7400 f_a. \end{aligned}$$

$\square$

Claim 10. *$\forall a$ with $f_a = 1/3$, $p_a \geq 0.7622 f_a$.*

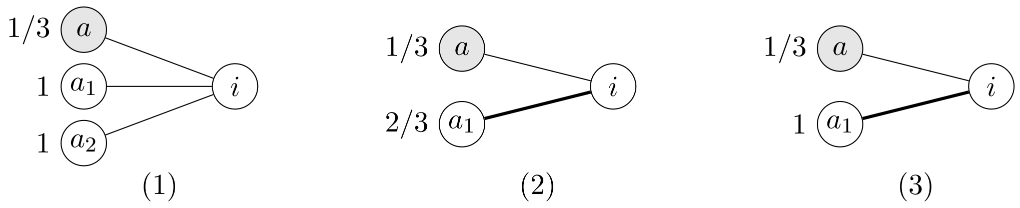


Figure 4-8: Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a.

Proof. There are 3 possible local configurations:

1. The probability of certificate event $\Pr(G_a^i) \geq 0.1756$, thus,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^i) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^i)) \\ &\geq 1 - e^{-0.1}(1 - 0.1756) = 0.2541 = 0.7622 f_a. \end{aligned}$$

2. $p_a \geq \Pr(B_a) = 1 - e^{-1/3} = 0.2835 = 0.8504 f_a$.

3. The probability of certificate event $\Pr(G_a^i) \geq 0.2275$, thus,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^i) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^i)) \\ &\geq 1 - e^{-0.1}(1 - 0.2275) = 0.3010 = 0.9030 f_a. \end{aligned}$$

□

Claim 11. *$\forall a$ with $f_a = 2/3$, $p_a \geq 0.7299 f_a$.*

Proof. Let us first compute the probabilities of certificate events.

α). $f_{a,i} = 1/3$,

$\alpha 1$.

$$\Pr(G_a^i) \geq 0.1748$$

$\alpha 2$.

$$\Pr(G_a^i) \geq 0.1443$$

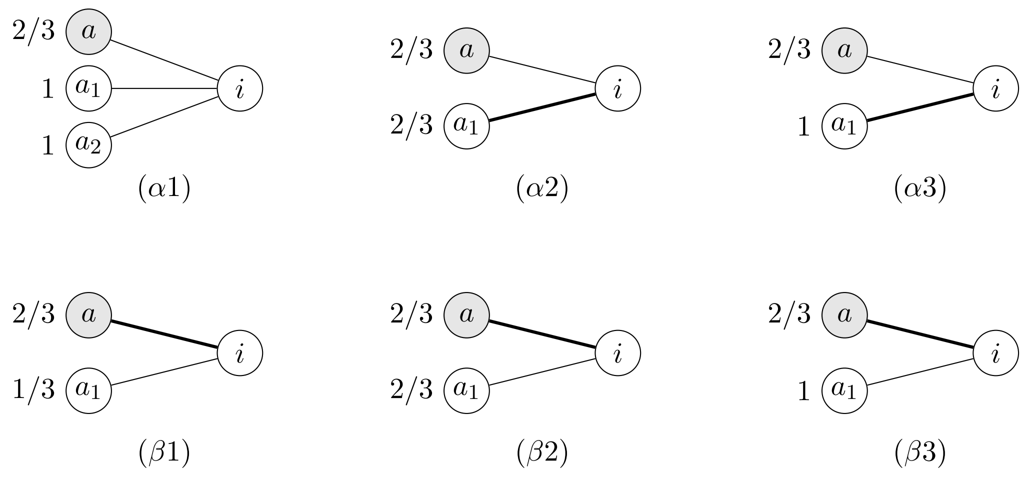


Figure 4-9: Possible configurations of i's neighborhood in the graph. Thin edges carry 1/3 flow, and thick edges carry 2/3 flow. The number next to advertiser a indicates f_a.

$\alpha 2$.

$$\Pr(G_a^i) \;\geq\; 0.1748$$

β). $f_{a,i} = 2/3$,

$\beta 1$.

$$\Pr(G_a^i) \;\geq\; 0$$

$\beta 2$.

$$\Pr(G_a^i) \;\geq\; 0$$

$\beta 3$.

$$\Pr(G_a^i) \;\geq\; 0.2016$$

We are now ready to bound p_a when $f_a = 2/3$.

1). If a has only one neighbor i,

1a. i is in case $\beta 1$ or 2, then $p_a \geq \Pr(B_a) = 1 - e^{-2/3} = 0.4866 = 0.7299 f_a$.

1b. i is in case $\beta 3$, then,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^i) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^i)) \\ &\geq 1 - e^{-0.6}(1 - 0.2016) = 0.5618 = 0.8427 f_a. \end{aligned}$$

2). If a has two neighbors i_1 and i_2,

2a. If neither of i_1 or i_2 is in case $\alpha 2$,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-0.3}(1 - 0.1748)^2 = 0.4955 = 0.7433 f_a. \end{aligned}$$

2b. If at least one of i_1 or i_2 is in case $\alpha 2$,

$$\begin{aligned} p_a &\geq \Pr(B_a \cup G_a^{i_1} \cup G_a^{i_2}) = 1 - (1 - \Pr(B_a))(1 - \Pr(G_a^{i_1}))(1 - \Pr(G_a^{i_2})) \\ &\geq 1 - e^{-0.15-1/3}(1 - 0.1443)^2 = 0.5484 = 0.8226 f_a. \end{aligned}$$

□

In conclusion, combining the results of Claims 5, 6, and 7, we obtain the proof of Lemma 7.

□

Summing these inequalities up, we have:

Theorem 28. $\mathbb{E}[ALG] = \sum_{a \in A} w_a \tilde{p}_a \geq 0.725 \sum_{a \in A} w_a f_a \geq 0.725 \mathbb{E}[OPT]$.

4.5 Unweighted Stochastic Matching Problems Revisit

In this section, we use a new approach to analyze the algorithm proposed in Section 4.3.

Let A' be the set of all nodes that are contained in cycles of length 4 in $G_{\mathbf{f}}$. As argued in Section 4.3.1, every cycle of length 4 in $G_{\mathbf{f}}$ is a connected component; and according to case 1 in the proof of Lemma 32,

$$\sum_{a \in A'} p_a \geq \sum_{a \in A'} (1 - 2e^{-2}) f_a.$$

Thus, we only need to consider bins in $A \backslash A'$. For simplicity, in the remaining part of the section, we remove all cycles of length 4 in $G_{\mathbf{f}}$.

We then divide all nodes and edges into different subgroups as follows: all ball type nodes are divided into the following 11 subgroups, according to their adjacent nodes in $G_{\mathbf{f}}$:

- $\mathcal{I}_1 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 1/3, f_{a_2,i} = 2/3$, and $f_{a_2} = 1\}$.

- $\mathcal{I}_2 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 2/3, f_{a_2,i} = 2/3$, and $f_{a_2} = 1\}$.

- $\mathcal{I}_3 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 1, f_{a_2,i} = 2/3$, and $f_{a_2} = 1\}$.

- $\mathcal{I}_4 = \{i | i$ has exactly one neighbor a_1, $f_{a_1,i} = 2/3$ and $f_{a_1} = 1\}$.

- $\mathcal{I}_5 = \{i | i$ has exactly three neighbors a_1, a_2, a_3, $f_{a_1,i} = f_{a_2,i} = f_{a_3,i} = 1/3$ and $f_{a_1} = f_{a_2} = f_{a_3} = 1\}$.

- $\mathcal{I}_6 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 1, f_{a_2,i} = 1/3$, and $f_{a_2} = 1\}$.

- $\mathcal{I}_7 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 1, f_{a_2,i} = 2/3$, and $f_{a_2} = 2/3\}$.

- $\mathcal{I}_8 = \{i | i$ has exactly one neighbor a_1, $f_{a_1,i} = 1/3$ and $f_{a_1} = 1\}$.

- $\mathcal{I}_9 = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 1/3, f_{a_2,i} = 2/3$, and $f_{a_2} = 2/3\}$.

- $\mathcal{I}_{10} = \{i | i$ has exactly two neighbors a_1, a_2, $f_{a_1,i} = 1/3, f_{a_1} = 2/3, f_{a_2,i} = 2/3$, and $f_{a_2} = 2/3\}$.

- $\mathcal{I}_{11} = \{i | i$ has exactly one neighbor a_1, $f_{a_1,i} = 2/3$ and $f_{a_1} = 2/3\}$.

and let y_k be the number of bins in subgroup $\mathcal{I}_k$.

All bin nodes are divided into the following subgroups, according to their adjacent nodes in $G_{\mathbf{f}}$:

- $\mathcal{A}_k^1 = \{a | a$ has exactly one neighbor i, $f_{a,i} = 1/3$ and $i \in \mathcal{I}_k\}, (k \in \{1, 9\})$.

- $\mathcal{A}_k^2 = \{a | a$ has exactly one neighbor i, $f_{a,i} = 2/3$ and $i \in \mathcal{I}_k\}, (k \in \{7, 9, 10, 11\})$.

- $\mathcal{A}_{k_1,k_2}^3 = \{a | a$ has exactly two neighbors i_1 and i_2, $f_{a,i_1} = f_{a,i_2} = 1/3$, $i_1 \in \mathcal{I}_{k_1}$, and $i_2 \in \mathcal{I}_{k_2}\}, (k_1 \leq k_2 \in \{2, 10\})$.

- $\mathcal{A}_{k_1,k_2}^4 = \{a | a$ has exactly two neighbors i_1 and i_2, $f_{a,i_1} = 1/3, f_{a,i_2} = 2/3$, $i_1 \in \mathcal{I}_{k_1}$, and $i_2 \in \mathcal{I}_{k_2}\}, (k_1 \in \{3, 5, 6, 7, 8\}, k_2 \in \{1, 2, 3, 4\})$.

- $\mathcal{A}^5_{k_1,k_2,k_3} = \{a | a$ has exactly three neighbors i_1, i_2 and i_3, $f_{a,i_1} = f_{a,i_2} = f_{a,i_3} = 1/3$, $i_1 \in \mathcal{I}_{k_1}$, $i_2 \in \mathcal{I}_{k_2}$, and $i_3 \in \mathcal{I}_{k_3}\}$, $(k_1 \leq k_2 \leq k_3 \in \{3, 5, 6, 7, 8\})$.

and let x^1_k, x^2_k, $x^3_{k_1,k_2}$, $x^4_{k_1,k_2}$, and $x^5_{k_1,k_2,k_3}$ be the number of balls in the respective groups.

All edges in $G_{\mathbf{f}}$ are divided into the following 17 subgroups, according to their endpoints:

- $\mathcal{E}_1 = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1/3$ and $i \in \mathcal{I}_1\}$.
- $\mathcal{E}_2 = \{a = (a, i) | f_{a,i} = 2/3, f_a = 1$ and $i \in \mathcal{I}_1\}$.
- $\mathcal{E}_3 = \{a = (a, i) | f_{a,i} = 1/3, f_a = 2/3$ and $i \in \mathcal{I}_2\}$.
- $\mathcal{E}_4 = \{a = (a, i) | f_{a,i} = 2/3, f_a = 1$ and $i \in \mathcal{I}_2\}$.
- $\mathcal{E}_5 = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1$ and $i \in \mathcal{I}_3\}$.
- $\mathcal{E}_6 = \{a = (a, i) | f_{a,i} = 2/3, f_a = 1$ and $i \in \mathcal{I}_3\}$.
- $\mathcal{E}_7 = \{a = (a, i) | f_{a,i} = 2/3, f_a = 1$ and $i \in \mathcal{I}_4\}$.
- $\mathcal{E}_8 = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1$ and $i \in \mathcal{I}_5\}$.
- $\mathcal{E}_9 = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1$ and $i \in \mathcal{I}_6\}$.
- $\mathcal{E}_{10} = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1$ and $i \in \mathcal{I}_7\}$.
- $\mathcal{E}_{11} = \{a = (a, i) | f_{a,i} = 2/3, f_a = 2/3$ and $i \in \mathcal{I}_7\}$.
- $\mathcal{E}_{12} = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1$ and $i \in \mathcal{I}_8\}$.
- $\mathcal{E}_{13} = \{a = (a, i) | f_{a,i} = 1/3, f_a = 1/3$ and $i \in \mathcal{I}_9\}$.
- $\mathcal{E}_{14} = \{a = (a, i) | f_{a,i} = 2/3, f_a = 2/3$ and $i \in \mathcal{I}_9\}$.
- $\mathcal{E}_{15} = \{a = (a, i) | f_{a,i} = 1/3, f_a = 2/3$ and $i \in \mathcal{I}_{10}\}$.
- $\mathcal{E}_{16} = \{a = (a, i) | f_{a,i} = 2/3, f_a = 2/3$ and $i \in \mathcal{I}_{10}\}$.
- $\mathcal{E}_{17} = \{a = (a, i) | f_{a,i} = 2/3, f_a = 2/3$ and $i \in \mathcal{I}_{11}\}$.

It is easy to see, the number of edges in any edge subgroup can be counted in two ways, using bin nodes or using ball type nodes:

- $|\mathcal{E}_1| = y_1 = x^1_1$.

- $|\mathcal{E}_2| = y_1 = \sum_{k_1} x^4_{k_1,1}$.
- $|\mathcal{E}_3| = y_2 = \sum_{k_2} x^3_{2,k_2} + \sum_{k_1} x^3_{k1,2}$.
- $|\mathcal{E}_4| = y_2 = \sum_{k_1} x^4_{k_1,2}$.
- $|\mathcal{E}_5| = y_3 = \sum_{k_2} x^4_{3,k_2} + \sum_{k_2,k_3} x^5_{3,k_2,k_3} + \sum_{k_1,k_3} x^5_{k_1,3,k_3} + \sum_{k_1,k_2} x^5_{k_1,k_2,3}$.
- $|\mathcal{E}_6| = y_3 = \sum_{k_1} x^4_{k_1,3}$.
- $|\mathcal{E}_7| = y_4 = \sum_{k_1} x^4_{k_1,4}$.
- $|\mathcal{E}_8| = 3y_5 = \sum_{k_2} x^4_{5,k_2} + \sum_{k_2,k_3} x^5_{5,k_2,k_3} + \sum_{k_1,k_3} x^5_{k_1,5,k_3} + \sum_{k_1,k_2} x^5_{k_1,k_2,5}$.
- $|\mathcal{E}_9| = 2y_6 = \sum_{k_2} x^4_{6,k_2} + \sum_{k_2,k_3} x^5_{6,k_2,k_3} + \sum_{k_1,k_3} x^5_{k_1,6,k_3} + \sum_{k_1,k_2} x^5_{k_1,k_2,6}$.
- $|\mathcal{E}_{10}| = y_7 = \sum_{k_2} x^4_{7,k_2} + \sum_{k_2,k_3} x^5_{7,k_2,k_3} + \sum_{k_1,k_3} x^5_{k_1,7,k_3} + \sum_{k_1,k_2} x^5_{k_1,k_2,7}$.
- $|\mathcal{E}_{11}| = y_7 = x^2_{11}$.
- $|\mathcal{E}_{12}| = y_8 = \sum_{k_2} x^4_{8,k_2} + \sum_{k_2,k_3} x^5_{8,k_2,k_3} + \sum_{k_1,k_3} x^5_{k_1,8,k_3} + \sum_{k_1,k_2} x^5_{k_1,k_2,8}$.
- $|\mathcal{E}_{13}| = y_9 = x^1_9$.
- $|\mathcal{E}_{14}| = y_9 = x^2_9$.
- $|\mathcal{E}_{15}| = y_{10} = \sum_{k_2} x^3_{10,k_2} + \sum_{k_1} x^3_{k1,10}$.
- $|\mathcal{E}_{16}| = y_{10} = x^2_{10}$.
- $|\mathcal{E}_{17}| = y_{11} = x^2_{11}$.

Similar to the approach in Section 4.3, we can derive lower bounds p^1_k, p^2_k, $p^3_{k_1,k_2}$, $p^4_{k_1,k_2}$, and $p^5_{k_1,k_2,k_3}$ on the probability that bins in respective bin subgroups are filled:

- $p^1_k = 1 - e^{-\frac{1}{3}}(1 - q^1_k)$, where $q^1_1 = 1 - 3e^{-\frac{2}{3}} + 2e^{-1}$ and $q^1_9 = 1 - \frac{5}{3}e^{-\frac{2}{3}}$.
- $p^2_k = 1 - e^{-\frac{2}{3}}(1 - q^2_k)$, where $q^2_7 = 1 - \frac{3}{2}e^{-\frac{1}{3}} + \frac{1}{2}e^{-1}$, $q^2_9 = 1 - \frac{4}{3}e^{-\frac{1}{3}}$, $q^2_{10} = 1 - 2e^{-\frac{1}{3}} + e^{-\frac{2}{3}}$, and $q^2_{11} = 1 - e^{-\frac{1}{3}}$.
- $p^3_{k_1,k_2} = 1 - e^{-\frac{2}{3}}(1 - q^3_{k_1})(1 - q^3_{k_2})$, where $q^3_2 = 1 - 3e^{-\frac{2}{3}} + 2e^{-1}$ and $q^3_{10} = 1 - \frac{5}{3}e^{-\frac{2}{3}}$.

- $p^4_{k_1,k_2} = 1 - e^{-1}(1-p^5_{k_1})(1-p^4_{k_2})$, where $p^4_1 = 1 - \frac{4}{3}e^{-\frac{1}{3}}$, $p^4_2 = 1 - 2e^{-\frac{1}{3}} + e^{-\frac{2}{3}}$, $p^4_3 = 1 - \frac{3}{2}e^{-\frac{1}{3}} + \frac{1}{2}e^{-1}$, $p^4_4 = 1 - e^{-\frac{1}{3}}$, $q^5_3 = 1 - 3e^{-\frac{2}{3}} + 2e^{-1}$, $q^5_6 = 1 - \frac{7}{4}e^{-\frac{2}{3}} + \frac{3}{4}e^{-\frac{4}{3}}$, $q^5_5 = 1 - \frac{21}{8}e^{-\frac{2}{3}} + \frac{9}{4}e^{-\frac{4}{3}} - \frac{5}{8}e^{-2}$, $q^5_7 = 1 - \frac{5}{3}e^{-\frac{2}{3}}$, and $q^5_8 = 1 - e^{-\frac{2}{3}}$.

- $p^5_{k_1,k_2,k_3} = 1 - e^{-1}(1-q^5_{k_1})(1-q^5_{k_2})(1-q^5_{k_3})$ where $q^5_3 = 1 - 3e^{-\frac{2}{3}} + 2e^{-1}$, $q^5_6 = 1 - \frac{7}{4}e^{-\frac{2}{3}} + \frac{3}{4}e^{-\frac{4}{3}}$, $q^5_5 = 1 - \frac{21}{8}e^{-\frac{2}{3}} + \frac{9}{4}e^{-\frac{4}{3}} - \frac{5}{8}e^{-2}$, $q^5_7 = 1 - \frac{5}{3}e^{-\frac{2}{3}}$, and $q^5_8 = 1 - e^{-\frac{2}{3}}$.

Then, $\mathbf{p}'\mathbf{x}$ is a lower bound on the online objective value $\sum_{a\in A\setminus A'} p_a$. On the other hand, it is easy to see that

$$\sum_{a\in A\setminus A'} f_a = \frac{1}{3}\sum_k x^1_k + \frac{2}{3}\sum_k x^2_k + \frac{2}{3}\sum_{k_1,k_2} x^3_{k_1,k_2} + \sum_{k_1,k_2} x^4_{k_1,k_2} + \sum_{k_1,k_2,k_3} x^5_{k_1,k_2,k_3}.$$

For simplicity, let us scale both $\mathbf{x}$ and $\mathbf{y}$ such that $\sum_{a\in A\setminus A'} f_a = 1$. Note that the equalities due to counting edges in edge subgroups are not affected by scaling.

Since the optimal solution to LP

$$\begin{array}{rl} \max & \mathbf{p}'\mathbf{x} \\ s.t. & \text{equalities constraints due to counting edges in each edge subgroup} \\ & \frac{1}{3}\sum\limits_k x^1_k + \frac{2}{3}\sum\limits_k x^2_k + \frac{2}{3}\sum\limits_{k_1,k_2} x^3_{k_1,k_2} + \sum\limits_{k_1,k_2} x^4_{k_1,k_2} + \sum\limits_{k_1,k_2,k_3} x^5_{k_1,k_2,k_3} = 1 \\ & \mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0} \end{array}$$

is 0.7363, we can show that

$$\sum_{a\in A\setminus A'} p_a \geq 0.7363 \sum_{a\in A\setminus A'} f_a.$$

Combined with

$$\sum_{a\in A'} p_a \geq (1-2e^{-2}) \sum_{a\in A'} f_a,$$

we get our result

$$\sum_{a\in A} p_a \geq (1-2e^{-2}) \sum_{a\in A} f_a.$$

4.6 Stochastic Matching with General Arrival Rates

In this section, we assume that $r_i \leq 1$ for all i. The algorithm and basic ideas here are very similar to [57]: in the offline stage, we approximate the expected offline optimal solution;

then, in the online stage, use the approximation solution to generate lists of length two. However, our algorithm is different in two aspects. First, we use a max flow problem instead of Monte Carlo methods to approximate the offline solution; second, the way lists are generated is different. The first difference leads to much less computation in the offline stage, while the second difference results in a slightly better competitive ratio.

4.6.1 Offline solution

One possible approach to find useful offline information in the general case is to use sampling methods to estimate the optimal offline solution, as described in Manshadi et al. [57]. However, some properties that hold for the optimal offline solution may not hold for the estimated one. Furthermore, a large number of samples may be needed in order to estimate the offline optimal solution within a desirable accuracy, which takes a long time. Therefore, we consider the following LP instead:

$$\begin{aligned} \max \quad & \sum_{a,i} f_{a,i} & & \\ s.t. \quad & \sum_{i\sim a} f_{a,i} \le 1 & & \forall a \in A \\ & \sum_{a\sim i} f_{a,i} \le r_i & & \forall i \in I \\ & \sum_{i\sim a} (2f_{a,i} - r_i)^+ \le 1 - \ln 2 + \tfrac{1}{n} & & \forall a \in A \\ & f_e \ge 0 & & \forall e \in E \end{aligned} \tag{4.4}$$

Note that that LP(4.4) is equivalent to a single-source s single-destination t maximum flow problem on a directed network $\hat{G} = \{\hat{V}, \hat{E}\}$ with $|A| + 2|I| + 2$ vertices and $2|E| + |A| + 2|I|$ arcs. The vertex set $\hat{V} = \{s, t\} \cup A \cup I \cup I'$, where I' is a duplicate of I, and the arc set $\hat{E} = \{(s, a), (a, i), (i', i), (a, i'), (i, t) | a \in A, i \in I, i \text{ is a duplicate copy of } i\}$. The capacity of (s, a) is 1; the capacity of (a, i) is $r_i/2$; the capacity of (i', i) is $1 - \ln 2 + 1/n$; the capacity of (i, t) is r_i; (a, i') have infinite capacities.

4.6.2 Upper bound on the optimal offline solution

Let $\mathbf{f}^*$ be an optimal offline solution. All but the third constraints in (4.4) are trivially valid for $\mathbf{f}^*$. The third constraint has been proven in [57]:

Lemma 35. *[[57], Lemma 5]* $\sum_{i\sim a} (2f^*_{a,i} - r_i)^+ \le 1 - \ln 2 + \frac{1}{n}, \forall a \in A.$

4.6.3 Randomized algorithm

For simplicity, let us again first add a dummy advertiser a_d with $f_{a_d} \triangleq 1$, and dummy edges (a_d, i) for all i with $f_{a_d,i} \triangleq r_i - \sum_{a\in A} f_{a,i}$. The dummy advertiser is full at the very beginning. Every time an impression of type i arrives, a random list consisting of two advertisers will be generated as follows. Assume $a_1, ..., a_k$ are the advertisers interested in i. Choose a random number x uniformly over $[0, r_i]$. If $x \in [\sum_{l=1}^{j-1} f_{a_l,i}, \sum_{l=1}^{j} f_{a_l,i}]$, then a_j is the first advertiser in the list to be considered; if $x \pm r_i/2 \in [\sum_{l=1}^{k-1} f_{a_l,i}, \sum_{l=1}^{k} f_{a_l,i}]$ then a_k is the second in the list to be considered. Worth noting is the possibility that a_j and a_k correspond to the same advertiser; in that case, the list degenerates to a singleton.

Let $m^i_{a_j,a_k}$ be the expected number of requests for impressions of type i and corresponding lists given by $\langle a_j, a_k\rangle$. Since all lists are i.i.d., the probability that an impression is of type i and its corresponding list is $\langle a_j, a_k\rangle$ is $m^i_{a_j,a_k}/n$. From the construction of the lists, we have $m^i_{a_j,a_k} = m^i_{a_k,a_j}$. As we mentioned in Section 4.2, from a given realization of the sequence of random lists, we can find the cardinality of the corresponding online matching. Since the random list associated with the j^{th} request only depends on the impression type of that request, and not on types and assignments of earlier requests, these random lists are all i.i.d.. Thus, we can focus on the lists themselves, rather than on the impression types that they are associated with. Then, $m_{a_j,a_k} \triangleq \sum_{i\in I} m^i_{a_j,a_k}$ is the expected number of lists that are $\langle a_j, a_k\rangle$, irrespective of the impression types which they are associated with. Furthermore, because $m^i_{a_j,a_k} = m^i_{a_k,a_j}$, we have $m_{a_j,a_k} = m_{a_k,a_j}$. Since all lists are i.i.d., the probability that a list is $\langle a_j, a_k\rangle$ is $m_{a_j,a_k}/n$.

4.6.4 Lower bound on the online algorithm

The analysis here is almost the same as in [57] except for some minor changes due to the different ways we generate random lists, e.g. $m_{a_j,a_k} = m_{a_k,a_j}$. To help better understand the arguments, we present the full proof in this section. The following main result is proved by way of successive claims.

Theorem 29. $\sum\limits_{a\in A} p_a \geq 0.706 \sum\limits_{a\in A} f_a$.

Let $A_a = A\backslash\{a\}$, $A^* = A \cup \{a_d\}$, and $A^*_a = A^*\backslash\{a\}$. $\forall a \in A^*, a_1 \in A^*_a$, define events $B_a = \{\exists j$, such that $L_j = \langle a, .\rangle\}$, $E_{a_1,a_2} = \{\exists j < k$ such that $L_j = \langle a_1, .\rangle, L_k = \langle a_1, a_2\rangle\}$, and $E_{a_d,a} = \{\exists j$, such that $L_j = \langle a_d, a\rangle\}$. If any of B_a, $E_{a_1,a}$, and $E_{a_d,a}$ happens, then advertiser a is matched. Thus, the probability p_a that advertiser a is matched is at least:

$$\begin{aligned}
p_a &\geq \Pr(B_a) + \Pr(\bar{B}_a)\Pr(\bigcup_{a_1 \in A_a^*} E_{a_1,a}|\bar{B}_a) \\
&\geq 1 - e^{-f_a} + e^{-1}\Big(\sum_{a_1 \in A_a^*} \Pr(E_{a_1,a}) - \tfrac{1}{2}\sum_{a_1 \neq a_2 \in A_a^*} \Pr(E_{a_1,a}, E_{a_2,a})\Big) \\
&\geq 1 - e^{-f_a} + e^{-1}\Big(\sum_{a_1 \in A_a^*} \Pr(E_{a_1,a}) - \tfrac{1}{2}\sum_{a_1 \neq a_2 \in A_a^*} \Pr(E_{a_1,a})\Pr(E_{a_2,a})\Big),
\end{aligned}$$

where the last two inequalities are due to asymptotic independence. The proof of asymptotic independence is similar to the proof of Lemma 31, and is omitted in the thesis.

Let us now provide a way to compute $\Pr(E_{a_1,a})$.

Claim 12. *We have $\Pr(E_{a_1,a}) = g(f_{a_1}, m_{a_1,a})$ for all $a_1 \in A$ and $a \in A_{a_1}^*$, and $\Pr(E_{a_d,a}) \geq g(f_{a_d}, m_{a_d,a})$ for all $a \in A$, where:*

$$g(y,x) = h(y,0) - h(y,x), \quad \textit{and} \quad h(y,x) = \begin{cases} \dfrac{y}{y-x}(e^{-x} - e^{-y}), & \text{if } x \neq y \\ ye^{-y}, & \text{if } x = y \end{cases}$$

Proof. Define $F_{a_1}^j = \{\text{the } j^{th} \text{ list is } \langle a_1, . \rangle\}$ and $G_{a_1,a}^j = \{\text{there exists } k \geq j \text{ such that the } k^{th} \text{ list is } \langle a_j, a\rangle\}$. Then,

$$\begin{aligned}
\Pr(E_{a_1,a}) &= \sum_j \Pr(F_{a_1}^j)\Pr(G_{a_1,a}^{j+1}) \\
&= \sum_j \big(1 - \frac{f_{a_1}}{n}\big)^{j-1}\frac{f_{a_1}}{n}\Big(1 - \big(1 - \frac{m_{a_1,a}}{n}\big)^{n-j}\Big) \\
&\approx \sum_j \frac{f_{a_1}}{n} e^{-\frac{j}{n}f_{a_1}}\big(1 - e^{-\frac{n-j}{n}m_{a_1,a}}\big).
\end{aligned}$$

If $m_{a_1,a} \neq f_{a_1}$,

$$\begin{aligned}
\Pr(E_{a_1,a}|\bar{B}_a) &= \sum_j \frac{f_{a_1}}{n}\big(e^{-\frac{j}{n}f_{a_1}} - e^{-m_{a_1,a}}e^{-\frac{j}{n}(f_{a_1} - m_{a_1,a})}\big) \\
&= 1 - e^{-f_{a_1}} - \frac{f_{a_1}}{f_{a_1} - f_{a_1,a}} e^{-f_{a_1,a}}\big(1 - e^{-(f_{a_1} - f_{a_1,a})}\big) = g(f_{a_1}, f_{a_1,a}).
\end{aligned}$$

If $m_{a_1,a} = f_{a_1}$,

$$\begin{aligned}
\Pr(E_{a_1,a}|\bar{B}_a) &= \sum_j \frac{f_{a_1}}{n}\big(e^{-\frac{j}{n}f_{a_1}} - e^{-f_{a_1}}\big) \\
&= 1 - e^{-f_{a_1}} - f_{a_1}e^{-f_{a_1}} = g(f_{a_1}, f_{a_1}).
\end{aligned}$$

□

We have transformed a probabilistic problem into an algebraic problem. In the remaining part of the section, we only use algebraic manipulations and the following properties of functions g and h to find a lower bound of the competitive ratio.

Claim 13. *For $y \in [0,1]$, $h(y,x)$ is convex and decreasing in $x \in [0,y]$; $g(y,x)$ is concave and increasing in $x \in [0,y]$; $g(y,x)$ is increasing in $y \in [x,\infty)$.*

Proof. The claim can be easily verified by taking first and second order partial derivatives.

□

Because of the convexity of h in the second argument, we have

$$\Pr(E_{a_1,a}) = h(f_{a_1},0) - h(f_{a_1},m_{a_1,a}) \le -m_{a_1,a} \cdot \frac{\partial h}{\partial y}(f_{a_1},0) \le e^{-1} m_{a_1,a}, \tag{4.5}$$

implying that $\sum_{a_1 \in A_a} \Pr(E_{a_1,a}|\bar{B}_a) \le e^{-1}$. Combined with $\Pr(E_{a_1,a}|\bar{B}_a) \ge g(f_{a_1},m_{a_1,a})$ for all $a_1 \in A_a^*$, we have

$$p_a \ge 1 - e^{-f_a} + e^{-1}\Big(\sum_{a_1 \in A_a^*} g(f_{a_1},m_{a_1,a}) - \frac{1}{2}\big(\sum_{a_1 \in A_a^*} g(f_{a_1},m_{a_1,a})\big)^2 + \frac{1}{2}\sum_{a_1 \in A_a^*} g(f_{a_1},m_{a_1,a})^2\Big).$$

Since g is increasing in the first argument, we have $g(f_{a_d},m_{a_d,a}) = g(f_{a_d},m_{a,a_d}) \ge g(f_a,m_{a,a_d})$ for all $a \in A$. Thus,

$$\sum_{a \in A}\sum_{a_1 \in A_a^*} g(f_{a_1},m_{a_1,a}) \quad \ge \quad \sum_{a \in A}\sum_{a_1 \in A_a^*} g(f_a,m_{a,a_1})$$

and

$$\sum_{a \in A}\sum_{a_1 \in A_a^*} g(f_{a_1},m_{a_1,a})^2 \quad \ge \quad \sum_{a \in A}\sum_{a_1 \in A_a^*} g(f_a,m_{a,a_1})^2 \ .$$

Therefore, by switching the order of summation, we have

$$\begin{aligned}
\frac{\sum\limits_{a\in A} p_a}{\sum\limits_{a\in A} f_a} &\geq \frac{\sum\limits_{a\in A}\Big(1-e^{-f_a}+\frac{1}{e}\sum\limits_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})-\frac{1}{2e}\big(\sum\limits_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})\big)^2+\frac{1}{2e}\sum\limits_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})^2\Big)}{\sum\limits_{a\in A} f_a} \\
&\geq \frac{\sum\limits_{a\in A}\Big(1-e^{-f_a}+\frac{1}{e}\sum\limits_{a_1\in A_a^*} g(f_a,m_{a,a_1})-\frac{1}{2e}\big(\sum\limits_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})\big)^2+\frac{1}{2e}\sum\limits_{a_1\in A_a^*} g(f_a,m_{a,a_1})^2\Big)}{\sum\limits_{a\in A} f_a} \\
&\geq \min_{a\in A}\frac{1-e^{-f_a}+\frac{1}{e}\sum\limits_{a_1\in A_a^*} g(f_a,m_{a,a_1})-\frac{1}{2e}\big(\sum\limits_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})\big)^2+\frac{1}{2e}\sum\limits_{a_1\in A_a^*} g(f_a,m_{a,a_1})^2}{f_a}
\end{aligned}$$

Let $\beta_a = \max\limits_{a_1\in A_a^*} m_{a,a_1} \triangleq m_{a,a_1^*}$, $s_a = \sum\limits_{a_1\in A_a^*} m_{a,a_1}$. Then, we have $\sum\limits_{a_1\in A_a^*} g(f_a,m_{a,a_1})^2 \geq g(f_a,\beta_a)^2$. Furthermore, because g is concave in the second argument and $g(f_a,0)=0$,

$$\sum_{a_1\in A_a^*} g(f_a,m_{a,a_1}) \geq \sum_{a_1\in A_a^*}\frac{m_{a,a_1}}{\beta_a}g(f_a,\beta_a) = \frac{s_a}{\beta_a}g(f_a,\beta_a).$$

On the other hand, from inequality (4.5),

$$\sum_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a}) = \sum_{a_1\in A_a^*\setminus\{a_1^*\}} g(f_{a_1},m_{a_1,a}) + g(f_{a_1},\beta_a) \leq e^{-1}(s_a-\beta_a)+g(1,\beta_a).$$

Therefore,

$$\begin{aligned}
&\tfrac{1}{f_a}\big(1-e^{-f_a}+\tfrac{1}{e}\sum_{a_1\in A_a^*} g(f_a,m_{a,a_1})-\tfrac{1}{2e}\big(\sum_{a_1\in A_a^*} g(f_{a_1},m_{a_1,a})\big)^2+\tfrac{1}{2e}\sum_{a_1\in A_a^*} g(f_a,m_{a,a_1})^2\big) \\
\geq\ &\tfrac{1}{f_a}\big(1-e^{-f_a}+\tfrac{1}{e}\tfrac{s_a}{\beta_a}g(f_a,\beta_a)-\tfrac{1}{2e}\big(e^{-1}(s_a-\beta_a)+g(1,\beta_a)\big)^2+\tfrac{1}{2e}g(f_a,\beta_a)^2\big) \triangleq R(f_a,\beta_a,s_a).
\end{aligned}$$

From the definitions of f_a, β_a, and s_a, we have $f_a \geq s_a \geq \beta_a$, and $f_a - s_a$ is the expected number of lists that are singletons $\langle a\rangle$. From the construction of lists, the expected number of singletons $\langle a\rangle$ associated with impressions of types i is $(2f_{a,i}-r_i)^+$. Thus, $f_a - s_a = \sum_{i\sim a}(2f_{a,i}-r_i)^+ \leq (1-\ln 2)+1/n$. We can numerically show that, for $n \geq 100$:

Claim 14. *Subject to $1 \geq f_a \geq s_a \geq \beta_a \geq 0$ and $f_a - s_a \leq (1-\ln 2)+1/n$, $R(f_a,\beta_a,s_a) \geq 0.706$.*

Proof. We divide the feasible region into cubes of side length 0.001. In each small region S, define $f_a^{\max} \triangleq \sup f_a$ and $f_a^{\min} \triangleq \inf f_a$. Define $s_a^{\max}$, $s_a^{\min}$, $\beta_a^{\max}$, and $\beta_a^{\min}$ similarly. Then, from Claim 13, we can show that $\forall(f_a, s_a, \beta_a) \in S$, $R(f_a, s_a, \beta_a)$ is bounded from below by

$$\frac{1}{f_a^{\max}}\big(1-e^{-f_a^{\min}}+\frac{1}{e}\frac{s_a^{\min}}{\beta_a^{\max}}g(f_a^{\min},\beta_a^{\min})-\frac{1}{2e}\big(e^{-1}(s_a^{\max}-\beta_a^{\min})+g(1,\beta_a^{\max})\big)^2+\frac{1}{2e}g(f_a^{\min},\beta_a^{\min})^2\big).$$

We can numerically verify $R(f_a, s_a, \beta_a) \geq 0.706$ in each region. The lower bound is achieved when $f_a \in [0.999, 1]$, $s_a \in [0.692, 0.693]$, and $\beta_a \in [0.564, 0.565]$. Hence, $R(f_a, s_a, \beta_a) \geq 0.706$ is a valid inequality for the whole feasible region. $\square$

Theorem 29 follows from Claim 14.

4.7 Poisson Arrivals

In the preceding sections, the number of arriving requests is assumed to be fixed and known in advance. However, in most applications, such an assumption is too strong. Thus, in this section, we attempt to relax this assumption.

In this section, we consider the following scenario. A set of advertisers express their interests in impressions of different types. Advertisers are fixed and known ahead of time while requests for impressions come online. Impression types are i.i.d., and the distribution may be known or unknown. The arrival of impressions is a Poisson Process with arrival rate $\lambda = n$. The task is to maximize the cardinality of matching by the end of a given fixed period $T = 1$.

4.7.1 Algorithms

The expected number of arrivals is $\lambda T = n$. We show that, greedy algorithms designed for stochastic matching with given number of arrivals works well for the one with Poisson arrivals (e.g. the ranking algorithm for problems with unknown distribution, our proposed algorithms in the previous sections for problems with known distribution). More specifically, we will show that a c-competitive "greedy-type" algorithm (where c is the ratio of expectation) for fixed arrivals is $c - \epsilon$ competitive for Poisson arrivals.

Because the number of Poisson arrivals concentrates around its mean, we expect both online and offline objective to concentrate around their means.

Lemma 36. *Let N be the number of arrivals within* $[0, T]$. *Then,* $\Pr((1-\epsilon)\lambda T < N < (1+\epsilon)\lambda T) \to 1$ *as $\lambda T \to \infty$ for any $\epsilon > 0$.*

Let OPT_m be the expected offline optimal solution given $N = m$ and OPT be the expected offline optimal solution.

Lemma 37. *$\forall (1-\epsilon)n < m \leq n$, $OPT_m \leq OPT_n$; $\forall n \leq m < (1+\epsilon)n$, $OPT_m \leq (1+\epsilon)OPT_n$.*

Proof. $\forall (1-\epsilon)n < m \leq n$, an instance τ_m of m arrivals can be generated in the following way: generate an instance τ_n of n arrivals first, and then remove $n-m$ arrivals uniformly at random. Since τ_m is a subset of τ_n, $OPT(\tau_m) \leq OPT(\tau_n)$. By taking expectation, we have $OPT_m \leq OPT_n$.

$\forall n \leq m < (1+\epsilon)n$, an instance τ_n of n arrivals can be generated in the following way: generate an instance τ_m of m arrivals first, and then remove $m-n$ arrivals uniformly at random. A feasible solution of τ_n can be induced by the optimal solution of τ_m, by removing pairs corresponding to removed arrivals and not adding any other pairs. The feasible solution of τ_n has expected value of $\frac{n}{m}OPT(\tau_m)$. Thus, $OPT(\tau_m) \leq \frac{m}{n}OPT(\tau_n) \leq (1+\epsilon)OPT(\tau_n)$. $\square$

As a consequence we have:

Corollary 2. *$OPT \leq (1+\epsilon)OPT_n$.*

The unweighted case

Let ALG_m be the expected online solution given $N = m$ and ALG be the expected online solution.

Lemma 38. *$\forall (1-\epsilon)n < m \leq n, ALG_m \geq (1-\epsilon)ALG_n$; $\forall n \leq m < (1+\epsilon)n, ALG_m \geq ALG_n$.*

Proof. $\forall n \leq m < (1+\epsilon)n$, an instance τ_n of n arrivals can be generated in the following way: generate an instance τ_m of m arrivals first, and then remove the last $m-n$ arrivals. Because of the greediness of the algorithm, $ALG_m \geq ALG_n$.

$\forall (1-\epsilon)n < m \leq n$, let r_i be the probability that the i^{th} arrival is matched. Because of the greediness of the algorithm and the fact that less and less bins are unmatched, r_i is non-increasing. Since $ALG_m = \sum_{i=1}^{m} r_i$ and $ALG_n = \sum_{i=1}^{n} r_i$, we have $ALG_m \geq \frac{m}{n}ALG_n \geq (1-\epsilon)ALG_n$. $\square$

As a consequence we have:

Corollary 3. $ALG \geq (1-2\epsilon)ALG_n$.

Because of the assumption of c-competitiveness, $ALG_n \geq c \cdot OPT_n$. Therefore Corollaries 2 and 3 imply that $ALG \geq (1-2\epsilon)c \cdot OPT$.

The weighted case

Let ALG_m be the expected online solution given $N = m$ and ALG be the expected online solution. Let R_i be the marginal revenue in the i^{th} step. For the algorithm proposed in Section 4.4.1 and 4.4.2, we can show that although $\mathbb{E}[R_i]$ is not non-increasing as in the unweighted case, R_j cannot be too large compared to $\mathbb{E}[R_i]$ for $i < j$. Specifically:

Lemma 39. $\mathbb{E}[R_j] \leq 9\mathbb{E}[R_i], \forall i < j$.

Proof. Let I be the indicator vector of availability of advertisers right after step $i-1$. Given I, if an advertiser has zero probability to be matched to a query at step i, he has zero probability to be matched to a query at step j. Given I, if he has non-zero probability to be matched at step i, then the probability is at least $1/3n$; on the other hand, with probability at most $3/n$, he is matched at step j. From the discussion above, we have $\mathbb{E}[R_j|I] \leq 9\mathbb{E}[R_j|I]$. By taking expectation over I, we have our lemma. □

Lemma 40. *$\forall (1-\epsilon)n < m \leq n, ALG_m \geq (1-9\epsilon)ALG_n$; $\forall n \leq m < (1+\epsilon)n, ALG_m \geq ALG_n$.*

Proof. $\forall n \leq m < (1+\epsilon)n$, an instance τ_n of n arrivals can be generated in the following way: generate an instance τ_m of m arrivals first, and then remove the last $m-n$ arrivals. Because of the greediness of the algorithm, $ALG_m \geq ALG_n$.

$\forall (1-\epsilon)n < m \leq n$, $ALG_m = \sum_{i=1}^{m} \mathbb{E}[R_i]$ and $ALG_n = \sum_{i=1}^{n} \mathbb{E}[R_i]$. From the above lema, $ALG_m \geq \frac{m}{m+9(n-m)} ALG_n \geq (1-9\epsilon)ALG_n$. □

As a consequence:

Corollary 4. $ALG \geq (1-9\epsilon)ALG_n$.

Because of the assumption of c-competitiveness, $ALG_n \geq c \cdot OPT_n$. Therefore Corollaries 2 and 4 imply that $ALG \geq (1-10\epsilon)c \cdot OPT$.

Remarks

As we can see, the only property of the Poisson distributed random variables we use is that they concentrate around their means. Hence, if the number of arriving queries is a random variable concentrating around its means, the results in this section still apply.

Chapter 5

Online Resource Allocation Problems

5.1 Introduction

As discussed in Chapter 1, online optimization is attracting wide attention from computer science and operations research communities. It has many applications, including those dealing with dynamic resource allocation problems. In many real-world problems, information about the instance to optimize is not completely known ahead of time, but revealed in an online fashion. For example, in typical revenue management problems, customers arrive sequentially offering a price for a subset of commodities, e.g. multi-leg flights. The seller must make irrevocable decisions to accept or reject customers at their arrivals, and try to maximize long-term overall revenue while respecting various resource constraints. Another example is the so-called AdWords problem, also known as the display ads problem. From keyword search queries arriving online, the problem is to sequentially allocate ad slots to budget-constrained bidders/advertisers. Similar problems appear in online routing problems, online packing problems, online auctions, and various internet advertising display applications.

In this chapter, we consider a general online linear programming that covers many of the examples mentioned above. To be precise about the problem, we need to introduce some notations. Let I be a set of m resources; associated with each resource $i \in I$ is a capacity b_i. The set of resources and their capacities are known ahead of time. Let J be a set of n customers; each customer has a set of options O_j and arrival time t_j. We assume that every customer has a bounded number of options, i.e. there exists a constant q such that $|O_j| \leq q$ for all j. Each option $o \in O_j$ has a value π_{jo} and requires a_{ijo} units of resources i for each $i \in I$, also written $\mathbf{a_{jo}}$ as a vector of dimension m. The set of options O_j and associated $(\pi_{jo}, \mathbf{a_{jo}})$ are revealed at time t_j when customer j arrives. Upon arrival, the online algorithm must decide immediately and irrevocably whether or not to satisfy the customer, and, if yes, which option to choose. The goal is to find a solution that maximizes the overall revenue from customers while respecting resource constraints. More precisely, we consider the following linear program:

$$\begin{array}{lll} \max & \sum_j \sum_{o \in O_j} \pi_{jo} x_{jo} & \\ s.t. & \sum_{j,o} a_{ijo} x_{jo} \leq b_i, & \forall i \\ & \sum_{o \in O_j} x_{jo} \leq 1, & \forall j \\ & x_{jo} \geq 0, & \forall j, o \end{array} \tag{5.1}$$

where $\forall j, \boldsymbol{\pi}_j \in (0,1]^{|O_j|}, \mathbf{a_j} \in [0,1]^{m \times |O_j|}$, and $\mathbf{b} \in \mathbb{R}^m_+$. In the online version of this problem, $(\boldsymbol{\pi}_j, \mathbf{a_j})$ is revealed only when customer j arrives at time t_j. Upon that arrival, and constrained by irrevocable decisions $x_{j'o}$ made for customers arriving earlier, the online algorithm must then make decisions x_{jo}, such that

$$\begin{array}{ll} \sum_{j': t_{j'} \leq t_j} \sum_{o \in O_{j'}} a_{ij'o} x_{j'o} \leq b_i, & \forall i \\ \sum_{o \in O_j} x_{jo} \leq 1 & \\ x_{jo} \geq 0, & \forall o \in O_j \end{array} \tag{5.2}$$

The goal is to choose the variables $\mathbf{x}$ such that the objective function $\sum_j \sum_{o \in O_j} \pi_{jo} x_{jo}$ is maximized.

Several models (on how online instances are chosen) can be used to evaluate online algorithms, including the adversarial model, the i.i.d. model with or without knowledge of distributions, and the random permutation model. In the adversarial setting, no further assumption is made on the model. In that case, no online algorithm can achieve better than $O(1/n)$ fraction of the optimal offline solution [14]. However, as the adversarial setting is too conservative, it is natural to consider stochastic models. In the i.i.d. model with known distribution about future customers, positive results have been obtained for various problems. For many practical problems, such an assumption may be too strong, and the i.i.d. model without knowledge of the distribution would be more suitable. A weaker model, but easier to analyze, the random permutation model, has been considered more frequently. In that model, the order of customers is a uniform random permutation, and many near-optimal results have been obtained for it.

In this chapter, we propose a new model, closer to the random permutation model, but removing a fundamental, yet practically questionable, assumption behind it. In all results using the random permutation model, the exact knowledge about the total number of customers to come is a key assumption, essential for ensuring near-optimality results. Without such information, no non-trivial result can be achieved. In many practical settings, including all the applications discussed above, this assumption is however far from being realistic. We consider instead a more realistic and natural setting, initially using the following two assumptions (the consequences of the relaxations of these two initial assumptions will also be considered in this chapter):

Assumption 1. *Customers have i.i.d. random arrival times.*

The assumption is reasonable in many practical problems where customers' arrival rates are homogeneous throughout time. Ignoring the specific arrival times, the order of customers is essentially equivalent to the random permutation model. Later in the chapter, we will relax the assumption and take heterogeneity of arrival rates into account.

Assumption 2. *The distribution governing random arrival times is known to the online algorithm.*

The assumption is necessary to estimate the total number of customers in case no past data is available. However, as discussed in Section 5.4, if a limited amount of past data is available, this assumption is not needed anymore.

For simplicity in the presentation of the results, we make two additional technical assumptions, which can be removed without compromising the validity of our results, as we explain below.

Assumption 3. *The arrival time is modeled as a continuous random variable.*

No matter what the nature of the original random variable is, we can add an auxiliary random variable t_j^a, uniformly distributed between $[0, 1]$ for every customer j upon his arrival. We define a total ordering on pairs (t_j, t_j^a) based on lexical order. Note that the order of customers is preserved except for those who arrive exactly at the same time. The artificial ordering imposed on these customers does not help an online algorithm.

Assumption 4. *There are no degeneracies among all points* $\{(\pi_{jo}, \mathbf{a_{jo}})\}_{j,o}$ *and* $(0, \mathbf{0})$*, i.e. no* $m+2$ *points share the same* m*-dimensional hyperplane.*

If this is not the case, we can introduce a random perturbation on π_{jo}: every π_{jo} is multiplied by an i.i.d. random variable uniformly distributed between $[1, 1+\epsilon]$. After the perturbation, there are no degeneracies almost surely. On the other hand, because the perturbation is small enough, the optimal value of the solution is affected by no more than a multiple factor of $1+\epsilon$.

5.1.1 Our techniques and contributions

The online algorithms proposed in the chapter share similar ideas with some other papers [26][4][33]: the algorithms first observe (without making any allocation) customers arriving early over a given period of time, and solve an offline LP problem over those customers. The corresponding optimal dual solution then works as a pricing mechanism for making online allocations on the following set of customers. The dual prices are updated from time to time to depict customers' preference more accurately as time moves along. We prove that such algorithms are $1-\epsilon$ competitive under several different scenarios if resource capacities are large enough.

Our results significantly improves previous results by removing the need to know a priori the number of customers n, a critical assumption in [26][4][33]. To the best of our knowledge, this is the first attempt to do so. As pointed out in several papers, knowing n is so essential that no near-optimal online algorithms can be obtained even under a probabilistic version of that assumption. So a new model, with near-optimal online algorithm aspiration, would need to introduce alternative assumptions.

We believe our model with arrival time fits reality better: In practice, the setting that an online problem is more likely to face typically involves a known fixed period of time over which the customers are considered, rather than a known fixed number of customers to come. The question that a company usually asks is how to maximize revenue over a given period of time instead of how to maximize revenue over a given fixed number of future customers. The arrival time of a customer is also more natural and informative than his rank order. Furthermore, our model is more flexible as it allows, depending on specific applications, various extensions which can better fit real-world scenarios. For example, in airline revenue management problems, business customers and casual customers have different price-sensitivity and arrival time. The random permutation model cannot capture the heterogeneity among customers well. In contrast, our model can easily be extended to such scenarios, as demonstrated in Section 5.5.

We first consider problems where the distribution of arrival time is known in advance. Although similar in form, the previous approaches with fixed number of customers do not address our model well. One could first estimate the number of total arrivals in the early stage, and then use the estimation for the fixed-number algorithm as in [4]. However, the performance of this approach depends on the quality of the estimation. In order to keep the loss due to the estimation below ϵ-fraction, the estimation error must be within ϵ-fraction. According to concentration laws, it requires the total number of customers be at least $O(1/\epsilon^3)$. Noting that b_i's are only required to be $O(1/\epsilon^2)$, this new requirement on the total number of customers is quite restrictive. On the other hand, our approach works for any number of customers, even if the number is smaller than $O(1/\epsilon^2)$.

We then consider two scenarios for which our initial assumptions are relaxed. In the first scenario, we do not assume the knowledge of the exact distribution, but some past observations instead. Instead of estimating the cumulative distribution function (CDF) on every point, we only make estimation on only a few critical points. This approach requires much less data points than the naive one. In the second scenario, we consider heterogeneous customers.

5.2 One-Time Learning

Let $F(\cdot)$ be the cumulative distribution function of the random arrival time of customers. Assumption 3 ensures that its inverse $F^{-1}(\cdot)$ is well defined. Consider $S_\epsilon = \{j : t_j \leq F^{-1}(\epsilon)\}$, the set of customers arriving earlier than $F^{-1}(\epsilon)$. From Assumption 1, every customer belongs to S_ϵ with probability ϵ.

The online algorithm observes customers in S_ϵ, rejects them all, and then computes dual prices by solving the following primal dual LPs:

$$
\begin{array}{llll}
\max & \sum_{j\in S_\epsilon, o\in O_j} \pi_{jo}x_{jo} & \min & \sum_i (1-\epsilon)\epsilon b_i p_i + \sum_{j\in S_\epsilon} q_j \\
s.t. & \sum_{j\in S_\epsilon, o\in O_j} a_{ijo}x_{jo} \le (1-\epsilon)\epsilon b_i, \forall i & s.t. & \sum_i a_{ijo}p_i + q_j \ge \pi_{jo}, \forall j\in S_\epsilon, o\in O_j \\
 & \sum_{o\in O_j} x_{jo} \le 1, \forall j \in S_\epsilon & & p_i \ge 0, \forall i \\
 & x_{jo} \ge 0, \forall j\in S_\epsilon, o\in O_j & & q_j \ge 0, \forall j \in S_\epsilon
\end{array}
\tag{5.3}
$$

Let $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$ be the optimal primal and dual solutions for these problems.

For customers arriving later, they are accepted if payments exceed the threshold set by $\hat{\mathbf{p}}$:

$$
x_{jo}(\hat{\mathbf{p}}) = \begin{cases} 1, \text{ if } \pi_{jo} - \sum_i a_{ijo}\hat{p}_i > \max_{o'\ne o}\{\pi_{jo'} - \sum_i a_{ijo'}\hat{p}_i, 0\} \\ 0, \text{ otherwise} \end{cases} \tag{5.4}
$$

The proposed online algorithm is then as follows:

Algorithm 7 (Online Learning Algorithm(OLA))**.**

1. *Reject all customers arriving earlier than* $F^{-1}(\epsilon)$.

2. *Let* $\mathbf{x_j} = \mathbf{x_j}(\hat{\mathbf{p}})$ *for customers arriving after* $F^{-1}(\epsilon)$.

We call a customer j degenerate if the maximizer for $\max_o\{\pi_{jo} - \sum_i a_{ijo}\hat{p}_i\}$ is not unique or $\pi_{jo} - \sum_i a_{ijo}\hat{p}_i = 0$. Degeneracies may lead to undesired results. Fortunately, due to Assumption 4, there are at most $m+1$ degenerate customers, and all of them, if any, are in S_ϵ. For degenerate j, the decision rule $\mathbf{x_j}(\hat{\mathbf{p}}) = \mathbf{0}$. For non-degenerate j, using complementary slackness, $\mathbf{x_j}(\hat{\mathbf{p}})$ equals the optimal solution $\hat{\mathbf{x}}_\mathbf{j}$ to LP (5.3), as stated in the following lemma:

Lemma 41. *For non-degenerate* $j \in S_\epsilon$, $\hat{x}_{jo} = x_{jo}(\hat{\mathbf{p}})$ *for all* $o \in O_j$.

Proof. If there exists $o \in O_j$ such that $\hat{x}_{jo} > 0$. From complementary slackness, we have $\sum_i a_{ijo}\hat{p}_i + \hat{q}_j = \pi_{jo}$. Since $\forall o' \in O_j$,

$$
\pi_{jo'} - \sum_i a_{ijo'}\hat{p}_i < \hat{q}_j = \pi_{jo} - \sum_i a_{ijo}\hat{p}_i,
$$

we have $x_{jo}(\hat{\mathbf{p}}) = 1$ and $\hat{q}_j > 0$. Since j is non-degenerate, then $\hat{q}_j > 0$. Combined with complementary slackness, we have $\sum_{o'\in O_j} \hat{x}_{jo'} = 1$. Note that $\forall o' \ne o$, $\hat{x}_{jo'} = 0$ because $\pi_{jo'} - \sum_i a_{ijo'}\hat{p}_i < \hat{q}_j$. Hence, $\hat{x}_{jo} = 1 = x_{jo}(\hat{\mathbf{p}})$.

If $\hat{x}_{jo} = 0$ for all $o \in O_j$, then $\hat{q}_j = 0$. Since $\pi_{jo} - \sum_i a_{ijo}\hat{p}_i \leq \hat{q}_j$ and j is non-degenerate, $\pi_{jo} - \sum_i a_{ijo}\hat{p}_i < 0$. Therefore, $x_{jo}(\hat{\mathbf{p}}) = 0$ for all $o \in O_j$. □

In this chapter, we repeatedly use concentration laws to show that some undesired events rarely happen. In particular, we use Bernstein inequalities:

Bernstein Inequalities [19]: Let $X_1, ..., X_n$ be independent zero-mean random variables. Suppose there exists $M > 0$ such that $|X_i| \leq M$ almost surely for all i. Then, $\forall t$,

$$\Pr(\sum_{i=1}^{n} X_i > t) \leq \exp\big(-\frac{t^2/2}{\sum_i \mathbb{E}[X_i^2] + Mt/3}\big).$$

Toward the analysis of our online algorithms, we first show that the resulting solution is feasible with high probability:

Lemma 42. *If* $\min_i b_i \geq 5m\ln(nq/\epsilon)/\epsilon^3$*, then w.p.* $1-\epsilon, \sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \leq b_i$ *for all* i*.*

Proof. From Lemma 41, we have

$$\sum_{j\in S_\epsilon, o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \leq \sum_{j\in S_\epsilon, o} a_{ijo}\hat{x}_{jo} \leq \epsilon(1-\epsilon)b_i.$$

We would like to apply Bernstein inequalities to show that $\sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \geq b_i$ rarely happens. The difficulty is that the random variables $\{a_{ijo}x_{jo}(\hat{\mathbf{p}})\}_{j,o}$ depend on the realization of S via $\hat{\mathbf{p}}$. To get around the issue, let us first fix $\mathbf{p}$ and i, and consider the event

$$\{\sum_{j,o} a_{ijo}x_{jo}(\mathbf{p}) \geq b_i, \sum_{j\in S_\epsilon, o} a_{ijo}x_{jo}(\mathbf{p}) \leq (1-\epsilon)\epsilon b_i\} \tag{5.5}$$

For every customer j, because the arrival time is uniformly distributed between $[0, T]$, $j \in S_\epsilon$ with probability ϵ. Hence,

$$\mathbb{E}[\sum_{j\in S_\epsilon, o} a_{ijo}x_{jo}(\mathbf{p})] = \epsilon \cdot \mathbb{E}[\sum_{j,o} a_{ijo}x_{jo}(\mathbf{p})]$$

Using Bernstein's inequalities, we have

$$\begin{aligned} & \Pr(\textstyle\sum_{j,o} a_{ijo}x_{jo}(\mathbf{p}) \geq b_i, \sum_{j\in S_\epsilon,o} a_{ijo}x_{jo}(\mathbf{p}) \leq (1-\epsilon)\epsilon b_i) \\ \leq\ & \Pr(\textstyle\sum_{j,o} a_{ijo}x_{jo}(\mathbf{p}) \geq b_i, \sum_{j\in S_\epsilon,o} a_{ijo}x_{jo}(\mathbf{p}) - \mathbb{E}[\sum_{j\in S_\epsilon,o} a_{ijo}x_{jo}(\mathbf{p})] \leq -\epsilon^2 b_i) \\ \leq\ & \exp(-\epsilon^3 b_i/4) \leq \epsilon/mn^m. \end{aligned}$$

According to [62], $\mathbb{R}^{|I|}$ can be divided into no more than $(nq)^m$ regions such that all $\mathbf{p}$ in a region lead to the same $\mathbf{x}(\mathbf{p})$. By taking union bounds over all possible $\mathbf{p}$ and i, we have with probability ϵ, there exist i and $\mathbf{p}$ such that (5.5) is true. So:

$$\begin{aligned}
& \Pr(\exists i, \sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \geq b_i) \\
= \quad & \Pr(\exists i, \sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \geq b_i, \sum_{j \in S_\epsilon, o} a_{ijo}x_{jo}(\hat{\mathbf{p}}) \leq (1-\epsilon)\epsilon b_i) \\
\leq \quad & \Pr(\exists i, \mathbf{p}, \sum_{j,o} a_{ijo}x_{jo}(\mathbf{p}) \geq b_i, \sum_{j \in S_\epsilon, o} a_{ijo}x_{jo}(\mathbf{p}) \leq (1-\epsilon)\epsilon b_i) \leq \epsilon.
\end{aligned}$$

By taking the complement, we conclude the lemma. □

After obtaining feasibility, we now compare the online solution with the offline optimal solution OPT. Note that OPT is the optimal solution to LPs:

$$\begin{array}{rlrl}
\max & \sum_{j,o} \pi_{jo}x_{jo} & \min & \sum_i b_i p_i + \sum_j q_j \\
s.t. & \sum_{j,o} a_{ijo}x_{jo} \leq b_i, \forall i & s.t. & \sum_i a_{ijo}p_i + q_j \geq \pi_{jo}, \forall j, o \\
 & \sum_{o \in O_j} x_{jo} \leq 1, \forall j & & p_i \geq 0, \forall i \\
 & x_{jo} \geq 0, \forall j, o & & q_j \geq 0, \forall j
\end{array} \tag{5.6}$$

We now show that the objective value of online solution $\mathbf{x}(\hat{\mathbf{p}})$ is close to OPT:

Lemma 43. *If* $\min_i b_i \geq 5m\ln(nq/\epsilon)/\epsilon^3$*, then w.p.* $1-\epsilon$*,*

$$\sum_{j,o} \pi_{jo}x_{jo}(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT \tag{5.7}$$

Proof. Let us consider the following LP:

$$\begin{array}{rll}
\max & \sum_{j,o} \pi_{jo}x_{jo} & \\
s.t. & \sum_{j,o} a_{ijo}x_{jo} \leq \hat{b}_i, & \forall i \\
 & \sum_{o \in O_j} x_{jo} \leq 1, & \forall j \\
 & x_{jo} \geq 0, & \forall j, o
\end{array} \tag{5.8}$$

where $\hat{b}_i = \sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}})$, if $\hat{p}_i > 0$ and $\hat{b}_i = \max\{\sum_{j,o} a_{ijo}x_{jo}(\hat{\mathbf{p}}), b_i\}$, if $\hat{p}_i = 0$. By complementary slackness, $\mathbf{x}(\hat{\mathbf{p}})$ is the optimal solution to this LP.

We then show that with probability $1-\epsilon$, $\hat{b}_i \geq (1-3\epsilon)b_i, \forall i$. For i such that $\hat{p}_i = 0$, it is trivially true from the definition of $\hat{b}_i$. For i such that $\hat{p}_i > 0$, by complementary slackness, we have $\sum_{j\in S_\epsilon,o}\pi_{jo}\hat{x}_{jo} = (1-\epsilon)\epsilon b_i$. Furthermore, according to Assumption 4 and Lemma 41, at most $m+1$ different j make $x_{jo}(\hat{\mathbf{p}}) \neq \hat{x}_{jo}$. Noting that $\pi_{jo} \in (0,1]$, we have

$$\sum_{j\in S_\epsilon,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}}) \geq \sum_{j\in S_\epsilon,o}\pi_{jo}\hat{x}_{jo} - (m+1) = (1-\epsilon)\epsilon b_i - (m+1) \geq (1-2\epsilon)\epsilon b_i$$

Using the same technique as in the proof of Lemma 42, we can show that

$$\Pr\left(\exists i, s.t. \sum_{j\in S_\epsilon,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}}) \geq (1-2\epsilon)\epsilon b_i, \sum_{j\in S,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}}) \leq (1-3\epsilon)b_i\right) \leq \epsilon.$$

Hence, with probability $1-\epsilon$, $\hat{b}_i \geq (1-3\epsilon)b_i, \forall i$.

In that case, we argue that $\sum_{j,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT$. In fact, assuming $\mathbf{x}^*$ is the optimal solution to LP (5.6), then $(1-3\epsilon)\mathbf{x}^*$ is feasible to LP (5.8). As the optimal solution to LP (5.8), $\hat{\mathbf{x}}(\hat{\mathbf{p}})$ is no worse than $(1-3\epsilon)\mathbf{x}^*$, which concludes the lemma. □

Note that the left hand side of (5.7) includes revenue from customers in S_ϵ, which should be excluded. It is upper bounded by OPT_ϵ, the optimal solution to LP (5.3).

Lemma 44. $\mathbb{E}[OPT_\epsilon] \leq \epsilon \cdot OPT$.

Proof. Note that the optimal dual solution $(\mathbf{p}^*, \mathbf{q}^*)$ to (5.6) is also feasible to the partial dual problem (5.3). Hence, the optimal solution to (5.3): $OPT_\epsilon \leq (1-\epsilon)\epsilon\sum_i b_i p_i^* + \sum_{j\in S} q_j^*$. By taking expectation on both sides, we can conclude our lemma. □

We are now ready to prove the main theorem.

Theorem 30. *If* $\min_i b_i \geq 5m\ln(nq/\epsilon)/\epsilon^3$*, OLA is* $1-O(\epsilon)$ *competitive.*

Proof. From the lemmas above, with probability $1-2\epsilon$ (denoted by $\mathcal{E}_1$), the solution $\mathbf{x}(\hat{\mathbf{p}})$ is feasible and $\sum_{j,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT$. Then,

$$\begin{aligned}
\mathbb{E}[\textstyle\sum_{j\notin S_\epsilon,o\in O_j}\pi_{jo}x_{jo}(\hat{\mathbf{p}})] &\geq \mathbb{E}[\textstyle\sum_{j\notin S_\epsilon,o\in O_j}\pi_{jo}x_{jo}(\hat{\mathbf{p}})|\mathcal{E}_1]\cdot\Pr(\mathcal{E}_1)\\
&\geq (\mathbb{E}[\textstyle\sum_{j,o}\pi_{jo}x_{jo}(\hat{\mathbf{p}})|\mathcal{E}_1] - \mathbb{E}[OPT_\epsilon|\mathcal{E}_1])\Pr(\mathcal{E}_1)\\
&\geq (1-3\epsilon)(1-2\epsilon)OPT - \epsilon OPT \geq (1-6\epsilon)OPT
\end{aligned}$$

□

5.3 Dynamic Pricing

The basic idea of OLA is to compute dual prices for resources, based on customers who arrive early. However, because of the limited number of customers, $\min_i b_i$ is required to be as large as $O(1/\epsilon^3)$ to have a small error probability. A natural question is if sampling more customers can help. The answer is affirmative as showed in this section.

Let $\epsilon = 2^{-E}$, where $E \in \mathbb{N}$. Let $S_l = \{j : t_j < F^{-1}(l)\}$ be the set of customers arriving no later than $F^{-1}(l)(l \in L = \{\epsilon, 2\epsilon, 4\epsilon, ...\})$. Let $\hat{\mathbf{p}}_\mathbf{l}$ denote the optimal dual solution to the following partial LPs:

$$\begin{array}{lll} \max & \sum_{j\in S_l, o\in O_j} \pi_{jo}x_{jo} & \\ s.t. & \sum_{j\in S_l, o\in O_j} a_{ijo}x_{jo} \le (1-h_l)lb_i, & \forall i \\ & \sum_{o\in O_j} x_{jo} \le 1, & \forall j \in S_l \\ & x_{jo} \ge 0, & \forall j \in S_l, o \in O_j \end{array} \tag{5.9}$$

where $h_l = \epsilon\sqrt{1/l}$.

Unlike OLA, DPA updates dual prices multiple times to have better performance:

Algorithm 8 (Dynamic Pricing Algorithm(DPA))**.**

1. *Reject all customers arriving earlier than ϵT.*

2. *Update dual prices $\hat{\mathbf{p}}_\mathbf{l}$ at time $\epsilon T, 2\epsilon T, 4\epsilon T, ...$*

3. *Let $x_j = x_j(\hat{\mathbf{p}}_\mathbf{l})$ for customers arriving between lT and $2lT$.*

The analysis of DPA is very similar to the one of OLA. We show that with high probability, the resulting solution is feasible, the resulting solution is near optimal, and the loss caused by observation process is small.

Lemma 45. *If $\min_i b_i \ge 10m\ln(nq/\epsilon)/\epsilon^2$, then w.p. $1-\epsilon, \sum_{j\in S_{2l}\setminus S_l, o\in O_j} a_{ijo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \le lb_i, \forall i, l$*

Proof. The proof is very similar to the one of Lemma 42. Let us first consider the probability of event

$$\{\sum_{j\in S_l}^{o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \le (1-h_l)lb_i, \sum_{j\in S_{2l}\setminus S_l}^{o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \ge lb_i\} \tag{5.10}$$

for all fixed l, $\mathbf{p_l}$, and i:

$$\begin{array}{rl} & \Pr(\sum_{j\in S_l,o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \le (1-h_l)lb_i, \sum_{j\in S_{2l}\setminus S_l,o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \ge lb_i) \\ \le & \Pr(\sum_{j\in S_l,o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \le (1-h_l)lb_i, \sum_{j\in S_{2l},o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \ge (2-h_l)lb_i) \\ + & \Pr(\sum_{j\in S_{2l}\setminus S_l,o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \ge lb_i, \sum_{j\in S_{2l},o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \le (2-h_l)lb_i). \end{array}$$

Note that $\Pr(j \in S_l | j \in S_{2l}) = 1/2$. From Bernstein inequalities, the first term is upper bounded by $\exp(-\epsilon^2 b_i/10)$. Similarly, the second term is also upper bounded by $\exp(-\epsilon^2 b_i/10)$. Thus,

$$\Pr(\sum_{j\in S_l}^{o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \le (1-h_l)lb_i, \sum_{j\in S_{2l}\setminus S_l}^{o\in O_j} a_{ijo}x_{jo}(\mathbf{p_l}) \ge lb_i) \le 2\exp(-\epsilon^2 b_i/10).$$

Note that for each l, there are at most $(nq)^m$ distinct $\mathbf{p_l}$ regions. By union bounds, we have that with probability ϵ, there exist i, l, and $\mathbf{p_l}$, such that (5.10) is true. On the other hand, from Lemma 41, we have $\sum_{j\in S_l,o\in O_j} a_{ijo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \le \sum_{j\in S_l}^{o\in O_j} a_{ijo}\hat{x}_{jo} \le (1-h_l)lb_i$. By letting $\mathbf{p_l} = \hat{\mathbf{p}}_\mathbf{l}$, we can conclude our lemma. $\square$

Lemma 46. *If* $\min_i b_i \ge 10m\ln(nq/\epsilon)/\epsilon^2$, *then w.p.* $1-\epsilon, \sum_{j\in S_{2l},o\in O_j} \pi_{jo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \ge (1-2h_l-\epsilon)OPT_{2l}, \forall l$.

Proof. Let us consider the following LP:

$$\begin{array}{rll} \max & \sum_{j\in S_{2l},o} \pi_{jo}x_{jo} & \\ s.t. & \sum_{j\in S_{2l},o} a_{ijo}x_{jo} \le \hat{b}_i, & \forall i \\ & \sum_{o\in O_j} x_{jo} \le 1, & \forall j \in S_{2l} \\ & x_{jo} \ge 0, & \forall j \in S_{2l}, o \end{array}, \tag{5.11}$$

where $\hat{b}_i = \sum_{j\in S_{2l}} a_{ijo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l})$, if $\hat{p}_{l,i} > 0$ and $\hat{b}_i = \max\{\sum_{j\in 2l} a_{ijo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l}), b_i\}$, if $\hat{p}_{l,i} = 0$. By complementary slackness, $\mathbf{x}(\hat{\mathbf{p}}_\mathbf{l})$ is the optimal solution to this LP.

On the other hand, using the same argument as in the proof of Lemma 43, we can show that with probability $1-\epsilon$, $\hat{b}_i \ge 2l\cdot(1-2h_l-\epsilon)b_i$ for all i and l. In that case, $\sum_{j\in S_{2l},o} \pi_{jo}x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \ge (1-2h_l-\epsilon)OPT_{2l}$. $\square$

Lemma 47. *Let* OPT_l *be the optimal value to (5.9), then* $\mathbb{E}[OPT_l] \le l\cdot OPT$.

Proof. Consider the optimal dual solution $(\mathbf{p}^*, \mathbf{q}^*)$ to LP (5.1). We can easily check that it is feasible to the dual problem of LP (5.9):

$$\begin{array}{rl} \min & \sum_i (1-h_l) l \epsilon b_i p_i + \sum_{j \in S_l} q_j \\ s.t. & \sum_i a_{ijo} p_i + q_j \geq \pi_{jo}, \forall j \in S_l, o \in O_j \\ & p_i \geq 0, \forall i \\ & q_j \geq 0, \forall j \in S_l \end{array}.$$

Thus, $OPT_l \leq (1-h_l) l \sum_i b_i p_i^* + \sum_{j \in S_l} q_j^*$. Note that for every j, $\Pr(j \in S_l) = l$. By taking expectation on both sides, we can conclude the lemma. □

Combining Lemma 45, 46, and 47, we conclude the main result:

Theorem 31. *If* $\min_i b_i \geq 10m \ln(nq/\epsilon)/\epsilon^2$, *then DPA is* $1 - O(\epsilon)$ *competitive.*

Proof. Let $\mathcal{E}_2$ denote the event that both inequalities in Lemma 45 and 46 are true, then $\Pr(\mathcal{E}_2) \geq 1 - 2\epsilon$.

$$\begin{array}{rl} & \mathbb{E}[\sum\limits_{l \in L} \sum\limits_{j \in S_{2l} \setminus S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l}) | \mathcal{E}_2] \\ \geq & \sum\limits_{l \in L} \mathbb{E}[\sum\limits_{j \in S_{2l}} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l}) | \mathcal{E}_2] - \sum\limits_{l \in L} \mathbb{E}[\sum\limits_{j \in S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l}) | \mathcal{E}_2] \\ \geq & \sum\limits_{l \in L} (1 - 2h_l - \epsilon) \mathbb{E}[OPT_{2l} | \mathcal{E}_2] - \sum\limits_{l \in L} \mathbb{E}[OPT_l | \mathcal{E}_2] \\ \geq & OPT - \sum\limits_{l \in L} 2h_l \mathbb{E}[OPT_{2l} | \mathcal{E}_2] - \epsilon \sum\limits_{l \in L} \mathbb{E}[OPT_{2l} | \mathcal{E}_2] - \mathbb{E}[OPT_\epsilon | \mathcal{E}_2] \\ \geq & OPT - 4 \sum\limits_{l \in L} h_l l \cdot OPT - 2\epsilon \sum\limits_{l \in L} l \cdot OPT - \epsilon OPT \\ \geq & OPT - 13\epsilon OPT \end{array}.$$

Therefore, $\mathbb{E}[\sum_{l \in L} \sum_{j \in S_{2l} \setminus S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l})] \geq (1 - 15\epsilon) OPT$. □

5.4 Learning From the Past

The previous two sections discuss problems where the distribution of customers' arrival time is known to the online algorithm ahead of time. However, the assumption may not be true in many applications. Instead, the online algorithm is more likely to have access to past data rather than the exact distribution. For example, from observation on previous days, a retail store owner may expect that roughly two-thirds of the customers arrive in the afternoon. Specifically, in this section, we assume customers have i.i.d. arrival time with unknown distribution. Furthermore, information about the k past customers $\{t'_k, \mathbf{a'_k}, \boldsymbol{\pi}'_k\}$ is given to the online algorithm. The algorithm proposed in this section only uses arrival times of past customers.

Intuitively, by concentration laws, the distribution $f(\cdot)$ can be estimated arbitrarily well point-wise as k grows. However, point-wise accuracy is unnecessary for our algorithm, and requires a huge amount of data. Note that, only at time $F^{-1}(\epsilon), F^{-1}(2\epsilon), F^{-1}(4\epsilon), ...$ does DPA update its pricing policy. Thus, if we could estimate those quantile points well, we would expect the resulting algorithm has similar performance as DPA.

First, let us show that t'_{lk} is a good estimate of $F^{-1}(l)$. To be more precise,

Lemma 48. *If $k \geq 5\ln(1/\epsilon)/\epsilon^2$, w.p. $1-\epsilon$, $F^{-1}((1-h_l)l) \leq t'_{lk} \leq F^{-1}((1+h_l)l), \forall l \in L = \{\epsilon, 2\epsilon, 4\epsilon, ...\}$. Here $h_l = \epsilon\sqrt{1/l}$.*

Proof. Let N_l^1 be the number of customers arriving between $[0, F^{-1}((1-h_l)l)]$. Then,

$$\Pr(N_l^1 \geq lk) = \Pr(N_l^1 - \mathbb{E}[N_l^1] \geq h_l lk) \leq \exp(-\epsilon^2 k/2).$$

Let N_l^2 be the number of customers arriving between $[0, F^{-1}((1+h_l)l)]$. Then,

$$\Pr(N_l^2 \leq lk) = \Pr(N_l^2 - \mathbb{E}[N_l^1] \leq -h_l lk) \leq \exp(-\epsilon^2 k/4).$$

Noting that $N_l^1 \geq lk$ is equivalent to $t'_{lk} \geq F^{-1}((1-h_l)l)$ and $N_l^2 \leq lk$ is equivalent to $t'_{lk} \leq F^{-1}((1+h_l)l)$. Therefore, by union bound, $F^{-1}((1-h_l)l) \leq t'_{lk} \leq F^{-1}((1+h_l)l), \forall l = \epsilon, 2\epsilon, 4\epsilon, ...$ w.p. $1 - 2\ln(1/\epsilon)\exp(-\epsilon^2 k/4) \geq 1-\epsilon$. □

After obtaining estimates of $F^{-1}(l)$, let us present the online algorithm DPAD. The only difference from DPA is that instead of updating at $F^{-1}(l)$, DPAD updates its pricing policy at t'_{lk}.

Algorithm 9 (Dynamic Pricing Algorithm with Data(DPAD))**.**

1. *Reject all customers arriving earlier than $t'_{\epsilon k}$.*

2. *Update dual prices $\hat{\mathbf{p}}_\mathbf{l}$ at time $t'_{\epsilon k}, t'_{2\epsilon k}, t'_{4\epsilon k}, ...$ according to LP (5.12) given below.*

3. *Let $x_j = x_j(\hat{\mathbf{p}}_\mathbf{l})$ for customers arriving between t'_{lk} and t'_{2lk}.*

$$\begin{array}{lll} \max & \sum_{j\in S_l, o\in O_j} \pi_{jo} x_{jo} & \\ s.t. & \sum_{j\in S_l, o\in O_j} a_{ijo} x_{jo} \leq (1-6h_l) l b_i, & \forall i \\ & \sum_{o\in O_j} x_{jo} \leq 1, & \forall j \in S_l \\ & x_{jo} \geq 0, & \forall j\in S_l, o\in O_j \end{array} \tag{5.12}$$

where $h_l = \epsilon\sqrt{1/l}$ and $S_l = \{j : t_j \leq t'_{lk}\}$.

Let event $\mathcal{E}_{est}$ denote the event where $F^{-1}(\epsilon), F^{-1}(2\epsilon), F^{-1}(4\epsilon), ...$ are well-estimated as in Lemma 48. Given $\mathcal{E}_{est}$, we would expect DPAD has many similar properties as DPA. Indeed, it is the case, and the analysis is almost identical. We show that with high probability, the resulting solution is feasible, the resulting solution is near optimal, and the loss due to observation is small.

Lemma 49. *Given $\mathcal{E}_{est}$, if $\min_i b_i \geq 3m\ln(nq/\epsilon)/\epsilon^2$, then with probability $1-\epsilon$,*

$$\sum_{j\in S_{2l}\setminus S_l, o\in O_j} a_{ijo} x_{jo}(\mathbf{\hat{p}_l}) \leq l b_i, \forall i, l.$$

Proof. Fix $\mathbf{\hat{p}}$, i, and l. Let $X_j = \sum_{o\in O_j} a_{ijo} x_{jo}(\mathbf{\hat{p}})$. Then,

$$\begin{array}{ll} & \Pr(\sum_{j\in S_{2l}\setminus S_l} X_j > l b_i, \sum_{j\in S_l} X_j \leq (1-6h_l) l b_i) \\ \leq & \Pr(\sum_{j\in S_l} X_j \leq (1-6h_l) l b_i, \sum_{j\in S_{2l}} X_j \geq 2(1-3h_l) l b_i) \\ & +\Pr(\sum_{j\in S_{2l}\setminus S_l} X_j \geq l b_i, \sum_{j\in S_{2l}} X_j \leq 2(1-3h_l) l b_i) \end{array}.$$

Since $\Pr(j\in S_l | j \in S_{2l}) = F(t'_{lk})/F(t'_{2lk}) \leq (1+2h_l)/2$, the first term

$$\begin{array}{ll} & \Pr(\sum_{j\in S_l} X_j \leq (1-6h_l) l b_i, \sum_{j\in S_{2l}} X_j \geq 2(1-3h_l) l b_i) \\ \leq & \Pr(\sum_{j\in S_l} X_j - \mathbb{E}[\sum_{j\in S_l} X_j] \leq \min\{-h_l \sum_{j\in S_{2l}} X_j/2, -h_l l b_i\}, \sum_{j\in S_{2l}} X_j \geq 2(1-3h_l) l b_i) \\ \leq & \exp(-\epsilon^2 b_i/3) \end{array}.$$

Since $\Pr(j \in S_{2l} \backslash S_l | j \in S_{2l}) = 1 - \Pr(j \in S_l | j \in S_{2l}) \geq (1-2h_l)/2$, the second term

$$\begin{aligned}
&\Pr(\sum_{j \in S_{2l} \backslash S_l} X_j \geq lb_i, \sum_{j \in S_{2l}} X_j \leq 2(1-3h_l)lb_i) \\
&\leq \Pr(\sum_{j \in S_{2l} \backslash S_l} X_j - \mathbb{E}[\sum_{j \in S_{2l} \backslash S_l} X_j] \geq h_l lb_i, \sum_{j \in S_{2l}} X_j \leq 2(1-3h_l)lb_i) \ . \\
&\leq \exp(-\epsilon^2 b_i/3)
\end{aligned}$$

Therefore, $\Pr(\sum_{j \in S_{2l} \backslash S_l} X_j > lb_i, \sum_{j \in S_l} X_j \leq (1-6h_l)lb_i) \leq 2\exp(-\epsilon^2 b_i/3)$. By taking union bounds over all possible $\mathbf{p_l}$, i, and l, we can conclude the lemma. □

Lemma 50. *Given $\mathcal{E}_{est}$, if $\min_i b_i \geq 3m \ln(nq/\epsilon)/\epsilon^2$, then with probability $1-\epsilon$,*

$$\sum_{j \in S_{2l}, o \in O_j} \pi_{jo} x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \geq (1 - 9h_l - \epsilon) OPT_{2l}, \forall l.$$

Proof. Let us consider the following LP:

$$\begin{array}{lll}
\max & \sum_{j \in S_{2l}, o} \pi_{jo} x_{jo} & \\
s.t. & \sum_{j \in S_{2l}, o} a_{ijo} x_{jo} \leq \hat{b}_i, & \forall i \\
& \sum_{o \in O_j} x_{jo} \leq 1, & \forall j \in S_{2l} \\
& x_{jo} \geq 0, & \forall j \in S_{2l}, o
\end{array}, \tag{5.13}$$

where $\hat{b}_i = \sum_{j \in S_{2l}} a_{ijo} x_{jo}(\hat{\mathbf{p}}_\mathbf{l})$, if $\hat{p}_{l,i} > 0$ and $\hat{b}_i = \max\{\sum_{j \in 2l} a_{ijo} x_{jo}(\hat{\mathbf{p}}_\mathbf{l}), b_i\}$, if $\hat{p}_{l,i} = 0$. By complementary slackness, $\mathbf{x}(\hat{\mathbf{p}}_\mathbf{l})$ is the optimal solution to this LP.

On the other hand, using the same argument as in the proof of Lemma 43, we can show that with probability $1-\epsilon$, $\hat{b}_i \geq 2l \cdot (1 - 9h_l - \epsilon) b_i$ for all i and l. In that case, then $\sum_{j \in S_{2l}, o} \pi_{jo} x_{jo}(\hat{\mathbf{p}}_\mathbf{l}) \geq (1 - 9h_l - \epsilon) OPT_{2l}$. □

Lemma 51. *Given $\mathcal{E}_{est}$, $\mathbb{E}[OPT_l] \leq (1+h_l) l \cdot OPT$.*

Proof. Consider the optimal dual solution $(\mathbf{p}^*, \mathbf{q}^*)$ to LP (5.1). We can easily check that it is feasible to the partial dual problem (5.9)

$$\begin{array}{lll}
\min & \sum_i (1-6h_l) l\epsilon b_i p_i + \sum_{j \in S_l} q_j & \\
s.t. & \sum_i a_{ijo} p_i + q_j \geq \pi_{jo}, & \forall j \in S_l, o \in O_j \\
& p_i \geq 0, \forall i & \\
& q_j \geq 0, \forall j \in S_l &
\end{array}.$$

Thus, $OPT_l \leq (1-6h_l)l\sum_i b_i p_i^* + \sum_{j\in S_l} q_j^*$. Note that given $\mathcal{E}_{est}$, for every j, $\Pr(j \in S_l) \leq (1+h_l)l$. By taking expectation on both sides, we can conclude the lemma. □

From lemmas above, we can conclude:

Theorem 32. *If* $\min_i b_i \geq 3m\ln(nq/\epsilon)/\epsilon^2$ *and* $k \geq 5\ln\epsilon/\epsilon^2$, *the algorithm is* $1-O(\epsilon)$ *competitive.*

Proof. Let $\mathcal{E}_3$ denote the event that both inequalities in Lemma 49 and 50 are true, then $\Pr(\mathcal{E}_3 \cap \mathcal{E}_{est}) \geq 1-3\epsilon$.

$$\begin{array}{rl}
& \mathbb{E}[\sum\limits_{l\in L}\sum\limits_{j\in S_{2l}\setminus S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l})|\mathcal{E}_3\cap\mathcal{E}_{est}] \\
\geq & \sum\limits_{l\in L}\mathbb{E}[\sum\limits_{j\in S_{2l}} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l})|\mathcal{E}_3\cap\mathcal{E}_{est}] - \sum\limits_{l\in L}\mathbb{E}[\sum\limits_{j\in S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l})|\mathcal{E}_3\cap\mathcal{E}_{est}] \\
\geq & \sum\limits_{l\in L}(1-9h_l-\epsilon)\mathbb{E}[OPT_{2l}|\mathcal{E}_3\cap\mathcal{E}_{est}] - \sum\limits_{l\in L}\mathbb{E}[OPT_l|\mathcal{E}_3\cap\mathcal{E}_{est}] \\
\geq & OPT - \sum\limits_{l\in L} 9h_l\mathbb{E}[OPT_{2l}|\mathcal{E}_3\cap\mathcal{E}_{est}] - \epsilon\sum\limits_{l\in L}\mathbb{E}[OPT_{2l}|\mathcal{E}_3\cap\mathcal{E}_{est}] - \mathbb{E}[OPT_\epsilon|\mathcal{E}_3\cap\mathcal{E}_{est}] \\
\geq & OPT - 9\sum\limits_{l\in L}(2h_l l + h_l h_{2l} l)\cdot OPT - \epsilon\sum\limits_{l\in L} 4l\cdot OPT - 2\epsilon OPT \\
\geq & OPT - 42\epsilon OPT
\end{array}.$$

Therefore, $\mathbb{E}[\sum_{l\in L}\sum_{j\in S_{2l}\setminus S_l} \pi_j x_j(\hat{\mathbf{p}}_\mathbf{l})] \geq (1-45\epsilon)OPT$. □

The assumptions made in the theorem are reasonable. On the one hand, the lower bound on $\min_i b_i$ is the same as in DPA, which has been showed to be best possible in many occasions. On the other hand, the lower bound on k is even lower than the one on $\min_i b_i$, which means only a limited amount of past observations are required to obtained the near-optimal result.

DPAD does not take advantage of demands and prices information from past customers. In the random permutation model, such information is unlikely to improve the online algorithm. But for practical problems where demands and payments come from unknown i.i.d. distributions, these data may provide good estimation on the distributions, and lead to better results.

5.5 Heterogeneous Customers

As we may see in many applications, customers are not all homogeneous. Customers with different preference may have different arrival time. For instance, in the airline revenue management problems, casual travelers, whose reserve prices are probably lower, usually arrive long before their scheduled departure time; while business travelers, who tend to be price-insensitive, are more likely to appear shortly before intended trips. In this section, we take this heterogeneity into account.

Assume all customers are categorized into K groups: $N = \bigcap_{k=1}^{K} N_k$. Furthermore, we assume there exists a constant c such that $\forall k, k', t, F_k(t) \leq cF_{k'}(t)$. Let t_0 be the ϵ-quantile point, i.e. $F_1(t_0) = \epsilon$. Assume $F_k(t_0) = r_k\epsilon$. From the assumption on the CDFs, we have $r_k \in [1/c, c]$.

Let S_k be the set of customers from group k that arrive before t_0. The online algorithm observes customers arriving before t_0 and solves the following LPs:

$$\begin{array}{rl} \max & \sum_k (r_k\epsilon)^{-1} \sum_{j\in S_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k \\ s.t. & \sum_k (r_k\epsilon)^{-1} \sum_{j\in S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k \leq \epsilon(1-\epsilon)b_i, \quad \forall i \\ & \sum_{o\in O_j} x_{jo}^k \leq 1, \forall j, k \\ & \mathbf{x} \geq 0 \end{array} \qquad \begin{array}{rl} \min & \epsilon(1-\epsilon)\sum_i b_i p_i + \sum_{j,k} q_j^k \\ s.t. & r_k\epsilon q_j^k + \sum_i a_{ijo}^k p_i \geq \pi_j^k, \forall j, o, k \\ & \mathbf{p}, \mathbf{q} \geq 0 \end{array} \tag{5.14}$$

Similar to arguments in the previous sections, We show that with high probability, the resulting solution is feasible, the resulting solution is near optimal, and the loss due to observation is small:

Lemma 52. *If* $\min_i b_i \geq 3cm\ln(nq/\epsilon)/\epsilon^3$, *then w.p.* $1-\epsilon, \forall i, \sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}}) \leq b_i$.

Proof. The proof is very similar to the one of Lemma 42. Let us first consider the probability of the event

$$\{\sum_k (r_k\epsilon)^{-1} \sum_{j\in S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\mathbf{p}) \leq \epsilon(1-\epsilon)b_i, \sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\mathbf{p}) \geq b_i\} \tag{5.15}$$

for all fixed $\mathbf{p}$ and i. Since $\Pr(j \in S_k | j \in N_k) = r_k\epsilon$, we expect $\sum_k (r_k\epsilon)^{-1} \sum_{j\in S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k$ close to its mean $\sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k$. Therefore, event (5.15) should be a rare event. More precisely,

$$\Pr(\sum_k (r_k\epsilon)^{-1} \sum_{j\in S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\mathbf{p}) \leq \epsilon(1-\epsilon)b_i, \sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\mathbf{p}) \geq b_i) \leq \exp(-\epsilon^3 b_i/3c).$$

Note that there are at most $(nq)^m$ distinct $\mathbf{p}$. By taking union bounds over all distinct $\mathbf{p}$

and i, we can conclude the lemma. $\square$

Lemma 53. *If* $\min_i b_i \geq 3cm\ln(nq/\epsilon)/\epsilon^3$, *then w.p.* $1-\epsilon, \sum_k \sum_{j\in N_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT$.

Proof. Let us consider the following LP:

$$\begin{array}{lll} \max & \sum_k \sum_{j\in N_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k & \\ s.t. & \sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k \leq \hat{b}_i, & \forall i \\ & \sum_{o\in O_j} x_{jo}^k \leq 1, & \forall j,k \\ & x_{jo}^k \geq 0, & \forall j,k,o \end{array}, \tag{5.16}$$

where $\hat{b}_i = \sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}})$, if $\hat{p}_i > 0$ and $\hat{b}_i = \max\{\sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}}), b_i\}$, if $\hat{p}_i = 0$. By complementary slackness, $\mathbf{x}(\hat{\mathbf{p}})$ is the optimal solution to this LP.

On the other hand, using the same argument as in the proof of Lemma 42, we can show that with probability $1-\epsilon$, $\hat{b}_i \geq (1-3\epsilon)b_i$ for all i and l. If such an event happens, then $\sum_k \sum_{j\in N_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT$. $\square$

Lemma 54. $\mathbb{E}[\sum_k \sum_{j\in S_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k] \leq \max_k r_k \epsilon OPT \leq c\epsilon OPT$.

Proof. Let OPT_ϵ be the optimal value to the partial LP (5.14). Let $(\mathbf{p}^*, \mathbf{q}^*)$ be the optimal dual solution to the complete LP (5.16). It is easy to check that $(\mathbf{p}^*, \mathbf{q}^*)$ is a dual feasible solution to (5.14). Therefore, $\mathbb{E}[OPT_\epsilon] \leq OPT$.

On the other hand, for any realization, the lost revenue resulting from the first ϵ fraction customers is no more than OPT_1. Hence,

$$\mathbb{E}[\sum_k \sum_{j\in S_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k] \leq \mathbb{E}[\max_k r_k \epsilon \cdot OPT_\epsilon] \leq \max_k r_k \epsilon \cdot OPT.$$

$\square$

Combining the three lemmas above, we can conclude that:

Theorem 33. *If* $\min_i b_i \geq 3cm\ln(nq/\epsilon)/\epsilon^3$, *the algorithm is* $1-O(\epsilon)$ *competitive.*

Proof. Let $\mathcal{E}_4$ denote the event that for all i

$$\sum_k \sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}}) \leq b_i$$

and

$$\sum_k \sum_{j\in N_k}^{o\in O_j} \pi_{jo}^k x_{jo}^k(\hat{\mathbf{p}}) \geq (1-3\epsilon)OPT.$$

Then, from Lemma 52 and 53, we have $\Pr(\mathcal{E}_4) \leq 1-2\epsilon$. Given $\mathcal{E}$, the online solution $\mathbf{x}(\hat{\mathbf{p}})$ is feasible. Therefore,

$$\begin{aligned}
\mathbb{E}[\textstyle\sum_k\sum_{j\notin S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}})] &\geq \mathbb{E}[\textstyle\sum_k\sum_{j\notin S_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}})|\mathcal{E}_4]\cdot\Pr(\mathcal{E}_4)\\
&\geq (\mathbb{E}[\textstyle\sum_k\sum_{j\in N_k}^{o\in O_j} a_{ijo}^k x_{jo}^k(\hat{\mathbf{p}})|\mathcal{E}_4] - \mathbb{E}[\sum_k\sum_{j\in S_k}^{o\in O_j}\pi_{jo}^k x_{jo}^k|\mathcal{E}_4])\cdot\Pr(\mathcal{E}_4)\\
&\geq (1-3\epsilon)(1-2\epsilon)OPT - c\epsilon OPT\\
&\geq (1-O(\epsilon))OPT
\end{aligned}$$

□

Unfortunately, dynamic pricing techniques used in DPA and DPAD do not apply for this problem, because the arrival process is not homogeneous here. Without dynamic pricing mechanism, in order to have near optimality result, the lower bound imposed on the model is much higher than the one we obtained in the previous sections. Worth noting that, this approach can only deal with problems where customers of each type are well represented in the early stages. Otherwise, we may need additional assumptions of information to obtain good results. For example, consider airline tickets sales, if all casual travelers arrive at least one week before departure and all business travelers only appear one week within departure. If the numbers of travelers of the two types are unrelated, then past information alone is unlikely to help us decide how many seats to reserve for business customers. To find a proper reserve level, we need good estimates on the numbers of travelers of the two types, which probably requires more assumptions.

5.6 Flexible Inventory

In this section, we consider resource allocation problems with flexible inventory. Instead of having exactly b_i units of resource $i \in I$ as described in Section 5.1, resources can be bought at non-decreasing rates: d_{ik} units of resource $i \in I$ are available at price c_{ik} ($1 \leq k \leq K$, and $c_{i1} < c_{i2} < \cdots < c_{iK}$). Those resources can be replenished at any time at any amount as long as they are still available. For the simplicity of the section, we assume that every customer comes with only one option, and the system either accepts and rejects his option.

It is easy to see that the offline problem can be formulated as the following pair of primal dual LPs:

$$\begin{array}{ll} \max & \sum_j \pi_j x_j - \sum_{i,k} c_{ik} y_{ik} \\ s.t. & \sum_j a_{ij} x_j - \sum_k y_{ik} = 0, \forall i \\ & x_j \leq 1, \forall j \\ & y_{ik} \leq d_{ik}, \forall i,k \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{array} \qquad \begin{array}{ll} \min & \sum_j q_j + \sum_{i,k} d_{ik} r_{ik} \\ s.t. & \sum_i a_{ij} p_i + q_j \geq \pi_j, \forall j \\ & -p_i + r_{ik} \geq -c_{ik} \\ & \mathbf{q}, \mathbf{r} \geq \mathbf{0} \end{array} \tag{5.17}$$

Like in Section 5.2, the online algorithm observes customers in S_ϵ, rejects them all, and computes dual prices by solving the following pair of primal dual LPs:

$$\begin{array}{ll} \max & \sum_{j \in S_\epsilon} \pi_j x_j - \sum_{i,k} c_{ik} y_{ik} \\ s.t. & \sum_{j \in S_\epsilon} a_{ij} x_j - \sum_k y_{ik} = 0, \forall i \\ & x_j \leq 1, \forall j \in \epsilon \\ & y_{ik} \leq \epsilon(1-\epsilon) d_{ik}, \forall i,k \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{array} \qquad \begin{array}{ll} \min & \sum_{j \in S_\epsilon} q_j + \epsilon(1-\epsilon) \sum_{i,k} d_{ik} r_{ik} \\ s.t. & \sum_i a_{ij} p_i + q_j \geq \pi_j, \forall j \\ & -p_i + r_{ik} \geq -c_{ik} \\ & \mathbf{q}, \mathbf{r} \geq \mathbf{0} \end{array} \tag{5.18}$$

Let $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathbf{r}})$ be the optimal solution.

For customers arriving later, they are accepted if their payments exceed the threshold set by $\hat{\mathbf{p}}$:

$$x_j(\hat{\mathbf{p}}) = \begin{cases} 1, \text{ if } \pi_j - \sum_i a_{ij} \hat{p}_i > 0 \\ 0, \text{ otherwise} \end{cases} \tag{5.19}$$

The proposed online algorithm is then as follows:

Algorithm 10 (Flexible Inventory Algorithm(FIA))**.**

1. *Reject all customers arriving earlier than* $F^{-1}(\epsilon)$.

2. *Let* $x_j = x_j(\hat{\mathbf{p}})$ *for customers arriving after* $F^{-1}(\epsilon)$.

To analyze the algorithm, let us first construct a pair of primal dual LPs similar to 5.17:

$$\begin{array}{rl} \max & \sum_j \pi_j x_j - \sum_{i,k} c_{ik} y_{ik} \\ s.t. & \sum_j a_{ij} x_j - \sum_k y_{ik} = 0, \forall i \\ & x_j \leq 1, \forall j \\ & y_{ik} \leq \hat{d}_{ik}, \forall i, k \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{array} \qquad \begin{array}{rl} \min & \sum_j q_j + \sum_{i,k} \hat{d}_{ik} r_{ik} \\ s.t. & \sum_i a_{ij} p_i + q_j \geq \pi_j, \forall j \\ & -p_i + r_{ik} \geq -c_{ik} \\ & \mathbf{q}, \mathbf{r} \geq \mathbf{0} \end{array} \tag{5.20}$$

where

$$\hat{d}_{ik} = \begin{cases} \max\{\sum_j a_{ij} x_j(\hat{\mathbf{p}}) - \sum_{k'=1}^{k-1} d_{ik'}, d_{ik}\}, \text{ if } \hat{r}_{ik} = 0 \\ \min\{\sum_j a_{ij} x_j(\hat{\mathbf{p}}) - \sum_{k'=1}^{k-1} d_{ik'}, d_{ik}\}, \text{ otherwise} \end{cases} \tag{5.21}$$

and its solution $(\mathbf{x}(\hat{\mathbf{p}}), \mathbf{y}^*, \hat{\mathbf{p}}, \mathbf{q}^*, \hat{\mathbf{r}})$ where $y^*_{ik} = \min\{(\sum_j a_{ij} x_j(\hat{\mathbf{p}}) - \sum_{k'=1}^{k-1} y^*_{ik'})^+, d_{ik}\}$, and $\hat{q}^*_j = \max\{0, \pi_j - \sum_i a_{ij} \hat{p}_i\}$.

Due to the choice of the solution, we can prove that

Lemma 55. *If $\min_i d_{i1} \geq m \ln(n/\epsilon)/\epsilon^3$, with probability $1-\epsilon$, $(\mathbf{x}(\hat{\mathbf{p}}), \mathbf{y}^*, \hat{\mathbf{p}}, \mathbf{q}^*, \hat{\mathbf{r}})$ is the optimal solution to LPs (5.20).*

Proof. The feasibility is obvious from our choice of $\mathbf{x}(\hat{\mathbf{p}}), \mathbf{y}^*, \hat{\mathbf{p}}, \mathbf{q}^*, \hat{\mathbf{r}}$ and $\hat{\mathbf{d}}$. We then check the optimality by verifying complementary slackness:

- $(x_j(\hat{\mathbf{p}}) - 1) q^*_j$. If $q^*_j > 0$, then from the definition of q^*_j, we have $\pi_j - \sum_i a_{ij} \hat{p}_i > 0$. Hence $x_j(\hat{\mathbf{p}}) = 1$.

- $(y^*_{ik} - \hat{d}_{ik}) \hat{r}_{ik}$. If $\hat{r}_{ik} > 0$, then $y^*_{ik} = \min\{(\sum_j a_{ij} x_j(\hat{\mathbf{p}}) - \sum_{k'=1}^{k-1} y^*_{ik'})^+, d_{ik}\} = \hat{d}_{ik}$.

- $x_j(\hat{\mathbf{p}})(\sum_i a_{ij} \hat{p}_j + q^*_j - \pi_j)$. If $x_j(\hat{\mathbf{p}}) > 0$, then $\pi_j > \sum_i a_{ij\hat{p}_j}$, therefore, $q^*_j = \pi_j - \sum_i a_{ij\hat{p}_j}$.

- $y^*_{ik}(-\hat{p}_i + \hat{r}_{ik} + c_{ik})$. If $-\hat{p}_i + \hat{r}_{ik} > -c_{ik}$, then because of the complementary slackness of (5.18), $\hat{y}_{ik} = 0$. Since $\forall k' > k$, $c_{ik'} > c_{ik}$, we have $\hat{y}_{ik'} = 0$. Hence, $\sum_{j \in S_\epsilon} x_j(\hat{\mathbf{p}}) = \sum_k \hat{y}_k = \sum_{k'<k} \hat{y}_{ik'} \leq \epsilon(1-\epsilon) \sum_{k'<k} d_{ik'}$. By arguments similar to those in Section 5.2, with probability $1-\epsilon$, $\sum_j x_j(\hat{\mathbf{p}}) \leq \sum_{k'<k} d_{ik'}$. Hence, $y^*_{ik} = 0$.

□

Lemma 56. *If* $\min_i d_{i1} \geq m \ln(n/\epsilon)/\epsilon^3$*, with probability* $1-\epsilon$*,* $\sum_j \pi_j x_j(\hat{\mathbf{p}}) - \sum_{i,k} c_{ik}\hat{y}_{ik} \geq (1-2\epsilon)OPT$*.*

Proof. Let us first show that with probability $1-\epsilon$, $\forall k$, $\sum_{k' \leq k} \hat{d}_{ik} \in [(1-2\epsilon)\sum_{k' \leq k} d_{ik}, \sum_{k' \leq k} d_{ik}]$. We again use the argument that fixes $\hat{\mathbf{p}}$ as in Section (5.2):

- If $\hat{r}_{ik} = 0$, then $\forall k' > k$, $c_{ik'} > c_{ik} \geq \hat{p}_i$. By complementary slackness of LP (5.18), $\hat{y}_{ik'} = 0$. Therefore, $\sum_{j \in S_\epsilon} a_{ij}x_j(\hat{\mathbf{p}}) \leq \sum_{k' \leq k} \epsilon(1-\epsilon)d_{ik}$. By concentration laws, with high probability, $\sum_j a_{ij}x_j(\hat{\mathbf{p}}) \leq \sum_{k' \leq k} d_{ik}$. In such cases, $\hat{d}_{ik} = \max\{\sum_j a_{ij}x_j(\hat{\mathbf{p}}) - \sum_{k'<k} d_{ik'}, d_{ik}\} = d_{ik}$.

- If $\hat{r}_{ik} > 0$, then because of complementary slackness of LP (5.18), $\hat{y}_{ik} = \epsilon(1-\epsilon)d_{ik}$, and $\forall k' \leq k$, $\hat{y}_{ik'} = \epsilon(1-\epsilon)d_{ik}$. Therefore, $\sum_{j \in S_\epsilon} a_{ij}x_j(\hat{\mathbf{p}}) \geq \sum_{k' \leq k} \epsilon(1-\epsilon)d_{ik}$. By concentration laws, with high probability, $\sum_j a_{ij}x_j(\hat{\mathbf{p}}) \geq (1-2\epsilon)\sum_{k' \leq k} d_{ik}$. Hence, $\sum_{k' \leq k} \hat{d}_{ik} \geq (1-2\epsilon)\sum_{k' \leq k} d_{ik}$.

Note that $\hat{r}_{ik}$ is non-increasing in k, and $(\hat{\mathbf{p}}, \mathbf{q}^*, \hat{\mathbf{r}})$ is also a feasible solution to the dual problem of LP (5.17). We can conclude that the optimal solution of LP (5.20) is at least $(1-2\epsilon)OPT$. □

Lemma 57. $\mathbb{E}[OPT_\epsilon] \leq \epsilon OPT$*.*

Proof. Note that the dual optimal solution to LP (5.17) is also feasible to LP (5.18). Thus, $\mathbb{E}[OPT_\epsilon] \leq \epsilon OPT$. □

Theorem 34. *If* $\min_i d_{i1} \geq m \ln(n/\epsilon)/\epsilon^3$*, then FIA is* $1 - O(\epsilon)$ *competitive.*

Proof. Due to Lemma 55 and 56, $\mathbb{E}[\sum_j \pi_j x_j(\hat{\mathbf{p}}) - \sum_{i,k} c_{ik}\hat{y}_{ik}] \geq (1-4\epsilon)OPT$. Then, we need to remove customers in S_ϵ as they are all rejected. The revenue collected from those customers is exactly $\sum_{j \in S_\epsilon} \pi_j x_j(\hat{\mathbf{p}})$. On the other hand, cost for those resources is at most $\sum_{i,k} c_{ik}\hat{y}_{ik}$. Therefore, the lost net revenue is at most OPT_ϵ. Hence, the online revenue should be at least $(1-5\epsilon)OPT$. □

Chapter 6

Conclusion and Future Work

In this chapter, we first summarize the thesis and then briefly discuss some open problems and possible directions for future research.

6.1 Summary of Thesis

In Chapter 1, we began with examples and applications to justify the purpose of the thesis. Then, we provided an overview on online optimization and methods of analyzing online algorithms. Finally, we gave a literature review on work related to this thesis, and presented our results and contributions.

In Chapter 2, we considered online TSP with rejection options. We first presented a best optimal online algorithm for the basic version, and showed its optimality for several generalization of the problem. We then designed online algorithms for the real time version of the problem on different metric spaces, including non-negative real line, real line, and arbitrary metric space.

In Chapter 3, we introduced generalized online assignment problems. We first showed that no algorithm has a positive competitive ratio for such problems in general. We then imposed two constraints on the model. We presented and numerically compared two best possible online algorithms for the restricted model. Finally, we showed how to modified algorithms for less restricted models.

In Chapter 4, we studied online stochastic matching problems. We began with a general class of algorithms. We then presented and analyzed an online algorithm for unweighted problems. The algorithm and analysis were then modified for vertex-weighted problems.

We finished the chapter with problems with Poisson arrivals.

In Chapter 5, we first considered online resource allocation problems with fixed inventory levels. A one-time learning algorithm was presented, followed by a dynamic pricing algorithm, which is better but more complicated. We then relaxed the constraints imposed on the model, and gave near optimal online algorithms. Finally, we considered problems with flexible inventory levels.

6.2 Future Research

The following are several directions that the research in this thesis can be extended to.

- For online TSP with rejection options, online algorithms proposed in the thesis are designed for worst case scenarios. However, in many applications, average case performance is of more interest. It would be great to propose realistic probability distributions of interest, and design online algorithms for such cases.

- Primal-dual based online algorithms could be very powerful for online optimization problems, as showed in Chapter 3 of this thesis and in [23]. It would be interesting to find the class of online problems for which primal-dual based algorithms perform well.

- Our algorithm for online stochastic matching problems is based on a discretization idea, i.e. flows on all edges are 0, 1/3, or 2/3. We conjecture that if an optimal solution to

$$\begin{array}{lll} \max & \sum_{a,i} w_a f_{a,i} & \\ s.t. & \sum_{i\sim a} f_{a,i} \le 1 & \forall a \in A \\ & \sum_{a\sim i} f_{a,i} \le 1 & \forall i \in I \\ & f_{a_1,i} + f_{a_2,i} \le 9/10 & (a_1,i) \in E, (a_2,i) \in E \\ & f_e \in [0, 7/10] & \forall e \in E \end{array}$$

  is used as the offline flow in the Random Lists Algorithm, the resulting online algorithm is 0.751-competitive .

Bibliography

[1] D. Adelman. Dynamic bid prices in revenue management. *Operations Research*, 55(4):647–661, July 2007.

[2] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *ACM Conference on Electronic Commerce*, pages 1–7, 2006.

[3] G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *SODA '11: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264, 2011.

[4] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. Working paper, Stanford University.

[5] M. Akan and B. Ata. Bid-price controls for network revenue management: Martingale characterization of optimal bid prices. *Mathematics of Operations Research*, 57(4):912–936, 2009.

[6] S. Albers. Competitive online algorithms. *OPTIMA: Mathematical Programming Society Newsletter*, 54:1–8, June 1997.

[7] L. Allulli, G. Ausiello, and L. Laura. On the power of lookahead in on-line vehicle routing problems. In *Proceedings of the Eleventh International Computing and Combinatorics Conference, Lecture Notes in Computer Science*, volume 3595, pages 728–736, 2005.

[8] E. Angelelli, M. Savelsbergh, and M. Speranza. Competitive analysis for dynamic multi-period uncapacitated routing problems. *Networks*, 49:308–317, 2007.

[9] E. Angelelli, M. Savelsbergh, and M. Speranza. Competitive analysis of a dispatch policy for a dynamic multi-period routing problem. *Operations Research Letters*, 35:713–721, 2007.

[10] N. Ascheuer, S. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, volume 1770, pages 639–650, 2000.

[11] G. Ausiello, V. Bonifaci, and L. Laura. The on-line prize-collecting traveling salesman problem. *Information Processing Letters*, 107(6):199–204, 2008.

[12] G. Ausiello, M. Demange, L. Laura, and V. Paschos. Algorithms for the on-line quota traveling salesman problem. *Information Processing Letters*, 92(2):89–94, 2004.

[13] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[14] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2):1–11, June 2008.

[15] B. Bahmani and M. Kapralov. Improved bounds for online stochastic matching. In *ESA '10: Proceedings of the 22nd Annual European Symposium on Algorithms*, pages 170–181, 2010.

[16] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[17] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Algorithmica*, pages 379–386, 1990.

[18] R. Bent and P. Van Hentenryck. Scenario based planning for partially dynamic vehicle routing problems with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

[19] S. Bernstein. On a modification of Chebyshev's inequality and of the error formula of Laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math.*, 1:38–49, 1924.

[20] D. Bertsimas and R. Demir. An approximate dynamic programming approach to multidimensional knapsack problems. *Manage. Sci.*, 48(4):550–565, April 2002.

[21] M. Blom, S. Krumke, W. de Paepe, and L. Stougie. The online TSP against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

[22] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, first edition, 1998.

[23] N. Buchbinder. *Designing competitive online algorithms via a primal-dual approach.* PhD thesis, 2008.

[24] N. Buchbinder, K. Jain, and S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA '07: Proceedings of the 15th Annual European Symposium on Algorithms*, 2007.

[25] C. Chekuri and S. Khanna. A ptas for the multiple knapsack problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '00, pages 213–222, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[26] N. Devanur and T. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *EC '09: Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 71–78, 2009.

[27] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[28] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[29] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS '09: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126, 2009.

[30] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *ACM Conference on Electronic Commerce*, 2007.

[31] E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

[32] A. Fiat and G. Woeginger. *Online Algorithms: The State of the Art.* Springer Verlag LNCS State of the Art Survey, 1998.

[33] J. Fledman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. In *Algorithms - ESA 2010*, pages 182–194, 2010.

[34] L. Fleischer, M. Goemans, V. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 611–620, New York, NY, USA, 2006. ACM.

[35] G. Gallego and G. van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45(1):24–41, 1997.

[36] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 982–991, 2008.

[37] B. Haeupler, V. Mirrokni, and M. Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *WINE '11: Workshop of Network and Internet Economics*, pages 170–181, 2011.

[38] P. Van Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization.* MIT Press, first edition, 2006.

[39] P. Jaillet and X. Lu. Generalized online assignment problems. Working paper, Massachusetts Institute of Technology.

[40] P. Jaillet and X. Lu. Near-optimal online algorithms for dynamic resource allocation problems. Working paper, Massachusetts Institute of Technology.

[41] P. Jaillet and X. Lu. Online stochastic matching: New algorithms with better bounds. Working paper, Massachusetts Institute of Technology.

[42] P. Jaillet and X. Lu. Online traveling salesman problems with rejection options. Working paper, Massachusetts Institute of Technology.

[43] P. Jaillet and X. Lu. Online traveling salesman problems with service flexibility. *Networks*, 58:137–146, 2011.

[44] P. Jaillet and M. Wagner. Online routing problems: value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210, 2006.

[45] P. Jaillet and M. Wagner. Generalized online routing: New competitive ratios, resource augmentation and asymptotic analyses. *Operations Research*, 56:745–757, 2008.

[46] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125–138, 1994.

[47] B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325, 2000.

[48] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *STOC '11: Proceedings of the 43nd Annual ACM Symposium on Theory of computing*, 2011.

[49] A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[50] R. Karp, U. Vazirani, and V. Varirani. An optimal algorithm for online bipartite matching. In *STOC '90: Proceedings of the 22nd Annual ACM Symposium on Theory of computing*, pages 352–358, 1990.

[51] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 630–631, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[52] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms.* Springer, second edition, 2002.

[53] S. Lahaie, D. Pennock, A. Saberi, and R. Vohra. *Algorithmic Game Theory, chapter 28 Sponsored Search.* Cambridge University Press, 2007.

[54] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. *The Traveling Salesman Problem, A Guided Tour of Combinatorial Optimization.* John Wiley & Sons Ltd., 1985.

[55] M. Lipmann. *On-line Routing.* PhD thesis, Technische Universiteit Eindhoven, 2003.

[56] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *STOC '11: Proceedings of the 43nd Annual ACM Symposium on Theory of computing*, 2011.

[57] V. Manshadi, S. Oveis Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[58] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics, London Math. Soc. Lect. Note Series 141*, pages 148–188, 1989.

[59] A. Mehta, A. Saberi, U. Vazirani, and V. Varirani. Adwords and generalized online matching. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, 2005.

[60] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. *Journal of the ACM*, 54(5):Article 22 (19 pages), 2007.

[61] V. Mirrokni, S. Oveis Gharan, and M. Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *SODA '12: Proceedings of the 23nd Annual ACM-SIAM Symposium on Discrete Algorithms*, 2012.

[62] P. Orlik and H. Terao. *Arrangement of Hyperplanes.* Springer-Verlag, 1992.

[63] P. Rusmevichientong and D. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Seventh ACM Conference on Electronic Commerce*, 2006.

[64] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, December 1993.

[65] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[66] L. Stougie and A. Vestjens. Randomized algorithms for online scheduling problems: How low can't you go. *Operations Research Letters*, 30(2):89–96, 2002.

[67] C. Therkelsen. Technical report. Technical report, Massachusetts Institute of Technology, 2011.

[68] H. Varian. Position auctions. *International Journal of Industrial Organization*, 25:1163–1178, 2007.

[69] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, Washington, DC, USA, 1977. IEEE Computer Society.

[70] D. Zhang and D. Adelman. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science*, 43(3):381–394, August 2009.