The Raymond and Beverly Sackler
Faculty of Exact Scinces
School of Computer Science

ONLINE AND STRATEGIC ASPECTS OF NETWORK RESOURCE MANAGEMENT ALGORITHMS

THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOHY

by
Nir Andelman

Under the supervision of Prof. Yishay Mansour

Submitted to the Senate of Tel-Aviv University
September 2006

To Yoav

Abstract

Today's communication networks provide a wide range of algorithmic challenges for managing network resources, such as memory buffers, computation power, bandwidth, as well as other resources. Challenges include developing algorithms, which are efficiently computable on one hand, and achieve optimal or near optimal optimization on the other hand. Additional aspects of algorithmic design in network environments include online computation, where decisions have to be taken before the entire input is known, and mechanism design, where the algorithm designer is aiming to achieve the cooperation of selfish agents, distributed over the network, who participate in the execution of the algorithm.

In the first chapters of this dissertation we consider the task of online packet scheduling of a single FIFO queue. For non preemptive queues and arbitrary packet values, an online policy that nearly matches the lower bound for online scheduling is given. Randomized algorithms are considered for preemptive queues, including an algorithm for two packet values, which performs better that any deterministic algorithm.

In the remaining chapters we consider mechanisms for allocation problems, such as machine scheduling and resource allocation. At first, resource allocation under budget constraints is considered. This is a special case of combinatorial auctions with compliment free valuations, as well as a relaxation of the general assignment problem [72, 24]. At first, we develop several non strategic algorithms for this problem. Following, we derive a general sufficient condition for designing incentive compatible mechanisms when the valuation functions of the participants are one dimensional. Using this, we present truthful algorithms for the auction with budget constraints problem, as well as for machine scheduling on uniformly related machines, when machines and job owners may be strategic.

Preliminary results of this dissertation appeared in [7, 8, 6, 5, 9].

Acknowledgements

The completion of this dissertation is a personal landmark, ending a notable period spent in research at Tel-Aviv University. This is an opportunity to thank a few of the people who filled keynote roles during this recent episode of my life:

First and foremost, I would like to express my sincere gratitude to my advisor, Yishay Mansour, for his dedicated supervision and exceptional guidance through the stormy sea of academic research. This work could not have been completed without him.

I would like to thank my co-authors Yossi Azar and Motti Sorani, who contributed to this work, as well as Michal Feldman, for her leading part in later research. Special thanks to Noam Nisan, who inspired me with some of the ideas that began this research, to Amir Ronen and Amos Fiat for their useful comments on early drafts of this work, and to Mooly Sagiv for his generous assistance. I would also like to thank Roman Manevich, Yossi Richter, Danny Hendler and Eyal Even-Dar for our fruitful discussions.

Last but not least, I would like to thank my beloved family for their moral support, and especially my wife Michal, and our son Yoav, who was born during this research period.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's Communication Networks span a wide realm of algorithmic challenges. For instance, buffering and scheduling algorithms are applied within switches and routers, routing algorithms are executed along the network's paths, and distributed tasks are assigned to the hosts on the network's endpoints. In a related yet different aspect, communication networks lay the grounds for electronic commerce, where combinatorial challenges meet economic considerations.

Efficiency of computation and online computation are two aspects, which commonly have significant importance in designing algorithms in network environments. Response time is critical in network behavior, which motivates favoring algorithms that can be computed efficiently. Since time complexity in network algorithms is in many cases a function of the number of network components, polynomial time complexity is usually a minimal requirement needed to attain the scalability of the system.

A common difficulty is that efficient polynomial time algorithms for certain optimization problems aren't known, and are unlikely to exist unless $P = NP$. This motivates searching for approximation algorithms, which guarantee solutions that although are not optimal, are close to the optimal solution up to a certain bounded factor. For instance, scheduling jobs over several processors such that the longest processing time is minimized is an example to a classical NP-Hard problem in the networking environment [36], for which many approximations exist (see, for example, [38, 43, 42]).

In networking environments, many algorithms are bound to operate online, meaning that actions are taken while the input arrives, but before the entire input is known [23]. For example, network switches are bound to decide on buffering and sending incoming packets before all the future traffic will arrive. In many cases it is impossible to assure optimal performance online, where optimality is defined by the offline version of the same problem, where the entire input is known in advance. Developing online algorithms with a worst case performance that is close to the offline optimal up to a certain bounded factor is a common solution concept in online computation. Another interest in this field is to lower bound the worst case performance of any online algorithm in comparison to the offline case, in order to derive an impossibility result that measures the potential of such algorithms.

In large networks and most notably, in the Internet, users with various and possibly contradicting interests meet by using the network as a virtual platform for interchanging merchandise, services and information. Thus, the role of optimization algorithms for usage over networks expands beyond the physical challenge of passing data from one end to another in methods that are efficient, secure and reliable. The vast and continuing growth in electronic commerce (E-Commerce, e.g. [29, 4]) is an incentive to develop algorithms for use in digital markets, which are efficient both computationally as well as economically, meaning that they optimize a collective social value.

Another aspect in algorithm design, which commonly emerges in digital markets, is that the participants cannot be obliged to follow a central protocol, and may strategically manipulate the algorithm's outcome by choosing operations that are inconsistent with the social objective function, yet promote their personal interests.

Mechanism design [76] is an approach for clearing the uncertainty in systems such as large networks, assuming that the participants' behavior is rational, meaning that they aim to maximize their personal valuation. Basically, in addition to decision rule, the mechanism uses a payment scheme, which enforces the cooperation of the participants. This goal is achieved if the payment scheme shifts the preferences of every participant in a way such that no action can benefit more than cooperation, regardless of the actions of the other participants.

In the algorithmic context [64], there are two major difficulties in applying mechanism design. The first is that in addition to the computational efficiency required from the algorithm, there is now another computational constraint of efficiently calculating the payments. The second is that not every algorithm has a matching payment scheme that enforces cooperation of rational participants. This may lead to a compromise in performance, settling with approximation algorithms that don't fully optimize the objective function.

1.1 Packet Scheduling

One of the main bottlenecks in the traffic flow inside communication networks is the management of queues in the connection points, such as switches and routers. If incoming traffic from several sources is directed toward the same destination, it may be impossible to immediately direct all the traffic toward the outgoing link, since the bandwidth of the outgoing link is limited, and packet loss is therefore unavoidable.

The traffic in communication networks tends to arrive in bursts, which is the motivation of buffering the packets in queues, located either at the incoming links, or the outgoing links, or both. The packets arriving in a burst are stored in queues, and are later sent in a speed determined by the bandwidth of both outgoing links and the backbone of the connection device. If the traffic is not too heavy, the queues will drain before more bursts arrive, avoiding packet loss. This best effort approach relies on statistical characteristics of communication traffic, assuming that the links are sometimes idle and not always used in full capacity.

The best effort approach is not sufficient for service providers who wish to guarantee

Quality of Service (QoS) to their users. QoS can be considered as a contract between the communication service provider and the network user. As long as the user's traffic does not exceed certain quotas agreed in the contract, the service provider guarantees specific characteristics of the network service, such as minimal bandwidth, maximal delay time, jitter, packet loss, etc. The demand for QoS results in a couple of problems that the service provider must solve. One problem is to determine what the service provider can guarantee to a user. The other is what to do with the user's traffic when it does not fulfill the terms of the contract.

One solution to the problem of QoS is *Premium Service* [21], in which the traffic is shaped upon the entry to the network. The service provider guarantees that subscribers of QoS will have a certain portion of the bandwidth allocated for their use. Packets that are not part of the Premium Service are treated by best effort mechanism, and might be dropped due to congestion. From the point of view of the user, the shared network is almost indistinguishable from a private link. From the point of view of the service provider, this is a solution with high utilization that allows dedicating private links, since the dedicated bandwidth may be used for general traffic when the traffic from QoS subscribers is idle. However, this service can be offered in parallel to a limited number of users who share a common link, since the total maximal bandwidth requirements may not exceed the link's capacity.

Assured Service [25] is a different approach that relies on *statistical multiplexing*, which takes into account the fact that usually worst case scenarios, where all users use the same network resources at once, do not occur. This assumption allows the service provider to apply an "overbooking" policy, which relaxes the constraints on the users' usage of the communication network, risking a potential buffer overflow, when congestion occurs. Assured Service supplies a relative guarantee to the user, which is a vague promise that certain traffic will be treated with higher priority at times of network congestion, achieving better throughput relative to the rest of the traffic. The concept that some packets are worth more than others is the basis of *Differentiated Services*. Packets that have QoS guarantee are distinguished from normal packets. Furthermore, the service provider may decide to treat differently packets of different paying customers (in a "pay more - get more" fashion).

Consider the following abstraction to analyze the performance of a single queue: The system is a queue that can hold up to B packets. The input packets that arrive to the queue have values of $v \in [1, \alpha]$. A packet's value represents the priority given to the packet by the service provider. The queue management is an online policy that decides for each packet, when it arrives, whether to place it in the queue or to reject it.

At arbitrary times a packet is sent from the queue (if it's not empty), at FIFO order, i.e. at the order of arrival. We assume that packet values are accumulative, so the benefit of the policy over a given input is the sum of values of accepted packets. We use competitive analysis [23] to analyze online policies, meaning that we derive bounds on the ratio between the total value of the optimal offline scheduling, to the total value of the online policy, over the worst case input.

An optional feature that may or may not be supported by the queue is preemption, which is the ability to extract previously accepted packets from the queue and drop them.

In preemptive queues, the online scheduler can accept packets more freely, compared to schedulers in non-preemptive queues, since while these packets accumulate in the queue, the option to preempt them remains, clearing space for future arriving packets.

Online packet scheduling with a non-preemptive queue was first analyzed in [2] for two packet values only, 1 and $\alpha > 1$. A general lower bound of $2 - 1/\alpha$ was derived, and several online policies were analyzed, deriving upper and lower bounds for each policy. An optimal online policy for the two value case with a competitive ratio of $2 - 1/\alpha$ was analyzed in [10]. The continuous case, were packet values may vary along the range $[1, \alpha]$ was also considered in [10]. A general lower bound of $\ln(\alpha) + 1$ was derived for any online policy, and an upper bound of $e\lceil \ln(\alpha) \rceil$ was proved.

When considering preemptive queues, improved competitive ratios are achievable, since while online policies can use preemption in order to improve their throughput, the optimal offline to which they are compared remains the same. Restricting the input to only two packet types, a natural greedy policy which accepts any packet given that there is sufficient queue space, and preempts low value packets when high value packets arrive to a full queue has a competitive ratio of asymptotically 2 [47]. In [48], an improved algorithm lowers the upper bound to 1.894 for the worst case α, and in [59] another algorithm sets the worst case upper bound to 1.304. This result is nearly optimal, since according to [74] no deterministic algorithm has a competitive ratio of less than 1.281. Englert and Westermann [30] have succeeded to close this gap with an online algorithm with a provable competitive ratio of asymptotically 1.281, as the queue's size B goes to infinity.

The asymptotically 2 competitive ratio of the greedy algorithm holds even under the extension of the model to arbitrary packet values. Breaking the 2 barrier for this model was first accomplished by Kesselman et al. [50], who presented a policy with a provable competitive ratio of 1.983, as well as a lower bound of 1.419. Mahdian et al. [20] have slightly modified this policy and have improved the upper bound to 1.75. Englert and Westermann [30] returned to the original algorithm of Kesselman et al. [50] and improved the competitive ratio to $\sqrt{3} \approx 1.732$. Their analysis is almost tight, as they have proved a lower bound of $\frac{2+\sqrt{2}}{2} \approx 1.707$ for this algorithm, which is still far from the general lower bound of 1.419.

Azar and Richter [19] have introduced the important role of *comparison based* policies, which rely only on the relative order of the packets' values, and not on the values themselves. The competitive ratio of any comparison based policy remains unchanged even when the possible inputs are restricted to packets with values of 0 and 1, under the constraint that ties can be broken either way. This property has simplified the analysis of online policies in various models. However, with the exception of the greedy policy, none of the online policies presented for the single queue model are comparison based. Since communication networks require high speed decision making, and since comparison based policies are usually simple to implement efficiently, it is interesting whether the 2 competitive ratio of the greedy policy can be broken again by a comparison based policy.

Policy	Input type	Bound
Ratio Partition	Two Classes	$2 - \frac{1}{\alpha}$
Round Robin	Arbitrary Classes	$e\lceil \ln(\alpha) \rceil$
Smooth SB		$\ln(\alpha) + 2 + O(\ln^2(\alpha)/B)$
General Lower Bound	Two Classes	$2 - \frac{1}{\alpha}$
	Arbitrary Classes	$\ln \alpha + 1$

Table 1.1: Summary of past and new results for non-preemptive queues

Policy	Input type	Bound	Randomized
Greedy	Two Classes	$2 - \frac{2}{1+\alpha}$	No
$\sqrt{\alpha}$-Preemptive		1.894	No
Mark & Flush		1.304	No
Optimal Online		1.281	No
Randomized Mark & Flush		1.25	One bit
Greedy	Arbitrary Classes	$2 - \frac{2}{1+\alpha}$	No
β-Preemptive Greedy		1.732	No
Randomized Half Greedy		1.75	One bit
General Lower Bound	Two Classes	1.281	No
	Two Classes	1.197	Yes
	Arbitrary Classes	1.419	No

Table 1.2: Summary of past and new results for preemptive queues

Our Contribution

The focus of Chapter 2 is online management of non-preemptive queues. The central result is an analysis of a RED-like [34] online policy for input packets with continuous values, with a provable upper bound of $\ln(\alpha)+2+O(\ln^2(\alpha)/B)$, which nearly matches the lower bound of $\ln(\alpha)+1$. Similarly to Random Early Detection, we suggest a policy that detects the possibility of congestion before the queue is full, and drops packets in advance. While RED uses probabilities to decide whether to drop a packet, we use the packet's value for this decision. Our analysis measures the exact influence of the queue size on the competitive ratio. For large values of α, we prove that $\ln(\alpha)+2$ is a lower bound for a large family of policies, which in addition to the value of the arriving packet, consider only the number of packets in the queue. For a low value of α, we analyze an alternative policy, which is an extension of the optimal online policy for the case of two packet values, and prove that its competitive ratio in this case is strictly lower than $\ln(\alpha)+2$, for low values of α. Table 1.1 presents a comparison of the central result in Chapter 2 to selected past results.

Chapter 3 considers algorithms for preemptive queues. Since all of the algorithms previously considered are deterministic, the work in this chapter is motivated by the question whether introducing randomization can improve the performance. Our work proves this is indeed true: We present an online algorithm for two packet values that uses

only a single random bit, yet it achieves a competitive ratio of 1.25 for the worst case α. This is not only an improvement to the current upper bound, but also shows that randomized algorithms strictly outperform deterministic ones, since the bound is below the lower bound of 1.281 for deterministic online algorithms.

Naturally, once randomization has been introduced, the lower bound of 1.281 for deterministic policies does not hold anymore. We prove a new lower bound of approximately 1.197 for any randomized online algorithm.

We present an algorithm for the general case of arbitrary packet values, which also uses only one random bit, and achieves a competitive ratio of 1.75. Although this result is slightly inferior to the deterministic policy of [30], it demonstrates the potential of randomization since our policy is comparison based, and since no deterministic comparison based policy that has a competitive ratio strictly lower than 2 is currently known. Table 1.2 summarizes our results in comparison to past results.

Related Models

Several models extend the FIFO queue model to a switch with several queues. A switch with m input queues and a single output (a *multiqueue*) has been modeled in [17, 19]. In this model, the online policy needs to select from which queue to send the next packet, as well as manage each queue separately. In [17], a 2ρ competitive policy is presented, where ρ is the competitive ratio of a policy for a single queue. Thus, a competitive ratio of 3.46 is implied by [30]. A 3 competitive policy, which is also comparison based, is presented in [19].

Scheduling within a multiqueue remains challenging even for uniform packets, although managing each queue separately becomes trivial. Azar and Richter [17] have shown that any reasonable algorithm, which serves any non-empty queue, is 2-competitive. Albers and Schmidt [3] were the first to break the bound of 2, presenting a 1.89 competitive algorithm. This was later improved by Azar and Litichevskey [16], who provided an algorithm whose competitive ratio approaches $\frac{e}{e-1} \approx 1.58$, for sufficiently large queue size B. Schmidt [71] has shown a randomized policy for this problem, which has a competitive ratio of 1.5, better than the deterministic lower bound of $\frac{e}{e-1}$, and near the randomized lower bound of approximately 1.46.

This model has been extended in [46, 18] to a fully *combined input-output queue* (CIOQ) switch, where several input queues may pass packets simultaneously to output queues, and each input may send more than one packet due to the speedup of the switch. Kesselman et al. [46] suggested two online algorithms, one with a competitive ratio that is linear to the switch speedup, and the other with a competitive ratio that is logarithmic to the range of packet values. Azar and Richter [18] presented an 8-competitive ratio algorithm, which is the first constant bound for this problem.

A different switching model analyzes policies for output queues that share the same memory. Different policies have been presented and analyzed for both preemptive [41] and non-preemptive queues [49].

Another variation on buffer management is the *delay bounded buffer* [10, 47], where packets may be sent in arbitrary order, and not by order of arrival as in FIFO queues.

Each packet arrives with a expiration time, and must be sent by its deadline, otherwise it expires and is lost.

1.2 Combinatorial Auctions and Allocation Problems

Auctions are a popular mechanism for selling and purchasing goods when traditional market mechanisms based on supply and demand are not satisfying, or are not implementable. In a combinatorial auction, a number of items is sold to a group of bidders whose valuation function may not be additive, meaning that a valuation of a bundle of items may express relations between subsets of items.

Mechanisms dealing with combinatorial auctions face several challenges. For instance, if there are k items then each bidder has to submit 2^k bids to fully express his valuation function. This exponential growth would make such an approach infeasible in practice. One alternative typical approach is to assume that bidders have simple preferences that can be expressed compactly.

Another important issue is computational, namely, deciding on the allocation of the items to the bidders. The allocation should maximize an objective function of the auctioneer, which is usually either the auctioneer's revenue or the economic efficiency. Finding an optimal allocation is computationally hard in general, although it is tractable in certain cases [63, 68, 75]. There are various methods to tackle this difficulty, such as finding an approximate allocation rather than the optimal one [55, 35] or developing mechanisms which work well in practice, though do not necessarily have a formal guarantee [35, 58, 63, 70].

Lehmann et al. [55] have concentrated on combinatorial auctions where bidders' valuations are known to be subadditive. A very natural subclass of subadditive valuations is decreasing marginal utilities (also known as submodular valuations), where the valuation a bidder gives to an item monotonically decreases as the set of items he already purchased grows. Formally, if $V(\cdot)$ denotes the valuation function of a bidder, then for any two bundles S and T such that $S \subseteq T$ and for any item x such that $x \notin T$, we have $V(S \cup \{x\}) - V(S) \geq V(T \cup \{x\}) - V(T)$. Lehmann et al. [55] presented a greedy algorithm for auctions with decreasing marginal utilities, and proved that it is a 2-approximation. This was slightly improved by Dobzinski and Schapira [28], who gave a randomized $2 - \frac{1}{n}$ approximation, where n is the number of bidders.

In general, combinatorial auctions are a family of assignment problems, where the valuation functions may not be additive, resulting in a difficulty in optimizing a given objective function. A different yet related class of problems is allocation problems where the valuations are additive, but not every assignment is feasible.

A classical assignment problem is the Generalized Assignment Problem (GAP), presented by Shmoys and Tardos [72], where items are allocated to agents. For each pair of an item and an agent, there is a cost for allocating that item to the agent, as well as a size constraint. Each agent has a limit on the total size of items that can be assigned to it. The goal is to minimize the total cost of a feasible assignment, restricted to the size constraints, given that such an assignment exists.

A natural variant of GAP is MAX-GAP [24], where items have positive values instead

of costs, and the goal is to maximize the total value of the allocation, subject to size constraints (not all items have to be allocated). MAX-GAP is a generalization of the multiple knapsack problem (MKP), where the value and size of an item is the same among all agents. MKP is a generalization of the knapsack problem [54], where a singe agent participates in the allocation. The knapsack problem generalizes the subset-sum problem [44], where item sizes are equal to their values.

Our Contribution

Chapter 4 concentrates on auctions with budget constraints, which are a special case of auctions with decreasing marginal utilities. In such auctions bidder (agent) i has a budget constraint of d_i, and the valuations are additive as long as the budget constraint is not met. Once the budget constraint is reached, the valuation equals to it. Namely, the valuation of bidder i for a bundle A is given by $\min(d_i, \sum_{j \in A} b_{ij})$, where b_{ij} is the value of item j to bidder i.

An auction with budget constraints can also be viewed as a relaxation of allocation problems. Instead of a strict size constraint for each agent, which cannot be exceeded, the budget constraint is a soft constraint, since allocations that exceed it are still feasible, but their valuation is no longer additive. Additionally, items are characterized only by by their values, and not by their sizes.

Bidding in auctions with budget constraints is a rather concise process, since each bidder has to submit only k bids, where k is the number of auctioned items, and also submit the budget constraint. Our goal is to maximize the total valuation of the bidders.

We prove that finding an optimal allocation that maximizes the total valuation is NP-hard even if there are only two bidders with identical valuations. We show that an exact solution can be found in time $O(\min(n4^k, k^2 4^k + nk))$, where k denotes the number of items and n denotes the number of bidders. If the number of bidders is constant, there is also a pseudo-polynomial algorithm.

Our main results are polynomial time approximation algorithms. We present a randomized algorithm which is a $\frac{e}{e-1} \approx 1.582$ approximation, and also derandomize it. We then exhibit improved approximation ratios when all bidders have the same budget constraint (but possibly different valuations), and prove that the approximation ratio is between 1.3837 and 1.3951. We also present an FPTAS for the case of a constant number of bidders. We remark that the greedy allocation algorithm of [55] for auctions with budget constraints, even if there are only two bidders, has approximation ratio 2.

Related Models

An auction with budget constraints is also a special case of the AdWords Assignment Problem (AAP) problem, which is a combination of a budget constrained problem with multiple copies of items with a two dimensional rectangular packing problem. The AdWord problem is motivated by the pricing scheme used in the sponsored search mechanism in Google's search engine [1].

In sponsored search mechanisms, advertisers submit a rectangular advertisement and select related keywords, for which their advertisement should appear, when users search for these keywords. The payment mechanism used by Google is as follows: Advertisers decide in advance on a fixed payment, which they pay whenever a user clicks on their ad, and also limit their total advertisement budget.

Assuming that Google has an estimation on the number of clicks each advertisement will receive when assigned to a keyword, there is an estimated profit for each assignment of an advertisement to a keyword. If for each keyword there was only enough advertisement space for a single ad, then in order to maximize its profit, Google had to solve an auction with budget constraints, where the bidders are the advertisers and the items are the keywords. In AAP, since for each keyword it is possible to publish several ads, there is an additional space constraint of being able to arrange the ads in the region of the web page that is devoted for sponsored ads.

Fleischer et al. [33] give a $\frac{2e}{e-1}$ approximation for AAD. If all ads have the same width, then the two dimensional packing constraint reduces to the knapsack problem, and the approximation ratio reduces to $\frac{e}{e-1}$, which is a generalization to the main result in Chapter 4.

While in auctions with budget constraints valuation functions can be compactly represented as a vector of $k+1$ numbers, this is not the case with submodular valuations in general, which may require an exponential communication complexity to be fully described. Therefore, allocation algorithms for submodular valuations, as well as algorithms for other families of valuation functions, use queries to gather partial information about the bidder's preferences. Value queries, where the bidder is asked to report its valuation for a given bundle of items are the most natural type of queries.

Another type of query is a *demand query*, which sets prices to single items and requests the bidder to report its most profitable bundle under these prices. Demand queries are stronger than value queries, since the latter can be simulated by the former using a polynomial number of queries [22]. The converse, however, is not necessarily true, as there are valuations where it is NP-Hard to simulate a demand query using value queries. Using demand queries, there exists a $\frac{e}{e-1}$ approximation for combinatorial auctions with submodular valuations, which also holds for a wider family of valuation functions, called XOS[1]. The approximation ratio for XOS valuations is tight, for any type of queries [28], while for submodular valuations, Feige and Vondrak [32] have slightly improved the approximation ratio to $\frac{e}{e-1} - \epsilon$, for a small yet constant ϵ.

1.3 Mechanism Design with Single Parameter Agents

At the core of Mechanism Design is the desire to define a system such that selfish agents interacting with the system would reach the desired outcome. Example of such setting is a centralized decision maker with regulatory power (government, employer, etc.) that wishes to optimize an objective function by aggregating signals on private information

[1] An XOS valuation is the maximum among a set of additive valuations. In [55], submodular valuations are shown to be a special case of XOS valuations.

collected from agents (buyers, sellers, workers, etc.). The private information of an agent, which is also referred as its type, may include its demand curve, income reports, potential labor power, etc. The information (e.g., bids) that the agents submit signals their types, however they are expected to strategically choose their signal such that they will affect the decision making in a manner that will optimize their own utility.

Mechanism design aims to give the agents an incentive to simply reveal their information to the decision maker, and avoid strategic signaling. This goal is achieved by assigning payments from or to the mechanism, which encourage the agents' cooperation by making truthtelling a (weakly) dominant strategy. I.e., for each agent, regardless of the other agents' actions, reporting its type is a best response strategy. Settings where each agent has a dominant strategy are more convenient to the central decision maker, since it is likely that all agents will indeed follow their dominant strategies, and this frees the central decision maker from considering the implications of strategic behaviors of the agents. By the revelation principle, if dominant strategy mechanisms exist, it is sufficient to consider mechanisms where truthtelling is the dominant strategy (see, e.g., [60]). Under the assumption that each agent is aware of its own type, truthtelling is the simplest strategy for an agent to follow, which motivates making it a dominant strategy.

Algorithmic mechanism design [64] extends mechanism design into the realm of discrete algorithms, by considering computational limitations on the algorithmic process that calculates the decision rule. Much of the algorithmic mechanism design literature is devoted to studying a wide range of different utility and objective functions and design appropriate efficient mechanisms for them.

One can hope that rather than verifying that a given algorithm and a payment scheme construct together a truthful mechanism, we would rather have a criteria for verifying whether a given algorithm can be melded into a truthful mechanism by adding a proper payment scheme (i.e., if it is rationalizable). Such criteria was given by Rochet in [67]. Additionally, if a decision rule is rationalizable, there is a mathematical formulation to the suitable payment functions. The drawback of Rochet's characterization of rationalizable decision rules is that it does not provide a computationally practical method for testing rationalizability. A simpler condition, known as *weak monotonicity* [53] or as *2-cycle inequality* [40], is a necessary condition for rationalizability, but not a sufficient condition, as demonstrated in [53].

Given a restricted domain of single parameter agents, we search for a sufficient condition, which is not a necessary one in general, although in some settings we can show it is also necessary. On one hand, we aim for a condition that is simple enough to be easily applied to test rationalizability. On the other hand, the condition should be general enough to be useful in constructing truthful mechanisms. We focus on mechanisms for single parameter agents, which although restricted, include a wide range of mechanisms, and weak monotonicity alone does not imply rationalizability by itself.

In the case that the output domain of the mechanism is continuous, the *Mirrlees-Spence condition* [73, 61] is a simple condition that characterizes truthful single parameter mechanisms. Unfortunately, the requirement that the output space is continuous is a severe limitation, since in algorithmic mechanism design, the outcome space for many problems is discrete (Example of such problems include auctions, machine scheduling,

routing, etc.). In such settings the Mirrlees-Spence condition is inapplicable, since in discrete spaces the preliminary requirements required for applying this condition immediately fail. Currently, algorithmic mechanism design requires deriving rationalizability conditions for each model, separately. A replacement to the Mirrlees-Spence condition that holds for discrete models would simplify this process in single parameter setups.

Archer and Tardos [12] suggested a simple characterization for single parameter agents, where the valuation (or cost) of an agent is linear with respect to its type, and therefore the type can be considered as representing a value (or cost) per unit. Unlike the Mirrlees-Spence condition, this characterization does not require the allocation function to be continuous, only monotone in the type. However, it significantly limits the structure of the valuation function.

Our Contribution

Chapter 5 presents a generalization to both the Mirrlees-Spence condition, and Archer and Tardos' characterization, which relaxes prior assumptions on the structure of the valuation functions and the decision rule. We define a property named *Halfway Monotone Derivative (HMD)*, prove that all HMD algorithms are rationalizable and characterize the structure of the payment function. We also prove that for some valuation functions, HMD is also a necessary condition.

We apply our condition on several sample algorithmic problems and derive simple conditions for truthfulness, as well as simple structures of the payment functions. Specifically, we show that in all single commodity auctions, the critical value condition is equivalent to the HMD condition. We define a setting where an agent's utility from getting service before some deadline is proportional to the time before the deadline that the service was given, and analyze conditions for rationalizability.

We then extend our work to models where the valuation of each agent is also affected by the true types of the other agents. Since in many cases it is not possible to achieve a reasonable truthful implementation in dominant strategies, we settle with an *ex-post truthful* implementation, where there is a Nash equilibrium where all agents reveal their true types. We show that the HMD condition holds for this model.

Another application is given in Chapter 6, where we revisit the model in Chapter 4 and analyze a single parameter version of an auction with budget constraints. We give a piecewise monotonicity condition for rationalizability, and then present rationalizable approximation algorithms for this problem.

Chapter 7 considers machine scheduling problems where machines or job owners may have a cost per unit utility function. When the only agents are the machines, the monotonicity condition of Archer and Tardos [12] is an outcome of the HMD condition. We present a truthful FPTAS for a fixed number of machine agents. In the dual scheduling problem where the job owners are strategic, the monotonicity condition in [15] can be derived using HMD. For strategic job owners, we present a 2 approximation, which is ex-post truthful, the first constant approximation ratio for the problem. Finally, we consider the problem where both machines and jobs are strategic, and present a 5 approximation that is ex-post truthful for this problem.

Related Work

Provided that each agent's valuation function depends only on its type and the outcome, a truthful mechanism for any optimization problem where the objective function of the algorithm is to maximize the sum of utilities (maximize the social welfare) is the *VCG* mechanism [26, 39, 76], which can also be generalized to any maximization problem of an affine function. However, for other objective functions VCG is not applicable. Additionally, optimally solving an affine maximization problem may be computationally hard, which motivates developing truthful mechanisms that only approximate the optimal solution. However, the VCG mechanism does not cope well with approximate solutions [65].

If the domain of valuation functions is unrestricted and there are at least 3 possible outcomes, Roberts [66] proved that a mechanism is truthful is and only if it is a weighted VCG mechanism. In particular, the allocation must maximize an affine function of the valuations.

Given that the valuation function of the agents is taken from some restricted domain, several papers have attempted to characterize function families for which weak monotonicity would also be a sufficient condition for truthfulness [40, 53]. Saks and Yu [69] have generalized these results to any finite outcome space and valuation functions defined on convex domains (in contrast, we do not require the outcome space to be finite).

Although weak monotonicity is a simple condition, in practice it is not always easy to prove this property holds for a given algorithm. We believe that a weaker sufficient yet not necessary condition may be easier to prove, yet most practical algorithms will not stumble into the pitfall of rationalizable algorithms that this condition does not hold for them.

The classic problem of scheduling jobs on parallel related machines was dealt by several approximation approaches. The known basic result of an LPT algorithm which sorts the jobs in non-increasing order of job size, then allocates a job to the machine which will be least busy afterwards is a 2-approximation [37]. An FPTAS was first introduced by Horowitz and Sahni for the case where the number of machines is constant. Their approach was based on rounding an exact solution by dynamic programming [43]. Finally, in the late 80's Hochbaum and Shmoys introduced a PTAS for the general case of an arbitrary number of machines [42, 31]. Since the problem is strongly NP-Complete, no FPTAS is possible for the general case, and their result remains the best possible, unless P=NP.

Scheduling with selfish machine agents was first analyzed by Ronen and Nisan. Their results mainly concern scheduling on unrelated machines, known also as $R||C_{max}$. The case of uniformly related machines was first tackled by Archer and Tardos [12] who showed that the former known approximation algorithms for the problem are not truthful. They introduced a truthful randomized mechanism which achieves a 3-approximation to the problem, which was later improved to a 2-approximation [11]. This approach achieves truthfulness with respect to the *expected profit* only. Thus it possible that even though the expected profit is maximized when telling the truth, there might exist a better (untruthful) strategy given a particular outcome of the mechanism's coin flips.

The first deterministic result is due to Auletta et al [13]. They show a deterministic truthful mechanism which is $(4+\varepsilon)$-approximation for any *fixed* number of machines. The best deterministic result for the case of an arbitrary number of machines was given by Kovacs [51], which is a 3-approximation, based on rounding the speeds of the machines and then executing LPT scheduling [37]. A better analysis of the same algorithm improved the approximation ratio to approximately 2.8 [52].

A different approach by Nisan and Ronen introduces another model in which the mechanism is allowed to observe the machines process their jobs and compute the payments afterwards. Using these *mechanisms with verification* [64, 14] allows application of penalty on lying agents, and was shown to cope well with the existing known approximation algorithms.

Scheduling with selfish job agents was first considered by Auletta et al. [15]. For uniformly related machines, a $r+\epsilon$ randomized approximation mechanism that is ex-post truthful in expectation is given, where r is the ratio between the speeds of the fastest machine and the slowest machine. Additionally, a lower bound of $\frac{\sqrt{17}+1}{4} \approx 1.28$ on the approximation ratio of any ex-post truthful mechanism is given, independent of its running time.

Chapter 2

Non-Preemptive Queue Management

This chapter considers online deterministic algorithms for managing packet admission into a single non-preemptive FIFO queue, with arbitrary packet values. It is known that no online algorithm has a competitive ratio lower than $\ln(\alpha)+1$ [10], where α is the ratio between the highest and lowest values of packets. The focus in this chapter is on online algorithms whose competitive ratio is near the general lower bound, given that the queue is significantly large.

2.1 Model and Notations

2.1.1 Packets

A packet p is a basic unit that describes the input of the system. Each packet is identified by its value $v(p)$, which is the value that the system gains by sending the packet. Packet values may vary from 1 to α, where $\alpha \geq 1$ is an arbitrary value, known in advance. When the packet p is clear from the context, we use v instead of $v(p)$ for simplicity. Additionally, v is used when referring in general to packet values, and not to a specific packet.

2.1.2 Input Streams

The input to the system is a finite stream of events, occurring at discrete times $t = 1, 2, ..., n$. There are two types of events, `arrive(`p`)`, which is the arrival of a packet with value $v(p)$ to the system and `send()`, which allows the system to send one of the packets that arrived earlier.

2.1.3 FIFO Queues

The system uses a queue for buffering the arriving packets. Let B denote the maximal number of packets that can be held in the queue. In our analysis we assume that $B \geq \ln(\alpha) + 2$. This is a reasonable assumption since usually buffers are fairly large. In

addition, if $B = o(\ln(\alpha))$, no online policy has a logarithmic order competitive ratio. Each `arrive(`p`)` event invokes a response from the system, which is either to accept the packet and place it inside the queue, or the reject it. The decision whether to accept or to reject a packet depends on the specific policy used to manage the queue. No more than B packets can be stored in the queue at the same time, and preemption is not allowed. During a `send()` event the system extracts the most recent packet in the queue and sends it, unless the queue is empty.

2.1.4 Online Policies

A policy is an algorithm that given an input stream decides which packets to accept and which packets to reject. Given an input stream σ and a policy A, let $h_A^\sigma(t)$ denote the number of packets in the queue at time t. Let $V_A^\sigma(t)$ denote the total value of packets accepted until time t. Notice that since we restrict ourselves to non-preemptive queues, each accepted packet is eventually sent, therefore we may consider $V_A^\sigma(t)$ as the total value assured by policy A until time t over the input stream. Throughout the chapter, we will use the notation $h_A(t)$ and $V_A(t)$ when σ is clear from the context.

A policy is considered an online policy if the decision to accept or reject a packet depends only on the previous and current events. We use competitive ratio [23] to analyze the performance of online policies. An online policy is *c-competitive* if for any input sequence σ and for some constant $b \geq 0$, $c \cdot V_{\mathrm{ON}}^\sigma + b \geq V_{\mathrm{OPT}}^\sigma$, where V_{ON}^σ is the total value of the online policy and V_{OPT}^σ is the total value of an optimal offline policy.

2.2 Smooth Selective Barrier Policy

We define a policy that is analogous to the mechanism of Random Early Detection (RED) [34] gateways for congestion avoidance in packet switching networks. The RED gateway measures the number of packets in the FIFO queue, and marks or drops packets with a certain probability, if this number exceeds a certain threshold. The probability of marking/droping a packet depends on the number of packets in the queue. RED gateways are designed to accompany a congestion control protocol such as TCP. By marking or dropping packets, the gateway hails the connections to reduce their windows, and thus keeps the number of packets in the queue low.

The `Selective Barrier Policy` is an online policy intuitively designed as a derandomized variation of RED. Like RED, the policy becomes more restrictive as the queue size increases. Unlike RED (which is ignorant to packet values), the decision to drop a packet is deterministically based on the packet's value, and the number of packets in the queue.

Formally, we define $\mathcal{F}(v) : [1, \alpha] \to [1, B]$ to be a monotone function that bounds the maximal number of packets that can be in the queue if a packet of value v was just accepted. If an `arrive(`p`)` event occurs at time t then the packet is accepted if $\mathcal{F}(v(p)) \geq h_{\mathrm{ON}}(t) + 1$ and is rejected otherwise. Intuitively, we define $\mathcal{F}^{-1}(h)$ to be the lowest value of a packet that will be accepted by the policy when the queue already holds $h - 1$ packets.

The competitive ratio of the Selective Barrier Policy depends on the exact mapping of $\mathcal{F}(\cdot)$. In [10] an upper bound of $e \cdot \lceil \ln(\alpha) \rceil$ was derived, where $\mathcal{F}(v) = \frac{i}{\lceil \ln(\alpha) \rceil} B$ for each $v \in [e^{i-1}, e^i)$. This mapping divides the queue equally into $\lceil \ln(\alpha) \rceil$ sub-queues, and increments the threshold for accepting packets in exponential steps. Due to the division to sub-queues, the policy might accept only a $1/\lceil \ln(\alpha) \rceil$ fraction of the packets in a certain range. The e factor in the competitive ratio is mainly due to the fact that packet values in $[e^{i-1}, e^i)$ may differ by a factor of e in their value, yet the policy treats them identically.

The main contributions in this chapter are a finer definition of $\mathcal{F}(\cdot)$, and more importantly, a proof of a tight bound of the competitive ratio. Specifically, we consider the mapping $\mathcal{F}(v) = \mathcal{F}(1) + \frac{\ln(v)}{\ln(\alpha)} S$, where $\mathcal{F}(1) = \lceil \frac{B}{\ln(\alpha)+2} \rceil$ and $S = B - \mathcal{F}(1)$. We name this variant of the Selective Barrier Policy as the `Smooth Selective Barrier Policy`. Intuitively, the first $\mathcal{F}(1)$ slots in the queue are unrestricted, meaning that the policy accepts any packet into these slots, regardless of the packet value. In the remaining slots we accept packets if their value is larger than a threshold that increases in an exponential rate, yet its increment is smooth, as can be observed by the following lemma:

Lemma 2.2.1 *For any $h \in [\mathcal{F}(1), B-1]$, we have $\mathcal{F}^{-1}(h+1) = \mathcal{F}^{-1}(h)\alpha^{1/S}$*

Proof: Let v be the value such that $\mathcal{F}(v) = h$. Since the mapping chosen for $\mathcal{F}(\cdot)$ is strictly monotone increasing, then v is unique and $\mathcal{F}^{-1}(\mathcal{F}(v)) = v$. By the definition of $\mathcal{F}(\cdot)$, we have the following:

$$\begin{aligned} \mathcal{F}(v)+1 &= \mathcal{F}(1) + \frac{\ln(v)}{\ln(\alpha)} S + 1 = \mathcal{F}(1) + \frac{\ln(v) + \frac{\ln(\alpha)}{S}}{\ln(\alpha)} S \\ &= \mathcal{F}(1) + \frac{\ln(v) + \ln(\alpha^{1/S})}{\ln(\alpha)} S = \mathcal{F}(1) + \frac{\ln(v\alpha^{1/S})}{\ln(\alpha)} S = \mathcal{F}(v\alpha^{1/S}) \end{aligned}$$

Therefore, we have

$$\mathcal{F}^{-1}(h+1) = \mathcal{F}^{-1}(\mathcal{F}(v)+1) = \mathcal{F}^{-1}(\mathcal{F}(v\alpha^{1/S})) = v\alpha^{1/S} = \mathcal{F}^{-1}(h)\alpha^{1/S}$$

which completes the proof. ∎

2.3 Analysis of Smooth Selective Barrier

We now prove an upper bound for the Smooth Selective Barrier Policy with $\mathcal{F}(\cdot)$ as defined in Section 2.2 by comparing it to an optimal offline policy over an arbitrary input. Our proof is constructed from two stages: First, we simplify the input by modifying the values of the packets in a conservative way such that the competitive ratio cannot decrease due to the modification. In the second stage we define a potential function that bounds the possible additional benefit that the offline can gain, without any gain to the benefit of the online. We use this potential function to prove inductively that given a modified input, the ratio between the benefit of the online and the benefit of the offline plus its potential is always bounded.

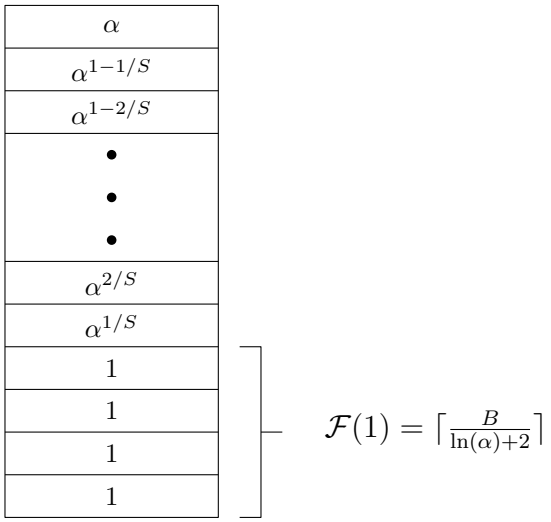

Figure 2.1: The relationship between the queue cells and the value of the function $\mathcal{F}(\cdot)$. The value written in each cell is the minimal value of a packet that may be stored in it.

In our proof we use ON to denote the Smooth Selective Barrier Policy while OPT denotes the optimal offline. We perform a simple relaxation on the input stream, defined as follows: If the online accepts a packet p with value $v = v(p)$ at time t, then we may reduce the value of v to $v' = \mathcal{F}^{-1}(h_{\mathrm{ON}}(t))$, i.e. the v' is the lowest value that the online policy still accepts. We derive the following claims for relaxed inputs:

Lemma 2.3.1 *The competitive ratio of a relaxed input is at least the same as the competitive ratio of the original input.*

Proof: We analyze the competitive ratio after each change of a packet value v to v', and notice that it does not reduce the competitive ratio. After changing the value of a single packet, the decisions of the Smooth Selective Barrier Policy remain unchanged, since the policy does not consider the values of previously accepted packets, only their amount, and therefore loses a benefit of exactly $v - v'$. The optimal offline is unaffected if it rejected v in the original schedule, and loses at most benefit $v - v'$ if it did accept the packet. Since the competitive ratio is at least 1, it cannot decrease over the relaxed input. ∎

In the following lemma we prove an upper bound on the values of the packets that the offline accepts, which depends on the state of the online queue.

Lemma 2.3.2 *In a relaxed input, if the offline accepts p at time t, then*

$$\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) \geq v(p)$$

Proof: If the online policy also accepts the packet p, then by the relaxation we have $\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) = v(p)$. Otherwise, the packet was rejected, and therefore $\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) > v(p)$. ∎

We now introduce a potential function $\phi(t)$. Roughly, $\phi(t)$ measures the extra benefit that the offline can gain without changing the online. The potential $\phi(t)$ is defined as follows:

$$\phi(t) = \begin{cases} \max\left(0, \frac{h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t)\right) & : \quad h_{\mathrm{ON}}(t) < \mathcal{F}(1) \\ (B - h_{\mathrm{OPT}}(t))\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) + \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t)-1} \mathcal{F}^{-1}(i) & : \quad h_{\mathrm{ON}}(t) \geq \mathcal{F}(1) \end{cases}$$

Notice that both functions used in the definition of $\phi(t)$ are equal at $h_{\mathrm{ON}}(t) = \mathcal{F}(1)$. When $h_{\mathrm{ON}}(t) \geq \mathcal{F}(1)$, $\phi(t)$ measures the maximal possible gain in the benefit of the offline, without increasing the benefit of the online. The offline can fill its queue with $B - h_{\mathrm{OPT}}(t)$ packets, which will be rejected by the online if their values are less than $\mathcal{F}^{-1}(h_{\mathrm{ON}}(t))$. Then, the offline can send packets in order to accept more packets, but the online sends packets too, so the new packets must have decreasing values, otherwise the online will accept them. When $h_{\mathrm{ON}}(t) < \mathcal{F}(1)$, $\phi(t)$ increments in linear steps, measuring how near the online queue is to $\mathcal{F}(1)$, which is the threshold for accepting packets with value 1. Using the potential $\phi(t)$ we prove the following invariant:

Theorem 2.3.3 *For any relaxed input sequence and for any time t,*

$$C \cdot V_{\mathrm{ON}}(t) \geq V_{\mathrm{OPT}}(t) + \phi(t),$$

where $C = \max\left\{\ln(\alpha) + 2, B\alpha^{1/S} - B + 1\right\}$.

Proof: The proof is by induction. For $t = 0$ we have $V_{\mathrm{ON}}(0) = 0 = V_{\mathrm{OPT}}(0)$. Since $h_{\mathrm{ON}}(0) = 0$ we also have $\phi(0) = 0$ and the claim holds. Assuming the claim holds for $t = t_0$, we will prove it holds for time $t + 1$.

If at time t_0 both queues are empty and the input has ended, then the inequality holds trivially for any $t \geq t_0$. Otherwise, the event at time $t+1$ is either `arrive(`p`)` or `send()`. We analyze each case separately.

If the next event is `arrive(`p`)`, we analyze its effect in two stages: First, the online policy decides whether to accept p or to reject it, then the offline decides. This separation obviously has no effect on the behavior of both policies, so it is sufficient to prove that the claim holds after each stage. Using this technique, at each state either $V_{\mathrm{ON}}()$ or $V_{\mathrm{OPT}}()$ remains unchanged. If at any stage the active policy rejects the packet, then the inequality remains unchanged and therefore the claim obviously holds. If the packet is accepted we analyze the inequality separately for each policy and for each status of $h_{\mathrm{ON}}(t)$.

We note that if $h_{\mathrm{ON}}(t) < \mathcal{F}(1)$, then the online must accept p, and by the relaxation, $v(p) = 1$. Since $V_{\mathrm{ON}}(t+1) - V_{\mathrm{ON}}(t) = v(p) = 1$, the left side of the inequality increases by C. As for the right side, we concentrate on the case where at the end of the event, $\phi(t+1) > 0$. Otherwise, the right side of the inequality can only increase by 1, which is less than C. With this assumption in mind, we now bound the change in the right side in two stages:

After the online policy accepts p, the left side of the inequality increases by C. The right side of the inequality changes by the increase of the potential, which is bounded by:

$$\begin{aligned}
&\frac{h_{\mathrm{ON}}(t+1)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t) - \frac{h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B + h_{\mathrm{OPT}}(t) \\
&= \frac{h_{\mathrm{ON}}(t+1) - h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B\frac{B}{\mathcal{F}(1)} = \frac{B}{\lceil \frac{B}{\ln(\alpha)+2} \rceil} \leq \frac{B}{\frac{B}{\ln(\alpha)+2}} = \ln(\alpha)+2
\end{aligned}$$

The claim holds for $C \geq \ln(\alpha) + 2$.

In case the second stage occurs, the offline policy accepts p and the left side of the inequality remains unchanged while the right side changes as follows:

$$\begin{aligned}
&v(p) + \frac{h_{\mathrm{ON}}(t+1)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t+1) - \frac{h_{\mathrm{ON}}(t+1)}{\mathcal{F}(1)}B + h_{\mathrm{OPT}}(t) \\
= \;& v(p) - [h_{\mathrm{OPT}}(t)+1] + h_{\mathrm{OPT}}(t) = 1 - h_{\mathrm{OPT}}(t) - 1 + h_{\mathrm{OPT}}(t) = 0,
\end{aligned}$$

and the claim holds.

We now move on to the case where $h_{\mathrm{ON}}(t) \geq \mathcal{F}(1)$. If the online policy accepts p, then due to the relaxation of the input, we have $v(p) = \mathcal{F}^{-1}(h_{\mathrm{ON}}(t))$. The benefit of the online increases by $v(p)$, therefore the left side of the equation increases by $v(p)C$. Since the online queue accepts p, we have $h_{\mathrm{ON}}(t+1) = h_{\mathrm{ON}}(t)+1$. The change in the potential is therefore:

$$\begin{aligned}
&\left(B - h_{\mathrm{OPT}}(t)\right)\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t+1)\right) + \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t+1)-1} \mathcal{F}^{-1}(i) \\
&\qquad - \left(B - h_{\mathrm{OPT}}(t)\right)\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t)\right) - \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t)-1} \mathcal{F}^{-1}(i) \\
= \;& \left(B - h_{\mathrm{OPT}}(t)\right)\left[\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t)+1\right) - \mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t)\right)\right] \\
&\quad + \mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t)+1-1\right) \\
= \;& \left(B - h_{\mathrm{OPT}}(t)\right)\left(v(p)\alpha^{1/S} - v(p)\right) + v(p) \\
= \;& v(p)\left[\left(B - h_{\mathrm{OPT}}(t)\right)\left(\alpha^{1/S} - 1\right) + 1\right] \\
\leq \;& v(p)\left[B\left(\alpha^{1/S} - 1\right) + 1\right],
\end{aligned}$$

where the second identity is by Lemma 2.2.1. The inductive claim holds for $C \geq B\alpha^{1/S} - B + 1$.

If the offline policy accepts p and $h_{\mathrm{ON}}(t) \geq \mathcal{F}(1)$: By Lemma 2.3.2, we have $v(p) \leq \mathcal{F}^{-1}(h_{\mathrm{ON}}(t+1))$. The left side of the inequality remains unchanged, where the change to

the right side is as follows:

$$\begin{aligned} & V_{\mathrm{OPT}}(t+1) - V_{\mathrm{OPT}}(t) + (B - h_{\mathrm{OPT}}(t+1))\,\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t+1)\right) \\ & \quad - (B - h_{\mathrm{OPT}}(t))\,\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t+1)\right) \\ & \quad + \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t+1)-1} \mathcal{F}^{-1}(i) - \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t+1)-1} \mathcal{F}^{-1}(i) \\ = \;& v(p) + (h_{\mathrm{OPT}}(t) - h_{\mathrm{OPT}}(t+1))\,\mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t+1)\right) + 0 \\ = \;& v(p) - \mathcal{F}^{-1}\left(h_{\mathrm{ON}}(t+1)\right) \leq 0 \end{aligned}$$

This implies that the right side of the inequality cannot increase, and therefore the inductive claim remains true.

In the case of a `send()` event, the value of both policies remains unchanged. Therefore, it is sufficient to prove that the potential does not increase. If the queue of the Smooth Selective Barrier Policy is already empty when the `send()` event occurs, then the potential is 0, and remains unchanged. If the queue of the optimal policy is empty when the `send()` event occurs, then the queue of the online must be empty as well, otherwise the offline has rejected at least one packet falsely, and can be improved. Therefore, it is sufficient to analyze the case where both queues drain by one packet.

If $h_{\mathrm{ON}}(t) < \mathcal{F}(1)$ then the potential is either 0 or $\frac{h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t)$. If after a packet is sent the potential is 0 then trivially the potential did not increase. Otherwise, the change in the potential is at most:

$$\begin{aligned} & \phi(t+1) - \phi(t) \\ \leq \;& \frac{h_{\mathrm{ON}}(t+1)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t+1) - \frac{h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B + h_{\mathrm{OPT}}(t) \\ = \;& \frac{h_{\mathrm{ON}}(t) - 1 - h_{\mathrm{ON}}(t)}{\mathcal{F}(1)}B - h_{\mathrm{OPT}}(t) + 1 + h_{\mathrm{OPT}}(t) = 1 - \frac{B}{\mathcal{F}(1)} \end{aligned}$$

Since $\mathcal{F}(1) \leq B$ the potential can only decrease, and therefore the inductive claim holds.

If $h_{\mathrm{ON}}(t) \geq \mathcal{F}(1)$ the change in the potential is:

$$\begin{aligned}
&\phi(t+1) - \phi(t) \\
&= (B - h_{\mathrm{OPT}}(t+1))\mathcal{F}^{-1}(h_{\mathrm{ON}}(t+1)) + \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t+1)-1} \mathcal{F}^{-1}(i) \\
&\quad - (B - h_{\mathrm{OPT}}(t))\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) - \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t)-1} \mathcal{F}^{-1}(i) \\
&= (B - h_{\mathrm{OPT}}(t) + 1)\mathcal{F}^{-1}(h_{\mathrm{ON}}(t) - 1) + \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t)-2} \mathcal{F}^{-1}(i) \\
&\quad -(B - h_{\mathrm{OPT}}(t))\mathcal{F}^{-1}(h_{\mathrm{ON}}(t)) - \sum_{i=\mathcal{F}(1)}^{h_{\mathrm{ON}}(t)-1} \mathcal{F}^{-1}(i) \\
&= (B - h_{\mathrm{OPT}}(t)) \left[\mathcal{F}^{-1}(h_{\mathrm{ON}}(t) - 1) - \mathcal{F}^{-1}(h_{\mathrm{ON}}(t))\right] \leq 0
\end{aligned}$$

As the potential does not increase, the claim holds.

Since the claim is not violated after any step of the induction, it holds for any appropriate value of C. The constraints on C are dictated by the values of B and α, and are set to:

$$C \geq \max\left(\ln(\alpha) + 2, B\alpha^{1/S} - B + 1\right)$$

■

Theorem 2.3.3 proves the existence of a competitive ratio C, for the Smooth Selective Barrier Policy with the suggested function $\mathcal{F}(\cdot)$. Assuming $B \geq \ln(\alpha) + 2$, we derive the following bound on C.

Theorem 2.3.4 *For $B \geq \ln(\alpha) + 2$ the competitive ratio of the Smooth Selective Barrier Policy is at most $C \leq \max\left\{8.155, \ln(\alpha) + 2 + \frac{9\ln^2(\alpha)}{B} + \frac{3\ln(\alpha)}{B}\right\}$.*

Proof: We first prove the bound for $\alpha \geq e$. We prove the claim separately for each part of the maximum expression derived in Theorem 2.3.3. For $C \geq \ln(\alpha) + 2$ The claim trivially holds. We notice that for $\alpha \geq e$ we have $\ln(\alpha) \geq 1$ and analyze $B\alpha^{1/S} - B + 1$.

$$B\alpha^{1/S} - B + 1 = B\alpha^{\frac{1}{B - \lceil B/(\ln(\alpha)+2)\rceil}} - B + 1 = \quad Be^{\frac{\ln(\alpha)}{B - B/(\ln(\alpha)+2) - 1}} - B + 1$$

Given the inequality $e^x \leq 1 + x + x^2$, which holds for any $0 \leq x \leq 1$, we substitute x with $\ln(\alpha)/[B - B/(\ln(\alpha) + 2) - 1]$. Note that $B - \frac{B}{\ln(\alpha)+2} - 1 \geq \ln(\alpha)$ holds for our assumption of $B \geq \ln(\alpha) + 2$.

$$\begin{aligned} & B\left(1+\frac{\ln(\alpha)}{B-B/(\ln(\alpha)+2)-1}+\left(\frac{\ln(\alpha)}{B-B/(\ln(\alpha)+2)-1}\right)^2\right)-B+1 \\ = & \frac{B\ln(\alpha)}{B-B/(\ln(\alpha)+2)-1}+\frac{B\ln^2(\alpha)}{(B-B/(\ln(\alpha)+2)-1)^2}+1 \\ = & 1+\frac{\ln(\alpha)}{1-1/(\ln(\alpha)+2)-1/B}+\frac{\ln^2(\alpha)}{B[1-1/(\ln(\alpha)+2)-1/B]^2} \end{aligned}$$

Since $\alpha \geq e$ and $B \geq \ln(\alpha)+2$, we have $\ln(\alpha) \geq 1$ and $B \geq 3$. Therefore, the last part of the expression is bounded by:

$$\frac{\ln^2(\alpha)}{B[1-1/(\ln(\alpha)+2)-1/B]} \leq \frac{\ln^2(\alpha)}{B[1-1/3-1/3]^2} = \frac{9\ln^2(\alpha)}{B}$$

The rest of the expression is bounded as follows:

$$\begin{aligned} 1 \quad + \quad & \frac{\ln(\alpha)}{1-1/(\ln(\alpha)+2)-1/B} = \ln(\alpha)+1+\frac{\frac{\ln(\alpha)}{\ln(\alpha)+2}+\frac{\ln(\alpha)}{B}}{1-1/(\ln(\alpha)+2)-1/B} \\ = & \ln(\alpha)+1+\frac{1-\frac{2}{\ln(\alpha)+2}+\frac{\ln(\alpha)}{B}}{1-1/(\ln(\alpha)+2)-1/B} \\ = & \ln(\alpha)+2+\frac{\frac{\ln(\alpha)}{B}-\frac{1}{\ln(\alpha)+2}+\frac{1}{B}}{1-1/(\ln(\alpha)+2)-1/B} \\ \leq & \ln(\alpha)+2+\frac{3\ln(\alpha)}{B} \end{aligned}$$

The last inequality holds since $B \geq \ln(\alpha)+2$ and $\alpha \geq e$. Adding the parts of the analysis together, we have

$$B\alpha^{1/S}-B+1 \leq \ln(\alpha)+2+\frac{9\ln^2(\alpha)}{B}+\frac{3\ln(\alpha)}{B} = \ln(\alpha)+2+O\left(\frac{\ln^2(\alpha)}{B}\right)$$

This completes the proof for $\alpha \geq e$ and $B \geq \ln(\alpha)+2$. For $1 \leq \alpha \leq e$ we use a different analysis. We rely on the following lemma from [2], relating to a greedy policy, which accepts packets as long as the queue is not full, but might use a smaller queue:

Lemma 2.3.5 *[2] A greedy policy with a queue of size xB, for some $0 < x \leq 1$ accepts at least a x fraction of the number of packets that any other policy with a queue of size B accepts.*

The Smooth Selective Barrier Policy accepts packets greedily until the number of packets in the queue reaches the threshold of $\mathcal{F}(1)$. Inductively, its queue always holds

more packets than a greedy policy with a queue of size $\mathcal{F}(1)$, and therefore, by Lemma 2.3.5 the competitive ratio of accepted packets (ignoring packet values) is at least:

$$\frac{B}{\mathcal{F}(1)} = \frac{B}{\lceil \frac{B}{\ln(\alpha)+2} \rceil} \leq \ln(\alpha) + 2$$

At a worst case scenario, the Smooth Selective Barrier Policy accepts only packets of value 1, while the offline accepts only packet of value α. Therefore, the competitive ratio (considering packet values) is at most the ratio of the number of accepted packets times the ratio between the largest and smallest packet values, meaning that for $\alpha \leq e$,

$$\alpha(\ln(\alpha) + 2) \leq e(\ln(e) + 2) = 3e \approx 8.155,$$

which completes the proof. ∎

2.4 Improved Bounds for Low α

For a sufficiently large B such that $B = \omega(\ln^2 \alpha)$ the factor of $\ln^2(\alpha)/B$ is negligible. The main bottleneck in Theorem 2.3.4 is when $1 \leq \alpha \leq e$, and causes the competitive ratio to be 8.155 rather than approximately 3. If α is sufficiently small, setting $\mathcal{F}(v) = B$ for any $v \in [1, \alpha]$ may be better. This policy is equivalent to a greedy policy which accepts any packet, and its competitive ratio, as shown is [2], is as follows.

Theorem 2.4.1 *[2] The competitive ratio of the greedy policy is* α

Since $\alpha \leq \ln(\alpha) + 2$ for $1 \leq \alpha \leq e$ we have the following corollary:

Corollary 2.4.2 *A combined policy that runs the Smooth Selective Barrier Policy for* $\alpha \geq e$ *and the Greedy Policy for* $1 \leq \alpha < e$ *has a competitive ratio of at most* $\ln(\alpha) + 2 + O(\frac{\ln^2(\alpha)}{B})$

Solving numerically, a competitive ratio of α is better than $\ln(\alpha) + 2$ for any $1 \leq \alpha \leq 3.146$. However, in the following section we present better bounds by introducing an alternative policy for small values of α.

2.4.1 The Rounded Ratio Partition Policy

The *Ratio Partition* (RP) policy was presented in [10] for the two value case, i.e. where packet values can be only 1 (low value) or α (high value). The competitive ratio of this policy is $(2\alpha-1)/\alpha$, which is optimal for an online policy. The RP policy accepts any high value packet (as long as the queue is not full), and marks $\alpha/(\alpha-1)$ low value packets that are in the queue (marking is done from bottom to top). A low value packet is accepted if by filling the remaining space in the queue with high value packets marks the new low value packet. In the analysis, rejected high value packets that were accepted by the offline

match the marked low value packets, that were rejected by the offline. Rejected low value packets accepted by the offline match the remaining low value packets accepted by the online.

The analysis, however, assumed the queue size B is large enough in order to ignore rounding errors, due to the fact that when the online first rejects a low value packet, the analysis may assume that a fraction of the packet was indeed accepted. The cost of this difference is proportional to $1/B$, so a more accurate upper bound to the policy is $(2\alpha - 1)/\alpha + O(1/B)$

We convert the Ratio Partition Policy to a multiple value policy by translating the inputs to two values. The translation function $\mathcal{T}(\cdot)$ changes the value of the packet as follows:

$$v(\mathcal{T}(p)) = \begin{cases} 1 & : \quad 1 \leq v(p) < \sqrt{\alpha} \\ \sqrt{\alpha} & : \quad \sqrt{\alpha} \leq v(p) \leq \alpha \end{cases}$$

We define the converted policy for multiple values as follows:

Definition 2.4.3 *The* Rounded Ratio Partition Policy *executes a simulation of the Ratio Partition Policy with values* 1 *and* $\sqrt{\alpha}$*. Each event* `arrive(`p`)` *is translated to the event* `arrive(`$\mathcal{T}(p)$`)`*. Packet* p *is accepted if the Ratio Partition accepts* $\mathcal{T}(p)$*, and rejected otherwise.*

Theorem 2.4.4 *The competitive ratio of the Rounded Ratio Partition Policy is at most* $2\sqrt{\alpha} - 1 + O(\sqrt{\alpha}/B)$

Proof: The competitive ratio of the Ratio Partition is $\frac{2\alpha-1}{\alpha} + O(\frac{1}{B})$. Since the simulation uses values of 1 and $\sqrt{\alpha}$, we substitute α with $\sqrt{\alpha}$ and get a competitive ratio of $\frac{2\sqrt{\alpha}-1}{\sqrt{\alpha}} + O(\frac{1}{B})$ for the simulation. The Rounded Ratio Partition loses a factor of at most $\sqrt{\alpha}$ due to the translation of packet values. Therefore, the competitive ratio is upper bounded by $2\sqrt{\alpha} - 1 + O(\sqrt{\alpha}/B)$ ∎

If B is large enough such that $O\left(\frac{\sqrt{\alpha}}{B}\right)$ is negligible, and assuming that α is small, the Rounded Ratio Policy is preferable over the greedy policy and the Smooth Selective Barrier Policy. Combining policies, we derive the following corollary:

Corollary 2.4.5 *A combined policy that based on the value of* α*, runs either the Smooth Selective Barrier Policy or the Rounded Ratio Partition Policy, has a competitive ratio of*

$$\min\left\{2\sqrt{\alpha} - 1, \ln(\alpha) + 2\right\} + O\left(\frac{\sqrt{\alpha}}{B}\right)$$

Solving numerically, the competitive ratio is improved for $1 \leq \alpha \leq 5.558$, since in that range $2\sqrt{\alpha} - 1 \leq \ln(\alpha) + 2$.

2.5 Lower Bound

By altering the function $\mathcal{F}(\cdot)$ used by the Smooth Selective Barrier Policy we can define any online policy that takes into account only the value of the arriving packet and the number of packets in the queue in order to decide whether to accept or reject a packet, ignoring the values of the packets in the queue, or the history of the input sequence. We refer to such policies as *Static Threshold Policies*. In this section we prove a lower bound of asymptotically $\ln(\alpha) + 2$ for the competitive ratio of Static Threshold policies. Specifically, we prove the following theorem:

Theorem 2.5.1 *For any value $\epsilon > 0$, there exists a value β such that for any $\alpha \geq \beta$, no Static Threshold policy has a competitive ratio of less than $\ln(\alpha) + 2 - \epsilon$.*

To prove Theorem 2.5.1, we observe the performance of an arbitrary Static Threshold policy on two sets of $n+1$ different inputs each, where n will be determined later. We denote the first set as $Up(i)$, and the second set as $UpDown(i)$, where $i \in [0, n]$. First, we prove several properties that are common to all online policies, using the Up series. We then define the $UpDown$ series as extensions to Up series, and observe how these extensions can increase the competitive ratio for Static Threshold policies. Before defining the input series, we introduce the notations that will be used in our analysis:

Definition 2.5.2 *Let V^i and $\widehat{V}^i$ denote the total values of an arbitrary Static Threshold policy on inputs $Up(i)$ and $UpDown(i)$, respectively. Let U^i and $\widehat{U}^i$ denote the total values of the optimal offline on inputs $Up(i)$ and $UpDown(i)$, respectively. Let $\rho_i = U^i/V^i$ and $\widehat{\rho}_i = \widehat{U}^i/\widehat{V}^i$.*

For convenience, this section uses a different measurement of time. The standard used in the previous sections incremented the time counter by one for each event. In this section, we increment the time counter only at `send()` events.

$Up(0)$ is defined as a burst of B packets with value 1 that arrive at $t = 0$. $Up(i)$ is defined as the same packets from $Up(i-1)$ followed by B packets with value $\alpha^{i/n}$. The optimal offline response to $Up(i)$ is to accept only the last B packets, gaining a benefit of $U^i = B\alpha^{i/n}$. The online will accept $x_0 = \lfloor \mathcal{F}(1) \rfloor$ packets with value 1, $x_1 = \max\left\{0, \lfloor \mathcal{F}(\alpha^{1/n}) \rfloor - x_0\right\}$ packets with value $\alpha^{1/n}$, etc. gaining a total value of $V^i = \sum_{0 \leq j \leq i} x_j \alpha^{j/n}$. The competitive ratio of the online over the i-th input is $\rho_i = U^i/V^i$.

The Up series was used in [10] to prove a general lower bound of $\ln(\alpha) + 1$, for the competitive ratio of any online policy. Without loss of generality, we assume that $\sum_{0 \leq i \leq n} x_i = B$, otherwise we can increase x_0 appropriately and lower all the ρ_i. The following convenient property holds for the sequence:

Given a series of constraints $\rho_i \leq a_i$, we can search for a feasible solution $x_0, x_1, ..., x_n$ in a greedy manner: Select x_0 such that $\rho_0 = a_0$, then select x_1 such that $\rho_1 = a_1$, etc. If at the end of the process we have $\sum_i x_i \leq B$ then we have a feasible solution. We define the outcome of this process as the *greedy solution*, and claim the following:

Lemma 2.5.3 *The greedy solution to a set of inequlities $\rho_i \leq a_i$ under the constraint $\sum_i x_i \leq B$ is feasible if and only if some feasible solution exists.*

Proof: The set of inequlities that the greedy solution satisfies is equivalent the following linear set of inequalities

$$\sum_{0\le j\le i} x_j \alpha^{j/n} \ge \frac{B\alpha^{i/n}}{a_i}.$$

Since the non zero coefficients are increasing by the index, if we have a feasible solution where the i-th inequality is strict, there exists a small $\epsilon > 0$ such that we can decrease x_i by ϵ and increase x_{i+1} by ϵ and still remain with a feasible solution. A series of these steps reduces any feasible solution to the greedy solution, and therefore the greedy solution is feasible if and only if a feasible solution exists. ∎

We now define the *UpDown* series. $UpDown(0)$ is identical to $Up(0)$. The beginning of $UpDown(i)$ is similar to $Up(i)$, however the sequence continues after $t = 0$, with i bursts of packets. At time $t = x_i$, x_i packets of value $\alpha^{(i-1)/n}$ arrive. At time $t = x_i + x_{i-1}$, x_{i-1} packet of value $\alpha^{(i-2)/n}$ arrive. The sequence continues with packets of decreasing value, until at time $x_1 + x_2 + ... + x_i$, the last burst containing x_1 packets with value 1 arrives. Since the number of packets in each burst equals the number of time units that passed since the last burst, there has to be enough space in the queue to accept all the packets in the burst, regardless of the policy. However, for a Static Threshold policy, when a burst arrives, the number of packets in the queue is equal to the threshold for which the policy rejects packets with this value. Therefore, we have the following observation:

Observation 2.5.4 *The total value $\widehat{V}^i$ gained by a Static Threshold policy for input $UpDown(i)$ is V^i, the same as for input $Up(i)$. The total value of the optimal policy for input $UpDown(i)$ is $\widehat{U}^i = U^i + \sum_{1\le j\le i} x_j \alpha^{j/n}$.*

Unfortunately, the greedy solution search technique of Lemma 2.5.3 does not fit *UpDown* series. For example, for large n and small α, increasing x_0 and decreasing x_1 appropriately, may lower all the ρ_i. Therefore, we cannot search for feasible solutions in a greedy manner. Instead, we measure how bad is the *Down* part in the *UpDown* series by comparing $\widehat{\rho}_i$ to ρ_i.

The following lemma lower bounds the gain to the competitive ratio derived by *UpDown* inputs, compared to the competitive ratio over *Up* inputs:

Lemma 2.5.5 $\widehat{\rho}_i \ge \max\left\{\rho_i, \rho_i + \alpha^{-1/n} - \rho_i \alpha^{-(i+1)/n}\right\}$

Proof: Since $\widehat{V}^i = V^i$ and $\widehat{U}^i \ge U^i$, $\widehat{\rho}_i \ge \rho_i$ trivially holds. We now prove the second part of the maximum expression:

$$\begin{aligned}
\widehat{\rho}_i &= \frac{\widehat{U}^i}{\widehat{V}^i} = \frac{U^i + \sum_{1\le j\le i} x_j \alpha^{(j-1)/n}}{V^i} = \rho_i + \frac{\left(\sum_{0\le j\le i} x_j \alpha^{j/n} - x_0\right)\alpha^{-1/n}}{V^i} \\
&= \rho_i + \frac{\left(V^i - x_0\right)\alpha^{-1/n}}{V^i} = \rho_i + \alpha^{-1/n} - \frac{x_0\alpha^{-1/n}}{V^i} \\
&= \rho_i + \alpha^{-1/n} - \frac{x_0\alpha^{-1/n}\rho_i}{U^i} = \rho_i + \alpha^{-1/n} - \frac{x_0\alpha^{-1/n}\rho_i}{B\alpha^{i/n}} \\
&\ge \rho_i + \alpha^{-1/n} - \frac{\rho_i}{\alpha^{(i+1)/n}}
\end{aligned}$$

■

We now observe how Lemma 2.5.5 contributes to proving Theorem 2.5.1: Obviously, if $\rho_i \geq \ln(\alpha)+2$, then also $\widehat{\rho}_i \geq \ln(\alpha)+2$. Otherwise, we have from Lemma 2.5.5 that $\widehat{\rho}_i \geq \rho_i + \alpha^{-1/n} - \frac{\ln(\alpha)+2}{\alpha^{(i+1)/n}}$. Due to the general lower bound of $\ln(\alpha)+1$ (from [10]), for sufficiently large n there exists an index i such that $\rho_i \geq \ln(\alpha)+1-\frac{1}{2}\epsilon$, where $\epsilon > 0$ is arbitrary small. If $\rho_i < \ln(\alpha)+2$ we have to prove that $\widehat{\rho}_i \geq \rho_i + 1 - \frac{1}{2}\epsilon \geq \ln(\alpha)+2-\epsilon$. For $n \geq \frac{4}{\epsilon}\ln(\alpha) \geq -\ln(\alpha)/\ln(1-\frac{1}{4}\epsilon)$, we have $\alpha^{-1/n} \geq 1-\frac{1}{4}\epsilon$. To complete our proof we have to show that $(\ln(\alpha)+2)\alpha^{-(i+1)/n} \leq \frac{1}{4}\epsilon$ holds for sufficiently large α. Although this holds for $i \geq cn-1$, for some constant $c > 0$, it is not true in general for any i. For $i < cn-1$ we can only conclude that $\widehat{\rho}_i \geq \rho_i$.

The following lemma states that for any integer $m \geq 4$, even if we allow ρ_0, ρ_1, ...,$\rho_{n/m}$ to be slightly larger than $\ln(\alpha)+1$ (but less than $\ln(\alpha)+2$), the competitive ratio for the remaining inputs cannot be much lower than $\ln(\alpha)+1$.

Lemma 2.5.6 *Assume $\alpha > e^2$. Let $m \geq 4$, let $\delta = \frac{2}{m} + \frac{1}{\ln(\alpha)} < 1$ and let $n > m\ln(\alpha)(\ln(\alpha)+1)$, such that $n = km$ for some integer k. There is no feasible solution for $x_0, x_1, ..., x_n$ such that $\rho_i < \ln(\alpha)+2-\delta$ for $0 \leq i \leq n/m$ and $\rho_i < \ln(\alpha)+1-\delta$ for $n/m+1 \leq i \leq n$.*

Proof: We shall attempt to find a solution for $x_0, x_1, ..., x_n$ which satisfies the constraints with equalities, i.e. $\rho_i = \ln(\alpha)+2-\delta$ for $i \leq \frac{n}{m}$ and $\rho_i = \ln(\alpha)+1-\delta$ otherwise. According to Lemma 2.5.3, if such a solution exists, it can be found in a greedy manner.

To satisfy $\rho_0 = U_0/V_0 = \ln(\alpha)+2-\delta$, since $U_0 = B$ and $V_0 = x_0$, we have $x_0 = B(\ln(\alpha)+2-\delta)^{-1}$. For $i > 0$, we have:

$$\begin{aligned} \rho_i^{-1} &= \frac{V^i}{U^i} = \frac{V^{i-1}+x_i\alpha^{i/n}}{B\alpha^{i/n}} = \frac{V^{i-1}}{B\alpha^{i/n}} + \frac{x_i\alpha^{i/n}}{B\alpha^{i/n}} \\ &= \frac{V^{i-1}}{U^{i-1}\alpha^{1/n}} + \frac{x_i}{B} = \rho_{i-1}^{-1}\alpha^{-1/n} + \frac{x_i}{B} \end{aligned}$$

With the exception of $\rho_{n/m+1}$, we have $\rho_i = \rho_{i-1}$ for any $i > 0$. Therefore, we have that $x_i = B\rho_{i-1}^{-1}(1-\alpha^{-1/n})$. For $1 \leq i \leq n/m$, where $\rho_{i-1} = \ln(\alpha)+2-\delta$, we have

$$x_i = B\frac{1-\alpha^{-1/n}}{\ln(\alpha)+2-\delta}.$$

For $n/m+2 \leq i \leq n$, where $\rho_{i-1} = \ln(\alpha)+1-\delta$, we have

$$x_i = B\frac{1-\alpha^{-1/n}}{\ln(\alpha)+1-\delta}.$$

For $i = n/m+1$ we have

$$x_{\frac{n}{m}+1} = B\left(\frac{1}{\ln(\alpha)+1-\delta} - \frac{\alpha^{-1/n}}{\ln(\alpha)+2-\delta}\right) > B\frac{1-\alpha^{-1/n}}{\ln(\alpha)+1-\delta}.$$

We continue by testing the constraint $\sum_i x_i \leq B$:

$$\frac{\sum_{i=0}^{n} x_i}{B} \geq \frac{1}{\ln(\alpha)+2-\delta} + \frac{n}{m}\frac{1-\alpha^{-1/n}}{\ln(\alpha)+2-\delta} + \left(n-\frac{n}{m}\right)\frac{1-\alpha^{-1/n}}{\ln(\alpha)+1-\delta}$$

We note that $\alpha^{-1/n} = e^{-\ln(\alpha)/n}$ and that for any $x \in [0,1]$, $e^{-x} \leq 1 - x + x^2$. After applying these substitutions, we have:

$$\begin{aligned}
&\frac{\sum_{i=0}^{n} x_i}{B} \\
&\geq \frac{1}{\ln(\alpha)+2-\delta} + \frac{n}{m}\left(\frac{\frac{\ln(\alpha)}{n} - \frac{\ln^2(\alpha)}{n^2}}{\ln(\alpha)+2-\delta}\right) + \left(n-\frac{n}{m}\right)\left(\frac{\frac{\ln(\alpha)}{n} - \frac{\ln^2(\alpha)}{n^2}}{\ln(\alpha)+1-\delta}\right) \\
&= \frac{1+\frac{\ln(\alpha)}{m}}{\ln(\alpha)+2-\delta} + \frac{\frac{m-1}{m}\ln(\alpha)}{\ln(\alpha)+1-\delta} - \frac{\frac{\ln^2(\alpha)}{m}}{n(\ln(\alpha)+2-\delta)} - \frac{\frac{m-1}{m}\ln^2(\alpha)}{n(\ln(\alpha)+1-\delta)}
\end{aligned}$$

Since $\delta < 1$, we can bound the second half of the expression above:

$$\frac{\frac{\ln^2(\alpha)}{m}}{n(\ln(\alpha)+2-\delta)} + \frac{\frac{m-1}{m}\ln^2(\alpha)}{n(\ln(\alpha)+1-\delta)} \leq \frac{\frac{\ln^2(\alpha)}{m}}{n\ln(\alpha)} + \frac{\frac{m-1}{m}\ln^2(\alpha)}{n\ln(\alpha)} = \frac{\ln(\alpha)}{n}$$

As for the first part of the expression, we have:

$$\begin{aligned}
&\frac{1+\frac{\ln(\alpha)}{m}}{\ln(\alpha)+2-\delta} + \frac{\frac{m-1}{m}\ln(\alpha)}{\ln(\alpha)+1-\delta} \\
&= 1 + \frac{\delta}{\ln(\alpha)+1-\delta} - \frac{1+\frac{\ln(\alpha)}{m}}{(\ln(\alpha)+1-\delta)(\ln(\alpha)+2-\delta)}
\end{aligned}$$

Since $1 > \delta = 2/m + 1/\ln(\alpha) > 0$,

$$\begin{aligned}
&\frac{1+\frac{\ln(\alpha)}{m}}{\ln(\alpha)+2-\delta} + \frac{\frac{m-1}{m}\ln(\alpha)}{\ln(\alpha)+1-\delta} \\
&> 1 + \frac{\frac{2}{m}+\frac{1}{\ln(\alpha)}}{\ln(\alpha)+1} - \frac{1+\frac{\ln(\alpha)}{m}}{\ln(\alpha)(\ln(\alpha)+1)} = 1 + \frac{\ln(\alpha)}{m\ln(\alpha)(\ln(\alpha)+1)}
\end{aligned}$$

Combining everything together we have

$$\frac{\sum_{i=0}^{n} x_i}{B} > 1 + \frac{\ln(\alpha)}{m\ln(\alpha)(\ln(\alpha)+1)} - \frac{\ln(\alpha)}{n},$$

which is more than 1 if $n > m\ln(\alpha)(\ln(\alpha)+1)$, which proves that no feasible solution exists. ∎

For any $0 < \epsilon < 1$ we choose $m = \frac{5}{\epsilon}$ and $\beta = m^{4m}$. This ensures that for any $\alpha \geq \beta$ we have $\frac{\epsilon}{2} > \delta = \frac{2}{m} + \frac{1}{\ln(\alpha)}$ and $\frac{\epsilon}{4} > (\ln(\alpha)+2)\alpha^{-1/m}$. For any fixed α we set $n > m\ln(\alpha)(\ln(\alpha)+1)$, which ensures that $\alpha^{-1/n} > 1 - \frac{\epsilon}{4}$.

If there exists an i such that $\rho_i > \ln(\alpha)+2-\epsilon$, we are done. Otherwise, by Lemma 2.5.6, there exists an $i > \frac{n}{m}$ such that $\rho_i > \ln(\alpha) + 1 - \frac{\epsilon}{2}$, and by Lemma 2.5.5, we have $\widehat{\rho}_i > \rho_i + 1 - \frac{\epsilon}{4} - \frac{\epsilon}{4} \geq \ln(\alpha) + 2 - \epsilon$, which completes the proof of Theorem 2.5.1. ∎

Chapter 3

Preemptive Queue Management

This chapter concentrates on online management of a single preemptive queue. The focus of the chapter is on randomized algorithms for two packet classes. An extension to arbitrary packet values is given at the end of the chapter.

3.1 Model and Notation Adjustments

The addition of a preemption operation to the queue requires refinements to the queue model presented in Chapter 2. Additionally, the analysis techniques for preemptive queues differ from those used for non-preemptive queues and therefore require a new set of notations.

3.1.1 Preemptive FIFO Queues

Similarly to Chapter 2, the system uses a queue of size B to buffer the arriving packets. To ignore rounding residuals, we assume that B, although finite, is fairly large. Whenever a packet arrives, an online scheduler has to decide whether to accept the packet and place it in the queue, or to reject it and lose it.

The time measurement that will be used is the same as in Section 2.5, where sending events occur at integral times. As is Chapter 2, without loss of generality, we consider only work conserving policies, that always send a packet when allowed, unless the queue is empty.

In contrast to Chapter 2, the online scheduler is also allowed to preempt packets from the queue. When a packet is preempted it is lost, and all the packets that follow it in the queue advance forward one slot. The scheduler is allowed to use preemption along with accepting and sending packets, as if they were an atomic action. Therefore, when a packet arrives to a full queue, the scheduler is allowed to first preempt packets and only then accept the new packet. Similarly, when the scheduler sends a packet, it may send a packet that is not in the head of the queue by preempting all the packets that precede it.

3.1.2 Streams and Queue Contents

In non-preemptive queues, once a packet enters the queue, it is bound to be sent later, and therefore we may account its contribution to the total value gained. This is not the case with preemptive queues, and therefore we need to distinguish between packets that were already sent to packets still buffered in the queue.

Given an input stream σ and a policy A, we define the following notations:

Definition 3.1.1 *Let $Q_A^\sigma(t)$ denote the set of packets contained in the queue at time t, let $S_A^\sigma(t)$ denote the packet that is sent at time t, and let $V_A^\sigma = \sum_t v(S_A^\sigma(t))$ denote the benefit of the policy, which is the total value of the packets it sends.*

In some cases we would like to refer to the contents of the queue restricted to the packets that actually get sent, ignoring packets that will be preempted later.

Definition 3.1.2 *Let $SQ_A^\sigma(t)$ denote the set of packets that occupy the queue at time t and are sent by the policy.*

3.1.3 Randomization

When considering a policy that uses randomization, we recall that a random online policy is a distribution over deterministic online policies. A random online policy A is therefore c-competitive if for every σ its *expected* total value $E(V_A^\sigma)$ is at least $\frac{1}{c}V_{\mathrm{OPT}}^\sigma$.

3.2 Online Policies

In this section we present online policies for the case of two packet classes: Low priority packets with value 1 and high priority packets with value $\alpha > 1$. The following definition is used to separate between packets of different classes:

Definition 3.2.1 *Given a set of packets T, let $L(T)$ (resp. $H(T)$) denote the number of low (resp. high) value packets in T.*

3.2.1 Mark and Flush

Our randomized policy is a combination of two deterministic policies, *mark & flush* (MF) and *greedy-high* (GH). The MF policy was first presented in [59]. Having a competitive ratio of at most 1.304 for the worst case α, MF is nearly the best deterministic online policy for preemptive queues possible, when the input is restricted to two packet classes. Upon packet arrival, the MF policy greedily accepts it, resolving overflows by preempting the packet from the lowest class in the queue, which is closest to the queue's head. Initially, all packets are *unmarked*, however every new high value packet that is accepted marks the nearest currently unmarked r low value packets in front of it (r is a non-negative parameter of MF). In case the marking ratio r is non-integral, a high value packet may

mark a fraction of a low value packet, and a low value packet may accumulate markings from several high value packets.

Upon a send event, if the packet at the head of the queue is unmarked, then it is simply sent. Otherwise, all low value packets between the marked packet at the head of the queue and the high value packet that marked it are preempted. For this purpose, partially marked packets are treated as unmarked packets. We refer to the action of preempting marked packets as *flushing*, hence the name of the policy (note that all the flushed packets must be also marked). As a result, the packet at the head of the queue is a high value packet, and it is sent. Figure 3.2.1 illustrates the marking process of the policy.

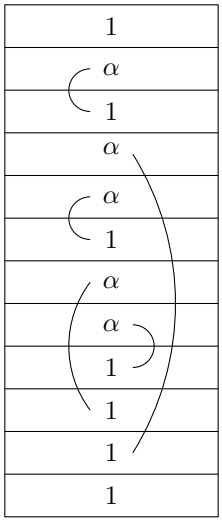

Figure 3.1: Marking example with $r = 1$.

Each arc connects a high value packet to the low value packet it marks. Assuming no more packets arrive, the next packet to be sent is a low value packet, since the packet a the head of the queue (the bottom of the figure) is unmarked. At the following time step, five low value packets will be flushed out to clear the way to the high value packets.

When the number of high value packets in the queue is at least $\frac{B}{r+1}$ then the next packet that MF sends is a high value packet, because if there are low value packets at the head of the queue they must be marked and therefore will be preempted at the next send event. This property of MF was proved in [59], and is formally presented as follows:

Lemma 3.2.2 *(Lemma 3.7 [59]) If* $H(Q^{\sigma}_{MF}(t)) \geq \frac{B}{r+1}$ *at some time* t*, then* $v(S^{\sigma}_{MF}(\lceil t \rceil)) = \alpha$.

3.2.2 Greedy High

The second deterministic policy that we use is GH. Policy GH ignores all low value packets and accepts only high value packets. Clearly, the following holds for GH:

Lemma 3.2.3 *For any input stream, the number of high value packets sent by GH is maximal.*

Proof: The correctness of this claim immediately follows from GH being a work conserving policy that accepts only high value packets and drops them only upon buffer overflows. ∎

When buffer overflows occur to GH, it does not matter which packets are accepted and which are dropped or preempted, since all high value packets are equivalent. However, to simplify the analysis, we assume without loss of generality that any high value packet sent by MF is also sent by GH. This assumption is formally stated as follows:

Observation 3.2.4 *For any input sequence, the high value packets sent by MF(r) are a subset of the high value packets sent by GH.*

Since MF also accepts high value packets greedily, and since GH rejects all low value packets, the following also holds:

Lemma 3.2.5 *For any input sequence σ and any time t, $|Q^{\sigma}_{MF}(t)| \geq |Q^{\sigma}_{GH}(t)|$.*

Proof: By proving a stronger property $H(Q^{\sigma}_{\mathrm{MF}}(t)) \geq H(Q^{\sigma}_{\mathrm{GH}}(t))$ the claim follows since $|Q^{\sigma}_{\mathrm{MF}}(t)| \geq H(Q^{\sigma}_{\mathrm{MF}}(t))$ while $|Q^{\sigma}_{\mathrm{GH}}(t)| = H(Q^{\sigma}_{\mathrm{GH}}(t))$. The proof is by induction on the sequence of packet arrivals and send events. At $t = 0$ both queues are empty and the claim holds. Assuming the inequality holds at some time $t \geq 0$, if the next event is an arrival of a low value packet then the number of high value packets in both buffers in unaffected. If a high value packet arrives then MF accepts it unless $H(Q^{\sigma}_{\mathrm{MF}}(t)) = B$. Either way the claim still holds. At send events, if $H(Q^{\sigma}_{\mathrm{GH}}(t)) = 0$ the claim trivially holds. Otherwise GH sends one high value packet while MF sends at most one high value packet (and may possibly flush low value packets, which does not affect the inequality), and again the claim holds. ∎

One may note that it is unlikely to apply GH in practice, since low value packets are never sent. An alternative policy that can replace GH in this chapter is MF$(B-1)$, which sends the same high value packets as GH, but also accepts low value packets as long as they don't interfere with high value packets. Since the analysis does not rely on any low value packets that MF$(B-1)$ may send, GH is preferred as it simplifies the analysis.

3.2.3 Randomized Mark and Flush

Policy *randomized-mark&flush* (RMF) is a randomized online policy, which requires only one random bit. Policy RMF(π, r) executes MF(r) with probability π (the parameters r and π will be determined later) and executes GH with probability $1-\pi$.

3.3 Augmented Offline Policy

The analysis of MF presented in [59] compared the performance of MF to an optimal offline schedule over an arbitrary input by degenerating the input into one of two simple

types of inputs. In order to analyze RMF we need to compare two different runs of RMF (MF(r) and GH) to the offline schedule. The proof that appears in [59] does not apply for RMF since the degeneration of the sequence may yield different input sequences for MF and for GH, which cannot be compared. Therefore, our analysis of RMF is based on an alternative analysis for MF.

Since there may be more than one optimal schedule for a given input sequence, we concentrate on a schedule that is similar to MF and GH, by augmenting the schedule of MF in an offline process. The augmentation consists of two phases. We use the following terms in the description of the augmentation phases:

Definition 3.3.1 *A packet p is* addable *to a schedule if p can be added to the schedule without dropping any other packet from the original schedule.*

Naturally, p is addable to a given schedule if and only if after the arrival of p at time t there is a time t_1 such that $SQ^{\sigma}_{A}(t_1) = \emptyset$ and $|SQ^{\sigma}_{A}(t_2)| < B$ for every $t \leq t_2 < t_1$, where A is a policy that outputs the schedule without p.

Definition 3.3.2 *A packet p is* replaceable *by a packet q if by removing p from the schedule it is possible to add q to the schedule without making any other changes.*

We note that if p arrived before q, and if q is not addable then p is replaceable by q if and only if the queue is never empty from the arrival of p and until the arrival of q.

The augmentation process consists of two phases:

Phase 1: For each high value packet q sent by GH but not by MF, find a low value packet p sent by MF such that p is replaceable by q and replace them.

Phase 2: For each marked low value packet p that is not sent, if p is addable then add p to the schedule.

We note that the decisions in phases 1 and 2 are adaptive, i.e. the decision whether packets are replaceable or addable is made according to the current status of the partially augmented sequence, and not with respect to the original sequence. Let OPT denote the augmented sequence. We now prove the optimality of OPT.

Theorem 3.3.3 *The total value of the augmented sequence OPT is maximal.*

Proof: After phase 1, OPT sends the same high value packets as GH, and therefore by Lemma 3.2.3 maximizes the number of high value packets it sends. If phase 2 was not restricted to marked packets, then clearly OPT would also maximize the total number of packets it sends, and therefore would be optimal.

It remains to prove that after phase 2 there are no unmarked low value packets that are addable. By the way of contradiction, let p be the first packet (by its arrival time) that is unmarked yet addable. Since p is unmarked, it surely wasn't flushed from the queue. Therefore p was either preempted or rejected. Packet p cannot be a packet that has been replaced during phase 1 since adding it back will surely overflow the queue. Let t_1 be the time when MF discards p. Let t_2 be the first time after t_1 where the $Q^{\sigma}_{\mathrm{MF}}(t_2) = \emptyset$.

If there exists a time t_3 such that $t_1 < t_3 < t_2$ where the queue is full with high value packets, then p cannot be addable to the original schedule, and each augmentation (replacement or addition of a marked packet) does change the fact that after t_1, the queue first refills with high value packets before it drains. Therefore p cannot be addable to OPT.

If the queue does not refill with high value packets before it drains, then surely in phase 1 there were no replacements of packets that arrived between t_1 and t_2. Packet p must have been preempted by MF as the oldest low value packet in a full queue. Therefore, p can be addable only if at least one of the packets that arrived after p had been flushed between t_1 and t_2, and not returned in phase 2. However, if such a packet exists, it must be an addable marked packet, which contradicts the definition of phase 2. ∎

3.4 Analysis of RMF

Given an input stream σ, let t_1 be a time before the last packet arrives where $Q^{\sigma}_{\mathrm{MF}}(t_1) = \emptyset$ (assuming such t_1 exists). By Lemma 3.2.5, $Q^{\sigma}_{\mathrm{GH}}(t_1) = \emptyset$. Let τ be the input stream of packets from σ that arrived before t_1. Similarly, let υ be the input stream of packets from σ that arrived after t_1.

Surely, $V^{\sigma}_{\mathrm{MF}} = V^{\tau}_{\mathrm{MF}} + V^{\upsilon}_{\mathrm{MF}}$ and $V^{\sigma}_{\mathrm{GH}} = V^{\tau}_{\mathrm{GH}} + V^{\upsilon}_{\mathrm{GH}}$. The optimal policy, however, may benefit from the division of the sequence, and therefore $V^{\sigma}_{\mathrm{OPT}} \leq V^{\tau}_{\mathrm{OPT}} + V^{\upsilon}_{\mathrm{OPT}}$.

The following holds for the ratio between the expected value of packets sent by RMF and OPT:

$$\begin{aligned}
&\frac{\pi \cdot V^{\sigma}_{\mathrm{MF}} + (1-\pi) \cdot V^{\sigma}_{\mathrm{GH}}}{V^{\sigma}_{\mathrm{OPT}}} \\
&\geq \frac{\pi \cdot (V^{\tau}_{\mathrm{MF}} + V^{\upsilon}_{\mathrm{MF}}) + (1-\pi) \cdot (V^{\tau}_{\mathrm{GH}} + V^{\upsilon}_{\mathrm{GH}})}{V^{\tau}_{\mathrm{OPT}} + V^{\upsilon}_{\mathrm{OPT}}} \\
&\geq \min\left\{ \frac{\pi \cdot V^{\tau}_{\mathrm{MF}} + (1-\pi) \cdot V^{\tau}_{\mathrm{GH}}}{V^{\tau}_{\mathrm{OPT}}}, \frac{\pi \cdot V^{\upsilon}_{\mathrm{MF}} + (1-\pi) \cdot V^{\upsilon}_{\mathrm{GH}}}{V^{\upsilon}_{\mathrm{OPT}}} \right\}
\end{aligned}$$

By the preceding inequality, we may concentrate on a subset of inputs:

Lemma 3.4.1 *For any input stream which ends at time t, if there is a time $\hat{t} < t$ such that $Q_{MF}(\hat{t}) = \emptyset$ then there is a shorter input, which has at least the same competitive ratio.*

By Lemma 3.4.1, we may assume that if MF is chosen, then the queue is never empty before the last packet is sent. We now divide the packets that MF has sent into sets MF_i, by defining a time interval for each set. A packet p belongs to the set MF_i if its arrival time occurred within the i-th interval and if MF has sent it (not necessarily during the same time interval). We slightly abuse the notation and use MF_i to denote also the i-th time interval and not only the subset of packets that arrived during this interval.

The first interval starts at $t = 0$ and the last interval ends with the arrival of the last packet. The arrival time t of a high value packet p that was sent by MF ends an

interval (and therefore the following packet starts a new interval) if after p was accepted, $H(Q_{\mathrm{MF}}(t)) = |SQ_{\mathrm{MF}}(t)| = B$, and if one of the following conditions is filled:

1. The queue is never refilled with high value packets after t, or
2. A low value packet is sent before the next refill with high value packets.

Except for possibly the last set, each set contains at least B high value packets, since the queue is full with high value packets when the interval ends, and the interval starts either with an empty queue (the first interval) or before the arrival of a low value packet that is sent later, which must be before the arrival of the B high value packets. Let n denote the number of intervals. We now assign the packets in GH and OPT to four sets GH_0, GH_1, OPT_0 and OPT_1.

First, any packet p such that $p \in \mathrm{OPT} \cap \mathrm{MF}_i$ is assigned to OPT_1 if $p \in \mathrm{MF}_n$ and to OPT_0 otherwise. Similarly, any packet p such that $p \in \mathrm{GH} \cap \mathrm{MF}_i$ is assigned to GH_1 if $p \in \mathrm{MF}_n$ and to GH_0 otherwise. Second, for each replacement of a low value packet p with a high value packet q during the augmentation that created OPT, we add q to the set OPT_0. By Lemma 3.2.3 we also have $q \in \mathrm{GH}$, and therefore we also assign q to GH_0. Finally, each low value packet in OPT that was added in the augmentation process is assigned to OPT_1.

We now bound the amount of the augmentations:

Lemma 3.4.2 *The number of high value packets added in the augmentations is at most* $\frac{(n-1)B}{r+1}$.

Proof: We *associate* each high value packet p that MF rejects with the interval MF_i that holds the packets that are in the queue when p is rejected (by the definition of intervals, when a high value packet is rejected, all the packets in the queue must belong to the same interval). We shall bound the number of associated high value packets in each of the first $n-1$ intervals by $\frac{B}{r+1}$. Since no high value packets are lost during the last interval, the claim will follow.

For any interval $i < n$, by Lemma 3.2.2, all low value packets that are replaced by high value packets associated with MF_i must have arrived before for last time t during MF_i such that $H(Q^{\sigma}_{\mathrm{MF}}(t)) \leq \frac{B}{r+1}$, since from then only high value packets are sent until the end of the interval. Therefore, before the augmentations are applied, $H(Q^{\sigma}_{\mathrm{MF}}(t)) = |SQ^{\sigma}_{\mathrm{MF}}(t)|$. Each replacement of packet p with packet q causes the packets that arrived between p and q to be sent one time unit earlier (and only them), and therefore reduces $|SQ^{\sigma}_{\mathrm{MF}}(t)|$ by 1. After $\frac{B}{r+1}$ replacements, $|SQ^{\sigma}_{\mathrm{MF}'}(t)| = 0$ (where MF' is the partially augmented schedule). Removing another low value packet will cause the queue to be idle in time $\lceil t \rceil$, and will not allow sending another high value packet during MF_i. Since each of the first $n-1$ intervals is associated with at most $\frac{B}{r+1}$ high value packets, the claim follows. ∎

Lemma 3.4.3 *The number of added low value packets is at most* $r \cdot H(MF_n)$.

Proof: Since the queue of MF is full when the last packet in MF_{n-1} arrives, and since all replacements end by adding a high value packet that fills the queue, when the last packet

in OPT_0 arrives, the queue is also full. Therefore, no low value packet that arrived before the last interval can be added without overflowing the queue.

Since addable low value packets arrived only during MF_n, and each high value packet in MF_n marked at most r low value packet, there are at most $r \cdot H(\mathrm{MF}_n)$ addable marked packets. Since by Theorem 3.3.3 we can complete the augmentations using only marked packets the claim holds. ∎

Combining these two Lemmas yields the following:

Theorem 3.4.4 *The ratio between the expected total value of RMF and the optimal total value is at least:*

$$\min\left\{\pi, \frac{\alpha}{\alpha+r}, \frac{\alpha(r+2-\pi)+\pi}{\alpha(r+2)}\right\}$$

Proof: The ratio between the total values is given by the following expression:

$$\begin{aligned}&\frac{\pi\left(\alpha\sum_{i=1}^{n} H(\mathrm{MF}_i)+\sum_{i=1}^{n} L(\mathrm{MF}_i)\right)}{\alpha H(\mathrm{OPT}_0\cup\mathrm{OPT}_1)+L(\mathrm{OPT}_0\cup\mathrm{OPT}_1)}\\ &+\frac{(1-\pi)\alpha\left(H(\mathrm{GH}_0\cup\mathrm{GH}_1)\right)}{\alpha H(\mathrm{OPT}_0\cup\mathrm{OPT}_1)+L(\mathrm{OPT}_0\cup\mathrm{OPT}_1)}\end{aligned} \tag{3.1}$$

Assuming that there is more than one interval, we can lower bound the expression in 3.1 by the minimum of the following expressions:

$$\begin{aligned}\mathcal{A} &= \frac{\pi\left(\alpha\sum_{i=1}^{n-1} H(\mathrm{MF}_i)+\sum_{i=1}^{n-1} L(\mathrm{MF}_i)\right)}{\alpha H(\mathrm{OPT}_0)+L(\mathrm{OPT}_0)}\\ &\quad+\frac{(1-\pi)\alpha H(\mathrm{GH}_0)}{\alpha H(\mathrm{OPT}_0)+L(\mathrm{OPT}_0)} \qquad (3.2)\\ \mathcal{B} &= \frac{\pi\left(\alpha H(\mathrm{MF}_n)+L(\mathrm{MF}_n)\right)+(1-\pi)\alpha H(\mathrm{GH}_1)}{\alpha H(\mathrm{OPT}_1)+L(\mathrm{OPT}_1)} \qquad (3.3)\end{aligned}$$

If there is only one interval, the sets that appear in (3.2) are empty and the original bound degenerates to (3.3).

In (3.2), let $(n-1)B+x$ denote the value of $\sum_{i=1}^{n-1} H(\mathrm{MF}_i)$, and let $(n-1)B+x+y$ denote the value of $H(\mathrm{OPT}_0)$ and of $H(\mathrm{GH}_0)$. Therefore, we have:

$$\begin{aligned}\mathcal{A} &= \frac{\pi\left(\alpha((n-1)B+x)+\sum_{i=1}^{n-1} L(\mathrm{MF}_i)\right)}{\alpha\left((n-1)B+x+y\right)+L(\mathrm{OPT}_0)}\\ &\quad+\frac{(1-\pi)\alpha\left((n-1)B+x+y\right)}{\alpha\left((n-1)B+x+y\right)+L(\mathrm{OPT}_0)}\\ &= \frac{\alpha\left((n-1)B+x\right)+\pi\sum_{i=1}^{n-1} L(\mathrm{MF}_i)+(1-\pi)\alpha y}{\alpha\left((n-1)B+x\right)+\alpha y+L(\mathrm{OPT}_0)}\end{aligned}$$

Restricting to $x \geq 0$, the bound is minimized when $x = 0$. Also, since y represents the number of replacements, we have $L(\mathrm{OPT}_0)+y = \sum_{i=1}^{n-1} L(\mathrm{MF}_i)$, and therefore:

$$\begin{aligned}
\mathcal{A} &\geq \frac{\alpha(n-1)B + \pi\left(y + L(\mathrm{OPT}_0)\right) + (1-\pi)\alpha y}{\alpha(n-1)B + \alpha y + L(\mathrm{OPT}_0)} \\
&\geq \min\left\{\frac{\alpha(n-1)B + \pi y + (1-\pi)\alpha y}{\alpha(n-1)B + \alpha y}, \frac{\pi \cdot L(\mathrm{OPT}_0)}{L(\mathrm{OPT}_0)}\right\} \\
&= \min\left\{\frac{\alpha(n-1)B + \pi y + (1-\pi)\alpha y}{\alpha(n-1)B + \alpha y}, \pi\right\}
\end{aligned}$$

The first part of the minimum is minimized when y is maximized. By Lemma 3.4.2 we have $y \leq \frac{(n-1)B}{r+1}$, and therefore:

$$\begin{aligned}
\mathcal{A} &\geq \min\left\{\frac{\alpha(n-1)B + \pi\frac{(n-1)B}{r+1} + (1-\pi)\alpha\frac{(n-1)B}{r+1}}{\alpha(n-1)B + \alpha\frac{(n-1)B}{r+1}}, \pi\right\} \\
&= \min\left\{\frac{\alpha + \frac{\pi}{r+1} + \frac{\alpha(1-\pi)}{r+1}}{\alpha + \frac{\alpha}{r+1}}, \pi\right\} \\
&= \min\left\{\frac{\alpha(r+2-\pi) + \pi}{\alpha(r+2)}, \pi\right\}
\end{aligned}$$

To complete the proof it remains to analyze (3.3). Since no high value packets are lost during MF_n, we have $H(\mathrm{MF}_n) = H(\mathrm{GH}_1) = H(\mathrm{OPT}_1)$. Let h denote this value, and let l denote $L(\mathrm{MF}_n)$. Since by Lemma 3.4.3 at most rh low value packets were addable, we have:

$$\begin{aligned}
\mathcal{B} &\geq \frac{\pi(\alpha h + l) + (1-\pi)\alpha h}{\alpha h + l + rh} = \frac{\alpha h + \pi l}{(\alpha + r)h + l} \\
&\geq \min\left\{\frac{\alpha h}{(\alpha + r)h}, \frac{\pi l}{l}\right\} = \min\left\{\frac{\alpha}{\alpha + r}, \pi\right\}
\end{aligned}$$

Since the ratio is at least $\min\{\mathcal{A}, \mathcal{B}\}$ the proof is completed. ∎

By optimizing the bound in Theorem 3.4.4, the ratio is minimized when all three parts of the expression are equal. This happens when $r = \sqrt{\alpha} - 1$ and $\pi = \frac{\alpha}{\alpha + \sqrt{\alpha} - 1}$. The following claim trivially follows:

Theorem 3.4.5 *The competitive ratio of RMF policy is at most* $1 + \alpha^{-1/2} - \alpha^{-1}$

The worst competitive ratio is achieved when $\alpha = 4$. The optimal values for the parameters r and π for this α are $r = 1$ and $\pi = 0.8$. The competitive ratio in this case is 1.25, which is lower than the general lower bound of approximately 1.281 for any deterministic policy.

3.5 Lower Bound

In this section we derive lower bounds for any deterministic or randomized online policy. Our lower bound construction assumes two packet values, but naturally holds for arbitrary packets values as well. For completeness, we first present the deterministic lower bound of [74], which is used as a basis for the randomized lower bound.

3.5.1 A Deterministic Lower Bound

Consider the following adversarial scenario: At $t = 0$, $B - 1$ low value packets followed by one high value packets arrive. As long as the next packet the online policy is about to send is a low value packet, another high value packet arrives after the low value packet is sent. Let zB be the number of low value packets the online policy sends before the next packet to be sent will be a high value packet. If $\frac{zB(\alpha+1)}{B-1+zB\alpha} \leq \frac{z+\alpha}{(z+1)\alpha}$, the scenario ends. Otherwise, a burst of B high value packets arrives.

In the first case, the online sends zB low value packets and at most zB high value packets, while the optimal schedule sends all $B-1$ low value packet as well as the zB high value packets. In the second case, the online sends zB low value packets and at most B high value packets, while the optimal policy drops all low value packets and sends $zB+B$ high value packets. Having the adversary choose a sufficiently large B and $\alpha = 4.01$, yields a lower bound of 1.281.

3.5.2 A Randomized Lower Bound

Theorem 3.5.1 *Any randomized online policy for two values has a competitive ratio of at least* 1.197*.*

Proof: Given an online policy that may use randomization, we construct the following adversarial scenario: At time $t = 0$, $B - 1$ low value packets followed by one high value packet arrive. After each send event, either no more packets arrive and the scenario ends, or another high value packet arrives. If zB high value packet have arrived (where $0 < z \leq 1$ will be determined later), either the scenario ends, or B high value packets arrive, and then the scenario ends.

We note that the decisions whether to end the scenario or to continue it can be based only on the probabilities used by the policy for coin tosses, and not by the results of the coin tosses. Without loss of generality, we assume that the online policy delays preemptions as much as possible, i.e. a low value packet is preempted either when a high value packet arrives to a full queue, or when sending a high value packet that is not in the head of the queue.

We denote by P_i the probability that after i high value packets have arrived, the online decides to send a high value packet *for the first time*. If the scenario ends after $k \leq zB$ high value packets have arrived, let V_{ON}^k be the expected total value of the online and V_{OPT}^k be the total value of the optimal offline. Without loss of generality, we assume that if the online has sent the first k low value packets by the end of the scenario, then it will

succeed sending all the remaining low and high value packets. The offline accepts all the packets, therefore the expected total values are:

$$\begin{aligned} V^k_{ON} &= k\alpha + (1 - \sum_{i=1}^{k} P_i)(B-1) + \sum_{i=1}^{k} P_i(i-1) \\ &= k\alpha + B - 1 - \sum_{i=1}^{k} P_i(B-i) \qquad (3.4) \\ V^k_{OPT} &= k\alpha + B - 1 \qquad (3.5) \end{aligned}$$

We denote by U_{ON} the expected total value of the online policy in case the scenario continues until the last B high value packets, and by U_{OPT} the optimal offline total value for this scenario. The online can always accept $zB + B$ packets, where the expected number of low value packets in this total depends on the probabilities P_i. An optimal offline schedule is to send all $zB + B$ high value packets. The expected total values are as follows:

$$\begin{aligned} U_{ON} &= B\alpha + zB + (\alpha - 1)\sum_{i=1}^{zB}(zB + 1 - i)P_i \qquad (3.6) \\ U_{OPT} &= \alpha B(1+z) \qquad (3.7) \end{aligned}$$

The ratio between the expected online total value and the offline total value is therefore bounded by the minimal of $\frac{U_{ON}}{U_{OPT}}$ and $\min_k\{\frac{V^k_{ON}}{V^k_{OPT}}\}$. Optimally, the online policy achieves the same ratio in all of the scenarios, otherwise the bound can be improved by slightly perturbing the probabilities P_i. Let ρ denote this bound. Since $\forall i\ \frac{V^i_{ON}}{V^i_{OPT}} = \rho$, we also have for every $2 \le i \le zB$:

$$\rho = \frac{V^i_{ON} - V^{i-1}_{ON}}{V^i_{OPT} - V^{i-1}_{OPT}} = \frac{\alpha - (B-i)P_i}{\alpha} = 1 - \frac{B-i}{\alpha}P_i,$$

and therefore, for $i \ge 2$ we have $P_i = \frac{\alpha(1-\rho)}{B-i}$. From $\frac{V^1_{ON}}{V^1_{OPT}} = \rho$ we can calculate P_1:

$$\rho = \frac{\alpha + B - 1 - P_1(B-1)}{\alpha + B - 1} = 1 - \frac{P_1(B-1)}{\alpha + B - 1},$$

and therefore, we have: $P_1 = (1-\rho)\frac{\alpha+B-1}{B-1} = \frac{\alpha(1-\rho)}{B-1} + (1-\rho)$.

The expressions for the probabilities P_i can help us compute the sum in (3.6):

$$\sum_{i=1}^{zB}(zB + 1 - i)P_i = \alpha(1-\rho)\sum_{i=1}^{zB}\frac{zB+1-i}{B-i} + (1-\rho)zB \qquad (3.8)$$

Concentrating on the first part of the sum in (3.8), we have

$$\begin{aligned}
&\alpha(1-\rho)\sum_{i=1}^{zB}\frac{zB+1-i}{B-i} \\
&= \alpha(1-\rho)\sum_{i=1}^{zB}\left(1-\frac{B-zB-1}{B-i}\right) \\
&= \alpha(1-\rho)\left(zB-(B(1-z)-1)\sum_{i=1}^{zB}\frac{1}{B-i}\right) \\
&= \alpha(1-\rho)\left(zB-(B(1-z)-1)\sum_{j=B-zB}^{B-1}\frac{1}{j}\right)
\end{aligned}$$

For large B, the expression $\sum_{j=B(1-z)}^{B-1}\frac{1}{j}$ is asymptotic to $\ln(B-1)-\ln B(1-z)=\ln\frac{B-1}{B(1-z)}$. Substituting in (3.6), we have that the lower bound is asymptotically:

$$\frac{B\alpha+zB+(\alpha-1)\left[\alpha(1-\rho)\chi+(1-\rho)zB\right]}{\alpha B(1+z)}, \tag{3.9}$$

where $\chi=\left(zB-(B(1-z)-1)\ln\frac{B-1}{B(1-z)}\right)$ We denote by ρ^* the ratio when B goes to infinity. Therefore, we have

$$\rho^*=\frac{\alpha+z+(\alpha-1)\left[\alpha(1-\rho^*)\chi^*+(1-\rho^*)z\right]}{\alpha(1+z)}, \tag{3.10}$$

where $\chi^*=z+(1-z)\ln(1-z)$. By rearranging (3.10) we get an expression for ρ^*:

$$\begin{aligned}
\rho^* &= \frac{\alpha+z+(\alpha-1)\alpha\chi^*+(\alpha-1)z}{\alpha+\alpha z+(\alpha-1)\alpha\chi^*+(\alpha-1)z} \\
&= 1-\frac{z(\alpha-1)}{\alpha(1+z)+(\alpha-1)\left(\alpha\chi^*+z\right)}
\end{aligned}$$

By having the adversary choose $z=0.66$ and $\alpha=3.38$, we get a lower bound of $\frac{1}{\rho^*}\approx 1.197$ ∎

3.6 Arbitrary Classes

In this section we analyze policies for input streams where packet values may be arbitrary. We concentrate on comparison based policies, where the best competitive ratio known is asymptotically 2 and is achieved by greedy admission [47]. The only policy that is known to beat the competitive ratio of 2 is the Preemptive Greedy policy [50], which with slight modifications is also studied in [20], and is not comparison based. The tightest analysis

of its competitive ratio sets it between $1 + \frac{1}{\sqrt{2}} \approx 1.707$ and $\sqrt{3} \approx 1.732$ [30]. Using randomization, we reach a near competitive ratio of 1.75 while restricting to comparison based policies.

3.6.1 Optimal Scheduling

Before introducing online FIFO policies, we first observe that an optimal schedule can be constructed as an online process, by waving the FIFO constraint for sending the packets. We define policy ONOPT (*online optimal*) as an online policy that accepts packets greedily, yet whenever it can send a packet it sends the most valuable packet in the queue, ignoring the FIFO constraint. The following lemma proves the optimality of ONOPT:

Lemma 3.6.1 *For any input sequence σ and any schedule A, $V^{\sigma}_{ONOPT} \geq V^{\sigma}_{A}$*

Proof: Let $N^{\sigma}_{A}(v)$ denote the total number of packet of value v or larger from sequence σ that schedule A sends. Since packets with lower values have no interference on ONOPT's decisions on accepting or sending packets with higher values, then ONOPT's greedy behavior infers that $N^{\sigma}_{\mathrm{ONOPT}}(v) \geq N^{\sigma}_{A}(v)$. Therefore, we have

$$V^{\sigma}_{\mathrm{ONOPT}} = \int_{0}^{\infty} N^{\sigma}_{\mathrm{ONOPT}}(v)dv \geq \int_{0}^{\infty} N^{\sigma}_{A}(v)dv = V^{\sigma}_{A} \tag{3.11}$$

■

An optimal FIFO schedule, equivalent to ONOPT, can be constructed by reordering the packets that ONOPT sends according to their arrival time. We denote by OPT this optimal offline FIFO schedule.

3.6.2 Online Policies

We present the randomized online policy RHG (*randomized half greedy*), which uses a single random bit to select between one of two deterministic algorithms. The first deterministic algorithm is GREEDY, which accepts packets greedily, preempting the less valuable packet in the queue if necessary (breaking ties arbitrary) and sends packets by FIFO order. By [47], GREEDY has a competitive ratio of asymptotically 2.

Algorithm HG (*half-greedy*) is an online deterministic policy that accepts packets greedily, just like GREEDY. It also maintains a simulation of ONOPT, and marks the packets that ONOPT has already sent and still remain in its own queue. Let $M(t)$ denote the number of marked packets in HG's queue. Whenever HG has to send a packet, it first updates the simulation of ONOPT and then recalculates $M(t)$. If $M(t) \leq \frac{B}{2}$, HG simply sends the packet at the head of the queue. Otherwise, HG sends the marked packet that is nearest to the head of the queue, by preempting all the packets that block its way to the head of the queue. We denote the operation of forwarding a packet in the queue by preempting packets that precede it as *flushing*.

The idea behind policy HG is a compromise between the policies GREEDY and ONOPT. This policy, however, may not perform very well by itself, as it might flush

half of the queue in order to promote packets which are only slightly more valuable. We therefore use the following combination between HG and GREEDY: Algorithm RHG executes GREEDY with probability $\frac{5}{7}$ and executes HG with probability $\frac{2}{7}$.

We note that policy RHG is a comparison based algorithm, since both GREEDY and HG never address the value of the packets, only their relative order. An important property of comparison based algorithms for switching networks presented in [19] is that the competitive ratio of any comparison based algorithm can be derived by analyzing its performance over input streams constructed only from packets with values of either 0 or 1. Therefore, by [19], we have the following observation:

Observation 3.6.2 *The competitive ratio of policy RHG is at most c if and only if the competitive ratio of RHG limited to inputs streams of packets with values of* 0 *and* 1 *is at most c.*

3.7 Analysis of Policy RHG

In the analysis, the term *preemption* refers only to packets preempted not during a flush. Therefore, when a packet arrives to a full queue, we may preempt a packet with a lower value, in contrast to flushing packets from the queue in order to send a marked packet. Additionally, when comparing two packet values, we assume that there is a consistent way to break ties, which all policies agree upon. Therefore, in certain places in our analysis we ignore the possibility of equal packet values.

We first observe several properties of GREEDY and HG that we will use later in the analysis. Let $Q_G^\sigma(t)$ and $Q_H^\sigma(t)$ denote the queues of GREEDY and HG (respectively) at time t, for some given input stream σ. The following lemmata present several properties of these policies.

Lemma 3.7.1 *For any time t and for any schedule A, the following holds:*

$$|Q_G^\sigma(t)| \geq |Q_A^\sigma(t)|$$

Proof: The proof follows the same ideas of Lemma 3.2.5. Comparing the queue of the greedy schedule to an alternative schedule. If the queue of the alternative grows then so will the greedy queue, unless it is already full. Similarly, when the greedy queue sends a packet then the alternative also drains its queue, unless it is already empty. ∎

Lemma 3.7.2 *If GREEDY rejects or preempts a packet with value v then in the next B time units GREEDY sends B packets with values of at least v.*

Proof: If GREEDY rejects or preempts a packet at time t then obviously GREEDY's queue contains B packets with value of at least v. The claim trivially holds if all of these packets are sent. Otherwise, when one of these packets is preempted, the queue must be again full with B packets of value v or larger. Either way, no packet with value less than v will be sent before B time units have passed, hence the claim holds. ∎

Lemma 3.7.3 *If GREEDY accepts a packet p, then so does HG, and if GREEDY sends p, then HG either sends p or flushes p.*

Proof: The lemma follows from the following stronger claim: If HG rejects a packet, then so does GREEDY. If at some time t, HG preempts a packet then either GREEDY also preempts the packet at the same time, or GREEDY has already rejected or preempted it earlier.

By way of contradiction let t be the time just before HG rejects or preempts a packet p which violates the claim for the first time. Since p is not flushed, it must be that $|Q_H^\sigma(t)| = B$, and therefore, by Lemma 3.7.1 $|Q_G^\sigma(t)| = B$. If packet p is rejected then all B packets in HG's queue have at least the value of $v(p)$. If p is preempted then the packet that entered HG's queue instead of p as well as the other $B-1$ packets in the queue have at least value $v(p)$. Either way, $v(p)$ is the lowest value among all $B+1$ relevant packets.

Since greedy chooses to hold p, it must reject or preempt a different packet q, such that $v(q) \leq v(p)$. It is impossible for q to be the packet that just arrived, since HG chooses to accept it instead of p, and GREEDY makes an opposite decision. Therefore q must already be in the GREEDY's queue. However, q cannot be in HG's queue, otherwise HG would have preempted q instead of p. Therefore HG has already sent or flushed q (if q has been rejected or preempted this would be an earlier violation of the claim).

Since HG sent q or flushed it, all the B packets in $Q_H^\sigma(t)$ must have arrived after q has arrived. Since $q \in Q_G^\sigma(t)$, at most $B-1$ of these packets can be in greedy's queue, and therefore GREEDY rejected or preempted at least one of these packets, while q was in its queue, although $v(q)$ is lower. This contradicts GREEDY's definition, therefore such a packet q cannot exist. ∎

Lemma 3.7.4 *If $p \in Q_H^\sigma(t)$ is a marked packet and if $q \in Q_H^\sigma(t)$ is an unmarked packet that arrived before packet p then $v(p) \geq v(q)$.*

Proof: By way of contradiction, assume $v(p) < v(q)$. Let $t_1 \leq t$ be the time that p was marked. This means that at t_1, p was the most expensive packet in ONOPT's queue. Since q has a higher value, it cannot be in ONOPT's queue at t_1, and since it is unmarked, ONOPT must have rejected or preempted it before t_1, at some time t_0. Therefore, right after t_0 ONOPT's queue must have been full with B packets which are worth more than $v(q)$. Since at t_1 p is the most expensive packet, ONOPT must have sent at least B packets from t_0 to t_1. Since q arrived before t_0, it cannot remain in HG's queue until t, which is more than B time units later. ∎

Lemma 3.7.5 *if just before time t HG flushes a packet p then in the next $|Q_H^\sigma(t)|$ time units HG sends at least $\frac{B}{2}$ packets with a value of at least $v(p)$.*

Proof: By Lemma 3.7.4 the marked packets in the queue are worth at least as p, and since a flush has just occurred, there must be $\frac{B}{2}$ marked packets. If HG sends all of those packets then the claim follows trivially. Otherwise some of these marked packet must have been preempted (since marked packets cannot be flushed). After the last preemption (and reception of an alternative packet) of these marked packets, the queue holds B packet with

at least value $v(p)$. Even if more flushes occur, after each flush the queue still holds $\frac{B}{2}$ packets with larger values than the flushed packets. Therefore, in the next $\frac{B}{2}$ HG sends packets which are worth at least $v(p)$, hence the claim follows. ∎

We now refer to the fact that policy RHG is comparison based. Therefore, as noted in Observation 3.6.2, in order to analyze policy RHG, it is sufficient to observe sequences of two packet values, 0 and 1. However, when comparing two packets with the same value (either both are 0 or both are 1), the comparison may break the tie either way. Therefore, an analysis of RHG based on 0/1 sequences must consider all possible consistent tie breaks. For the rest of the analysis we assume that the input stream σ is a 0/1-sequence. We refer to any packet with value 0 (1) as a 0-packet (1-packet).

In our analysis we compare the total value of GREEDY and HG to the total value of the optimal schedule, OPT. Obviously, since only 1-packets contribute to the total value, it is sufficient to count the number of 1-packets sent by each schedule, and then compare between them.

Without loss of generality, we may assume that the queue of GREEDY is idle for the first time only after the input ends. By Lemma 3.7.1 if GREEDY's queue is empty then the queues of ONOPT and HG are also empty. Both GREEDY and ONOPT are memoryless policies which don't depend on packets that are not present in their queues. The memory of HG consists only on the present status of ONOPT, and therefore it is invariant to packets that are in neither ONOPT's queue nor in HG's queue. Therefore, we can divide the input into subsequences according to the idle points of GREEDY's queue, and analyze each subsequence separately. The exact proof is identical to the proof of Lemma 3.4.1, with policies GREEDY and HG replacing policies MF and GH.

In our comparison, we divide the input streams into intervals and then compare subsets of the schedules of GREEDY, HG and OPT that arrived in the same interval. We prove that for each interval, the weighted average between the 1-packets in GREEDY's set and in HG's set is at least a certain fraction of the number of 1-packets in the matching set of OPT.

We define a *time interval* by declaring its starting and ending time. A packet belongs to the i-th set of a schedule if it was sent by the scheduler, and if its arrival time is within the time bounds of the i-th interval (the packet's sending time may be later). The bounds of the time intervals are defined by the structure of the GREEDY schedule: For each maximal set of B or more 1-packets sent consecutively by GREEDY, we define a separate time interval. If GREEDY's schedule continues with a 0-packet, then the arrival time of the last 1-packet that arrived before this 0-packet ends the current interval, and the arrival time of the following packet starts the next time interval (we consider all packets when determining time interval bounds, not only the packets in GREEDY's schedule). The starting time of the first interval is the arrival time of the first packet in the input sequence, and the ending time of the last interval is the arrival time of the last packet in the input sequence.

We match the i-th GREEDY set to the i-th OPT set and to the i-th HG set. Let G_i, O_i and H_i denote the number of 1-packets in the i-th sets of GREEDY, OPT and HG, respectively. Let $R_i = \frac{5}{7}G_i + \frac{2}{7}H_i$. By lower bounding $\frac{R_i}{O_i}$ for each i we will bound the

competitive ratio $\rho(\sigma)$, since $\frac{1}{\rho(\sigma)} \geq \min_i \left\{ \frac{R_i}{O_i} \right\}$. We observe 3 cases:

Case I: $G_i \geq O_i$
In this case we ignore the packets from HG, and therefore we have:

$$\frac{R_i}{O_i} = \frac{\frac{5}{7}G_i + \frac{2}{7}H_i}{O_i} \geq \frac{5}{7}\frac{G_i}{O_i} \geq \frac{5}{7} \tag{3.12}$$

Case II: $G_i + \frac{B}{2} \geq O_i$
Since GREEDY has scheduled less 1-packets than OPT, GREEDY has surely rejected or preempted at least one 1-packet that arrived during this interval, and therefore there are at least B consecutive 1-packets in GREEDY's interval (Lemma 3.7.2).

If HG does not flush any 1-packets then clearly $H_i \geq B$. Otherwise, by Lemma 3.7.5, HG sends at least $\frac{B}{2}$ 1-packets that arrived in this interval. Therefore, we have:

$$\begin{aligned} \frac{R_i}{O_i} &= \frac{\frac{5}{7}G_i + \frac{2}{7}H_i}{O_i} \geq \frac{\frac{5}{7}(B + (G_i - B)) + \frac{2}{7}\frac{B}{2}}{\frac{3B}{2} + (O_i - \frac{3B}{2})} \\ &\geq \frac{\frac{5}{7}B + \frac{2}{7}\frac{B}{2} + \frac{5}{7}(G_i - B)}{\frac{3B}{2} + (G_i - B)} \geq \frac{4}{7} \end{aligned} \tag{3.13}$$

Case III: $G_i + \frac{B}{2} < O_i$
We first lower bound G_i: By Lemma 3.7.2, since GREEDY rejected or preempted a 1-packet it sends at least B 1-packets starting from the first rejection, and that arrived within the interval. Let t_1 denote the rejection or the preemption of the first 1-packet in this interval.

We now upper bound the number of extra 1-packets that OPT accepted, compared to GREEDY: If we observe the send events, then GREEDY sends only 1 packets after t_1, and since $|Q_G^\sigma(t_1)| = B$, OPT cannot send more 1-packets than GREEDY after t_1. Therefore, OPT's extra 1-packets are sent before t_1. Since GREEDY does not drop or preempt any 1-packets before t_1, OPT must have sent packets that were still in GREEDY's queue until t_1. Since each packet can be delayed in GREEDY's queue at most $B-1$ time units more than OPT's queue, the number of extra 1-packets that OPT accepts during the i-th interval is at most $B-1$.

We now prove a lower bound on H_i: Let x denote the number of extra 1-packets OPT has sent and let y denote the number of packets that OPT has sent in the $B-1$ time units before t_1, while GREEDY has kept these packets in its buffer at least until t_1. Let y_1 denote the number of 1-packets in y. We note that y may include 0-packets as well as replace 1-packets with "larger" 1-packets, and therefore the inequalities $B - 1 \geq y \geq y_1 \geq x \geq \frac{B}{2}$ may be strict.

By the time OPT has sent $\frac{B}{2}$ of these y_1 packets, ONOPT must have also done so, otherwise ONOPT has not sent packets greedily. Let t_0 denote the time where ONOPT has sent $\frac{B}{2}$ of the y_1 packets. Since GREEDY delays OPT's y_1 packets until t_1, GREEDY sends different packets until t_1, while these y_1 packets remain in GREEDY's queue. Since

HG also accepted these packets (Lemma 3.7.3), then by t_0 (or earlier) HG must start sending ONOPT's y_1 packets (otherwise $M(t_0) > \frac{B}{2}$), so by t_1 HG manages to send at least $y_1 - \frac{B}{2}$ 1-packets.

Just before GREEDY starts to preempt or reject 1-packets at t_1, HG has already sent $y_1 - \frac{B}{2}$ 1-packets that are still in $Q_G^\sigma(t_1)$. Since OPT has sent y_1 1-packets, $Q_H(t_1)$ can contain at most $\frac{3B}{2} - y_1$ 1-packets, where at most $B - y_1$ of them are unmarked, and might be flushed later. Note that since GREEDY rejects and preempts at least x 1-packets after t_1, HG manages to avoid at least $y_1 - \frac{B}{2}$ of these rejections and preemptions. Therefore, along with the (at least) B 1-packets that GREEDY sends, by Lemma 3.7.3, the total number of 1-packets that HG accepts and later sends or flushes is at least $B + y_1 - \frac{B}{2} = y_1 + \frac{B}{2}$.

If after t_1 there are no flushes of packets that arrived after t_1, then after possibly flushing the $B - y_1$ unmarked packets that are still in $Q_H^\sigma(t_1)$, we have $H_i \geq (y_1 + \frac{B}{2}) - (B - y_1) = 2y_1 - \frac{B}{2}$. In this case, we have

$$\begin{aligned}
\frac{R_i}{O_i} &= \frac{\frac{5}{7}G_i + \frac{2}{7}H_i}{O_i} \geq \frac{\frac{5}{7}(B + (G_i - B)) + \frac{2}{7}(2y_1 - \frac{B}{2})}{x + B + (O_i - B - x)} \\
&\geq \frac{5(B + (G_i - B)) + 2(2x - \frac{B}{2})}{7(x + B + (G_i - B))} \\
&= \frac{4B + 4x + 5(G_i - B)}{7B + 7x + 7(G_i - B)} \geq \frac{4}{7}
\end{aligned} \tag{3.14}$$

Otherwise, after t_1 there are flushes of packets that arrived after t_1. These flushes cannot occur before $t_2 \geq t_1 + \frac{B}{2}$.

Since ONOPT also sends 1-packets between t_1 and t_2, HG must send until t_2 at least y_1 1-packets (at least $y_1 - \frac{B}{2}$ until t_1 and the rest until t_2) to maintain a low number of marked packets. By Lemma 3.7.5, HG sends at least another $\frac{B}{2}$ 1-packets after t_2.

Therefore, we have:

$$\begin{aligned}
\frac{R_i}{O_i} &= \frac{\frac{5}{7}G_i + \frac{2}{7}H_i}{O_i} \geq \frac{\frac{5}{7}(B + (G_i - B)) + \frac{2}{7}(y_1 + \frac{B}{2})}{x + B + (O_i - B - x)} \\
&\geq \frac{5(B + (G_i - B)) + 2(x + \frac{B}{2})}{7(x + B + (G_i - B))} \\
&= \frac{6B + 2x + 5(G_i - B)}{7B + 7x + 7(G_i - B)} \\
&\geq \frac{8B + 5(G_i - B)}{14B + 7(G_i - B)} \geq \frac{4}{7}
\end{aligned} \tag{3.15}$$

Since in each interval of OPT we match 1-packets from a weighted average of HG and GREEDY in a proportion of at least $\frac{4}{7}$, the following theorem follows:

Theorem 3.7.6 *The competitive ratio of RMF is at most* $\frac{7}{4}$.

Chapter 4

Auctions with Budget Constraints

The auction with budget constraints problem is an assignment problem where the bidders' valuations are additive until a certain limit is reached. This chapter concentrates on offline non-strategic algorithms for maximizing the total valuation in such settings, with most of the focus given to approximation algorithms.

4.1 Model and Notations

An auction with budget constraints consists of n bidders and k items. Let b_{ij} denote the bid of bidder i for item j, which is the bidder's valuation of this item alone. Each bidder i also has a budget constraint denoted by d_i.

Let $z_{ij} \in \{0,1\}$ denote the allocation of the items, where $z_{ij} = 1$ if bidder i receives item j and $z_{ij} = 0$ otherwise. Given an allocation, the valuation of each bidder is is $v_i = \min\{d_i, \sum_{j=1}^{k} z_{ij} b_{ij}\}$. The objective of allocation with budget constraints is maximizing the total valuation of all bidders.

This allocation problem can be presented formally as the following Integer Programming (IP) problem:

$$\begin{array}{lll} \max \sum_{i=1}^{n} v_i & \text{s.t.} & /* \text{ maximizing valuations } */ \\ v_i \leq \sum_{j=1}^{k} z_{ij} b_{ij} & 1 \leq i \leq n & /* \text{ additive valuations } */ \\ v_i \leq d_i & 1 \leq i \leq n & /* \text{ budget constraints } */ \\ \sum_{i=1}^{n} z_{ij} \leq 1 & 1 \leq j \leq k & /* \text{ one copy of each item } */ \\ z_{ij} \in \{0,1\} & 1 \leq j \leq k, 1 \leq i \leq n & /* \text{ integral allocation } */ \end{array} \tag{4.1}$$

Without loss of generality, we assume that the budget constraint is consistent with the bids, i.e. $d_i \geq b_{ij}$, $\forall i, j$, $1 \leq i \leq n$, $1 \leq j \leq k$, and that the budget constraint is effective, i.e. $d_i \leq \sum_{j=1}^{k} b_{ij}$, $\forall i$, $1 \leq i \leq n$, since for larger budgets we can always replace d_i with $\sum_{j=1}^{k} b_{ij}$. In several places we will normalize the bids and budgets such that they will all be integral.

4.2 Exact Solutions

Lehmann, Lehmann and Nisan [55] have analyzed auctions where the bidders have submodular valuations, meaning that the marginal utility that each bidder gains for any item decreases as the set of items already allocated to this bidder increases. They prove that finding an optimal allocation is NP-hard even if there are only two bidders with additive valuations up to budget constraints. For the special case considered in this chapter, the following theorem strengthens this result, proving that the problem is hard even for identical bidders.

Theorem 4.2.1 *Determining whether there exists an allocation for an auction with budget constraints such that the total valuation is at least M is NP-hard even for two bidders with identical bids and budget constraints.*

Proof: The proof is by reduction from the $PARTITION$ [45] problem. Given a set of integers $\{a_1, a_2, ..., a_k\}$ as an instance of $PARTITION$, the decision problem is to determine whether the input can be partitioned into two subsets with equal sums. We assume that $A = \sum_i a_i$ is even, otherwise this is trivially a NO instance. The reduction to budget constrained auctions is as follows:

We construct k items. The bid of every agent for the i-th item is a_i, and the budget constraint of each bidder is $A/2$. In case we wish to reduce to an instance with more than two bidders, for each bidder beyond the second we add another item with a bid of $A/2$. YES instances are mapped into auctions with an optimal allocation of $A \cdot \frac{n}{2}$, by partitioning the first k items between two of the bidders, and allocating one of the large additional items to each of the remaining bidders. NO instances are mapped into auctions with an optimal allocation of less than $A \cdot \frac{n}{2}$. Setting $M = A \cdot \frac{n}{2}$ completes the reduction. ∎

Using dynamic programming [27], an exact solution can be found in a time complexity that is exponential in the number of items. In the i-th stage of the dynamic programming, optimal allocations of any subset of the k items are computed over the first i bidders, by using the optimal allocations from the previous stage. This process yields the following:

Theorem 4.2.2 *An exact optimal allocation for an auction with budget constraints can be found in time complexity of $O(n4^k)$.*

Proof: The dynamic programming solves the following recursion, which determines the value of optimally allocating a set S of items to the first i bidders:

$$F(i, S) = \max_{T \subseteq S} \left\{ F(i-1, S \setminus T) + \min\{d_i, \sum_{j \in T} b_{ij}\} \right\}, \tag{4.2}$$

where $0 \leq i \leq n$ and $S \subseteq \{1, 2, \ldots, k\}$. Initially, $F(0, S) = 0$, and the other values of F are calculated by recursive references. The dynamic programming table includes $\Theta(n2^k)$ entries, and the calculation of each entry requires accessing at most $O(2^k)$ other entries, which yields a search process of $O(n4^k)$ accesses to the table.

To avoid additional overhead, calculating all possible values of $\sum_{j \in T} b_{ij}$ should be done in advance. The optimal value of the auction is stored in entry $F(n, \{1, 2, \ldots, k\})$. Optionally, we can also store in each entry a pointer to the set T that yielded the value stored in it, and by this also construct the optimal allocation itself. ∎

For $n >> k$, we can slightly decrease the time complexity, as follows:

Theorem 4.2.3 *An exact optimal allocation for an auction with budget constraints can be found in time complexity of $O(k^2 4^k + nk)$.*

Proof: We first observe that in an optimal allocation, each item is allocated to one of the highest k bidders. If an item is not allocated to one of the highest bidders, then since there are only $k-1$ remaining items, at least one of the k highest bidders is not allocated anything. Therefore, the allocation can be improved by passing that item to this bidder.

Therefore, if $n > k$, only the k highest bidders should be considered for each item, and at most k^2 bidders in total. Finding the k highest bidders for an item can be done in a time complexity of $O(n)$, and in $O(nk)$ in total. We now solve the dynamic programming in Theorem 4.2.2 with the subset of at most k^2 bidders. ∎

Theorem 4.2.3 proposes a reasonable algorithm for calculating the optimal allocation when k is small relatively to n, for example $k = O(\ln n)$. When the number of bidders n is a constant, then a pseudo-polynomial algorithm based on dynamic programming exists.

Theorem 4.2.4 *If the number of bidders is constant then there is a pseudo-polynomial algorithm for the auction with budget constraints problem.*

Proof: Without loss of generality we assume that all b_{ij} and d_i are integers. Let $D = \max_j d_j$ denote the largest budget constraint. Build an $n+1$ dimensional table with kD^n entries. Each entry $(i, d_1, d_2, \ldots, d_n)$ holds the allocation of a subset of the first i items, such that the valuation of bidder j is d_j if such an allocation exists. To calculate the value of entry $(i, d_1, d_2, \ldots, d_n)$ we need n references to the entries $(i-1, d_1, \ldots, d_{j-1}, d_j - b_{ij}, d_{j+1}, \ldots, d_n)$ and therefore the entire table can be calculated within nkD^n steps. The optimal allocation is the non-empty entry $(k, d_1, d_2, \ldots, d_n)$ such that $\sum_i d_i$ is maximal. If n is constant then the time complexity is $O(kD^n)$, which is polynomial in k and D. ∎

If the number of bidders is constant and all the bids b_{ij} are drawn from a set of constant size, then the problem is no longer NP-Complete, and can be solved in polynomial time.

Theorem 4.2.5 *If the number of bidders is constant and the number of different bids is constant then the optimal allocation can be found in polynomial time.*

Proof: Let c denote the number of different bids. Since there are n bidders and c possible bids, there are c^n possible types of items. Within $O(k + c^n)$ time we can count how many items are of each type. Note that with respect to the the allocation's value, items of the same type are equivalent. For a fixed type of items there are at most k items of this type and therefore there are $O(k^n)$ allocations of items of the same type we need to consider. A full enumeration over allocations of items of all types is upper bounded by $O(k^{nc^n})$ different configurations. Since c and n are constants, the computation time is polynomial in k. ∎

4.3 Approximate Solutions

In this section presents the central results of the chapter, which include approximation algorithms for the case that the number of bidders is constant and for the general case of an arbitrary number of bidders. We also present improved bounds when all bidders have the same budget constraint, yet possibly different valuations.

4.3.1 An FPTAS for a Constant Number of Bidders

If the number of bidders is constant then there exists a fully polynomial time approximation scheme (FPTAS), meaning that the approximate allocation is at least $1-\epsilon$ (for any $\epsilon > 0$) times the optimal allocation, and the running time is polynomial in the number of items k and in $\frac{1}{\epsilon}$. The algorithm is an adaptation of the FPTAS for the scheduling problem with unrelated machines of Horowitz and Sahni [43].

The algorithm uses sets of tuples $(v_1, a_1, v_2, a_2, \ldots, v_n, a_n, t)$ to construct the approximation. Each tuple represents an allocation of subsets of items to the bidders. Let v_i denote the valuation of bidder i for the items allocated to him, and let a_i denote a bit-vector indicating which items were allocated to this bidder (if we are only interested in the valuation's value and not in the structure of the allocation, then the a_i are redundant). Let t denote the total valuation of the partial allocation. The set S_j contains tuples representing allocations of the first j items.

Algorithm DPA is described in Figure 4.1 and its analysis yields the following:

Lemma 4.3.1 *Algorithm DPA is a $1+\epsilon$ approximation for the optimal allocation with budget constraints.*

Proof: If stage 4b was removed from the algorithm, then a full enumeration over all allocations would be executed, yielding an exact optimal solution. Consider a tuple $\tau' = (v'_1, a'_1, \ldots, v'_n, a'_n, t')$ that was removed in favor of tuple $\tau = (v_1, a_1, \ldots, v_n, a_n, t)$ in the selection stage 4b. The loss of dropping τ' is bounded by the total valuation differences between the tuples τ and τ', which is

$$\sum_{i=1}^{n} |v_i - v'_i| \leq n\frac{\gamma\epsilon}{nk} = \frac{\gamma\epsilon}{k} \ , \tag{4.3}$$

since the difference $|v_i - v'_i|$ is at most $\frac{\gamma\epsilon}{nk}$.

The total loss over k stages is at most $\gamma\epsilon$, and since the optimal value is at least γ, the returned allocation is at least $1-\epsilon$ times the optimal allocation, which is a $1+\frac{\epsilon}{1-\epsilon}$ approximation. For any $\hat{\epsilon} > 0$, by choosing ϵ such that $\hat{\epsilon} \geq \frac{\epsilon}{1-\epsilon}$, we have a $1+\hat{\epsilon}$ approximation ∎

Lemma 4.3.2 *The running time of Algorithm DPA is polynomial in the size of the input and in $\frac{1}{\epsilon}$, for a constant n.*

Input: a bid matrix $\{b_{ij}\}$, a vector of constraints $\{d_i\}$ and a desired approximation factor ϵ.

Output: an allocation of items to bidders.

1. Let $\gamma = \max\{d_i\}$.
2. Divide the segment $[0, n\gamma]$ into $\frac{n^2k}{\epsilon}$ equal intervals of length $\frac{\gamma\epsilon}{nk}$ each.
3. Initialize $S_0 = \{(0, \phi, 0, \phi, \ldots, 0, \phi, 0)\}$.
4. For each item j,
   (a) Construct S_j from S_{j-1}, by replacing each tuple $s \in S_{j-1}$ with n tuples $s_1, \ldots, s_n$, where tuple s_i represents the same allocation as s, with item j allocated to bidder i.
   (b) For any tuple $s = (v_1, a_1, \ldots, v_n, a_n, t)$, if there is a tuple $s' = (v'_1, a'_1, \ldots, v'_n, a'_n, t')$ such that v_i, and v'_i are in the same interval for every i, and $t' \leq t$, remove s' from S_j.
5. Return the allocation represented by the tuple from S_k with the largest total value.

Figure 4.1: Algorithm Dynamic Programming Allocation (DPA)

Proof: The main time consumer is the calculation of the S_j sets. Each set contains at most $\left(\frac{n^2k}{\epsilon}\right)^n$ tuples, so the calculation of the next set requires calculating $n\left(\frac{n^2k}{\epsilon}\right)^n$ tuples. Filtering the tuples can be accomplished in a running time proportional to the upper bound on the number of tuples, by dividing them into buckets and choosing a representative from each bucket. Since n is a constant, the running time of DPA is $O\left(\left(\frac{k}{\epsilon}\right)^n\right)$, i.e. polynomial in k and $\frac{1}{\epsilon}$. ∎

Combining Lemmata 4.3.1 and 4.3.2, we have:

Theorem 4.3.3 *Algorithm DPA is an* FPTAS *for the auction with budget constraints problem with a constant number of bidders.*

Since DPA requires space for representing $O\left(\left(\frac{k}{\epsilon}\right)^n\right)$ tuples, its space complexity is polynomial in $\frac{1}{\epsilon}$. We note that there exists an approximation algorithm with space complexity polynomial in $\log\frac{1}{\epsilon}$, which is the space required for representing the value ϵ. The approximation, however, is a *PTAS*, rather than an *FPTAS*, i.e. the running time is not polynomial in $\frac{1}{\epsilon}$. This PTAS shares similar ideas with algorithms used for an arbitrary number of bidders, and therefore its presentation follows them in Section 4.3.4.

Input: a bid matrix $\{b_{ij}\}$ and a vector of constraints $\{d_i\}$.

Output: an allocation of items to bidders.

1. Find an optimal fractional allocation using LP.
2. For each item j, assign it to exactly one bidder, by selecting bidder i with probability x_{ij}.

Figure 4.2: Algorithm Random Rounding (RR)

4.3.2 Arbitrary Number of Bidders

We now present and analyze an algorithm with a provable approximation ratio of $\frac{e}{e-1} \approx 1.582$ for an arbitrary number of bidders.

In order to find an approximation to this allocation problem, we use the following Linear Programming (LP), which solves a relaxed version of the original Integer Programming (4.1):

$$\begin{array}{ll} \max \sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij} b_{ij} & \text{s.t.} \\ \sum_{j=1}^{k} x_{ij} b_{ij} \leq d_i & 1 \leq i \leq n \\ \sum_{i=1}^{n} x_{ij} \leq 1 & 1 \leq j \leq k \\ x_{ij} \geq 0 & 1 \leq j \leq k, 1 \leq i \leq n \end{array} \tag{4.4}$$

The relaxation replaces the original Boolean variables z_{ij} by variables x_{ij}, indicating a fractional assignment of items. The equations are simplified by removing the v_i variables, which become redundant in the LP. As is the case with any relaxation method, our main task is to round the fractional assignments to an integral solution.

Algorithm *Random Rounding* (RR) is an approximation algorithm for the auction with budget constraints problem with an arbitrary number of bidders.

We note that in Step 2 each item is allocated only once to exactly one of the bidders who participate in its fractional allocation. Additionally, if for some item j, $\sum_i x_{ij} < 1$ we can artificially increase one of the non-zero x_{ij} to assure that the item is indeed allocated. Obviously, Algorithm RR is randomized, outputs a feasible allocation, and terminates in polynomial time complexity. It remains to analyze the approximation ratio of the algorithm.

Theorem 4.3.4 *The expected approximation ratio of Algorithm RR is at most* $\frac{e}{e-1} \approx 1.582$.

Proof: Let Z_i be a random variable indicating the valuation of bidder i after the allocation. The expected total valuation is $\sum_i E(Z_i)$. Therefore, it is sufficient to prove that for any bidder, the expected ratio between the valuation of the fractional allocation and the final integer allocation is at most $\frac{e}{e-1}$.

We analyze separately the expected valuation of each bidder i. Without loss of generality, when considering bidder i we normalize the bids and the budget constraint such that $d_i = 1$, in order to simplify the calculations. Let $B_i = \sum_j x_{ij} b_{ij}$ denote the normalized valuation of bidder i generated by the fractional allocation. We prove that $E(Z_i) \geq \frac{e-1}{e} B_i$. Without loss of generality, we assume that the indices of the items allocated (fully or fractionally) to bidder i are $1, \ldots, r$.

Let X_{ij} be a random variable, which indicates whether item j is allocated to bidder i. The random variable X_{ij} is 1 with probability x_{ij} and 0 otherwise. The expected valuation of bidder i is therefore $Z_i = \min(1, \sum_{j=1}^{r} X_{ij} b_{ij})$.

We now prove that we can decrease the expected valuation of bidder i without changing the valuation of the fractional assignment, by replacing a bid for some item j with a maximal bid and decreasing the value of x_{ij} appropriately.

Lemma 4.3.5 *Assuming $d_i = 1$, the expected valuation of bidder i can only decrease if for some item j we replace b_{ij} with $\hat{b}_{ij} = 1$ and replace x_{ij} with $\hat{x}_{ij} = b_{ij} x_{ij}$, while the fractional valuation remains unchanged.*

Proof: For convenience, without loss of generality, we will assume that the item's index is 1, meaning that we replace b_{i1} and x_{i1} with $\hat{b}_{i1} = 1$ and $\hat{x}_{i1} = b_{i1} x_{i1}$, respectively. The size of the fractional item assigned by the Linear Programming (4.4) is $\hat{b}_{i1} \hat{x}_{i1} = b_{i1} x_{i1}$, meaning that the valuation of bidder i in the fractional assignment remains unchanged. We observe the effect of replacing X_{i1} with the corresponding $\hat{X}_{i1}$ on Z_i, when the remaining variables are kept constant. We denote $Z_{i1} = \min(1, \sum_{j=2}^{r} X_{ij} b_{ij})$ and observe that $Z_i - Z_{i1}$ is a random variable that denotes the marginal contribution of X_{i1} to the total valuation. We examine how replacing X_{i1} with $\hat{X}_{i1}$ effects the possible values of $Z_i - Z_{i1}$.

1. If $0 \leq Z_{i1} \leq 1 - b_{i1}$: The marginal contribution of X_{i1} is either 0 or b_{i1}, so the expected contribution is $b_{i1} x_{i1}$. The contribution of $\hat{X}_{i1}$ is either 0 or $1 - Z_{i1} \leq 1 = \hat{b}_{ij}$, so the expected marginal contribution is at most $\hat{x}_{i1} = b_{i1} x_{i1}$

2. If $1 - b_{i1} < Z_{i1} \leq 1$: The marginal contribution of both variables is either 0 or $1 - Z_{i1}$. The expected marginal contribution of X_{i1} is $(1 - Z_{i1}) x_{i1}$. The expected marginal contribution of $\hat{X}_{i1}$ is $(1 - Z_{i1}) x_{i1} b_{i1} \leq (1 - Z_{i1}) x_{i1}$.

In both cases, by replacing X_{i1} with $\hat{X}_{i1}$ we can only decrease $E(Z_i)$, without changing B_i. ∎

By repeatedly applying Lemma 4.3.5 for each $1 \leq j \leq r$ we replace b_{ij} with $\hat{b}_{ij} = 1$, x_{ij} with $\hat{x}_{ij} = b_{ij} x_{ij}$ and X_{ij} with $\hat{X}_{ij}$. Since each replacement does not increase $E(Z_i)$, we have $E(\hat{Z}_i) = E(\min(1, \sum_{j=1}^{r} \hat{X}_{ij})) \leq E(\min(1, \sum_{j=1}^{r} X_{ij}))$.

Since for each j, $\hat{X}_{ij}$ is either 0 or 1, then $\hat{Z}_i$ is also either 0 or 1. Therefore:

$$E(\hat{Z}_i) = P(\hat{Z}_i = 1) = 1 - P(\hat{Z}_i = 0) = 1 - \prod_{j=1}^{r} (1 - \hat{x}_{ij}) \tag{4.5}$$

The expectation is minimized when $\prod_{j=1}^{r}(1-\hat{x}_{ij})$ is maximized. Under the constraint $B_i = \sum_{j=1}^{r} b_{ij}x_{ij} = \sum_{j=1}^{r} \hat{x}_{ij}$, the maximum is when all $\hat{x}_{ij}$ are equal to B_i/r. Therefore:

$$E(\hat{Z}_i) \geq 1 - \left(1 - \frac{B_i}{r}\right)^r \geq 1 - e^{-B_i} \geq B_i(1 - e^{-1}) \tag{4.6}$$

The last inequality follows since $1 - e^{-x} \geq x(1 - e^{-1})$ for $x \in [0, 1]$. Therefore, we have:

$$\frac{B_i}{E(Z_i)} \leq \frac{B_i}{E(\hat{Z}_i)} \leq \frac{e}{e-1} \approx 1.582 \tag{4.7}$$

Since the approximation holds for the expected valuation of each bidder separately, it also holds for the expected total valuation. ∎

The following theorem claims that in the worst case, RR has an approximation ratio of$\frac{e}{e-1}$.

Theorem 4.3.6 *The expected approximation ratio of Algorithm RR is at least* $\frac{e}{e-1} \approx 1.582$.

Proof: The lower bound of $\frac{e}{e-1}$ for the approximation ratio is achieved in the following setting: $n+1$ bidders, $A_0, A_1, \ldots, A_n$, compete on $2n$ items, $I_1, I_2, \ldots, I_{2n}$. Bidder A_0 has a budget constraint $d_0 = n$, bids $b_{0j} = n$ for items $I_1, \ldots I_n$, and bids $b_{0(n+j)} = 0$ for items $I_{n+1}, \ldots I_{2n}$. The other bidders all have a budget constraint of $d_i = n^{-1}$. For $1 \leq i \leq n$, bidder A_i bids $b_{ii} = n^{-1}$ for item I_i, $b_{i(n+i)} = n^{-2}$ for item I_{n+i}, and $b_{ij} = 0$ for the other items.

The Linear Programming (4.4) finds a unique fractional allocation which satisfies all budget constraints: for each $1 \leq i \leq n$, bidder A_i receives item I_{n+i} (fully) and a $\frac{n-1}{n}$ fraction of item I_i. The remaining fraction of $\frac{1}{n}$ of item I_i is given to bidder A_0. The total valuation of the fractional allocation is $n+1$.

The optimal integer solution nearly reaches the optimal fractional solution: For some arbitrary $1 \leq k \leq n$, bidder A_0 receives item I_k, bidder A_k receives I_{n+k}, and every other bidder $j \notin \{0, k\}$, bidder A_j receives items I_j and I_{n+j}. The only bidder that does not reach (or exceed) its budget constraint is A_k, and the total valuation is $n + 1 - \frac{1}{n} + \frac{1}{n^2}$.

Algorithm RR achieves an expected valuation of $n(1 - (1 - \frac{1}{n})^n) + \frac{n^2-n+1}{n^2}$ which is approximately $n(1 - e^{-1}) + 1$ for sufficiently large n. ∎

From Theorems 4.3.4 and 4.3.6 we have the following corollary.

Corollary 4.3.7 *the approximation ratio of Algorithm RR is exactly* $\frac{e}{e-1}$.

4.3.3 Derandomized Rounding

A natural derandomization of Algorithm RR which maintains the $\frac{e}{e-1}$ approximation ratio would be to sequentially assign each item such that the expected valuation maintains above the expectation. Although calculating exactly the expected valuation may be involved as we need to consider up to k random variables for a given bidder, this difficulty can be resolved by replacing the exact expected valuation with lower bounds, by techniques similar to those used in the proof of Theorem 4.3.4.

Input: a bid matrix $\{b_{ij}\}$ and a vector of constraints $\{d_i\}$.

Output: an allocation of items to bidders.

1. Find an optimal fractional allocation using LP.
2. For each item j that is fractionally allocated,
   (a) Compute $T = \{i | x_{ij} > 0\}$, the set of bidders that receive a fraction of item j.
   (b) $h = \arg\max_{i \in T}(B_i(S_i^1 \cup \{j\}, S_i^x \setminus \{j\}) + \sum_{\hat{i} \neq i} B_{\hat{i}}(S_{\hat{i}}^1, S_{\hat{i}}^x \setminus \{j\}))$.
   (c) $x_{hj} = 1,\ x_{ij} = 0,\ \forall i \neq h$.

Figure 4.3: A deterministic rounding algorithm

Theorem 4.3.8 *A deterministic allocation of an auction with budget constraint that is a $\frac{e}{e-1}$ approximation can be computed in polynomial time.*

Proof: For bidder i, let S_i^1 denote the set of items fully allocated to i, S_i^x denote the set of items fractionally allocated to i, and S_i^0 denote the remaining items, which are not allocated to this bidder. We can lower bound the expected valuation of this bidder as follows: Let $L_i(S_i^1) = \min\{d_i, \sum_{j \in S_i^1} b_{ij}\}$ denote the assured valuation from items in S_i^1. By applying Lemma 4.3.5 on the items in S_i^x, the total expected valuation is lower bounded by

$$B_i(S_i^1, S_i^x) = L_i(S_i^1) + (d_i - L_i(S_i^1))(1 - \prod_{j \in S_i^x} (1 - b_{ij} x_{ij})) \ . \tag{4.8}$$

Similarly, for an arbitrary item $j \in S_i^x$, we can calculate two additional lower bounds of the expected valuation from bidder i: $B_i(S_i^1 \cup \{j\}, S_i^x \setminus \{j\})$, where item j is fully allocated to bidder i and $B_i(S_i^1, S_i^x \setminus \{j\})$, where item j is not allocated to i at all. Observe that since these calculations do not invoke Lemma 4.3.5 for item j, the weighted average of these lower bounds (with weights x_{ij} and $1 - x_{ij}$, respectively) is at least as large as the previous lower bound $B_i(S_i^1, S_i^x)$ calculated with the invocation of Lemma 4.3.5 for item j.

We can efficiently find a deterministic allocation that assures a competitive ratio of $\frac{e}{e-1}$, by rounding the fractional assignments as described in Figure 4.3. In words, we lower bound the total expected valuation by adding the lower bounds for each bidder. Now, we repeatedly select an arbitrary item that is currently fractionally allocated and recalculate lower bounds for each possible allocation of this item. We round the allocation for this item such that the new lower bound of the total expected valuation is maximized. Since we start with a lower bound of at least $\frac{e-1}{e}$ of the original fractional assignment, which only improves through the rounding stages, we are bound to end with a deterministic allocation that is a $\frac{e}{e-1}$ approximation. ∎

We now discuss alternative deterministic algorithms for rounding the fractional assignments. Among all feasible allocations that can be achieved by rounding every $0 < x_{ij} < 1$ in a solution of the LP instance, to either 0 or 1, we refer to the rounding with the highest total valuation as an *optimal rounding*. Obviously, an algorithm that returns an optimal rounding has an approximation ratio of at most any other rounding algorithm, including RR.

A convenient method to observe the output of the LP is by constructing a bipartite graph $G = (N, K, E)$, where the nodes N and K correspond to the bidders and items, respectively, and the edges E indicate that a bidder was assigned an item, or a fraction of an item. An illustrative example is given in Figure 4.4.

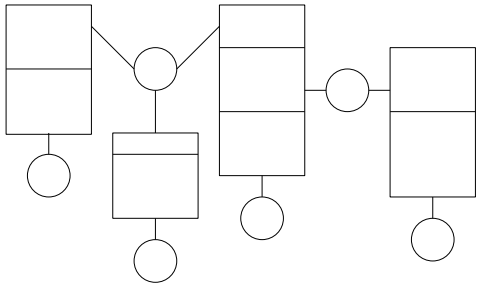

Figure 4.4: Graphical representation of a fractional assignment
The graph includes a single component, which represents an allocation of 6 items to 4 bidders. Items are represented by circular nodes while bidders are represented by rectangles, where the height of a rectangle is proportional to the budget. To illustrate the size of fractions, each rectangle is divided to segments in proportion to the values of the fractional allocation.

The original allocation problem can be divided into several subproblems, where each subproblem consists only bidders and items from the same component in G. Solving the Linear Programming separately for each subset of bidders and items should return the same allocation. Therefore, we may concentrate on each component of G separately.

The LP includes $nk + n + k$ constraints using nk variables. In the solution of the LP, at least nk of the constraints are satisfied with equality, meaning that at most $n + k$ of the x_{ij} variables are non-zero. Each non zero variable x_{ij} matches one edge in G, and therefore G contains at most $n + k$ edges. On the other hand, since we assume G is connected (as we concentrate on a single component) it must contain at least $n + k - 1$ edges, in order to connect $n + k$ nodes. Therefore, G is either a tree, or contains exactly one cycle.

The following lemma claims that if G contains a cycle, then the optimal solution to the LP can be modified, such that one edge will be deleted from G, and therefore G transforms to a tree, while maintaining the optimality of the fractional solution.

Lemma 4.3.9 *There is a node in the polyhedron of the LP which maximizes the objective function and induces a graph without cycles, and can be found in polynomial running time.*

Proof: Assume there is a cycle n_1, k_1, n_2,...,n_i, k_i, $n_i = n_1$. If we attempt to increase n_1's holdings in k_1 by a sufficiently small fraction δ_1 (for now, ignore the fact that the budget constraint of n_1 may be already met), we need to borrow δ_1 from n_2, which will in turn recover from its loss of $\delta_1 b_{n_2 k_1}$ by borrowing a δ_2 fraction of n_3's holdings in k_2. This routine continues until n_1 loses a fraction δ_i from k_1. There are two possibilities at this point:

1. $b_{11}\delta_1 \geq b_{1i}\delta_i$: The objective function does not decrease (if the inequality is strict then the budget constraint of n_1 must be satisfied to avoid contradiction to the LP's optimality). Therefore, δ_1 can be increased until some x_{ij} reaches 0, which results in removing an edge from the cycle.

2. $b_{11}\delta_1 < b_{1i}\delta_i$: The objective function decreases. Therefore, we shall reverse the process with a negative δ_1, until one edge is removed.

This process is equivalent to replacing the constraints in the LP that are strictly satisfied. One of the $n+k$ non trivial constraints is replaced with one of the nk constraints of type $x_{ij} = 0$. ∎

By applying Lemma 4.3.9, the fixed graph is a tree. Therefore, out of at most $n+k$ constraints in the LP that are not satisfied with equality, exactly $n+k-1$ of them are of type $x_{ij} \geq 0$. This means that at most one of the non-trivial constraints is not strict, as formally stated in the following observation:

Observation 4.3.10 *For each component in G, either there is at most one item which is not fully allocated, yet all bidders reach their budget constraint, or there is at most one bidder that hasn't met its budget constraint, yet all items are fully allocated.*

Algorithm *Semi Optimal Rounding* (SOR), returns an allocation that has a total valuation of at least $1-\epsilon$ times the optimal rounding, for any $\epsilon > 0$. The algorithm applies a recursive 'divide and conquer' process called *ROUND* on the tree graph representing the fractional allocations to choose a nearly optimal rounding.

We use the following notation in process ROUND: for a tree T and a node v, let $T_{v,i}$ denote the i-th subtree rooted at v. Let $T^+_{v,i}$ denote the tree containing the subtree $T_{v,i}$, the node v and the edge connecting v to $T_{v,i}$. When node v denotes a bidder, let u_i denote the i-th item node shared by v and bidders in $T_{v,i}$. Let $T^-_{v,i}$ denote the same tree as $T_{v,i}$, with item u_i replaced with a dummy item u^-_i, which has zero valuations from all bidders.

When the central node v represents an item, combining the partial allocations of the subtrees is a simple process, since only one subtree may receive item v. Formally, we enumerate on v's neighbors to calculate the following:

$$\max_j \left(\text{ROUND}(T^+_{v,j}, \epsilon) + \sum_{i \neq j} \text{ROUND}(T_{v,i}, \epsilon) \right) \tag{4.9}$$

However, if v represents a bidder, the number of rounding combinations is exponential in the degree of v, and this is why an approximation is preferred over an exact solution. The approximation process is as follows:

Input: a bid matrix $\{b_{ij}\}$, a vector of constraints $\{d_i\}$ and an approximation factor ϵ.

Output: an allocation of items to bidders.

1. Find an optimal fractional solution using LP.
2. Construct a bipartite graph G representing the LP allocation.
3. For each component G_i in G:
   (a) If G_i contains a cycle, convert G_i to a tree by modifying the LP solution.
   (b) Apply ROUND(G_i, ϵ)

Figure 4.5: Algorithm Semi Optimal Rounding (SOR)

Let r be the degree of node v. Let b_{vi} be the bid of bidder v for the item shared with bidders in the i-th subtree. Let $C_i = \text{ROUND}(T_{v,i}, \frac{\epsilon}{2})$ be the approximated valuation of rounding the i-th subtree when v does not get the i-th item it shares. Let $c_i = \text{ROUND}(T^-_{v,i}, \frac{\epsilon}{2})$ be the approximated valuation of rounding the i-th subtree when v gets the i-th item (while the bidders in $T^-_{v,i}$ share a dummy item with no value). We construct the following allocation problem with 2 bidders: The items are a subset of the original items, reduced to those allocated (partially or fully) to v, denoted as $I_1, I_2, \ldots, I_r$ plus r special items $I'_1, I'_2, \ldots, I'_r$. Bidder 1 has the same budget constraint as the bidder represented by v, and the same bids on $I_1, \ldots I_r$. Bidder 1 bids 0 on the special items $I'_1, \ldots, I'_r$. Bidder 2 has an unbounded budget constraint, and bids c_i for each special item I'_i, and $Cc_i = \max\{0, C_i - c_i\}$ for the original items I_i.

The new allocation problem is a reduction of the original rounding problem. Any rounding possibility matches an allocation with the same value. Therefore, if we approximate the reduced allocation solution, we get an approximation to the original rounding problem. Since there are only two bidders in the reduced problem, we use Algorithm DPA, of Theorem 4.3.3, to approximate an optimal allocation. Formally, we compute:

$$\text{DPA}\left(\begin{pmatrix} b_{v1} & b_{v2} & \ldots & b_{vr} & 0 & 0 & \ldots & 0 \\ c_1 & c_2 & \ldots & c_r & Cc_1 & Cc_2 & \ldots & Cc_r \end{pmatrix}, \begin{pmatrix} d_v \\ \infty \end{pmatrix}, \frac{\epsilon}{2}\right) \tag{4.10}$$

We now analyze the running time and approximation of SOR:

Lemma 4.3.11 *Algorithm* SOR*, returns an allocation whose value is at least* $1-\epsilon$ *times the value of the optimal rounding.*

Proof: At the end of the recursion there is at most only one bidder left, therefore we obviously round optimally. In any other stage, if the central node is an item, then we optimally combine several $1-\epsilon$ approximated partial solutions into one solution, therefore the combined rounding is also a $1-\epsilon$ approximation. When the central node is a bidder,

Input: An allocation tree T and an approximation factor ϵ.

Output: An allocation of items to bidders, for items and bidders induced by T.

1. If T Includes only one bidder, allocate all the items to this bidder. If there are no bidders in T, return a null assignment.
2. Otherwise, find vertex $v \in T$, which is a center of T.
3. If v represents an item, for each subtree $T_{v,i}$ recursively compute ROUND($T_{v,i}$, ϵ) and ROUND($T^{+}_{v,i}$, ϵ) Allocate v to a bidder such that the total valuation is maximized (see (4.9)).
4. If v represents a bidder, for each subtree $T_{v,i}$ recursively compute ROUND($T_{v,i}$, $\frac{\epsilon}{2}$) and ROUND($T^{-}_{v,i}$, $\frac{\epsilon}{2}$). Find a combination of the partial allocations, whose valuation is at least $1 - \frac{\epsilon}{2}$ times an optimal combination (see (4.10)).

Figure 4.6: Process ROUND

the rounding of each subtree is a $1 - \frac{\epsilon}{2}$ approximation, multiplied by a factor of $1 - \frac{\epsilon}{2}$ due to the combination, so the final approximation ratio is at least $1 - \epsilon$. ∎

Lemma 4.3.12 *The running time of* SOR *is polynomial in* n, k *and* $\frac{1}{\epsilon}$

Proof: The initial parts of solving the LP, constructing the bipartite tree graph and removing cycles are trivially polynomial. The rest of the algorithm is the recursive rounding. Without loss of generality, we assume that there is only one component in the bipartite graph, and that the number of nodes is $n + k$. Since we choose the center of the tree, the depth of the recursive process is at most $\log_2(n + k)$. Each edge of the center invokes two recursive calls, so the total number of nodes in all the subtrees in the recursive calls invoked at this stage is at most $2(n + k)$. Since the recursion must terminate when we reach a subtree with only one node, the total number of iterations in the recursive process is $O((n + k)^2)$.

Each iteration is either the end of the recursion, which is trivially polynomial, deals with an item shared by several bidders, which requires an additional computation time that is at most linear in the size of the tree, or deals with a central node representing a bidder, and requires the application of Algorithm DPA. The last case requires solving an allocation problem with 2 bidders and at most k items. The desired approximation ratio for an iteration at depth d of the recursion is approximately $1 + \frac{\epsilon}{2^d}$. Since d is at most $\log_2(n + k)$, the required approximation ratio is at most $1 + \frac{\epsilon}{n+k}$, Which DPA completes in a running time polynomial in n, k and $\frac{1}{\epsilon}$.

since the running time for each iteration of the recursion is polynomial, and the number of iterations is also polynomial, Algorithm SOR has a polynomial complexity running time. ∎

Combining Lemmas 4.3.11 and 4.3.12, we have the following:

Input: a bid matrix $\{b_{ij}\}$, a vector of constraints $\{d_i\}$ and a desired approximation factor ϵ.

Output: an allocation of items to bidders.

1. Let $\gamma = \max\{d_i\}$.
2. Classify bids as `HIGH` if $b_{ij} \geq \frac{\epsilon\gamma}{n}$ and `LOW` otherwise.
3. Enumerate over all partial allocations using only `HIGH` bids and allocating at most n/ϵ items to each bidder.
4. For each partial allocation,
   (a) Set `HIGH` bids of unallocated items to 0.
   (b) Set the budget of each bidder to the remaining unused budget.
   (c) Reduce `LOW` bids to the new budget if they are currently higher.
   (d) Find a fractional allocation of the remaining items using LP.
   (e) Round the fractional allocation to an integral one arbitrarily.
5. Of all generated allocations, return the allocation with the highest valuation.

Figure 4.7: A PTAS for the auction with budget constraints problem

Theorem 4.3.13 *For any c such that a c-approximation LP rounding algorithm for allocations with budget constraints exists, Algorithm* SOR *has an approximation ratio of at most $c + \epsilon$ and a polynomial running time for any $\epsilon > 0$.*

Since the tightest upper bound we have for LP rounding algorithms is $\frac{e}{e-1}$, we can only guarantee an approximation ratio of $\frac{e}{e-1} + \epsilon$ for SOR. By applying both SOR and the sequential rounding in figure 4.3 we can get rid of the additional factor of ϵ and guarantee an approximation ratio of $\frac{e}{e-1}$. This bound is not tight, as the lower bound of $\frac{4}{3}$ for rounding algorithms (presented in Section 4.3.6) holds also for this algorithm.

4.3.4 Space Efficient Allocations for PTAS

Using results from the previous sections we define and analyze an alternative approximation algorithm for the case of a constant number of bidders. The main advantage of this algorithm over Algorithm DPA (which is an FPTAS) is the required space, which for DPA is polynomial in k and in $\frac{1}{\epsilon}$, while the required space in the following PTAS is polynomial in k and $\log \frac{1}{\epsilon}$.

The general idea of the algorithm is to combine a partial enumeration on items with high valuations with a relaxed LP solution on the remaining items. A similar idea was implemented by Lenstra et al. [57] for the scheduling problem with unrelated machines.

Let γ be the highest bid of a single bidder for all of the items (without loss of generality, γ is the highest budget constraint). For any arbitrary $\epsilon > 0$, a bid b_{ij} is `HIGH` if $b_{ij} \geq \epsilon\gamma/n$ and `LOW` otherwise. The algorithm is constructed from three main processes: Enumerating over allocations of items with `HIGH` bids, finding a relaxed allocation for the rest of the items, and finally rounding the relaxed allocation.

Each bidder can receive at most n/ϵ items with `HIGH` bids, before reaching the budget constraint. We may ignore additional `HIGH` items, as they do not contribute to the bidder's valuation. Therefore, it is possible to enumerate all $O(k^{n^2/\epsilon})$ allocations of items with `HIGH` values in polynomial time.

Note that the enumeration over all allocations of `HIGH` valued items includes the partial allocation used by the optimal allocation for all items. For each partial allocation, we use the relaxed LP to allocate the remaining `LOW` items. For each instance of the LP, several adjustments need to be made according to the allocation of the `HIGH` items: First, all `HIGH` bids need to be removed from the LP, including bids for items that weren't allocated yet. Second, all budget constraints and `LOW` bids need to be adjusted to the actual remaining budget of each bidder, after the `HIGH` value items were already allocated in the enumeration phase.

The solution for the LP allocates the remaining $k' \leq k$ items, using $n + k'$ fractions (which can be reduced to $n + k' - 1$ according to Lemma 4.3.9). Since without loss of generality, each of the k' items is allocated to at least one bidder, fully or fractionally, there are at most n (or $n-1$, if cycles were removed) redundant fractions that need to be removed in the rounding process of the LP solution. For a constant n, this can be done optimally in polynomial time, but any rounding will suffice for the analysis. Finally, over all enumerated allocations, pick the allocation with the highest revenue.

The enumeration which allocates items with `HIGH` bids also includes a partial allocation that can be completed into an optimal allocation of all items. Therefore, after completing the allocation using LP for `LOW` value items, at least one of the allocations has a value that upper bounds the optimal allocation. In the rounding phase, we lose at most n fractions of `LOW` value items, with a total loss of at most ϵ/γ. For any ϵ, the guaranteed approximation ratio is at least:

$$\frac{ALG}{OPT} \geq \frac{OPT - \epsilon\gamma}{OPT} \geq 1 - \frac{\epsilon\gamma}{\gamma} = 1 - \epsilon. \tag{4.11}$$

Notice that γ is used as a lower bound for the optimum, thus any alternative better lower bound (such as the greedy allocation, which guarantees a 2-approximation) may improve the running time, as it reduces the size of the enumeration in the first stage.

4.3.5 Bidders with Identical Budget Constraints

We now derive improved approximation bounds for RR in the case where all bidders have the same budget constraint. The approximation ratio of $\frac{e}{e-1}$ is due to a bound of $1 - (1 - 1/r)^r$ on the expected valuation of each bidder, where r is the number of items owned or shared by a bidder, and the budget is normalized to 1. Actually, by considering

items fully assigned to the same bidder as one large item, we achieve a tighter bound of $1-(1-\frac{1}{a+1})^{a+1}$, where a is the number of partial assignments of items to a bidder. When a goes to infinity, the bound still goes to $\frac{e-1}{e}$, however not all bidders will have infinitely many fractional items. If all bidders have identical budget constraints, we can use this property to derive a better approximation ratio for RR.

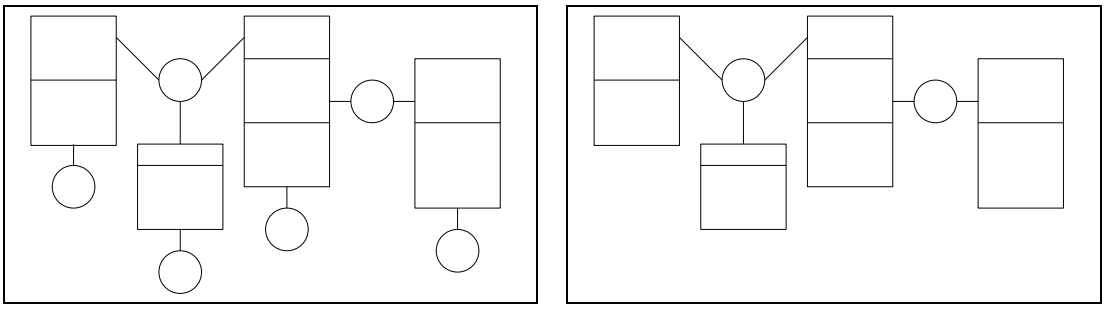

Figure 4.8: An allocation graph G and its subgraph G'.
Item nodes that are leaves in G are removed to construct G'. The bidder whose node is second to the right is classified to R_2 while the others are classified to R_1. The left item node is classified to S_2 while the right item node is classified to S_1.

Assuming the bipartite graph G, constructed from the solution to LP has only one component, let G' be the subgraph of G where nodes corresponding to items that are allocated only to one bidder are removed. Therefore, G' represents only the items shared among several bidders. Let N_i denote the node in G' corresponding to bidder i and let K_j denote the node in G' corresponding to item j. We define the following sets:

Definition 4.3.14 *Let R_a be the set of bidder nodes in G' such that $R_a = \{N_i | Deg(N_i) = a\}$, and let S_a be the set of item nodes in G' such that $S_a = \{K_j | Deg(K_j) = a\}$.*

The following relation holds for R_a and S_a, based on the fact that G' is a tree:

Lemma 4.3.15 $|R_1| = 2 + \sum_{a\geq 3}(a-2)|R_a| + \sum_{a\geq 3}(a-2)|S_a|$

Proof: The degree of nodes in R_a and in S_a is a. The sum of the degrees of all nodes is twice the number of edges, i.e.

$$\sum_{a\geq 1} a|R_a| + \sum_{a\geq 2} a|S_a| = 2|E|$$

We note that G' consists of nodes corresponding to items with fractional assignments, and therefore $|S_1| = 0$. Since G' is a tree, the number of edges is given by:

$$|E| = \sum_{a\geq 1} |R_a| + \sum_{a\geq 2} |S_a| - 1$$

Therefore, we have

$$\sum_{a\geq 1} a|R_a| + \sum_{a\geq 2} a|S_a| = 2\sum_{a\geq 1} |R_a| + 2\sum_{a\geq 2} |S_a| - 2$$

By subtracting common parts of both sides of the equation, we get the following desired equation:

$$\sum_{a\geq 3}(a-2)|R_a| + \sum_{a\geq 3}(a-2)|S_a| + 2 = |R_1|$$

■

According to Lemma 4.3.15, the number of bidders who have only one fraction of an item is fairly large: There are two of these bidders to begin with. Each bidder that has more than two fractions of items enforces another bidder in R_1 for each extra fraction. Also, Each item shared between three or more bidders adds another bidder to R_1 for each share beyond the second. We can use this property to prove the following:

Theorem 4.3.16 *When all bidders have the same budget constraint, the approximation ratio of algorithm RR is at most* $\frac{27}{19} \approx 1.421$.

Proof: Without loss of generality assume that the identical budget constraint is 1, and that G' is a tree. For each bidder from R_a ($a > 2$) we match $a-2$ bidders from R_1. By Lemma 4.3.15 this is possible, leaving at least 2 bidders from R_1 unmatched.

We first assume all bidders reach their budget constraint. Bidders from R_2 are unmatched, and have an expected valuation of at least $1-(1-1/3)^3 = \frac{19}{27}$ each. Bidders from set R_a have an expected valuation of at least $1-(1-1/(a+1))^{(a+1)}$, which is less than $\frac{19}{27}$ for $a \geq 3$, but they are matched with $a-2$ bidders from R_1 who have an expected valuation of at least $1-(1-1/2)^2 = 3/4$, each. On average, the expected valuation is larger than $\frac{19}{27}$, for any $a \geq 3$.

If not all bidders reach their budget constraint, by Observation 4.3.10 only one bidder u has not met the constraint. If u belongs to R_a, then the expected valuation of u is still at least $1-(1-1/(a+1))^{(a+1)}$ times the valuation achieved by the fractional assignment. If bidder u participates in a match, the ratio between the total expected valuation of the bidders in the match and the fractional valuation will remain above $\frac{19}{27}$ as long as u does not belong to R_1. If $u \in R_1$ it is possible to replace u with an unmatched bidder from R_1, since by Lemma 4.3.15 at least 2 bidders that are in R_1 are unmatched.

For each group of matched bidders, the ratio between the total expected valuation to the fractional valuation is at least $\frac{19}{27}$. Unmatched bidders remain only in R_1 or R_2 and therefore also have an expected valuation of at least $\frac{19}{27}$ times the fractional valuation. Therefore, the approximation ratio is at most $\frac{27}{19} \approx 1.421$. ■

Theorem 4.3.16 implies that bidders from R_2 are the bottleneck of the analysis, as they are not matched with bidders from R_1. If this bottleneck can be resolved, the approximation ratio could drop to 1.3951, which is induced by bidders from R_3, who are matched with bidders from R_1, therefore their average expectation is $\frac{1}{2}(\frac{3}{4} + \frac{175}{256}) \approx 0.7168 = (1.3951)^{-1}$. By using the sets S_a the following theorem claims that this improvement is indeed achievable.

Theorem 4.3.17 *When all bidders have the same budget constraint, the approximation ratio of algorithm RR is at most* 1.3951.

R_a	Ratio	Matched R_1	Average	1/Average
R_1	0.7500	n/a	0.7500	1.3333
R_2	0.7037	0	0.7037	1.4211
R_3	0.6836	1	0.7168	1.3951
R_4	0.6723	2	0.7241	1.3810
R_5	0.6651	3	0.7288	1.3722
R_6	0.6601	4	0.7320	1.3661

Table 4.1: Implied approximation ratios for various R_a

Proof: We assume that the budget constraint is 1, and concentrate on bidders from R_2, for which the analysis in Theorem 4.3.16 is insufficient. The main idea of the proof is to cluster bidders together, such that the average expected valuation in each cluster is at least 1.3951^{-1}. The clusters are constructed from bidders that share the same item, along with bidders from R_1 that were matched to them in the proof of Theorem 4.3.16. Since bidders from R_a participate in a clusters, we use a weighted average to compute the average expected valuation within a cluster, with a weight of $\frac{1}{a}$ for a bidder from R_a.

For now, we assume that in the fractional allocation, all bidders reach their budget constraint and that all items are fully allocated. At the end of the proof we deal with the case where these assumptions do not hold.

Items with many fractions: If a certain bidder is allocated a fraction of an item from S_a, where $a \geq 3$, then this item implies that there are $a-2$ bidders from R_1 that were not matched by the matching process described in the proof of Theorem 4.3.16. Since a bidders share the item, we can match each a $\frac{a-2}{a}$ fraction of a bidder from R_1. For a bidder from R_2, which has a weight of $\frac{1}{2}$ in this match (since the other half is dealt by the other fractional item allocated to it) we have a weighted average valuation of

$$\frac{\frac{1}{2}(1-(\frac{2}{3})^3)+\frac{a-2}{a}(1-(\frac{1}{2})^2)}{\frac{1}{2}+\frac{a-2}{a}} \ . \tag{4.12}$$

Table 4.2 presents the values of this expression for several S_a. The only case that does not reach the target approximation ratio is when a bidder from R_2 holds a fraction of an item from S_2.

S_a	Matched R_1	Average	1/Average
S_2	0	0.7037	1.4211
S_3	1	0.7222	1.3846
S_4	2	0.7269	1.3758
S_5	3	0.7290	1.3718
S_6	4	0.7302	1.3696

Table 4.2: Implied approximation ratios for various S_a

Bidders with many fractions: If a bidder from R_2 shares a fraction of an item

from S_2 with neighbor bidder from R_a ($a \geq 2$), then the average expected valuation of the bidders in the cluster containing the bidder from R_2 (with weight $\frac{1}{2}$), the bidder from R_a and $a-2$ matched bidders from R_1 (all with a weight of $\frac{1}{a}$) is

$$\frac{\frac{1}{2}(1-(\frac{2}{3})^3)+\frac{a-2}{a}(1-(\frac{1}{2})^2)+\frac{1}{a}(1-(\frac{a}{a+1})^{a+1})}{\frac{1}{2}+\frac{a-1}{a}}. \tag{4.13}$$

Table 4.3 presents the values of this expression for several R_a. The target approximation ratio is reached for any $a \geq 5$, while neighbor bidders from R_2, R_3 and R_4 require stronger arguments. The case where the neighbor bidder is from R_1 is special and will be analyzed separately.

R_a	Matched R_1	Average	1/Average
R_2	0	0.7037	1.4211
R_3	1	0.7112	1.4061
R_4	2	0.7159	1.3968
R_5	3	0.7191	1.3906
R_6	4	0.7214	1.3862

Table 4.3: Implied approximation ratios for a cluster of R_2 and R_a

Neighbor bidders from $R_2 \cup R_3 \cup R_4$: The worst case expected valuation of a bidder from R_a is achieved when the fractions' sizes are equal to $\frac{1}{a+1}$. When an item from S_2 is fully divided between two neighbor bidders, the worst case scenario cannot occur simultaneously for both bidders. By analyzing both bidders together, we can achieve an improved bound.

Assume that bidders i and i' share item j. By Lemma 4.3.5, we may assume for all fractionally allocated items except j, that their valuation is 1. We cannot apply Lemma 4.3.5 directly on item j, since the Lemma changes the sizes of the fractions, and then we would no longer be able to assume that the j-th item is fully allocated. However, the following argument bypasses this difficulty:

Let j' be the item that is fully allocated to bidder i, if such an item exists. Assume that for some $\delta > 0$, $b_{ij} \leq 1-\delta$ and $b_{ij'} \geq x_{ij}\delta$. Given an arbitrary realization of the randomized rounding algorithm (RR), except for item j, let D be the total valuation of bidder i for items other than j and j'. The expected marginal valuation contributed by the additional two items is:

$$x_{ij}\min\{1-D, b_{ij}+b_{ij'}\}+(1-x_{ij})\min\{1-D, b_{ij'}\} \tag{4.14}$$

If we increase b_{ij} by δ and decrease $b_{ij'}$ by $x_{ij}\delta$, the valuation of the fractional allocations remains the same, while the expected marginal valuation contributed by j and j' changes to:

$$x_{ij}\min\{1-D, b_{ij}+b_{ij'}+(1-x_{ij})\delta\}+(1-x_{ij})\min\{1-D, b_{ij'}-x_{ij}\delta\} \tag{4.15}$$

We note the first expression in (4.14) is at most equal to the first expression in (4.15), while the second expression in (4.14) is at least equal to the second expression in (4.15). If $1 - D \leq b_{ij} + b_{ij'}$ then the first expressions in both equations are equal to $1 - D$, and therefore (4.14) is larger or equal to (4.15). If $1 - D > b_{ij} + b_{ij'}$, then (4.14) equals $x_{ij}b_{ij} + b_{ij'}$ while (4.15) is at most the same. Either way, the expected valuation of i decreases as the valuation of item j increases and the valuation of item j' decreases appropriately. Therefore, we can choose a large enough δ such that either b_{ij} increases to 1, or $b_{ij'}$ decreases to 0, i.e., there are no items fully allocated to this bidder in the fractional assignment.

Given a bidder in R_a such that all items are allocated to it fractionally (in the fractional allocation), its expected valuation in the integral allocation is lower bounded by $1 - (1 - \frac{1}{a})^a$. If $a = 2$ then the lower bound is $\frac{3}{4}$, above the target ratio of 1.3951^{-1}. If the bidder in R_a shares an item from S_2 with a bidder from R_2, then the weighted average of the cluster is:

$$\frac{\frac{1}{2}(1 - (\frac{2}{3})^3) + \frac{a-2}{a}(1 - (\frac{1}{2})^2) + \frac{1}{a}(1 - (\frac{a-1}{a})^a)}{\frac{1}{2} + \frac{a-1}{a}} \ . \tag{4.16}$$

Table 4.4 presents the values of this expression for $a \in \{2, 3, 4\}$, which all reach the target approximation ratio.

R_a	Matched R_1	Average	1/Average
R_2	0	0.7269	1.3758
R_3	1	0.7169	1.3948
R_4	2	0.7182	1.3924

Table 4.4: Implied approximation ratios with only fractionally allocated items

We move on to the case where all fractionally allocated items in a cluster have valuations of 1, and closely observe a cluster containing a bidder i from R_2 and a bidder i' from R_a (for $a \in \{2, 3, 4\}$), sharing an item j from S_2. Let $x = x_{ij}$, given that the item is fully allocated, we have $x_{i'j} = 1 - x$. Let y and z denote the valuations of items fully allocated to i and i', respectively. Therefore, the size of the fraction of the third item allocated to i is $1 - x - y$, and the total size of the fractions of items allocated to i' are $x - z$. We may assume that they are all equal to $\frac{x-z}{a-1}$, since this minimizes the expected valuation of the integral allocation. Figure 4.9 demonstrates this structure for $a = 3$.

The expected valuation of bidder i in the integral allocation is $1-(1-x)(x+y)(1-y)$. The value of y which minimizes this expression is $y = \frac{1-x}{2}$, for which the expected valuation is $1 - (1 - x)(\frac{1+x}{2})^2$.

Similarly, the expected valuation of bidder i' in the integral allocation is

$$1 - x\left(1 - \frac{x - z}{a - 1}\right)^{a-1}(1 - z) \ . \tag{4.17}$$

The value of z which minimizes this expression is $y = \frac{x}{a}$, for which the expected valuation

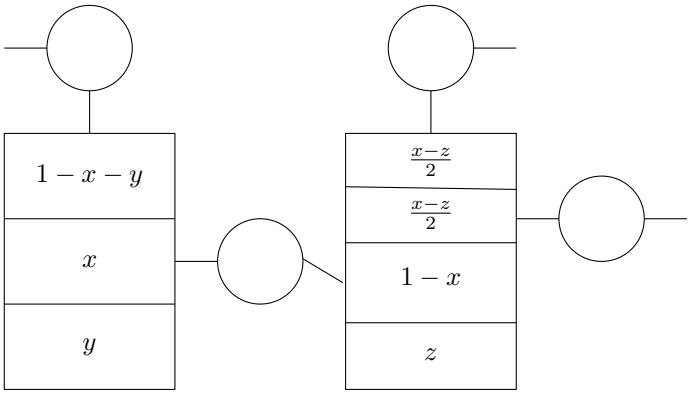

Figure 4.9: Fractional allocation with a neighbor from R_3

is $1-x(1-\frac{x}{a})^a$. The weighted average of the cluster (including matched bidders from R_1), is at least

$$\frac{\frac{1}{2}(1-(1-x)(\frac{1+x}{2})^2)+\frac{a-2}{a}(1-(\frac{1}{2})^2)+\frac{1}{a}(1-x(1-\frac{x}{a})^a)}{\frac{1}{2}+\frac{a-1}{a}}\ . \tag{4.18}$$

The values of x which minimize (4.18) are given in Table 4.5, showing that the target approximation ratio is reached for R_2, R_3 and R_4.

R_a	x	Average	1/Average
R_2	0.5000	0.7188	1.3913
R_3	0.4833	0.7252	1.3788
R_4	0.4649	0.7284	1.3728

Table 4.5: Implied approximation ratios for neighbor bidders

Neighbor bidders from R_1: The case where a bidder i from R_2 shares an item j from S_2 with a bidder i' from R_1 is special since it is possible that these bidders cannot be clustered together, because the bidder from R_1 may be already matched to another bidder. However, even if such a match exists, the bidder from R_1 contributes an expected valuation of $\frac{3}{4}$ to that match, and therefore any excess valuation can be transferred to its neighbor bidder from R_2.

Similarly to the previous case, where the neighbor bidder is from R_2, R_3 or R_4, we can concentrate on the case where all valuation of fractionally allocated items are 1. In this case, if $x_{ij}=x$ and $x_{i'j}=1-x$, then bidder i' also fully owns an item with value x. The worst case values for the other items held by i are as in the previous case, i.e. a fully owned item with value $\frac{1-x}{2}$, and a $\frac{1-x}{2}$ fraction of an item with value 1. Figure 4.10 illustrates this structure.

The expected valuation of bidder i is $1-(1-x)(\frac{1+x}{2})^2$. Bidder i' contributes an additional expected valuation of $1-x(1-x)-\frac{3}{4}$. Using a weight of $\frac{1}{2}$ for bidder i, we

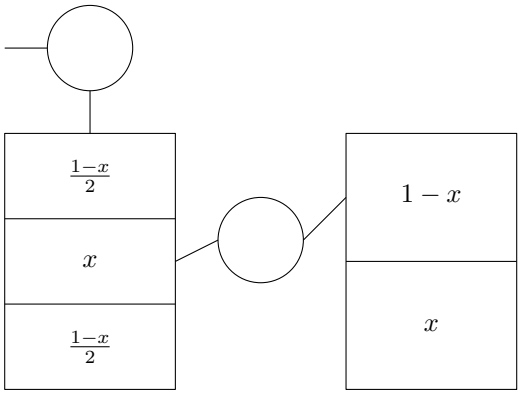

Figure 4.10: Fractional allocation with a neighbor from R_1

have in average:

$$\frac{\frac{1}{2}(1-(1-x)(\frac{1+x}{2})^2)+1-x(1-x)-\frac{3}{4}}{\frac{1}{2}} = \frac{5-9x+9x^2+x^3}{4} \tag{4.19}$$

The expression in (4.19) is minimized for $2\sqrt{3}-3 \approx 0.4641$, and equals to $0.7154 \approx 1.3978^{-1}$, slightly missing the target bound. However, an average with the other cluster that bidder i participates in would almost always suffice to reach the target bound, as demonstrated in Table 4.6. One case that does not appear in Table 4.6 is when bidder i shares both of its fractionally assigned items with neighbor bidders from R_1. However, in this case there are only three bidders, and therefore non of the bidders are already matched, and we can simply average between the expected valuation of the bidder from R_2 and one of the bidders from R_1,which easily reaches the target bound.

As can be seen from Table 4.6, there is still one exception where the target bound is slightly missed: If a bidder i from R_2 shares one item with a bidder from R_1 and another item with a bidder from R_3 who only owns fractions of items, we still need a stronger argument. Note that the bound is constructed from two worst case bounds that cannot occur simultaneously: To minimize (4.19), the fraction of the item that bidder i shares with its neighbor from R_1 is $2\sqrt{3}-3 \approx 0.4641$, while to minimize the average expected valuation in the cluster containing the bidder from R_3, the value $\frac{1}{3}$ is taken for the same fraction. If we force a single fraction size through the whole computation, then instead of (4.19), the expected valuation of bidder i, along with the excess valuation of the bidder from R_1, is

$$1-(1-x)\left(\frac{1+x}{2}\right)^2+1-x(1-x)-\frac{3}{4} = \frac{4-5x+5x^2+x^3}{4} \tag{4.20}$$

This expectation is minimized at $x = \frac{2\sqrt{10}-5}{3} \approx 0.4415$ and equals $0.7187 = 1.402^{-1}$. Averaging with the worst case expected valuation of a bidder from R_3 with only fractional

Table	Row	Weight	Orig. Avg.	New Avg.	1/Average
4.2	S_a	$1/2 + (a-2)/a$	(4.12)	-	-
4.2	S_3	5/6	0.7222	0.7197	1.3895
4.2	S_4	1	0.7269	0.7230	1.3831
4.2	S_5	11/10	0.7290	0.7247	1.3798
4.2	S_6	7/6	0.7302	0.7257	1.3779
4.3	R_a	$1/2 + (a-1)/a$	(4.13)	-	-
4.3	R_5	13/10	0.7191	0.7181	1.3926
4.3	R_6	4/3	0.7214	0.7198	1.3894
4.4	R_a	$1/2 + (a-1)/a$	(4.16)	-	-
4.4	R_2	1	0.7269	0.7230	1.3831
4.4	R_3	7/6	0.7169	0.7165	1.3957
4.4	R_4	5/4	0.7182	0.7174	1.3939
4.5	R_a	$1/2 + (a-1)/a$	(4.18)	-	-
4.5	R_2	1	0.7188	0.7176	1.3934
4.5	R_3	7/6	0.7252	0.7223	1.3845
4.5	R_4	5/4	0.7284	0.7247	1.3798

Table 4.6: Implied approximation ratios for bidders from R_2 with a neighbor from R_1. The weight column is the total weight used in calculating the original weighted average of the cluster. For the combined cluster, the average is given by New Avg. = (Orig. Avg. $\times$ Weight $+ 0.7154 \times 1/2)/($Weight $+ 1/2)$.

items and its matched bidder from R_1 (both taken with weight $\frac{1}{3}$), we get $0.7187 = 1.3914^{-1}$ and reach the target bound.

It remains to verify that the target bound can be reached even when our assumption that all budgets are reached and all items are fully assigned, is violated. By Observation 4.3.10, we may assume that there is only one bidder or item for which this assumption may not hold.

One item isn't fully assigned: The results summarized in Tables 4.5 and 4.6 rely on the assumption that certain items were fully allocated. Assume that there exists an item j with a fraction of size δ that wasn't allocated in the fractional allocation. The simplest solution to this difficulty is to arbitrarily allocate the δ fraction of item j to a bidder i who already has a fraction of the j-th item. If we then decrease the bid b_{ij} to $\frac{b_{ij}x_{ij}}{x_{ij}+\delta}$, the valuation of the i-th bidder in the fractional allocation remains 1, while the expected valuation of the integral allocation decreases, thus there exists a setting where the j-th item is fully allocated, and the expected valuation lower bounds the original valuation.

One bidder did not reach its budget constraint: If in the fractional assignment, a bidder i from R_a is allocated less than its budget, then the ratio between its expected valuation of the integral assignment and its valuation of the fractional assignment is still

at least $1-(a/(a+1))^{a+1}$. However, if bidder j participates in any clusters, its total weight would be its fractional valuation, which is less than 1, and it would have less effect on the average valuation compared to other bidders. This is only a problem if bidder j is from R_1, which is the only case where the ratio is guaranteed to be below the target of 1.3951^{-1}. Additionally, the analysis leading to the results in tables 4.5 and 4.6 strictly assumes that the budgets are reached. Therefore, new arguments need to be supplied when bidder i is in R_a, $a \in \{1,2,3,4\}$.

If bidder i is in R_1, then clearly, it is one of the two bidders that by Lemma 4.3.15 are redundant in the matching process. Therefore, there is another unmatched bidder i' from R_1, that did reach its budget constraint. Since it is unmatched, there can only be a problem with bidder i if its fractionally allocated item is from S_2 and is shared with a bidder i'' from R_2. By adding i' to the cluster containing i and i'' we easily reach the target bound.

If bidder i is in R_a, where $a \in \{2,3,4\}$, then there are potentially a bidders from R_2 who form clusters with it, as well as $a-2$ matched bidders from R_1. We can improve the weighted average ratio by adding to the cluster the two redundant unmatched bidders from R_1. As for any $a > 1$, the bound on this ratio is maximized when bidder i actually does reach its budget constraint, and equals:

$$\frac{1-(\frac{1}{2})^2+1-(\frac{2}{3})^3+\frac{1}{a}(1-(1-\frac{1}{a+1})^{a+1})}{2+\frac{1}{a}} \tag{4.21}$$

The weights of the bidders from R_2 used in (4.21) are 1 rather than $\frac{1}{2}$, in order to override a similar difficulty to the one solved in Table 4.6, where the bidders also share an item with matched bidders from R_1. Table 4.7 shows that all of the relevant values of (4.21) reach the target bound.

R_a	Matched R_1	Average	1/Average
R_2	2	0.7222	1.3846
R_3	3	0.7207	1.3876
R_4	4	0.7208	1.3874

Table 4.7: Implied approximation ratios with redundant bidders from R_1

■

The following lower bound nearly matches the upper bound:

Theorem 4.3.18 *The approximation ratio of Algorithm* RR *with identical budget constraints is at least* 1.3837

Proof: We define an allocation instance by directly constructing a fractional assignment, which we later argue that it is optimal. All budget constraints are set to 1. All bids for items with fractional assignments are also 1. All bids made by bidders to items that are not allocated to them are 0.

The bidders are arranged in a rooted tree, where the bidder nodes are arranged in m layers, and item nodes connect bidders from the same layer or adjacent layers. The tree

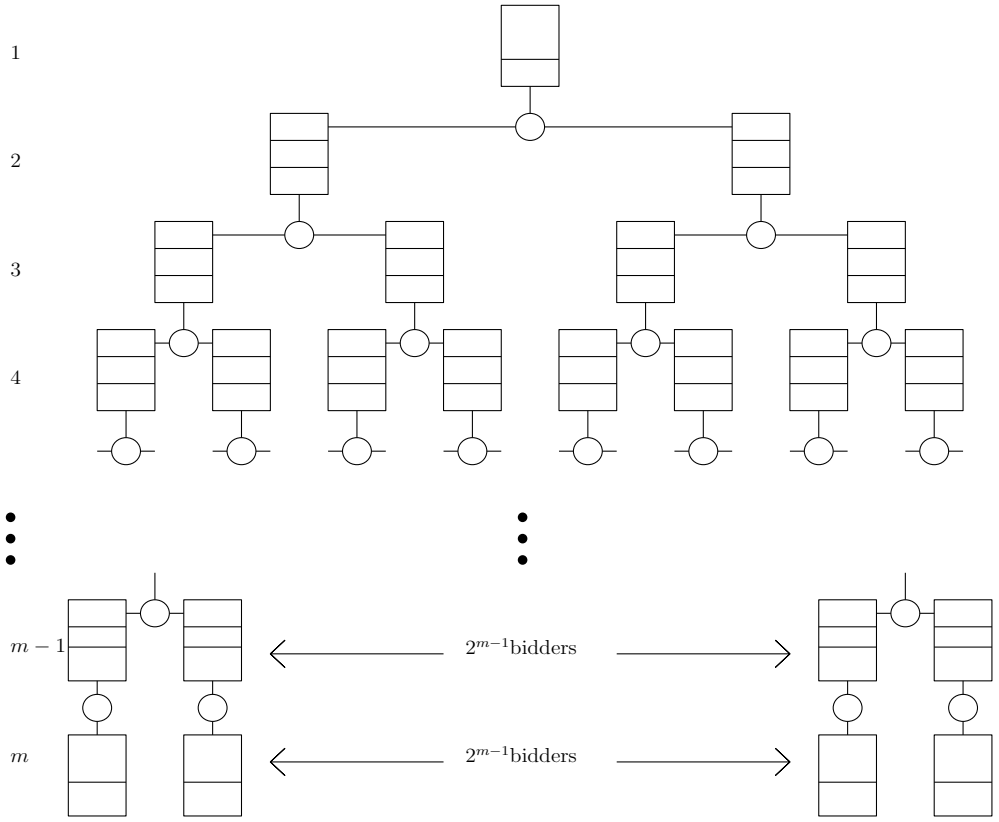

Figure 4.11: Graphical representation of the lower bound construction

has $3 \cdot 2^{m-2} - 1$ bidder nodes. The tree is symmetrical in the sense that bidders in the same layer of the tree have the same bids and fractional assignments, but for different items.

The bidder at the root layer fully owns one item with value $\frac{2}{3}$, and shares a $\frac{1}{3}$ fraction of an item with its two bidders at layer 2.

The nodes at layers 2 to $m-3$ represent $2^{m-3} - 2$ bidders. In layer i there are 2^{i-1} bidders. Each bidder in layer i holds fully one item with value $\frac{1}{3}$, a $\frac{1}{3}$ fraction of an item shared with two other bidders from layers $i-1$ and i, and a $\frac{1}{3}$ fraction of an item shared with two bidders at layer $i+1$.

At layer $m-2$ there are 2^{m-3} bidders. Each bidder holds one item with value $\frac{2}{3} - 0.364 \approx 0.30267$, a $\frac{1}{3}$ fraction of an item shared with two bidders from layers $m-3$ and $m-2$, and a 0.364 fraction of an item shared with its two bidders at layer $m-1$.

Layer $m-1$ consists of 2^{m-2} bidders. Each bidder holds one item with value 0.2465, a 0.318 fraction of an item shared with bidders from layers $m-1$ and $m-2$, and a 0.4355 fraction of an item shared with one bidder at the next layer m.

Each of the 2^{m-2} bidders at layer m holds one item with value 0.4355, and a 0.5645 fraction of an item shared with a bidder from level $m-1$. Figure 4.11 illustrates the structure of the layers.

The fractional assignment is optimal since all bidders satisfy their budget constraint. It is also the only optimal assignment and therefore is the solution of the LP. The total valuation of all bidders is $3 \cdot 2^{m-2} - 1$. The expected valuation for RR is: $\frac{7}{9}$ for the root bidder, $0.7037(2^{m-3} - 2)$ for bidders at layer 2 to $m-3$, $0.70433 \cdot 2^{m-3}$ for bidders at layer $m-2$, $0.70991 \cdot 2^{m-2}$ for layer $m-1$ and $0.75416 \cdot 2^{m-2}$ for the bottom bidders at layer m. As m goes to infinity, the average expected valuation reaches 0.7227, yielding a lower bound of $0.7227^{-1} = 1.3837$. ∎

4.3.6 Fractional Versus Integral Allocations

To conclude this section we derive a general lower bound for algorithms that are based on solving the LP and rounding the fractional assignments. The following theorem proves a lower bound for any algorithm that uses the relaxed LP.

Theorem 4.3.19 *LP has an integrality ratio of at least $\frac{4}{3}$. Also, the optimal solution of the IP can be $\frac{4}{3}$ times the optimal rounding of the LP.*

Proof: The integrality ratio if $\frac{4}{3}$ is achieved in the following case: Observe the following auction with 2 bidders, A and B, and 3 items x, y and z. Bidder A bids 1 for x, 0 for y and 2 for z and has a budget constraint of 2. Bidder B bids 0 for x, 1 for y and 2 for z and has a budget constraint of 2. Optimally, A get x, B gets y and either bidder gets z, and the revenue is 3. However, the LP produces an optimal fractional assignment, which divides z between both bidders, and achieves a revenue of 4.

The ratio of $\frac{4}{3}$ between the optimal integral solution and the optimal rounding of the LP solution is achieved in a similar auction, but now A bids 1, $1-\epsilon$ and 2 for x, y and z, respectively. B bids $1-\epsilon$, 1 and 2 for x, y and z, respectively. Both budget constraints remain 2. Optimally, A gets both x and y while B gets z (or vice versa), and the revenue is $4-\epsilon$. However, the fractional assignment that the LP will return allocates x to A, y to B, and divides z between both bidders. The revenue is 4, but any rounding technique will either grant z to A or to B, either way the revenue is 3, achieving a ratio that is asymptotically $\frac{4}{3}$. ∎

Bidder	x	y	z	Budget
A	1	0	2	2
B	0	1	2	2

Bidder	x	y	z	Budget
A	1	$1-\epsilon$	2	2
B	$1-\epsilon$	1	2	2

Table 4.8: Two settings leading to approximation ratio 4/3.
The left setting leads to an optimal integer solution, yet the integrality ratio is 4/3, while the right setting leads to a rounded solution that is asymptotically a 4/3-approximation of the optimal.

Chapter 5

Mechanisms for Single Parameter Agents

The budget constrained auction model in Chapter 4 is an example of a non-strategic approach to an optimization problem, where the entire input is publicly known to the algorithm, including the personal preferences of the participating bidders. In contrast, a strategic approach assumes that some of the information concerning the bidders' preferences is private information, and the algorithm's execution is based on bids that may not represent truthfully the actual preferences. The following chapter presents a simple sufficient condition for identifying when mechanism design is an applicable solution for such strategic settings, given that the bidders' private information type is one dimensional.

5.1 Preliminaries

5.1.1 Mechanisms, Agents and Bids

A mechanism is a combination of a decision rule or an algorithm[1] A, and a payment scheme. The input to the mechanism is received from n agents, who each submit a bid $b_i \in T$ to the mechanism. The aggregated vector of all bids is denoted by b. The output of the mechanism is constructed from two parts: An outcome $A(b) = \omega \in \Omega$, and a payment for each agent. We note that payments may be negative, meaning that the mechanism pays the agents. These terms are formally defined as follows:

Definition 5.1.1 *A* decision rule *is a function $A : T^n \mapsto \Omega$ that given a vector b of n bids returns an outcome $\omega \in \Omega$.*
A payment scheme *P is a set of payment functions $P_i : T^n \mapsto \mathbb{R}$, where P_i determines the payment of agent i to the mechanism, given the output ω and the bid vector b.*
A mechanism *$M = (A, P)$ is a combination of a decision rule A and a payment scheme P.*

[1]This Chapter characterizes decision rules with desired properties, where our goal later is to design algorithms that calculate them. Therefore, we use the terms decision rule and algorithm interchangeably.

The bid vector without the i-th bid is denoted by b_{-i}. Additionally, (b_{-i}, β) denotes the bid vector b with the i-th bid replaced with β. When it is clear from the context that A and b_{-i} are fixed, we shall let $\omega_{b_i} = A(b_{-i}, b_i)$ denote the outcome when agent i bids b_i.

5.1.2 Utilities

Each agent has a private value $t_i \in T_i$, which is called its *type*. To simplify the notation, we will assume that all agents have their types taken from the same space as the bids space, T.

Each agent has a valuation function $v_i : \Omega \times T \mapsto \mathbb{R}$ that reflects the utility from an outcome $\omega \in \Omega$, given that the type of the agent is t_i. The agents have a *quasi-linear* utility, meaning that their utility can be shifted linearly by monetary payments. Therefore, the utility of agent i from an outcome ω and a payment p_i is $u_i(\omega, t_i, p_i) = v_i(\omega, t_i) - p_i$.

In some cases we will be interested in the partial derivative of a valuation function by the agent's type. Therefore, for simplicity, we denote $v_i' = \frac{\partial v_i}{\partial t_i}$.

5.1.3 Truthfulness

Definition 5.1.2 *Algorithm A* admits a truthful payment *if and only if there exists a payment scheme P such that for any set of fixed bids b_{-i}, and for any two types $s, t \in T$*

$$v_i(\omega_t, t) - P_i(b_{-i}, t) \geq v_i(\omega_s, t) - P_i(b_{-i}, s) \tag{5.1}$$

In this case, A is also called rationalizable*, and the mechanism $M = (A, P)$ is called* truthful.

In words, a mechanism is truthful if for every agent, reporting its true type as a bid is a weakly dominant strategy. A mechanism is *strongly truthful* if truthtelling is the only dominant strategy.

For truthful mechanisms we will concentrate on payment functions P_i that depend only indirectly on the i-th bid, through the outcome ω_{b_i}, since only these types of mechanisms are truthful. If the dependence is direct, then there are two bids (b_{-i}, t) and (b_{-i}, s) such that $\omega_s = \omega_t$ yet $P_i(b_{-i}, t) > P_i(b_{-i}, s)$. Such a mechanism cannot be truthful since the i-th agent will prefer bidding s over t to reduce its payment, even if its real type is t.

If an algorithm A is randomized, then it can be viewed as a distribution over deterministic algorithms. Therefore, a randomized mechanism $M = (A, P)$, which is a distribution over truthful mechanisms is also truthful (this property is also referred as *universal truthfulness*). A weaker notion of truthfulness is to carry the randomization into the mechanism, and require truthtelling to always be a best response strategy in expectation over the random choices of the algorithm. A mechanism of this type is *truthful in expectation*.

Rochet [67] presented a necessary and sufficient condition for rationalizable decision rules. We refer to a slightly different presentation of this condition, which appears in [40]:

Theorem 5.1.3 *[67, 40] Given an agent i and having all other bids b_{-i} held fixed, let $G(i, b_{-i}) = (V, E)$ be a weighted directed graph such that $V = T$, $E = T \times T$ and the weight of every edge is $w(s,t) = v_i(\omega_t, t) - v_i(\omega_s, t)$. An algorithm is rationalizable if and only if for every agent i and for every vector of fixed bids b_{-i}, there are no cycles with a finite number of arcs and a negative weight in the graph $G(i, b_{-i})$.*

In addition to Theorem 5.1.3, if an algorithm is rationalizable, the following is a suitable payment function for the i-th agent, as proved in [67, 40]: For every vector of fixed bids b_{-i}, choose an arbitrary type t_0. The payment from agent i to the mechanism if it bids t is:

$$P_i(b_{-i}, t) = \inf \left\{ \sum_{j=0}^{k} w(t_j, t_{j+1}) | k \geq 0, \quad \begin{array}{l} t_1, \ldots, t_{k+1} \in T \\ t_{k+1} = t \end{array} \right\} \tag{5.2}$$

In words, the payment function is the infimum on the length all finite paths from an arbitrary type t_0 to the actual bid t.

Theorem 5.1.3 does not provide an efficient computational method for testing whether an algorithm is rationalizable. A simpler condition, which should be easier to test, is whether the graph contains a cycle with two arcs and a negative weight. Formally, a graph $G(i, b_{-i})$ does not have negative cycles of length 2 if and only if for every two types $t, s \in T$,

$$v_i(\omega_t, t) - v_i(\omega_s, t) \geq v_i(\omega_t, s) - v_i(\omega_s, s) \tag{5.3}$$

This necessary (but not sufficient) condition is referred as *weak monotonicity* [53] or as *2-cycle inequality* [40].

5.1.4 Single Parameter

Definition 5.1.4 *An agent i is a* single parameter *agent with respect to Ω if there exists an interval $S_i \subseteq \mathbb{R}$ and a bijective transformation $r_i : T \mapsto S_i$ such that for any $\omega \in \Omega$, the function $\hat{v}_i(\omega, s_i) = v_i(\omega, {r_i}^{-1}(s_i))$ is continuous and differentiable almost everywhere in s_i.*

The purpose of $r_i(\cdot)$ is to overcome the difficulty of having different representations for the same type space. The single parameter property should be indifferent to the chosen representation. For simplicity, we shall assume the present type representation allows ignoring $r_i(\cdot)$. For example, if the type space is $\mathbb{R}$ itself, or an interval $T = [t_0, t_1]$, and the utility function is continuous and differentiable almost everywhere in T for every $\omega \in \Omega$, then $r_i(\cdot)$ is the identity function and can be ignored. We therefore, slightly abuse the notation and assume $v_i = \hat{v}_i$.

5.2 Halfway Monotone Derivative

The *Mirrlees-Spence condition* [73, 61, 67] can be used to characterize all truthful single parameter mechanisms, where the notion of single parameter is more restricted than the

one we use. Assume the output space Ω is a continuous interval in $\mathbb{R}$, the type space T is a segment $T = [t_0, t_1]$. The condition requires that v_i is twice differentiable, and that

$$\forall \omega \in \Omega, t \in T \qquad \frac{\partial^2}{\partial t \partial \omega} v_i(\omega, t) > 0 \tag{5.4}$$

then the mechanism is truthful if and only if ω is non-decreasing in the bid b_i of the agent.

Our condition is a generalization of this notion of the Mirrlees-Spence condition, as it relaxes the assumptions on Ω and the differentiability of the valuation and decision functions. It is a sufficient but not necessary condition, which we call *Halfway Monotone Derivative (HMD)* condition.

Definition 5.2.1 *A valuation function v_i satisfies the* Halfway Monotone Derivative *(HMD) condition with respect to a given decision rule if for every fixed bid vector b_{-i},* one of the following conditions *holds for every two types $s, t \in T$ such that $s < t$, except for a set of measure zero:*

1. *$\forall u \geq s$, we have $v_i'(\omega_s, u) \leq v_i'(\omega_t, u)$, or*

2. *$\forall u \leq t$, we have $v_i'(\omega_s, u) \leq v_i'(\omega_t, u)$.*

Note that for a given vector b_{-i}, the same condition should hold for every pair of types s and t. The set of points where the inequality may not hold is required since the valuation function may not be differentiable everywhere. Note that by integrating the derivatives in the definition over $[s, t]$, HMD also implies weak monotonicity.

The following condition gives a characterization to a family of rationalizable decision rules.

Theorem 5.2.2 *A single parameter decision rule $A(b) : T^n \mapsto \Omega$ is rationalizable when all valuation functions satisfy the HMD condition.*

Proof: We shall prove only that the first condition of HMD leads to rationalizability, as the proof for the second condition is analogous. Assume by way of contradiction that the condition holds yet A is not rationalizable. By Theorem 5.1.3 there is a some graph $G(i, b_{-i})$ which has a negative cycle $t_0, t_1, \ldots, t_k, t_{k+1} = t_0$. We shall first show that the graph must also contain a negative 2-cycle, and later infer that the HMD condition is violated.

If $k = 1$ then clearly a negative 2-cycle exists. Otherwise, let t be the minimal type node in the cycle[2], i.e. $\forall 0 \leq i \leq k,\ t \leq t_i$. Let s and u be the adjacent nodes in the cycle which have cycle arcs into and from t, respectively. Since t has the minimal type in the

[2]In case the second HMD condition holds, we select t to be the maximal type node.

cycle, we have $t \leq u$ and $t \leq s$. The length of the path from s to u through t is:

$$\begin{aligned}
&w(s,t) + w(t,u) \\
&\quad = v_i(\omega_t, t) - v_i(\omega_s, t) + v_i(\omega_u, u) - v_i(\omega_t, u) \\
&\quad = v_i(\omega_s, u) - v_i(\omega_s, t) - v_i(\omega_t, u) + v_i(\omega_t, t) - v_i(\omega_s, u) + v_i(\omega_u, u) \\
&\quad = \int_t^u v_i'(\omega_s, x)dx - \int_t^u v_i'(\omega_t, x)dx - v_i(\omega_s, u) + v_i(\omega_u, u) \\
&\quad = \int_t^u \left(v_i'(\omega_s, x) - v_i'(\omega_t, x)\right) dx + w(s,u) \geq w(s,u)
\end{aligned}$$

The last integral is non-negative since $t \leq u$ and since $v_i'(\omega_s, x) \geq v_i'(\omega_t, x)$ for every $x \geq t$ (in particular, $t \leq x \leq u$), due to the first HMD condition. From the inequality we infer that a shorter negative cycle with $k-1$ nodes can be constructed with a shortcut from s to u. By induction, we get that the graph has a negative 2-cycle. Let s and u be the nodes in this cycle, and assume without loss of generality that $s < u$. We infer from HMD, that:

$$\begin{aligned}
&w(s,u) + w(u,s) \\
&\quad = \quad v_i(\omega_u, u) - v_i(\omega_s, u) + v_i(\omega_s, s) - v_i(\omega_u, s) \\
&\quad = \quad \int_s^u v_i'(\omega_u, x)dx - \int_s^u v_i'(\omega_s, x)dx \\
&\quad = \quad \int_s^u (v_i'(\omega_u, x) - v_i'(\omega_s, x))dx \geq 0,
\end{aligned}$$

which contradicts the cycle being of negative length. ∎

For some types of valuation functions, HMD is also a necessary condition. The following Theorem proves this property for a simple case where the partial derivative is a constant in the agent's type.

Theorem 5.2.3 *If for every agent i, fixed bid vector b_{-i}, and bid b_i, $v_i'(\omega_{b_i}, x)$ does not depend on x, then HMD is a necessary and sufficient condition for rationalizability.*

Proof: Sufficiency is by Theorem 5.2.2, thus it remains to prove necessity. Assume by the way of contradiction that HMD is violated. Then there is an agent i, a bid vector b_{-i} and types $s < t$ such that ${v_i}'(\omega_s, x) > {v_i}'(\omega_t, x)$ for some x. Since both functions don't depend on x, the inequality holds for every x, and specifically in the interval $[s,t]$. By integrating both sides of the inequality, we get:

$$\int_s^t {v_i}'(\omega_s, x)dx \quad > \quad \int_s^t {v_i}'(\omega_t, x)dx \tag{5.5}$$

$$v_i(\omega_t, s) - v_i(\omega_s, s) \quad > \quad v_i(\omega_t, t) - v_i(\omega_s, t) \tag{5.6}$$

Which is a violation of weak monotonicity. Hence, the decision rule cannot be rationalizable. ∎

In Section 5.3.1 as well as in Chapter 7 we demonstrate the importance of Theorem 5.2.3 by reproving classical truthfulness results as special cases of the theorems 5.2.2 and 5.2.3. In contrast to this result, we present in Section 6.5 an example of a truthful mechanism where neither HMD condition holds.

We now show a simple structure for the payment function in truthful HMD mechanisms:

Theorem 5.2.4 *A suitable payment scheme for agent i in a single parameter rationalizable decision rule $A : T^n \mapsto \Omega$ that is HMD is*

$$P_i(b_{-i}, t) = c(b_{-i}) + v_i(\omega_t, t) - \int_{t_0}^{t} v_i'(\omega_x, x)dx \tag{5.7}$$

Where b_{-i} is held fixed, $t_0 \in T$ is an arbitrary type and c is an arbitrary function of b_{-i}.

Proof: The following proof holds for the first HMD condition. The proof for the second HMD condition is analogous. We first prove the Theorem for $t_0 = \inf T$,[3] and later extend the result for any arbitrary $t_0 \in T$. For now we assume that $\inf T \in T$. At the end of the proof we refer to the case where $\inf T \notin T$.

By Theorem 5.1.3, a suitable payment function is

$$P_i(t, b_{-i}) = \inf \left\{ \sum_{j=0}^{k} w(t_j, t_{j+1}) | k \geq 0, \quad \begin{matrix} t_1, \ldots, t_{k+1} \in T \\ t_{k+1} = t \end{matrix} \right\} \tag{5.8}$$

We call an arc (t_j, t_{j+1}) a *forward arc* if $t_j < t_{j+1}$ and a *backward arc* if $t_j > t_{j+1}$. We first prove that within a path, adding an intermediate node inside a forward arc cannot make it longer. Let (t_j, t_{j+1}) be a forward arc in a given path. Let $s \in [t_j, t_{j+1}]$ be an intermediate node. We have that

$$\begin{aligned}
&w(t_j, s) + w(s, t_{j+1}) \\
&\quad = v_i(\omega_s, s) - v_i(\omega_{t_j}, s) + v_i(\omega_{t_{j+1}}, t_{j+1}) - v_i(\omega_s, t_{j+1}) \\
&\quad = -\int_s^{t_{j+1}} v_i'(\omega_s, x)dx + \int_s^{t_{j+1}} v_i'(\omega_{t_j}, x)dx + v_i(\omega_{t_{j+1}}, t_{j+1}) - v_i(\omega_{t_j}, t_{j+1}) \\
&\quad = \int_s^{t_{j+1}} \left(-v_i'(\omega_s, x) + v_i'(\omega_{t_j}, x)\right) dx + w(t_j, t_{j+1}) \leq w(t_j, t_{j+1})
\end{aligned}$$

The last inequality holds since the integrated expression is negative within the integral's range, due to the first HMD condition. For the first forward arc leaving the interval $[t_0, t]$ (if such an arc exists) we add an intermediate node at t. The resulting path is not longer, and all the nodes outside the interval $[t_0, t]$ are contained in a cycle, starting and ending at t (as the entire path ends at t). Since there are no negative cycles in the graph, we can

[3] In case the second HMD condition holds, we select $t_0 = \sup T$.

remove the cycle and end with a shorter path that uses only nodes within the interval $[t_0, t]$.

In a similar technique we remove all the backward arcs (t_j, t_{j+1}) such that $t_{j+1} < t_j$. Let $l > j$ be the index of the first node t_l that appears after t_{j+1} such that $t_l \geq t_j$. The arc that ends at t_j must be a forward arc (i.e. $t_{l-1} < t_l$), since by definition of l, $t_{l-1} \leq t_j$. By adding t_j as an intermediate node within the arc (t_{l-1}, t_l), we get a cycle from t_j to itself, which can be removed. Removing the cycle also removes the backward arc (t_j, t_{j+1}).

We now have that the payment is the infimum on paths from t_0 to t that only use forward arcs. By adding intermediate points to all arcs, we get that

$$\begin{aligned}
&P_i(b_{-i}, t) \\
&\quad = \lim_{\Delta \to 0} \sum_{j=t_0/\Delta}^{t/\Delta} w(j\Delta, (j+1)\Delta) \\
&\quad = \lim_{\Delta \to 0} \sum_{j=t_0/\Delta}^{t/\Delta} (v_i(\omega_{(j+1)\Delta}, (j+1)\Delta) - v_i(\omega_{j\Delta}, (j+1)\Delta)) \\
&\quad = v_i(\omega_t, t) - v_i(\omega_{t_0}, t_0) - \lim_{\Delta \to 0} \sum_{j=t_0/\Delta}^{t/\Delta} (v_i(\omega_{j\Delta}, (j+1)\Delta) - v_i(\omega_{j\Delta}, j\Delta)) \\
&\quad = v_i(\omega_t, t) - v_i(\omega_{t_0}, t_0) - \lim_{\Delta \to 0} \sum_{j=t_0/\Delta}^{t/\Delta} \int_{j\Delta}^{(j+1)\Delta} v_i'(\omega_{j\Delta}, x) dx \\
&\quad = v_i(\omega_t, t) - v_i(\omega_{t_0}, t_0) - \int_{t_0}^{t} v'(\omega_x, x) dx
\end{aligned}$$

Since $v_i(\omega_{t_0}, t_0)$ does not depend on t it can be replaced by any other function $c(b_{-i})$ without affecting the truthfulness, as this only shifts all payments by a constant.

Additionally, the integral can be changed to $\int_{\hat{t}}^{t} v'(\omega_x, x) dx$, where $\hat{t}$ is an arbitrary type, since this change only subtracts $\int_{t_0}^{\hat{t}} v'(\omega_x, x) dx$ from the payment, which does not depend on t and therefore is part of $c(b_{-i})$.

If $t_0 = \inf T \notin T$, we can naturally add t_0 to T by setting $v_i(\omega, t_0) = \lim_{t \to t_0} v_i(\omega, t)$ and $v_i(\omega_{t_0}, s) = \lim_{t \to t_0} v_i(\omega_t, s)$. If these limits are not well defined we can settle for any arbitrary values. Since t_0 is not a legal bid nor a legal type, these values affect neither the rationalizability of the algorithm, nor the value of the payment function. ∎

We describe a large class of cases where the integral in Theorem 5.2.4 reduces into an alternative representation that may be more convenient to implement. In many combinatorial problems the outcome space Ω is finite or countable, and combinatorial algorithms have the following property: For each agent i, when b_{-i} is fixed, the type space T can be divided into a finite or countable number of intervals, such that for each interval, the algorithm outputs the same outcome for any bid within the interval. We denote such algorithms as *piecewise continuous*. Let $T_{b_{-i}}$ denote the set of endpoints of these intervals. For any endpoint $s \in T_{b_{-i}}$, let s^- and s^+ denote the intervals whose upper and lower

endpoints, respectively, are s. Clearly, for any endpoint s, either $s \in s^-$ or $s \in s^+$. Given an interval $[t_1, t_2]$, $T_{b_{-i}} \cap [t_1, t_2]$ denotes the set of endpoints whose surrounding intervals intersect with $[t_1, t_2]$ (i.e., all the endpoints within (t_1, t_2), and possibly t_1 and t_2, if they are endpoints and $t_1 \in t_1^+$, $t_2 \in t_2^-$).

Corollary 5.2.5 *A suitable payment scheme for agent i in a single parameter piecewise continuous rationalizable decision rule $A : T^n \mapsto \Omega$ that is HMD is*

$$P_i(b_{-i}, t) = c(b_{-i}) + \begin{cases} \sum_{s \in T_{b_{-i}} \cap [t_0, t]} v_i(\omega_{s^+}, s) - v_i(\omega_{s^-}, s) & t > t_0 \\ \sum_{s \in T_{b_{-i}} \cap [t, t_0]} v_i(\omega_{s^-}, s) - v_i(\omega_{s^+}, s) & t < t_0 \\ 0 & t = t_0 \end{cases}, \tag{5.9}$$

where b_{-i} is held fixed, $t_0 \in T$ is an arbitrary type.

In words, the non constant part of the payment is given by accumulating the values of the transitions between two consecutive outcomes, over all the endpoints between t_0 and t.

Example: To illustrate the payment function in Corollary 5.2.5 consider the following case for player i, when all the other bids b_{-i} are fixed. Consider a decision rule that for any type $t \in \mathbb{R}^+$ outputs $A(x, b_{-i}) = \omega_{\lceil x \rceil}$, and consider a valuation $v_i(\omega_{\lceil x \rceil}, t) = t + \lceil x \rceil$.

Choosing $t_0 = 0$ and $c(b_{-i}) = 0$, the payment would be $P_i(x) = \lceil x \rceil$. If we change the valuation function to $v_i(\omega_{\lceil x \rceil}, t) = t \cdot \lceil x \rceil$ then the payment would be $P_i(x) = \sum_{k=0}^{\lceil x \rceil - 1} k = O(x^2)$.

5.3 HMD Applications

Following are a few applications of the HMD condition for several single parameter mechanisms. In Section 5.3.1 we rederive well known results by applying the HMD condition on the extensively studied single commodity auction, and show that it is actually a special case of Theorem 5.2.3. In Section 5.3.2 we present a new single parameter mechanism and demonstrate the usage of the HMD condition on it. Further usage of the HMD condition is given in Chapters 6 and 7.

5.3.1 Single Commodity Auction

In an auction, an agent's type is a private value t_i, which is the value the agent associates with a desired outcome. for any other outcome, the agent's value is 0, regardless of its type.

For simplicity, we concentrate on single commodity auctions, where there is only one item for sale. However, the results hold for any setup where agents divide the outcomes into winning outcomes where they gain a value of t_i, and losing outcomes where their value is 0, as long as the structure of the division is public knowledge. For example, an auction with known single minded bidders [62], is a multi commodity combinatorial auction where

each agent is interested in a specific bundle of commodities, which is publicly known. However, if the structure of the bundle is part of the agent's type, as in [56], then the valuations are no longer single parametered.

The second price auction [76] is a classical method for auctioning a single commodity: The highest bidder wins and pays the bid of the second highest bid. Although it is a popular example, it is not the only truthful mechanism for auctioning.

A well known result on rationalizable deterministic auctions is the existence of a *critical value* for each bidder, unless the bidder has no winning bid. A bidder's critical value is determined by the bids of the other agents. The bidder wins if its bid is above the critical value and loses if the bid is under it. Given that losing bidders pay 0, the winning bidders pay exactly their critical value. The second price auction is truthful since the critical value of each bidder is the highest bid among the other bidders.

The HMD condition is equivalent to the critical value condition in auctions, and extends to randomized auctions:

Claim 5.3.1 *In deterministic auctions the critical value condition is equivalent to HMD.*

Proof: In a deterministic auction, when winning, the value of the i-th agent is t_i, and the derivative is 1. In the losing outcome, the value is the constant 0, and so is the derivative. Since both derivatives are constant, by Theorem 5.2.3, HMD is a necessary and sufficient condition.

For any type t_i the derivative of the winning outcome is higher than the losing outcome. Having b_{-i} fixed, all deterministic HMD mechanisms must either decide that the i-th agent never wins, or have a critical value c_i, for which the i-th agent loses if it's type is lower than c_i, and wins if it is higher (in case of equality, both outcomes are acceptable). By Corollary 5.2.5, the payment in the winning outcome should be c_i plus the payment in the losing outcome (which is usually set to 0. However, this is not a requirement for achieving truthfulness). ∎

In randomized auctions, the relevant outcome for an agent is a probability $0 \leq p \leq 1$ for winning the good. The randomized auctioning can be viewed as a distribution over deterministic auctions. If for each and every one of these deterministic auctions we have a critical value, then we have a universal truthfulness. Otherwise, we can settle for truthfulness in expectation. In this case, the following holds:

Claim 5.3.2 *The following are equivalent:*

1. *A randomized auction is rationalizable,*
2. *each bidder's probability for winning is weakly monotone in its bid,*
3. *HMD conditions hold.*

Proof: If the probability for winning is p the expected valuation of an agent with type t is pt, and the derivative is p. In order to satisfy the HMD condition, the probability for winning has to be weakly monotone in the valuation. Since for a given probability p the derivative is constant, the HMD condition is a necessary and sufficient condition for rationalizability, and it is equivalent to a weakly monotone increase of p. ∎

5.3.2 Scheduling with Deadlines

The following model is an example for single parameter agents, where the derivative of the valuation function is not a constant, but still simple: n agents apply to get service from a central mechanism. An agent's type is a deadline $t_i \in \mathbb{R}^+$, which it must be served by to have a positive valuation. The output is a service time $\omega_i \in \mathbb{R}^+ \cup \{\infty\}$. The valuation function of an agent is $v_i(\omega_i, t_i) = \max\{0, t_i - \omega_i\}$, i.e., it linearly decreases until it drops to 0 at the deadline, and then remains fixed. The infinity outcome represents the case where an agent is never served, in which case the valuation is zero.

Since the HMD condition relates only to the valuation function, we don't consider here the objective function of the mechanism, or the set of feasible outcomes, which are needed for the algorithmic design, but settle on deriving the rationalizability condition. The only assumption we make on the behavior of the mechanism is that an agent is never served after its declared deadline. This assumption simplifies the condition for rationalizability, and is also a reasonable assumption since a cooperative agent is indifferent between late service and denial of service.

Theorem 5.3.3 *Given that a server never serves an agent after its declared deadline, then it is rationalizable if and only if for each agent, either $\omega_{b_i} = \infty$ for every b_i, or it has a critical time c_i such that if $b_i < c_i$ then $\omega_{b_i} = \infty$, and if $b_i > c_i$ then $\omega_{b_i} \leq c_i$. For $b_i = c_i$ both outcomes are possible.*

Proof: Sufficiency is proved by deriving the first HMD condition from Theorem 5.2.2: If $\omega_{b_i} = \infty$, then v_i is 0 for every t_i, and so is the derivative v_i'. If ω_{b_i} is finite, then $v_i(\omega_i, t_i) = t_i - \omega_{b_i}$ if $t_i > \omega_i$ and 0 otherwise. The derivative is therefore 1 if $t_i > \omega_{b_i}$, 0 if $t_i < \omega_{b_i}$, and undefined for $t_i = \omega_{b_i}$.

If an agent is never served, then the derivative is 0 for any bid, and HMD holds trivially. Otherwise, assume that for some deadline t_0, the agent is served at $\omega_0 \leq t_0$. Therefore, $v_i'(\omega_0, t) = 1$ for every $t > t_0$ (see also Figure 5.1). For the first HMD condition to hold, for any bid $b_i > t_0$, the agent has to be served before or at t_0. Therefore, ω_0 upper bounds the critical time. The exact critical time is the infimum over bids that receive service.

Necessity of critical time, under the constraint of never serving after the declared deadline, exists since the violation of the critical time also violates weak monotonicity: Let $b_0 < b_1$ be two optional bids, and let ω_0 and ω_1 be their service times, respectively. Additionally, assume that $\omega_0 \leq b_0 < \omega_1$. If the critical time constraint is violated then either $\omega_1 = \infty$ or $b_0 < \omega_1 \leq b_1$.

If $\omega_1 = \infty$ then $v_i(\omega_1, b_1) - v_i(\omega_0, b_1) = \omega_0 - b_1 < \omega_0 - b_0 = v_i(\omega_1, b_0) - v_i(\omega_0, b_0)$. Otherwise, if $b_0 < \omega_1 \leq b_1$ then $v_i(\omega_1, b_1) - v_i(\omega_0, b_1) = \omega_0 - \omega_1 < \omega_0 - b_0 = v_i(\omega_1, b_0) - v_i(\omega_0, b_0)$. ∎

5.4 Ex-Post Equilibrium

The theory of mechanism design assumes that the valuation function of each agent depends on its true type and on the output of the mechanism, which may depend on the

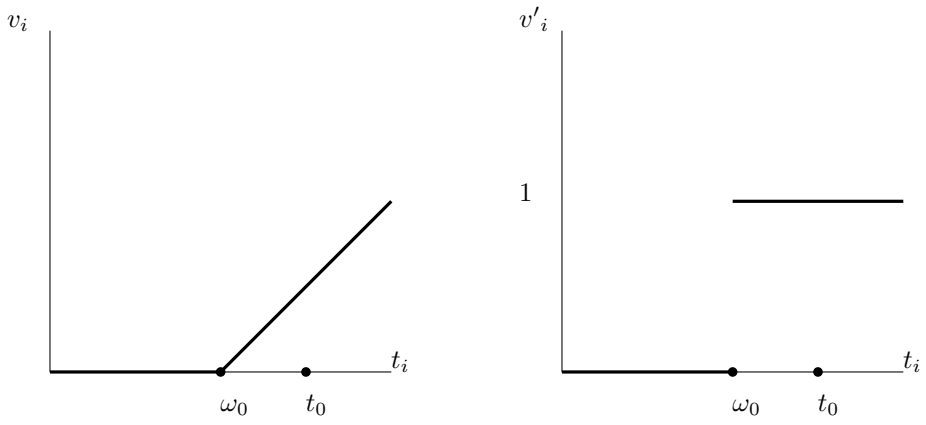

Figure 5.1: HMD under service with deadlines model.
Valuation function and its partial derivative assuming service at time ω_0. If bidding $t_0 \geq \omega_0$ assures this service time, then in order that the first HMD condition will hold, the service time for any bid $b_i > t_0$ cannot be after t_0.

bids of the other agents. The true types of all other agents, have no effect on the valuation function, once the outcome is fixed. In such setups, it is reasonable to try and construct a mechanism where truthtelling is a dominant strategy, since each agent is invariant to whether the other agents bid truthfully or strategically.

This assumption fails to hold when valuation functions are affected by the true types of other bidders and not only their actual bids and the mechanism outcome. When an agent lies about its type, it not only affects the outcome, but also inserts great uncertainty into the system, since the valuations of the other agents depend on its true type[4].

Since requiring truthtelling to be a dominant strategy may be an unreasonable aim, we settle on designing mechanisms which are ex-post truthful, meaning that there is an equilibrium where all agents reveal their true type. Relaxing the dominance requirement is necessary due to the complexity of the valuation functions, which depend on the private parameters of the other agents. Formally, the valuation function of an agent is now $v_i : \Omega \times T^n \mapsto \mathbb{R}$. From an agent's point of view, the outcome space is actually $\Omega \times T^{n-1}$ instead of Ω, i.e. a combination of the algorithm's decision and the true types of the other agents. We define an *ex-post truthful mechanism* as follows:

Definition 5.4.1 *A mechanism $M = (A, P)$ is* ex-post truthful *if for every agent i, for every fixed set of bids for the other agents b_{-i} and for every types $s, t \in T$,*

$$v_i(A(b_{-i}, t), (b_{-i}, t)) - P_i(b_{-i}, t) \geq v_i(A(b_{-i}, s), (b_{-i}, t)) - P_i(b_{-i}, s). \tag{5.10}$$

In words, a mechanism is ex-post truthful if any setting where all agents report their true type is a Nash Equilibrium. For any agent, assuming that all other agents report

[4]A motivating example is given in Chapter 7.

their true values, its valuation function reduces to a function that does not depend directly on the other agents' types, as those are assumed to be equal to the bids, and are now part of the extended outcome space. Let $\Omega' \subset \Omega \times T^{n-1} = \{(\omega, b_{-i}) | \exists b_i,\ A(b_{-i}, b_i) = \omega\}$ denote the set of possible outcomes if indeed the other agents bid truthfully. Replacing Ω with Ω', we can now extend the HMD conditions for valuation functions which depend on the types of the other agents, and give a ex-post truthful characterization and payment functions similar to Theorems 5.2.2 and 5.2.4.

Theorem 5.4.2 *If a decision rule A supports the HMD condition then $M = (A, P)$ is an ex-post truthful mechanism, where*

$$\begin{aligned} & P_i(b_{-i}, t) \\ &= \quad c(b_{-i}) + v_i(A(t, b_{-i}), (t, b_{-i})) - \int_{t_0}^{t} v_i'(A(x, b_{-i}), (x, b_{-i}))dx \end{aligned} \tag{5.11}$$

HMD is also a necessary condition if for each agent, the partial derivative of the valuation function by the agent's type does not depend on its type.

Chapter 6

Budget Constrained Auctions Revised

The approach presented in Chapter 4 for Auctions with Budget Constraints assumes a non-strategic setting, where all valuations and budgets are publicly known. However, it is likely that at least part of this information is privately known only to the agents. In light of Chapter 5, this chapter presents truthful mechanisms for a strategic version of auctions with budget constraints, where the agents' types are single parametered.

6.1 Strategic Budget Constraints

A straightforward characterization of agents' types is as a vector of $k+1$ indices, where k of the indices hold the valuations for each of the k items, and the last index holds the budget constraint. Unfortunately, this characterization of the agents' types is not a single parameter one, and the only known truthful mechanisms for this case are VCG based [76, 26, 39]. Applying VCG requires calculating in exact the allocation that maximizes the total valuation of agents, which is an NP-Hard task.

Therefore, we settle on a simpler characterization of agents' types, which is a single parameter one: Assume that all valuations are known, yet the budget constraints are privately held by the agents. Although this is a somewhat artificial setting, it demonstrates the HMD condition well: We will show that the valuation function resembles the valuation function in Section 5.3.2, yet the rationalizability conditions are different. As in Chapter 4, we assume that an agent's budget is at least equal to its largest valuation of a single item.

Given that agent i bids t as its budget and assuming the bids of the other agents are fixed, ω_t denotes the allocation. We slightly abuse the notation and also use ω_t as the value of the items allocated to the i-th bidder, before capping their valuation by the budget constraint. Using this notation, the valuation function of bidder i given a bid t and an actual budget of d_i is $v_i(\omega_t, d_i) = \min\{\omega_t, d_i\}$.

Definition 6.1.1 *An allocation scheme for auctions with budget constraints is* piecewise

monotone *if one of the following holds for every agent i:*

1. *For every budget constraint t_0 such that $\omega_{t_0} > t_0$, we have $\omega_{t_1} \geq \omega_{t_0}$ for every $t_1 > t_0$*
2. *For every budget constraint t_0 such that $\omega_{t_0} < t_0$, we have $\omega_{t_1} \leq \omega_{t_0}$ for every $t_1 < t_0$*

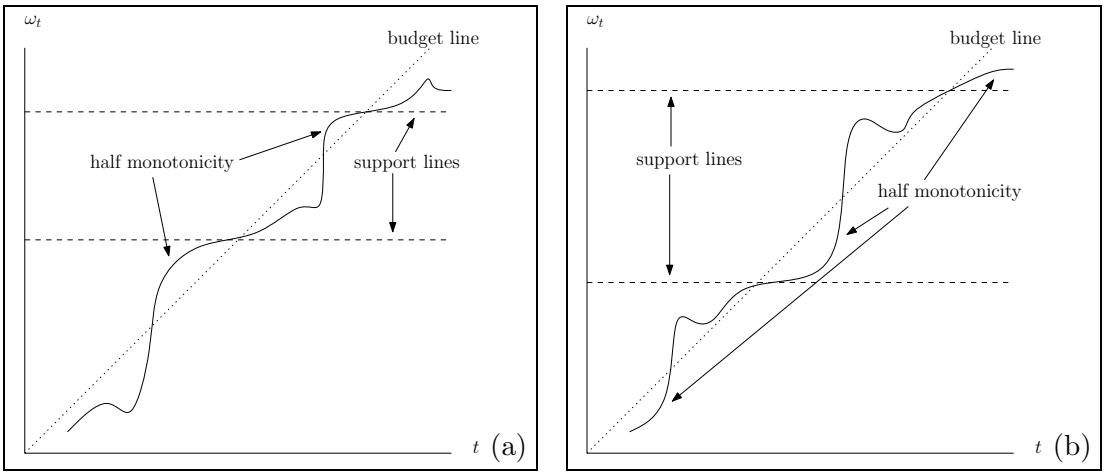

Figure 6.1: Piecewise monotone allocations
Examples of piecewise monotonicity, showing the value of the allocation as a function of the declared budget: (a) The valuation is monotone when above the budget line; When below, it cannot go below the last support line, which marks the last time the valuation was above the budget. (b) Dual piecewise monotonicity requires the valuation to be monotone when below the budget line; When above, it cannot exceed the next budget line, which marks the next time the valuation goes under the budget.

In words, The first characterization requires that whenever the value of the allocation meets or exceeds the reported budget constraint, it sets a lower bound on the value of the allocation for higher bids. This means that the allocation is weakly monotone when the budget constraint is exceeded. The allocation need not be monotone when its value is below the budget, but is lower bounded by the last budget exceeded. The second and dual definition requires weak monotonicity when the valuation is below the budget constraint. When above the budget, it is upper bounded by the valuation on the next time it goes below the budget. Figure 6.1 demonstrates these characterizations.

Theorem 6.1.2 *Any piecewise monotone allocation rule is rationalizable.*

Proof: From the point of view of the i-th agent, the exact allocation is not important, only the total value of items assigned to him (denoted by ω), and the budget constraint. For any ω, $v_i(\omega, t_i) = t_i$ if $t_i < \omega$ and $v_i(\omega, t_i) = \omega$ otherwise. The derivative is therefore 1 if $t_i < \omega$ and 0 otherwise.

We now prove that in order to satisfy the first HMD condition, it is sufficient to have the first type of piecewise monotonicity: Given a budget constraint b_0, first consider

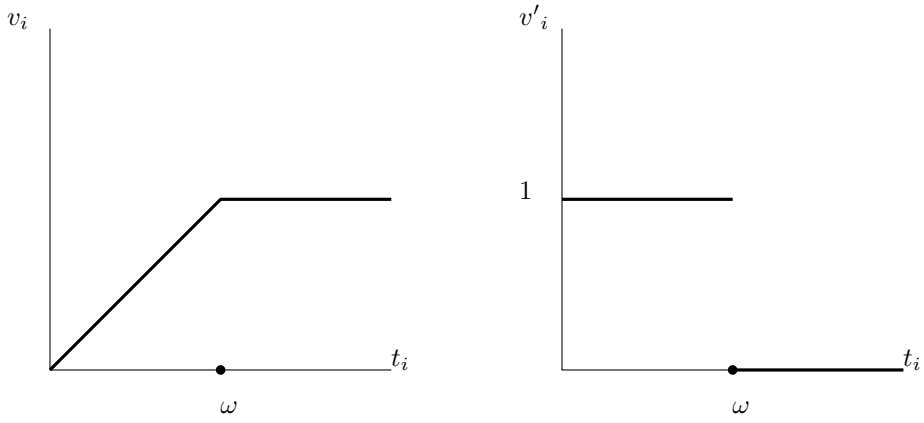

Figure 6.2: HMD under budget constraints
Valuation function and its partial derivative assuming an allocation valuation of ω.

the case where $\omega_{b_0} \leq b_0$. For any type $t > b_0$, $v'_i(\omega_{b_0}, t) = 0$, which guarantees that $v'_i(\omega_{b_0}, t) \leq v'_i(\omega_{b_1}, t)$ for any $b_1 > b_0$. If $\omega_{b_0} > b_0$, then the derivative is 1 until ω_{b_0}, after which it drops to 0. Therefore, to fulfill the HMD condition, for any bid $b_1 > b_0$ the outcome must be at least ω_{b_0}, in order that the partial derivative would also remain 1 until at least ω_{b_0}. Since this is exactly what the first definition of piecewise monotonicity guarantees, piecewise monotonicity leads to fulfilling the first HMD condition, which by Theorem 5.2.2 is a sufficient condition for rationalizability.

In case the second characterization of piecewise monotonicity holds, it leads to the second HMD condition. The proof is symmetric: Given a budget constraint b_0, first consider the case that $\omega_{b_0} \geq b_0$. For any type $t < b_0$, $v'_i(\omega_{b_0}, t) = 1$, which guarantees that $v'_i(\omega_{b_0}, t) \geq v'_i(\omega_{b_1}, t)$ for any $b_1 < b_0$. If $\omega_{b_0} < b_0$, then the derivative is 1 until ω_{b_0}, after which it drops to 0. Therefore, to fulfill the HMD condition, for any bid $b_1 < b_0$ the outcome must be at most ω_{b_0}, in order that the partial derivative would drop to 0 at ω_{b_0} or before it. This is exactly what the second definition of piecewise monotonicity guarantees, which leads to fulfilling the second HMD condition. ∎

We can now use Theorem 6.1.2 in order to develop rationalizable allocation algorithms. We note that in the non-strategic form in the problem, an objective funtion of maximizing the total valuation (the social welfare) was congruent to maximizing the revenue, under the assumption that each agent can be charged up to its valuation. This is not the case in the strategic setting, since payments are determined by Thorem 5.2.4, and may be strictly lower than the actual valuation.

Our main concern in the next sections is maximizing the total valuation, however we also analyze the revenue collected by the mechanisms we suggest. A mechanism that returns optimal allocations would be the VCG mechanism, however finding the optimal allocation is NP-Hard. Therefore, we show that there are piecewise monotone (hence, rationalizable) allocation algorithms that output approximate allocations.

Known approximation algorithms for the non-strategic form of the problem are not

necessarily monotone. The LP based approach in Chapter 4 does not guarantee monotonicity due to the rounding phase. A counter example for the greedy algorithm in [55] is given in Section 6.3. The piecewise monotone allocation algorithms we present in the next sections are based on the following corollaries, derived directly from Theorem 6.1.2:

Corollary 6.1.3 *Any allocation algorithm that never exceeds the budget constraint is rationalizable.*

Corollary 6.1.4 *Any allocation algorithm for which the total value of items it allocated to each agent is weakly monotone increasing in its budget constraint is rationalizable.*

The key idea in Corollary 6.1.3 is to avoid allocation patterns that actually require monotonicity, while in Corollary 6.1.4 monotonicity is always maintained.

6.2 MAX-GAP: Below the Budget Algorithm

Under the restriction of never exceeding the budget constraint, the allocation problem reduces to a subcase of the MAX-GAP (Generalized Assignment Problem). MAX-GAP is a generalization of the Multiple Knapsack problem [24], where each item has a size and profit, and the items are packed into bins (knapsacks) of limited capacity. The goal is to maximize the total profit of the packed items, under the capacity constraints. In the MAX-GAP problem, each item has a different size and profit for each bin.

MAX-GAP is NP-Hard to solve, but there are several approximation algorithms [24, 33]. An instance of an auction with budget constraints can be reduced to a MAX-GAP instance by having a bin for each agent, with a capacity equal to the budget constraint of that agent. The value of an assigning the j-th item to the i-th item is both the size and the profit of packing the j-th item in the i-th bin. By Corollary 6.1.3, these approximation algorithms are also rationalizable. However, the approximation ratio achieved for MAX-GAP is not necessarily preserved in this reduction, since the optimal allocation may benefit from exceeding the budgets, while in the reduction to MAX-GAP this is not allowed.

Chekuri and Khanna [24] give a 2-approximation to MAX-GAP, based on a reduction to MIN-GAP [72], which is a similar problem where items have a cost instead of a profit. The approximation is achieved by rounding an LP, which resembles the LP used in Chapter 4. The 2-approximation ratio of the MAX-GAP problem is preserved under the auction with budget constraint problem, since the approximation is in comparison to a optimal fractional allocation returned by the LP, whose optimality is preserved through the reduction.

Fleischer et al. [33] achieve a better approximation ratio of $\frac{e}{e-1} \approx 1.58$ for a family of allocation problems, including MAX-GAP, named Separable Assignment Problems (SAP). However, they use a different LP, such that the analysis does not preserve this approximation ratio.

Choosing that an agent with an empty assignment won't pay the mechanism, the payment function in Corollary 5.2.5 reduces to paying the exact value of the allocated

Input: a matrix $\{b_{ij}\}$ of values of k items for n agents
a bid vector $d = (d_1, d_2, .., d_n)$ of budget constraints, such that $d_i \geq b_{ij}$, $\forall i, j$

Output: An allocation $\{S_i\}_{i=1}^n$ of items to agents

1. $I \leftarrow \{1, 2, \ldots, n\}$, $S_i \leftarrow \emptyset$, $\forall\ 1 \leq i \leq n$
2. For $j \leftarrow 1$ to k;
   (a) If $I = \emptyset$ Return $\{S_i\}_{i=1}^n$
   (b) $i \leftarrow \arg\max_{i \in I}\{b_{ij}\}$
   (c) $S_i \leftarrow S_i \cup \{j\}$
   (d) If $\sum_{j \in S_i} b_{ij} > d_i$ Then $I \leftarrow I \setminus \{i\}$
3. Return the allocation $\{S_i\}_{i=1}^n$

Figure 6.3: Algorithm Lazy Greedy

items. This is due to the fact that since the allocation's value never exceeds the budget, the agent's valuations used in the payment function equals the actual value of the allocated items.

6.3 Lazy Greedy: A Monotone Algorithm

It may seem at first that the greedy algorithm [55] that greedily allocates each item to the highest bidder, with respect to the residual valuations derived from the current partial allocation, would be rationalizable since it appears to be monotone with respect to the budget constraint. However, the following example disproves this intuition:

Example 6.3.1 *Consider 3 items* $\{1, 2, 3\}$ *and 2 agents, with valuations* $\{3, 5, 6\}$ *and* $\{0, 4, 2\}$ *respectively. If both agents have a budget constraint of* 6 *then a greedy allocation gives the first agent items* 1 *and* 3*. If the first agent changes his budget constraint to* 9 *his allocation will be items* 1 *and* 2*. The greedy allocation is not rationalizable due to a violation of week monotonicity:*

$$v_1(\{1,3\},6) - v_1(\{1,2\},6) = 0 < 1 = v_1(\{1,3\},9) - v_1(\{1,2\},9)$$

Algorithm *Lazy Greedy* outputs a weakly monotone allocation and requires only $O(nk)$ operations, which is linear in the input size. Lazy Greedy allocates each item to the highest bidder, ignoring the budget constraint until after the assignment. If the budget constraint of an agent has been exceeded, then the agent is removed from the rest of the allocation process. Ignoring the budget constraint until after the allocation is required for piecewise monotonicity, since this guarantees that by increasing an agent's budget constraint it will receive at least the same items. By Corollary 6.1.4, this implies rationalizability.

However, the critical value of the budget constraint needed for winning each additional item is exactly the value of the preallocated items, and therefore the marginal valuation of this item is zero. Therefore, the payment function in Corollary 5.2.5 reduces to a constant payment, such as the agent's valuation of the allocation it receives when bidding its minimal possible budget.

In the analysis of Lazy Greedy, we use the following definition:

Definition 6.3.2 *The* virtual value *of an agent is the difference between the total value of the items assigned and their valuation, i.e.* $\sum_{j \in S_i} b_{ij} - v_i(S_i, d_i)$. *Similarly, the virtual value of an allocation is the sum of virtual values over all agents.*

We now prove several properties on the virtual value:

Lemma 6.3.3 *If all agents bid truthfully, the virtual value of the Lazy Greedy allocation is at most the total valuation of the allocation.*

Proof: An agent who hasn't exceeded its budget constraint does not have a virtual value. Once an agent exceeds its budget constraint, it is not allocated any more items, and therefore the virtual value of an agent is at most the value of the last item allocated to it, which is at most equal to the agent's budget. The actual valuation of the agent is exactly the budget constraint, since it has been exceeded. Since the claim holds for each agent separately, it holds for the entire allocation, too. ∎

Lemma 6.3.4 *If all agents bid truthfully, the value of the optimal allocation is at most twice the value of Lazy Greedy plus its virtual value.*

Proof: Given an allocation of Lazy Greedy, we alter the input by splitting each item that contributes to the virtual profit into two items whose combined valuation, for each agent, equals the valuation of original item. The division point is chosen such that the agent who receives it would exactly have its budget constraint met when receiving the first part of the split item. We set the valuations of the other agents for these fractional items proportionally to the valuations of the winning agent. For the altered input, Lazy Greedy outputs an allocation with the same value, since instead of allocating items that exceed the budget constraint, we allocate one fractional item that meets it, and another item entirely beyond the budget. In contrast to Lazy Greedy, the optimal allocation may only gain from the change in the input.

We can now divide the items into two groups: The first group includes both items that were allocated within the budget constraints, and items that were not allocated at all. The second group includes the items that contribute entirely to the virtual profit. We observe how an optimal allocation performs with respect to Lazy Greedy, by observing the marginal contribution to the valuation, when allocating the items in the first group before the items in the second group. The first group contains unallocated items only if the Lazy greedy allocation exceeded all budget constraints. In this case the Lazy Greedy allocation is optimal. Without loss of generality, we assume that Lazy Greedy allocated all of the items in the first group.

By the end of the allocation of the items in the first group, the marginal valuation of the optimal allocation (optimal with respect to the items in both groups) is at most twice the marginal valuation of Lazy Greedy. The proof is by induction on the number of items in the first group: If the first group is empty then the claim trivially holds. Assuming the claim holds for $n-1$ items, if there are n items then let i denote the agent who wins the first item in the Lazy Greedy allocation, and let p denote the item's valuation. If the optimal allocation also selects the i-th agent, then both allocations gain the same marginal valuation of p, while for the remaining items in the first group, by the inductive claim, the optimal allocation benefits at most twice that Lazy Greedy.

If the optimal allocation assigns the first item to a different agent $i' \neq i$, then its gain is $p' \leq p$, since p is the maximal bid for this item. By changing the optimal allocation to select the i-th agent, the marginal value from the first item cannot decrease, but the marginal valuation from optimally allocating the rest of the items can decrease by at most p, due to the decrease in the available budget of agent i. By artificially compensating the optimal allocation with an additional value of p, we have that the value of the optimal allocation is at most $2p$ plus an optimal allocation of the remaining items given that the budget of agent i has decreased by p. By the inductive claim, an optimal allocation gains on the rest of the items in the first group at most twice than the rest of the value of Lazy Greedy, and therefore the claim holds.

We have that the valuation of the optimal assignment is at most twice the value of allocating the first group of items according to the Lazy Greedy allocation, followed by an optimal allocation of the remaining items in the second group. We observe these items by the order of their allocation: For each item in the second group, Lazy Greedy allocates it to an agent who already reached its budget constraint, and gains a virtual value that is at least as the largest marginal valuation of any other agent, otherwise a different agent would have been selected. Additionally, the allocation of this item has no effect on the marginal valuations of the remaining items. In contrast, the optimal allocation of each item in the second group considers lower valuations, due to the allocation of a larger subset of items.

The optimal allocation of the second group of items contributes at most the virtual value of Lazy Greedy. Therefore, the value of the optimal allocation is bounded by twice the value of Lazy Greedy plus its virtual value. ∎

We now prove a tight bound on the approximation ratio of Lazy Greedy:

Theorem 6.3.5 *The approximation ratio of Lazy Greedy is* 3

Proof: The upper bound is directly derived from Lemmata 6.3.3 and 6.3.4. The following setup gives a tight lower bound:

Given 3 agents and 3 items, let the price vectors of the agents be $(1,1,1)$, $(1,0,0)$ and $(0,1,0)$, and let all budget constraints be 1. An optimal allocation allocates one item to each agent, achieving a value of 1 for each agent and a total value of 3. Lazy Greedy allocates the first two items two the first agent (with a total value of 1) and the third item to one of the other agents (who bid 0 for it). The total value of this allocation is 1. We note that one can avoid the tie breaks in this example by slightly perturbing the bids, and achieving a ratio that asymptotically reaches 3. ∎

6.4 Strongly Truthful Mechanisms

One may argue that the algorithms presented is Section 6.2 and 6.3 give a weak incentive for cooperation, since the mechanisms are not strongly truthful. There are strategies different than truthtelling, that may change the allocation, without affecting the utility.

In MAX-GAP based mechanisms, the budget constraint is never exceeded and the agent's payment equals its valuation. Therefore, there is an incentive to avoid overbidding, since the overbidding agent risks paying beyond its real budget constraint. However, there is no incentive to avoid underbidding, since as long as the allocation remains below the actual budget constraint, the agent pays exactly its valuation and always has a utility of zero.

The Lazy Greedy based mechanism charges a constant payment and exceeds the agent's budget constraint unless it is sufficiently high, such that the allocation cannot change by raising the budget even further. There is no incentive to underbid, since by this an agent may lose items that were allocated for free. However, there is no incentive to avoid overbidding, and receive a larger allocation, which although is above the budget constraint and does not contribute to the agent's valuation, does not have any additional cost.

This difficulty is an outcome of the structure of the valuation functions, as well as the allocation mechanisms, which either always charge the full cost of each allocated item, or always allocate everything for free. One approach that bypasses this obstacle is mixing different mechanisms, resulting in a randomized mechanism, which is strongly truthful *in expectation*. Another approach is to slightly modify the deterministic mechanisms while maintaining piecewise monotonicity.

Randomized Solutions

A classical solution is to apply a simple strongly truthful mechanism with a small probability ϵ, and the original mechanism (MAX-GAP or Lazy Greedy) with probability $1-\epsilon$. A possible candidate mechanism is as follows: Randomly select an agent i and a price p_i chosen uniformly from $[\max_j b_{ij}, \sum_j b_{ij}]$. Agent i receives all of the items for price p_i if its bid is at least p_i. This candidate mechanism is truthful since it is simply a randomized single commodity auction, and as required in Section 5.3.1, the probability of winning is monotone in the bid, and the payment is the critical value. The mechanism is strongly truthful since if an agent bids untruthfully, there is always a positive probability that the items will be offered at a price between the actual budget and the bid, an outcome where truthtelling is a strictly better strategy.

The approximation ratio of the randomized single commodity auction mechanism may be unbounded, but since it is executed with a small probability, the approximation ratio of the combined mechanism, *in expectation*, is either $\frac{2}{1-\epsilon}$ (if we combine with MAX-GAP) or $\frac{3}{1-\epsilon}$ (if we combine with Lazy Greedy). The combined mechanism is universally truthful (a distribution over truthful mechanisms), and in expectation, is strongly truthful.

A different solution is to use a convex combination of Lazy Greedy and MAX-GAP mechanisms, i.e. a randomized mechanism that with probability p applies the MAX-GAP

reductions, and with probability $1-p$ applies Lazy Greedy. This mechanism is universally truthful, has an approximation ratio of 3 in the worst case, and $3-p$ in expectation. In expectation, this mechanism is also strongly truthful.

A Deterministic Solution

A possible deterministic solution is to slightly scale the budget constraints. If we scale the budgets by a factor of $1-\epsilon$ and run Lazy Greedy, each allocated item now has a small value at the critical budget value where it is added to the agent's allocation, which now has a strong incentive to avoid overbidding. Calculating the payments is a slightly more involved process, as it requires finding the critical values of budgets where the allocation changes. The monotonicity of Lazy Greedy is preserved on scaling, thus the mechanism is indeed (strongly) truthful. The cost of the budget scaling is that the approximation ratio slightly degrades to $\frac{3}{1-\epsilon}$.

In contrast, a similar approach would not necessarily work on MAX-GAP mechanisms: Scaling the budget by a factor of $1+\epsilon$ may indeed return allocations that slightly exceed the original budget constraint, resulting in small discounts in the payment function. However, since MAX-GAP allocations are only piecewise monotone and not completely monotone, there is no guarantee that the piecewise monotonicity property will be preserved.

6.5 Mechanisms Beyond the HMD Condition

Unlike previous examples from Section 5.3, HMD is not a necessary condition for budget constrained auctions. For example, consider the VCG [76, 26, 39] mechanism, which requires that for each input we calculate an outcome that maximizes the total valuation. Since this is also the objective function here, any algorithm that outputs an optimal allocation is rationalizable.

If there is more than one optimal outcome, VCG may choose among them arbitrarily. However, An inconsistent choice rule between optimal outcomes in different settings may lead to violation of both versions of the HMD condition. An example of such a case follows:

Example 6.5.1 *Consider 2 agents and 5 identical items. Both agents have a valuation of* 1 *for each item. Given that the second agent reports a budget constraint of* 2*, consider the following allocation scheme, as a function of the budget constraint that the first agent bids: If the first agent bids* 2*, he receives two items, if he bids* 4*, he receives four items, and for any other bid he receives three items. All other items are allocated to the second agent. All of these allocations maximize the total valuation for the reported budgets, and are therefore consistent with the VCG mechanism.*

However, the decrease in the value of the allocation when bidding 2 *instead of* 1 *violates the first HMD condition, while the increase while lowering the bid from* 5 *to* 4 *violates the second HMD condition. Therefore, there exists a truthful VCG-based mechanism which does not comply to either HMD condition.*

Chapter 7

Incentive Compatible Scheduling

This chapter presents truthful approximation algorithms for scheduling jobs on uniformly related machines, when the machines and jobs may be controlled by strategic players. To assure rationalizability, all mechanisms satisfy monotonicity conditions, which were independently proved previously [12, 15], but can be easily derived from the HMD condition presented in Chapter 5.

7.1 Preliminaries

We consider the problem of scheduling jobs on uniformly related machines. In the non-strategic form of the problem, we are given n jobs with processing requirements of $p = \{p_1, p_2, \ldots, p_n\}$, and m machines with speeds $s_1, s_2, \ldots, s_m$. A feasible schedule σ is a division of all jobs in p to disjoint sets $\sigma_1, \sigma_2, \ldots, \sigma_m$. By allocating the jobs in σ_j to the j-th machine, the total work assigned to that machine is $w_j(\sigma) = \sum_{i \in \sigma_j} p_i$, and it incurs a load of $l_j(\sigma) = \frac{w_j(\sigma)}{s_j}$, which is the total time required to process all of the jobs in σ_j. The makespan, $\max_j l_j(\sigma)$, which is the latest completion time of a machine, is the objective function that we wish to minimize.

If the machines are owned by strategic agents (machine agents), then each machine incurs a cost proportional to its load $l_j(\sigma)$. A machine's speed is its private value, however, it would be more convenient to use the cost per unit of work, $c_j = \frac{1}{s_j}$ as the machine's type, instead of using speeds. Thus, the cost of the j-th machine is also given by $w_j(\sigma) \cdot c_j$.

If the jobs are owned by strategic agents (job agents), then each job incurs a cost proportional to the time that passes until its execution is completed. In this case, the processing requirement p_i is the private value of the i-th job, and also denotes its type. If machines execute jobs in parallel, the cost observed by the i-th job, when executed on the j-th machine is l_j. Given an assignment σ, let $\hat{s}_i(\sigma)$ denote the speed of the machine that the i-th job is assigned to. Assuming $i \in \sigma_j$, let $W_i(\sigma) = w_j(\sigma) - p_i$ denote the sum of processing requirements other than p_i that are allocated on the same machine with the i-th job. Therefore, the cost observed by the i-th job can also be presented as $\frac{p_i + W_i(\sigma)}{\hat{s}_i(\sigma)}$.

Alternatively, if the jobs are processed serially, then $W_i(\sigma)$ denotes the sum of pro-

cessing requirements completed before the i-th job is executed, on the same machine that executes it. For the remaining of the chapter, the only difference between parallel execution and serial execution is the definition of $W_i(\sigma)$.

Note that the cost structure of a job may depend on the actual processing requirement of other jobs. Therefore, if the jobs are owned by strategic agents, we settle with ex-post truthful mechanisms. If only the machine owners are strategic, then we aim for truthful mechanisms, since the actual speed of one machine does not affect the cost of another machine.

Given two sorted vectors, we define the following relation:

Definition 7.1.1 *A vector $v = (v_1, v_2, .., v_n)$ is lexicographically smaller than a vector $u = (u_1, u_2, .., u_n)$ if for some i, $v_i < u_i$ and $v_k = u_k$ for all $1 \leq k < i$.*

7.2 Monotone Algorithms

7.2.1 Strategic Machine Owners

Given strategic machine owners and public knowledge of the processing requirements p, the mechanism receives an input of bids $b = (b_1, b_2, \ldots, b_m)$, while the actual types of the machine agents are $(c_1, c_2, \ldots, c_m)$. Holding p fixed, the allocation σ that the mechanism outputs is uniquely defined by the bid vector b. Therefore, we slightly abuse the notation and replace σ with b.

Archer and Tardos [12] showed necessary and sufficient conditions to obtain a truthful mechanism for strategic machine owners.

Definition 7.2.1 *Fixing the other agents' bids b_{-j}, we define the* work-curve *for machine agent j as $w_i(b_{-j}, b_j)$, namely a single-variable function of b_j. A work-curve is decreasing if $w_j(b_{-j}, b_j)$ is a decreasing function of b_j for all b_{-j}.*

A decreasing work-curve means that when a machine agent bids lower costs per unit of work (declaring a faster speed), more or equal amounts of work should be allocated to his machine, given that the other agents' bids are fixed. A *monotone algorithm* for strategic machine owners is an algorithm that produces an assignment which preserves the decreasing work-curve property for all machine agents. Several classic approximation algorithms fail to keep this monotonicity, among them the *LPT* algorithm [37], and the classic PTAS of Hochbaum and Shmoys [12, 42].

Definition 7.2.2 *A mechanism satisfies the* voluntary participation *condition if agents who bid truthfully never incur a net loss, i.e. for all agents j, true values c_j and other agents' bids b_{-j}, $v_j(w_j(b_{-j}, c_j), c_j) - P_j(b_{-j}, c_j) \geq 0$.*

Theorem 7.2.3 *[12] A mechanism for machine agents is truthful and admits a voluntary participation if and only if the work-curve of each agent is decreasing, for all j and b_{-j}, $\int_0^\infty w_j(b_{-j}, x)dx < \infty$ and the payments are as in (7.1):*

$$P_j(b_{-j},b_j) = -b_j w_j(b_{-j}, b_j) - \int_{b_j}^{\infty} w_j(b_{-j}, x)dx \ . \tag{7.1}$$

The negative payment in (7.1) infers that the mechanism pays the machine agents. Assuming truthful bidding, i.e. $b_j = c_j$, the first term in P_j equals the cost of the j-th machine agent, thus the agent's utility is necessarily non negative, and voluntary participation trivially follows.

The monotonicity characterization in Theorem 7.2.3 can be also derived from the HMD condition presented in chapter 5: The valuation of a machine with type c_j executing $w_j(b)$ work is $v_j(b, c_j) = -c_j \cdot w_j(b)$. The partial derivative $\frac{\partial v_j}{\partial c_j} = -w_j(b)$ does not depend on c_j, and therefore, by Theorem 5.2.3, the HMD condition is sufficient and necessary, and reduces to a requirement that $-w_j(b_{-j}, b_j)$ would be monotone in b_j. This is equivalent to a decreasing work curve, since as the work curve decreases, $-w_j(b)$ becomes less negative. Using the payment function in Theorem 5.2.4, we get:

$$P_j(b_{-j}, b_j) = -b_j w_j(b_{-j}, b_j) + \int_{t_0}^{b_j} w_j(b_{-j}, x)dx \tag{7.2}$$

Substituting t_0 with ∞, we get the same payment as in (7.1).

7.2.2 Strategic Job Owners

Given strategic job owners and public knowledge of the machine speeds, the mechanism receives an input of bids $q = (q_1, q_2, \ldots, q_n)$, while the actual types of the job agents are $(p_1, p_2, \ldots, p_m)$. Symmetrically to Section 7.2.1, we slightly abuse the notation and replace σ with q.

First, we'd like to show that unless the scheduling algorithm is degenerate, there is no dominant strategy for the job owners. The degeneration of the scheduler reflects in always assigning the same jobs together on the same machine (or on a machine with the same speed), regardless of their bids, given that the jobs are assigned on the same machine for some bid (jobs that are always assigned alone can change machines). Clearly, degenerate scheduling may be considerably inefficient.

Assume that given the bids q_{-i} of the other agents, if job owner i bids its true type p_i then it is assigned to a machine together with some other jobs including the k-th job, while if the bid is $\hat{p}_i$ then the assignment is to a (possibly different) machine with a different set of jobs that does not include the k-th job. Since the real processing requirement of the k-th job is unknown to the mechanism and may be arbitrary large, any payment to job agent i may be insufficient to make the i-th job agent favor reporting p_i over $\hat{p}_i$, which avoids sharing a machine with the k-th job. Therefore truthtelling cannot be a dominant strategy, unless the decision to assign any two jobs to the same machine is independent of the bids.

similarly, assume that the i-th job is assigned with at least another job to a machine with speed s_j, but can move to a faster machine $s_{j'}$ by reporting some false bid. Again, even if the same jobs that accompany job i on s_j move with it to $s_{j'}$, the increase in the processing speed may dominate any difference in the payment function, given that the actual processing requirements of the other jobs are sufficiently high. This motivates searching for mechanisms that are ex-post truthful.

Definition 7.2.4 *Fixing the other agents' bids q_{-i}, we define the* speed-curve *for job agent i as $\hat{s}_i(q_{-i}, q_i)$, namely a single-variable function of q_i. A speed-curve is increasing if $\hat{s}_i(q_{-i}, q_i)$ is an increasing function of q_i for all q_{-i}.*

An increasing speed-curve means that when a job agent bids a higher processing requirement, it will be assigned to a machine with at least the same speed, given that the other agents' bids are fixed. A *monotone algorithm* for strategic job owners is an algorithm that produces an assignment which preserves the decreasing work-curve property for all machine agents.

Auletta et al. [15] showed that it is necessary to use a monotone algorithm to construct an ex-post truthful mechanism for strategic job owners.

Theorem 7.2.5 *[15] A mechanism for job agents is ex-post truthful only if the speed-curve of each agent is increasing.*

Using the HMD condition, it is possible to strengthen this result, by proving that an increasing speed-curve is also a sufficient condition, and provide a suitable payment function (in [15], Auletta et al. manage to avoid characterizing a general payment function by concentrating on algorithms with constant speed-curves, which have degenerate payment functions).

Theorem 7.2.6 *A mechanism for job agents is ex-post truthful if and only if the speed-curve of each agent is increasing, and a suitable payment function is:*

$$P_i(q_{-i},q_i) = -\frac{q_i + W_i(q_{-i}, q_i)}{\hat{s}_i(q_{-i}, q_i)} + \int_0^{q_i} \frac{dx}{\hat{s}_i(q_{-i}, x)} \ . \tag{7.3}$$

Proof: Given a valuation function of $v_i(q, p_i) = \frac{p_i + W_i(q)}{\hat{s}_i(q)}$, the partial derivative is $\frac{\partial v_i}{\partial p_i} = -\frac{1}{\hat{s}_i(q)}$. The partial derivative does not depend on p_i, and therefore, by Theorem 5.2.3, the HMD condition is both sufficient and necessary. Monotonicity of $-\frac{1}{\hat{s}_i(q)}$ requires that the speed of the machine that executes the i-th job will increase when p_i increases, which is exactly an increasing speed-curve.

By substituting t_0 with 0 in the payment function in Theorem 5.2.4, we get exactly the payment function in (7.3). ∎

Similarly to the payment function in (7.1), in the first term in (7.3) the mechanism pays back the agent its costs. However, in the second term, the agent pays the mechanism, and therefore, in contrast to Section 7.2.1, we cannot guarantee voluntary participation for job agents. Since the number of machines is finite, the second term in (7.3) is unbounded if q_i can be arbitrary large. Therefore, for any constant payment that we may add to the payment function, there exists a processing time for which the payment would not cover the agent's costs.

7.2.3 Double Sided Strategies

In case that both machine owners and job owners are strategic, we can achieve an ex-post truthful mechanism by using an allocation algorithm that satisfies both types of monotonicity: decreasing work-curve from the point of view of the machines, and increasing speed-curve from the point of view of the jobs. We use use the payment function in (7.1) for machine agents, and (7.3) for job agents.

To conclude, since (ex-post) truthfulness is reduced to designing a monotone algorithms, we may assume, for the sake of the monotonicity proofs, that the bids are equal to the real types.

7.3 An FPTAS for a Fixed Number of Machine Agents

We now show a truthful mechanism for any fixed number of machine agents. The mechanism uses a c-approximation algorithm as a black box, to generate a $c(1+\epsilon)$-approximation monotone algorithm. Using an FPTAS as the black box (for example, the FPTAS of Horowitz and Sahni [43]) outputs a monotone FPTAS. Adding a payments scheme as in (7.1), ensures truthful mechanism. The algorithm, *Monotone-Black-Box*, is shown in Figure 7.1.

Theorem 7.3.1 *Algorithm Monotone Black Box is a $c(1+\epsilon)$ approximation algorithm.*

Proof: The output assignment is a c-approximation for the vector d, since d is tested in the enumeration, and since sorting the assignment can only improve the makespan. As for the original bid vector b, rounding the bids adds a multiplicative factor of $1+\epsilon$ to the approximation ratio. Normalizing the vector has no effect, as well as trimming the largest bids, since any non zero assignment to a machine with a bid of at least $(1+\epsilon)^l$ cannot be a c-approximation, since the load on that machine will be more than c times the load of assigning all the jobs to the fastest machine. ∎

Theorem 7.3.2 *If the black box in algorithm Monotone Black Box is an FPTAS then the algorithm itself is also an FPTAS.*

Proof: By Theorem 7.3.1, If the black box in algorithm Monotone Black Box is an FPTAS, i.e. $c = 1+\epsilon$ then the algorithm is a $(1+\epsilon)^2$ approximation. It remains to prove that the running time is polynomial in the input, including $\frac{1}{\epsilon}$. In each iteration of the enumeration, applying the black box, sorting the output assignment and testing it on the vector d can be completed in polynomial time, by the assumption that the black box is an FPTAS. The size of the enumeration is $O(l^m)$, where m is a constant and l is polynomial in the input. ∎

Theorem 7.3.3 *Algorithm Monotone Black Box is monotone.*

Proof: We prove that if a machine j raises its bid (lowers its speed) then the amount of work assigned to it cannot increase. We increment the bid in steps, such that in each

Input: a job sequence p, a bid vector $b = (b_1, b_2, .., b_m)$, a parameter ϵ and a black box, which is a polynomial time c-approximation.

Output: an monotone allocation of jobs to the m machines that achieves a $c(1+\epsilon)$-approximation.

1. Construct a new bid vector $d = (d_1, d_2, \ldots, d_m)$, in the following way:
   (a) for $j = 1 \ldots m$, let $d_j = (1+\epsilon)^{\lceil \log_{1+\epsilon} b_j \rceil}$.
   (b) normalize vector d such that $\min_j d_j = 1$.
   (c) for each bid, if $d_j \geq (1+\epsilon)^{l+1}$ where $l = \lceil \log_{1+\epsilon} (cn \cdot \frac{\max_i p_i}{\min_i p_i}) \rceil$ then set $d_j = (1+\epsilon)^{l+1}$.
2. Enumerate over $d' \in \{((1+\epsilon)^{i_1}, (1+\epsilon)^{i_2}, \ldots, (1+\epsilon)^{i_m})\}$, where $i_j \in \{0, 1, \ldots, l+1\}$. For each vector:
   (a) apply the black box algorithm to d'.
   (b) sort the output assignment such that the i-th fastest machine in d' will get the i-th largest amount of work.
3. Test all the sorted assignments on d, and return the one with the minimal makespan. In case of a tie, choose the assignment with the lexicographically maximum schedule (i.e. allocating more to the faster machines).

Figure 7.1: Monotone Black Box

step the power of $1+\epsilon$ that equals the rounded bid rises by one. We prove the claim for a single step, and therefore, the claim also holds for the entire increment.

We first assume that d_j is not the unique fastest machine (i.e., there is a machine $k \neq j$ such that $d_k = 1$). If $d_j \geq (1+\epsilon)^l$ then by the proof of Theorem 7.3.1, the assignment to machine j must be null, otherwise the approximation ratio is not achieved. Clearly, by raising the bid the assignment will remain null, and the claim holds. Therefore, we assume that the normalized rounded bid vector changes from $d = (d_{-j}, d_j)$ to $d' = (d_{-j}, d_j(1+\epsilon))$, the assignment changes from $\sigma = \sigma(d)$ to $\sigma' = \sigma(d')$, and the amount of work allocated to machine j changes from $w_j = w_j(d)$ to $w'_j = w_j(d')$. By way of contradiction, we assume $w_j < w'_j$.

We use $T(\sigma, d)$ to denote the makespan of assignment σ, assuming that the real types are d. Since the algorithm chooses the optimal assignment with respect to d among a set of assignments that contains both σ and σ', we have that $T(\sigma, d) \leq T(\sigma', d)$ and that $T(\sigma', d') \leq T(\sigma, d')$. Additionally, since the bids in d are smaller than or equal to the bids in d', we have that $T(\sigma, d) \leq T(\sigma, d')$ and $T(\sigma', d) \leq T(\sigma', d')$.

Suppose that machine j is the bottleneck in $T(\sigma, d')$, meaning that the load on machine j is the highest. Since $w'_j > w_j$, we have $T(\sigma, d') < T(\sigma', d')$, as the load on machine j

increases even more. This is a contradiction to $T(\sigma', d') \leq T(\sigma, d')$, and therefore machine j cannot be the bottleneck in $T(\sigma, d')$.

If machine j is not the bottleneck in $T(\sigma, d')$, we have that $T(\sigma, d) = T(\sigma, d')$. Therefore, the inequalities $T(\sigma, d) \leq T(\sigma', d) \leq T(\sigma', d') \leq T(\sigma, d')$ hold with equality. Therefore, we have a tie between the makespan of σ and σ' for both d and d'. Since ties must be broken consistently (choosing the lexicographically maximum) it must be that $\sigma = \sigma'$. Since the assignment is sorted (the faster machine is assigned more work), if a machine decreases its speed then the amount of work assigned to it by the same assignment cannot increase, which is a contradiction to $w'_j > w_j$.

If machine j is the unique fastest, then due to the normalization of the rounded bids and trimming of high bids, after it raises its bid by one step the new bid vector d' will be as follows: d_j remains 1, bids between $1+\epsilon$ and $(1+\epsilon)^l$ decrease by a factor of $1+\epsilon$, and bids equal to $(1+\epsilon)^{l+1}$ can either decrease to $(1+\epsilon)^l$ or remain $(1+\epsilon)^{l+1}$.

Let $\hat{d}$ be the same bid vector as d', with all the bids of $(1+\epsilon)^{l+1}$ replaced with $(1+\epsilon)^l$. Since machines that bid $(1+\epsilon)^l$ or more must get a null assignment, then the optimal assignment (among all assignments that are considered by the algorithm) for $\hat{d}$ is the same as d'. The same assignment remains the optimum for vector $\hat{d}(1+\epsilon)$, where all bids are multiplied by $1+\epsilon$. The bid vector $\hat{d}(1+\epsilon)$ is exactly the bid vector d, with d_j replaced with $1+\epsilon$ (instead of 1). By the same argument from the case where machine j is not the unique fastest, the work assigned to machine j in $\hat{d}(1+\epsilon)$ is at most the same as the work assigned in d, and therefore the algorithm is monotone for the unique fastest machine as well. ∎

To conclude, we claim that the payments for each agent can be calculated in polynomial time. Consider the work-curve inside the integral expression in (7.1). The number of points (breakpoints) where the work-curve changes is bounded by the number of possible allocations considered by the algorithm, which is polynomial in the input size, including $\frac{1}{\epsilon}$.

7.4 An Approximation for Job Agents

A classic approximation algorithm for machine scheduling, forms a "valid" fractional allocation of the jobs to the machines and then uses a simple rounding to get a 2-approximation for the problem. In [12] it has been shown that this simple algorithm is not monotone with respect to the machines, thus not truthful for strategic machine owners. However, this algorithm is monotone with respects to the jobs, and therefore ex-post truthful for strategic job owners.

This is an improvement to [15], which presented non constant approximation algorithms, unless the ratio between the fastest and slowest machine speeds is small. Closing the gap between this upper bound and the lower bound of approximately 1.28 in [15] remains an open problem.

Given a threshold T, we can treat the machines as bins of size T/c_j. A *fractional assignment* of the jobs to the machines, is a partition of each job i into pieces whose sizes sum to p_i and an assignment of these pieces to the machines (bins). A fractional

Input: a bid vector $q = (q_1, q_2, \ldots, q_n)$, and machine costs $c = (c_1, c_2, .., c_m)$.

Output: a monotone 2-approximation allocation of the jobs to the machines.

1. Sort the jobs in non increasing order.
2. Set a threshold of maximal load using (7.4).
3. Create a fractional allocation by assigning jobs to machines, from fastest to slowest, splitting a job whenever the threshold is reached.
4. Round the assignment such that each job is allocated to the fastest machine that received a fraction of the job.

Figure 7.2: A monotone 2-approximation algorithm

assignment is *valid* if each bin is large enough to contain the sum of all pieces assigned to it, and for every piece assigned to it, it is capable of containing the entire job the piece belongs to. The smallest threshold for which there exists a valid fractional assignment, T^f is a lower bound to the optimal solution, and can be calculated exactly in the following manner (see [12]):

$$T^f(c,p) = \max_{i\in[n]} \min_{j\in[m]} \max \left\{ c_j p_i, \frac{\sum_{k=1}^{i} p_k}{\sum_{l=1}^{j} \frac{1}{c_l}} \right\}. \tag{7.4}$$

This valid fractional assignment with respect to this threshold is obtained by sorting the jobs in non-increasing order, and allocating them to the machines (ordered in non-increasing order of their speeds). Some jobs are sliced between two machines when the threshold is exceeded in the middle of a job.

By rounding each fractionally assigned job toward the faster machine we get a 2-approximation, since each machine gets at most one extra fraction, and the load it incurs is lower than the threshold. Although this approximation is not rationalizable for strategic machines, it can be used to construct an ex-post mechanism for strategic job owners.

Theorem 7.4.1 *There exists an ex-post truthful mechanism for scheduling strategic jobs on uniformly related machines, which is a* 2*-approximation of the makespan.*

Proof: We show that the approximation described in 7.2 is monotone with respect to the jobs, meaning that when a job increases in processing requirement, it cannot move to a slower machine. The existence of an ex-post mechanism follows from Theorem 5.4.2, using the payment function in (7.3).

Given an increase in the processing requirement of the i-th job, we observe that the threshold of the fractional assignment can only increase, increasing the capacity of every machine.

As long as the order of the sorted jobs does not change, The total processing requirement of the largest $i-1$ jobs is unchanged, and therefore the machine where the first

fraction (whether there are one or two fractions) of the i-th job is allocated is either a faster machine, or the same one. The integral solution rounds to the fastest machine with a fraction of the job, which is the machine that holds the first fraction, and therefore the chosen machine cannot be slower.

In breakpoints where the increase in the processing requirement of the i-th job cause it to swap places in the sorted job vector with the job before it, the same allocation is generated, except for swapping between two jobs. Obviously, the i-th job is shifted toward the faster machines, and cannot be allocated to a slower one. ∎

It remains to show that the payment function in (7.3) can be computed in polynomial time. For the i-th job, computing the integral in (7.3) requires finding the critical bids of q_i where the job moves from one machine to the other. Since the allocation is monotone, there are at most $m-1$ critical bids where these transitions occur, but a larger number of candidate bids need to be enumerated in order to find them all. One group of candidate bids includes the $n-1$ values where the i-th job switches positions in the sorted bid vector. Between every two adjacent bids of this group we need to consider other candidate bids where jobs move from one machine to another due to changes in the threshold computed in (7.4). Transitions between machines occur on intersections between the different values of $c_j q_i$ and $\frac{\sum_{k=1}^{i} q_k}{\sum_{l=1}^{j} \frac{1}{c_l}}$. Since q_{-i} is fixed, the intersections occur between a polynomial number of linear functions of the forms xq_i and $\frac{y+q_i}{z}$, resulting in a polynomial number of intersections points, which are candidates for critical bids.

7.5 A Double Sided Monotone Approximation

The main result of this section is a deterministic algorithm, which is monotone from both the perspectives of both machine agents and job agents. Algorithm *Monotone-RF (Monotone Rounded Fractional)*, shown in Figure 7.3, is shown to be a 5-approximation algorithm.

The majority of the effort in the analysis of *Monotone-RF* is to prove that it is monotone with respect to the machine agents, meaning that machine agents observe a non-increasing work-curve. An important property of the valid fractional assignment used in step 4 of the algorithm is the monotonicity of T^f: as we increase the speed of a machine, T^f is not increased. Let $T^f((b_{-j}, b_j), q)$ be the the smallest threshold for which there exist a valid fractional assignment, given the bids vectors (b_{-j}, b_j) and q.

Observation 7.5.1 *[12]* $T^f((b_{-j}, \alpha b_j), q) \leq \alpha T^f((b_{-j}, b_j), q)$ *for all* $\alpha > 1$ *and* $j \in [m]$.

Note that an index of a machine in vector d can be different than the one in vector b due to the method of breaking ties in step 2 of the algorithm. From now one we refer to indices of the machines in vector d only.

Lemma 7.5.2 *For any machine j which is not the fastest ($j > 1$), for any rounded machines bids vector d, for any jobs bid vector q, and for all $\beta \geq d_j$:*

$$T^f((d_{-j}, d_j), q) \leq T^f((d_{-j}, \beta), q) \leq \frac{5}{4} T^f((d_{-j}, d_j), q)$$

Input: a bid vector of jobs q, and a non-decreasing sorted bid vector of machines $b = (b_1, b_2, \ldots, b_m)$.

Output: an allocation of the jobs to the m machines.

1. Set $d_1 = \frac{5}{8} b_1$.
2. For $j \geq 2$, round up the bids to the closest value of $b_1 \cdot 2.5^i$, which is larger than the original bid, i.e. $d_j = b_1 \cdot 2.5^i$, where $i = \left\lfloor \log_{2.5} \frac{b_j}{b_1} \right\rfloor + 1$. Order d_j in non-decreasing order, breaking ties by an external machine-id.
3. Sort the job sequence in non-increasing order.
4. Calculate $T^f(d, q)$, the minimum value for which a valid fractional assignment exists, according to (7.4).
5. For each machine $j = 1, \ldots, m$, assign jobs in non-increasing order of job-size to machine j (using bid d_j), until machine j exceeds threshold $T^f(d, q)$ or equals it.
6. Return the assignment.

Figure 7.3: Monotone-RF

Proof: The first inequality is straight-forward as the allocation for (d_{-j}, β) is also a valid fractional assignment for (d_{-j}, d_j), given any fixed threshold T.

As for the second inequality, we generate a valid fractional assignment which allocates zero work to machine j. This allocation would use a threshold at most $\frac{5}{4} T^f(d_{-j}, d_j)$. Since this allocation is a valid fractional assignment for (d_{-j}, β), the minimal threshold for (d_{-j}, β) might only be smaller than the generated one.

To form an allocation which does not use machine j, for every $2 \leq k \leq i$ take all the pieces previously assigned to machine k and assign them to machine $(k-1)$. The first machine is now allocated the pieces originally assigned to the second machine, along with its original assignment. Since the algorithm assures that $4d_1 \leq d_2$, the assignment is clearly valid, with a threshold which is at most $\frac{5}{4} T^f(d_{-j}, d_j)$. ∎

We note that Lemma 7.5.2 holds for rounded bids vectors created by *Monotone-RF*, but does not hold in general. The following lemmata consider several scenarios in which machine j increases its bid (slows down). We denote by d' the rounded bid vector obtained after the machine's bid-increase. Let j' be the index of the bid-increasing machine in d'; notice that j' might be different than j. We denote by w'_k the total work allocated to machine j given the new bids vector d'. We denote by v_j the rounded speed of machine i, i.e. $v_j = 1/d_j$.

Lemma 7.5.3 *Using Algorithm Monotone-RF, when machine j which is not the fastest increases its bid, the total amount of work assigned to the machines faster than it cannot*

decrease, i.e.

$$\sum_{k=1}^{j-1} w_k \leq \sum_{k=1}^{j-1} w'_k \leq \sum_{k=1}^{j'-1} w'_k.$$

Proof: If the rounded bid of machine j is the same as before the bid-increase, the assignment is not changed. Thus we consider the case where the new rounded bid is different than the one before the bid-increase. Let β be the rounded bid of machine j where $\beta > d_j$. Let j' be the new index of the bid-increasing machine in d'. Clearly $j \leq j'$.

By Lemma 7.5.2, $T^f((d_{-j}, \beta), q) \geq T^f((d_{-j}, d_j), q)$, i.e. the new threshold used by algorithm *Monotone-RF* can only increase due to the bid-increase. By induction the index of the last job assigned to each of the first $j-1$ machines can be equal, or higher after the bid-increase. Thus the total amount of work assigned to the first $j-1$ machines is the same or higher, and the amount of work assigned to the first $j'-1$ machines can only be higher than that. ∎

Lemma 7.5.4 *If the fastest machine increases its bid yet remains the fastest, the amount of work assigned to it can only decrease.*

Proof: We observe how the bin size of the first machine changes as its bid increases gradually. As long as for all $k \geq 2$ the value of $\lfloor \log_{2.5} \frac{b_k}{b_1} \rfloor$ does not change, all rounded speeds change proportionally, i.e. there is some constant $\alpha > 1$ such that $d' = \alpha \cdot d$. Therefore, the same fractional assignment is calculated (with a new threshold of $\alpha T^f(d', q) = T^f(d, q)$) with the same sizes for bins. In case $\lfloor \log_{2.5} \frac{b_k}{b_1} \rfloor$ changes for some k, i.e. the rounded bid of at least one machine is cut down by 2.5, by Observation 7.5.1 the threshold cannot increase, and therefore the bin size of the first machine can only decrease.

Since the fastest machine is always assigned the first jobs, a decrease in its bin size can only decrease the number of jobs assigned to it, and therefore the amount of work assigned to it in the integral assignment also decreases. ∎

Definition 7.5.5 *Given an assignment of jobs by algorithm Monotone-RF, we classify the machines in the following way:*

- Full machine - *a machine (bin) which is assigned with jobs whose total processing time is at least the machine's threshold.*
- Empty machine - *a machine with no jobs allocated to it.*
- Partially-full machine - *a non-empty machine (bin) which is not full. (There is at most one partially-full machine).*

Lemma 7.5.6 *When machine j decreases its speed (increases its bid) the total work allocated to it by algorithm Monotone-RF cannot increase.*

Proof: Lemma 7.5.4 proves the claim when machine j is the fastest machine and remains the fastest. If machine j is not the fastest machine but its rounded bid d_j does not change, then the bid-increase has no effect since the same assignment is generated. It remains to prove the claim for the breakpoints: when the bid value of the fastest machine becomes larger than b_2 and when the rounded bid is multiplied by 2.5. We prove the claim for each step, thus the claim holds for the entire bid-increase.

Consider the following cases for the class of machine j before the bid-increase:

1. Machine j is the fastest machine ($j = 1$), but after the bid-increase another machine k becomes the fastest. We observe the breakpoint where both machines have the same bid and add an artificial stage to the bid-increase where the title of the fastest machine passes from j to k (without having the bids change). The same threshold is calculated in both cases, only the order of the machines changes. The amount of work allocated to machine j when it is considered the fastest is at least $\frac{8}{5} \cdot v_1 \cdot T^f(d, q)$, while after machine k becomes the fastest it is at most $2 \cdot \frac{v_1}{2.5} \cdot T^f(d, q) = \frac{4}{5} \cdot v_1 \cdot T^f(d, q)$, and therefore decreases.

2. Machine j is a full machine which is not the fastest. The threshold used for assigning jobs to the machine is $T^f(d, q)$. Due to Lemma 7.5.2, $T^f(d, q) \leq T^f(d', q) \leq \frac{5}{4} T^f(d, q)$, where d is the rounded bids vector, and d' is the rounded vector after the bid-increase. Before the bid-increase the amount of work allocated to it was at least $T^f(d, q) \cdot v_j$, whereas after increasing the bid it can become at most $2 \cdot (\frac{5}{4} \cdot T^f(d, q)) \cdot \frac{v_j}{2.5} = T^f(d, q) \cdot v_j$. If the machine became partially-full or empty after increasing its bid, the amount of work allocated to it can only be smaller.

3. Machine j is partially-full. If it becomes empty then the claim is trivial, otherwise some jobs are allocated to machine j. Let $j' \geq j$ be the new index of the machine in the sorted order. Due to Lemma 7.5.3 the amount of work allocated to machines with a lower index than j' can be no less than the amount before the bid-increase (i.e. $\sum_{k=1}^{j-1} w_k \leq \sum_{k=1}^{j'-1} w'_k$). Since machines $j+1 \dots m$ are empty before the bid-increase, machine j' is left with at most the same amount of work as machine j had.

4. Machine j is empty. The machine stays empty due to Lemma 7.5.3.

■

Lemma 7.5.6 shows that the work-curve of machine agent j is non increasing. Hence, the following theorem is immediate:

Theorem 7.5.7 *Algorithm Monotone-RF provides a monotone assignment, with respect to machine agents. Hence a mechanism design for strategic machine owners based on algorithm Monotone-RF and payment scheme as in (7.1) is truthful.*

We now prove that *Monotone-RF* is also monotone from the perspective of job agents, achieving a non-decreasing speed-curve:

Theorem 7.5.8 *Algorithm Monotone-RF provides a monotone assignment, with respect to job agents. Hence a mechanism design for strategic job owners based on algorithm Monotone-RF and payment scheme as in (7.3) is ex-post truthful.*

Proof: The proof is nearly the same as the proof of Theorem 7.4.1. For every job q_i, holding q_{-i} and d fixed, the threshold $T^f(d,(q_{-i},q_i))$ is monotone in q_i. The i-th job observes a non-decreasing speed-curve due to both the monotonicity of the threshold and its advancement toward the beginning of the sorted bid vector q, which determines the order of the allocation of the jobs. ∎

From the last two theorems it immediately follows that a there is a mechanism that applies *Monotone-RF* and is ex-post truthful for both machine agents and job agents.

We now analyze the approximation provided by algorithm *Monotone-RF*.

Denote by $k^f(j)$ the index of the last job (or a fraction of a job) assigned to machine j in the fractional assignment. Respectively let $k(j)$ be the index of the last job assigned to machine j by *Monotone-RF*.

Lemma 7.5.9 *For all j, $k^f(j) \leq k(j)$.*

Proof: By induction. The claim clearly holds for $j = 1$ since due to step 5, the total processing time of jobs assigned to the this machine is at least $T^f(d,q)$. Assume the argument is true for machine j. By induction hypothesis $k^f(j) \leq k(j)$. Since allocation is done in non-increasing order of job size, the first job to be allocated to $j+1$ by our algorithm might be only smaller than the one allocated by the fractional assignment. Moreover, since the allocation exceeds the fractional threshold, at least the same number of jobs will be assigned to machine j. Thus $k^f(j+1) \leq k(j+1)$. ∎

Theorem 7.5.10 *Algorithm Monotone-RF computes a 5-approximation.*

Proof: Lemma 7.5.9 assures that when algorithm *Monotone-RF* terminates, all jobs are allocated. Since the speeds were decreased by at most a factor of 2.5, the threshold $T^f(d,q)$ may be at most 2.5 times the value of the optimal allocation using the unrounded speeds. Since the speed of the fastest machine is increased by a factor of 1.6, the amount of work assigned to the fastest machine in the fractional solution may be at most $1.6 \cdot 2.5 = 4$ times the value of the optimal allocation.

In the integral solution, since the amount of work assigned to the first machine can exceed the bin's size by at most the size of the second bin, and since the first bin is at least 4 times larger than the second bin, the load on the fastest machine can be at most $1.25 \cdot T^f(d,q)$, and therefore the load on this machine is at most $1.25 \cdot 4 = 5$ times the optimal. For any other machine, the last job can exceed the threshold by at most $T^f(d,q)$, and therefore the load on any other machine is at most $2 \cdot T^f(d,q)$, which is at most $2 \cdot 2.5 = 5$ times the optimal. Therefore, a 5-approximation is achieved. ∎

To conclude, we need to show that calculating the payments given by (7.1) and by (7.3) can be accomplished in polynomial time.

Consider the expression in (7.1): We analyze the number of points (breakpoints) where the value of the work-curve changes in that expression. According to Lemma 7.5.9 the

work curve for machine j becomes zero furthermost when the valid fractional assignment does not use machine j. There is no use in assigning jobs to a machine when its bid β is too high even for the smallest job, i.e. $\beta q_n > T^f(d,q)$. Using the upper bound $T^f(d,q) \leq nq_1d_1 < nq_1b_1$, we get a zero assignment for $\beta \geq n\frac{q_1}{q_n}b_1$. The only exception is when the integral is calculated for the fastest machine, where we get a higher bound of $\beta \geq n\frac{q_1}{q_n}b_2$. While $\beta \leq b_2$, there is a breakpoint whenever $b_k = 2.5^i\beta$, for some i and for any machine $k > 1$. Therefore, for each factor of 2.5, there are at most $m-1$ breakpoints (one for each of the other machines), while for $\beta > b_2$, there is one breakpoint for each step.

Thus the number of iterations is $O\left(\log_{2.5}(n\frac{q_1}{q_n}) + m\log_{2.5}(\frac{b_2}{b_1})\right)$ for the fastest machine, and $O\left(m\log_{2.5}(n\frac{q_1}{q_n}\frac{b_2}{b_1})\right)$ for the rest of the machines all together. Hence the computation of (7.1) is polynomial in the input size.

We proceed to analyze the computation time of (7.3): Similarly to the calculation of the payment described in Section 7.4, for the i-th job, holding q_{-i} and d fixed, we can calculate exactly $T^f(d,(q_{-i},q_i))$, which is constructed from a polynomial sized set of intervals of linear functions of the forms xq_i and $\frac{y+q_i}{z}$. Unlike Section 7.4, transitions of jobs between machines may occur within the intervals and not only on intersection points, since the integral allocation of *Monotone-RF* is not merely rounding the fractional allocation. However, for any value of T^f within a given interval, we can efficiently find the next value where the integral allocation changes, (and the i-th job potentially moves to a faster machine), by comparing T^f to the load on each machine, and then compute the value of q_i for which this threshold value is attained.

It remains to prove that we need to compute only a polynomial number of additional changes in the integral allocation: Since we are only interested in the assignment of the i-th job, it is sufficient to only compute the allocation of the first i jobs. As q_i increases, the i-th job observes a monotone speed-curve, due to the increase in T^f. Moreover, for the same reason, the larger $i-1$ jobs also observe a monotone speed-curve, as long as they remain larger than q_i, which is guaranteed within a given interval. Since each job can move at most $m-1$ times while observing a non-decreasing speed-curve, we need to compute an additional number of only $O(nm)$ allocations.

Bibliography

[1] `https://adwords.google.com`.

[2] William Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosén. Competitive queue policies for differentiated services. In *Proceedings of IEEE INFOCOM, The 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 431–440, 2000.

[3] Susanne Albers and Markus Schmidt. On the performance of greedy algorithms in packet buffering. *SIAM Journal on Computing*, 35(2):278–304, 2005.

[4] `http://www.amazon.com`.

[5] Nir Andelman. Randomized qeeue management for diffserv. In *Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pages 1–10, 2005.

[6] Nir Andelman, Yossi Azar, and Motti Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS), Proceedings*, pages 69–82, 2005.

[7] Nir Andelman and Yishay Mansour. Competitive management of non-preemptive queues with multiple values. In *Distributed Computing, 17th International Conference (DISC), Proceedings*, pages 166–180, 2003.

[8] Nir Andelman and Yishay Mansour. Auctions with budget constraints. In *9th Scandinavian Workshop on Algorithm Theory (SWAT), Proceedings*, pages 26–38, 2004.

[9] Nir Andelman and Yishay Mansour. A sufficient condition for truthfulness with single parameter agents. In *Proceedings 7th ACM Conference on Electronic Commerce (EC)*, pages 8–17, 2006.

[10] Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for qos switches. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 761–770, 2003.

[11] Aaron Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, 2004.

[12] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *42nd Annual Symposium on Foundations of Computer Science (FOCS), Proceedings*, pages 482–491, 2001.

[13] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS), Proceedings*, pages 608–619, 2004.

[14] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. The power of verification for one-parameter agents. In *Automata, Languages and Programming: 31st International Colloquium (ICALP), Proceedings*, pages 171–182, 2004.

[15] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Pino Persiano. How to route and tax selfish unsplittable traffic. In *Proceedings of the 16th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pages 196–205, 2004.

[16] Yossi Azar and Arik Litichevskey. Maximizing throughput in multi-queue switches. In *Algorithms, 12th Annual European Symposium (ESA), Proceedings*, pages 53–64, 2004.

[17] Yossi Azar and Yossi Richter. Management of multi-queue switches in qos networks. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 82–89, 2003.

[18] Yossi Azar and Yossi Richter. An improved algorithm for cioq switches. In *Algorithms, 12th Annual European Symposium (ESA), Proceedings*, pages 65–76, 2004.

[19] Yossi Azar and Yossi Richter. The zero-one principle for switching networks. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 64–71, 2004.

[20] Nikhil Bansal, Lisa Fleischer, Tracy Kimbrel, Mohammad Mahdian, Baruch Schieber, and Maxim Sviridenko. Further improvements in competitive guarantees for qos buffering. In *Automata, Languages and Programming: 31st International Colloquium (ICALP), Proceedings*, pages 196–207, 2004.

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W Weiss. An architecture for differentiated services. RFC 2475, IETF, December 1998.

[22] Liad Blumrosen and Noam Nisan. On the computational power of iterative auctions. In *Proceedings 6th ACM Conference on Electronic Commerce (EC)*, pages 29–43, 2005.

[23] Alan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[24] Chandra Chekuri and Sanjeev Khanna. A ptas for the multiple knapsack problem. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 213–222, 2000.

[25] D. Clark and J. Wroclawski. An approach to service allocation in the internet. Internet-Draft Version 00, IETF, July 1997.

[26] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[27] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, 2001.

[28] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1064–1073, 2006.

[29] `http://www.ebay.com`.

[30] Matthias Englert and Matthias Westermann. Lower and upper bounds on fifo buffer management in qos switches. In *Algorithms, 14th Annual European Symposium (ESA), Proceedings*, 2006. To appear.

[31] Leah Epstein and Jiri Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *Algorithms, 7th Annual European Symposium (ESA), Proceedings*, pages 151–162, 1999.

[32] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *47th Symposium on Foundations of Computer Science (FOCS), Proceedings*, 2006. To appear.

[33] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 611–620, 2006.

[34] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[35] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, 1999.

[36] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman, San-Francisco, 1979.

[37] Teofilo F. Gonzalez, Oscar H. Ibarra, and Sartaj Sahni. Bounds for lpt schedules on uniform processors. *SIAM Journal on Computing*, 6(1):155–166, 1977.

[38] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[39] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[40] Hongwei Gui, Rudolf Müller, and Rakesh V. Vohra. Dominant strategy mechanisms with multidimensional types. In *Computing and Markets, Dagstuhl Seminar Proceedings*, 2005.

[41] Ellen L. Hahne, Alexander Kesselman, and Yishay Mansour. Competitve buffer management for shared-memory switches. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 53–58, 2001.

[42] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[43] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the Association for Computing Machinery*, 23:317–327, 1976.

[44] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[45] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenium Press, New York, 1972.

[46] Alexander Kesselman, Zvi Lotker, Yishay Mansour, and Boaz Patt-Shamir. Buffer overflows of merging streams. In *Algorithms, 11th Annual European Symposium, Proceedings (ESA)*, pages 349–360, 2003.

[47] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in qos switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[48] Alexander Kesselman and Yishay Mansour. Loss-bounded analysis for differentiated services. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms (SODA)*, pages 591–600, 2001.

[49] Alexander Kesselman and Yishay Mansour. Harmonic buffer management policy for shared memory switches. In *Proceedings IEEE INFOCOM, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[50] Alexander Kesselman, Yishay Mansour, and Rob van Stee. Improved competitive guarantees for qos buffering. In *Algorithms, 11th Annual European Symposium, Proceedings (ESA)*, pages 361–372, 2003.

[51] Annamária Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *Algorithms, 13th Annual European Symposium (ESA), Proceedings*, pages 616–627, 2005.

[52] Annamária Kovács. Tighter approximation bounds for lpt scheduling in two special cases. In *Algorithms and Complexity, 6th Italian Conference (CIAC), Proceedings*, pages 187–198, 2006.

[53] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *44th Symposium on Foundations of Computer Science (FOCS), Proceedings*, pages 574–583, 2003.

[54] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

[55] Benny Lehmann, Daniel J. Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings 3rd ACM Conference on Electronic Commerce (EC)*, pages 18–28, 2001.

[56] Daniel J. Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC)*, pages 96–102, 1999.

[57] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[58] Kevin Leyton-Brown, Yoav Shoham, and Moshe Tennenholtz. Bidding clubs in first-price auctions. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 373–378, 2002.

[59] Zvi Lotker and Boaz Patt-Shamir. Nearly optimal fifo buffer management for diffserv. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 134–142, 2002.

[60] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University press, 1995.

[61] James A. Mirrlees. Optimal tax theory: A synthesis. *Journal of Public Economics*, 6:327–358, 1976.

[62] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 379–384, 2002.

[63] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC)*, pages 1–12, 2000.

[64] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, pages 129–140, 1999.

[65] Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC)*, pages 242–252, 2000.

[66] Kevin Roberts. The characterization of implementable choice rules. In Jean Jacques Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–348. North Holland, Amsterdam, 1979.

[67] Jean Charles Rochet. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[68] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[69] Michael E. Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings 6th ACM Conference on Electronic Commerce (EC)*, pages 286–293, 2005.

[70] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 542–547, 1999.

[71] Markus Schmidt. Packet buffering: Randomization beats deterministic algorithms. In *22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS), Proceedings*, pages 293–304, 2005.

[72] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[73] Michael Spence. Competitive and optimal responses to signals: An analysis of efficiency and distribution. *Journal of Economic Theory*, 7:296–332, 1974.

[74] Maxim Sviridenko. A lower bound for on-line algorithms in the fifo model. Unpublished Manuscript, 2001.

[75] Moshe Tennenholtz. Tractable combinatorial auctions and b-matching. *Artificial Intelligence*, 140(1/2):231–243, 2002.

[76] William Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.