Online Resource Management

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
Doctor rerum naturalium
der Georg-August-Universität Göttingen

im Promotionsprogramm Mathematik
der Georg-August University School of Science (GAUSS)

vorgelegt von
Morten Tiedemann
aus Stade

Göttingen, 2015

Tag der mündlichen Prüfung: 16.04.2015

Contents

1 Introduction **1**
1.1 Preliminaries . . . 2
1.2 Outline and Contribution . . . 4
1.3 Credits . . . 7
1.4 Acknowledgement . . . 7

2 Online Knapsack Problems with Dynamic Capacity **9**
2.1 Introduction . . . 9
2.1.1 Previous Work . . . 10
2.2 Problem Definition . . . 11
2.3 Lower Bounds . . . 13
2.3.1 A Lower Bound for Deterministic Online Algorithms . . . 13
2.3.2 A Lower Bound for Randomized Online Algorithms . . . 15
2.4 Competitive Algorithms . . . 17
2.4.1 A Greedy Algorithm . . . 18
2.4.2 A Balancing Algorithm . . . 18
2.4.3 A Randomized Greedy Algorithm . . . 20
2.5 Limited Weights . . . 24
2.5.1 Deterministic Online Algorithms . . . 25
2.5.2 Randomized Online Algorithms . . . 27
2.6 Increasing the Power of the Online Player . . . 28
2.6.1 Resource Augmentation . . . 29
2.6.2 Removable Items . . . 30
2.6.3 Bounded Values . . . 43
2.7 Empirical Analysis . . . 45
2.8 Extension to Multiple Knapsacks . . . 49
2.9 Independent Knapsacks . . . 50

2.9.1 A Lower Bound for Deterministic Online Algorithms 52
2.9.2 Competitive Online Algorithms 53
2.10 Conclusion and Future Research 56

3 Multi-Objective Online Optimization 57
3.1 Introduction 57
3.1.1 Previous Work 57
3.2 Competitive Analysis for Multi-Objective Online Algorithms 58
3.2.1 Multi-Objective Online Problems 59
3.2.2 The Competitive Ratio and Competitiveness 59
3.3 The Multi-Objective Time Series Search Problem 62
3.3.1 Worst-Component Competitive Analysis 63
3.3.2 Mean-Component Competitive Analysis 70
3.3.3 A Randomized Bi-Objective Online Algorithm 73
3.4 The Bi-Objective Ski Rental Problem 81
3.5 The Bi-Objective 2-Server Problem in the Plane 87
3.6 The Multi-Objective k-Canadian Traveller Problem 94
3.7 Relations between Single- and Multi-Objective Online Problems 99
3.8 Yao's Principle for Multi-Objective Online Problems 104
3.9 Conclusion and Future Research 109

4 The Linear Search Problem with Turn Costs 111
4.1 Introduction 111
4.2 Tradeoff between Competitive Factor and Turn Costs 112
4.3 Conclusion 117

5 Optimization in the Wood Cutting Industry 119
5.1 Introduction 119
5.2 The Cutting Problem 121
5.2.1 Classification of the Cutting Problem 123
5.3 Modeling the Deterministic Cutting Problem 124
5.3.1 Variables and Constraints 124
5.3.2 Objectives 128
5.3.3 Computational Results 130
5.4 Robust Cutting Patterns 131
5.4.1 Uncertain Multi-Objective Optimization 131
5.4.2 A Simplified Version of the Multi-Objective Cutting Problem . . . 133
5.4.3 Minmax Robust Efficiency Applied to the Optimiziation Model . . 135
5.5 Conclusion and Future Research 139

Bibliography 141

1 Introduction

Decision making is part of our daily routine. Some decisions are easy to make, others carry you off into an energy-sapping tedious decision making process; and still others just have to be made, even though you cannot decide which option is the best.

Imagine you want to sell your car and you are facing a sequence of offers by potential buyers. You have to accept or reject an offer immediately since the potential buyers are not willing to wait for a decision at a later time and your goal is to maximize the selling prize for your car. The decision about each offer is a double-edged sword: If you decide to reject an offer, you gain the possibility to come across an offer with a higher price, but you bear the risk that all following offers feature a lower price. Otherwise, if you decide to accept an offer, you obtain the offered price, but you miss out on the opportunity of better offers in the future.

The decision making problem presented above features certain characteristics: For one thing, a decision about the optimal use of a resource (the car) has to be reached. For another thing, the decision has to be finalized without knowledge of future events (if you would know all offers and the corresponding prices in advance, the optimal decision would be to accept the highest price).

If input data for a decision problem is modeled as a sequence that is revealed step by step and decisions have to be made without any knowledge of future inputs, the decision problem is referred to as an *online problem* and an algorithm for such a problem is called *online algorithm*. In this thesis, the well-established concept of *competitive analysis* is applied in order to measure the quality of an online algorithm. Here, the quality of an online algorithm on each input sequence is given by comparing its objective value to the objective value of an optimal algorithm that has complete knowledge of the input sequence in advance.

Many resource management problems are indeed online problems in that they require immediate decisions based on uncertain data, such as stock trading, portfolio selection, routing, or search problems. By means of this thesis, we aim at providing better insight into decision processes related to resource management problems without secure information about the future, i.e., *online resource management*.

1.1 Preliminaries

In this section, we give an introduction to the field of online optimization, based on (Borodin and El-Yaniv, 1998), and provide some basic definitions used throughout this thesis. Other definitions, for example, definitions related to the field of multi-objective optimization, are given when necessary in the corresponding chapters.

When considering a mathematical optimization problem, usually complete knowledge of all input data is assumed. However, data is often entailed with uncertainty and decisions have to be made without complete knowledge of all necessary information. In *online optimization*, input data is modeled as a sequence that is revealed step by step to an algorithm. With each new bit of information, the algorithm has to make a decision that will have an impact on the quality of its overall solution. Additionally, each decision must be made without knowledge of future items of the input sequence. Such an algorithm is labeled as *online algorithm*.

The analysis of online algorithms is mainly motivated by the question: "Which is the better (or the best) algorithm for a given problem?". The concept of *competitive analysis* has become a well-established theory for measuring the quality of an online algorithm. Here, the quality of an online algorithm on each input sequence is given by comparing its objective value to the objective value of an *optimal offline algorithm* that has complete knowledge of the input sequence in advance. Since all input sequences are taken into account in order to measure the quality of an online algorithm, competitive analysis is a worst-case measure.

We proceed by giving the formal definition of a competitive online algorithm for a minimization problem. The cost of an online algorithm ALG associated with the input sequence σ is denoted by $\text{ALG}(\sigma)$ and the cost of an optimal offline algorithm on the input sequence σ is denoted by $\text{OPT}(\sigma)$.

Definition 1.1.1 (Deterministic Competitive Algorithm). *A deterministic online algorithm* ALG *is called* c-competitive *for a constant* $c \geq 1$ *if there is a constant* α *such that*

$$\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma) + \alpha$$

for all request sequences σ*.*

Definition 1.1.1 is given for minimization problems. For maximization problems, the inequality in Definition 1.1.1 is replaced by $\text{ALG}(\sigma) \geq {}^1\!/_c \cdot \text{OPT}(\sigma) + \alpha$. The *competitive ratio* of a deterministic online algorithm is defined as the infimum over all c such that the algorithm is c-competitive.

An illustrative depiction of analyzing online algorithms is given by viewing the problem as a game between an online player (representing the online algorithm) and a malicious adversary (representing the optimal offline algorithm). The online player runs the algorithm on a sequence presented by the adversary, whereby the adversary chooses the sequence in order to maximize the competitive ratio.

In order to enhance the competitive edge of the online player, the online algorithm can be based on random decisions: A *randomized online algorithm* is a probability distribution over deterministic algorithms. The solution quality obtained by a randomized online algorithm is measured by considering its competitive ratio against an *oblivious adversary*, who must construct the input sequence in advance based only on the description of the online algorithm but before any moves are made:

Definition 1.1.2 (Competitive Algorithm against an Oblivious Adversary). *A randomized online algorithm* ALG*, distributed over a set* $\{\text{ALG}_y\}$ *of deterministic algorithms, is* c-competitive (against an oblivious adversary) *for a constant* $c \geq 1$ *if there is a constant* α *such that*

$$\mathbb{E}_Y\left[\text{ALG}_y(\sigma)\right] \leq c \cdot \text{OPT}(\sigma) + \alpha$$

for all request sequences σ*. Here,* $\mathbb{E}_Y[\cdot]$ *denotes the expectation with respect to the probability distribution* Y *over* $\{\text{ALG}_y\}$ *that defines* ALG*.*

As above, Definition 1.1.2 is given for minimization problems. For maximization problems, the inequality in Definition 1.1.2 is replaced by $\mathbb{E}_Y[\text{ALG}_y(\sigma)] \geq {}^1/_c \cdot \text{OPT}(\sigma) + \alpha$. The *competitive ratio* of a randomized online algorithm is analogously defined as the infimum over all c such that the algorithm is c-competitive (against an oblivious adversary) and is denoted by $\overline{\mathcal{R}}_{\text{OBL}}\left(\text{ALG}\right)$.

The competitive ratio of a randomized online algorithm depends on the adversary's knowledge of the randomized decisions of the online algorithm and the adversary's ability to react to them. In addition to the model of the oblivious adversary, there are two further well-known adversary models in the literature of online optimization, namely the adaptive-online and the adaptive-offline adversary: The adaptive-online adversary chooses the next request based on the online algorithm's decisions so far and computes its solution in an online manner, whereas the adaptive-offline adversary also chooses the next request based on the online algorithm's decisions so far, but computes its solution for the complete resulting sequence. However, in this thesis, we only consider the oblivious adversary.

For the computation of lower bounds on the competitive ratio of any randomized online algorithm, we make use of *Yao's principle.* According to this, it suffices to choose a distribution over input sequences and show that no *deterministic* online algorithm performs well in expectation in order to establish a lower bound on the competitive ratio of any randomized online algorithm. In the following, we formally state Yao's principle for minimization problems. The proof is given in Yao (1977).

Theorem 1.1.1 (Yao's Principle for Minimization Problems (Yao, 1977)). *Let* ALG *be any randomized online algorithm for an online minimization problem. Let* $\{\text{ALG}_j : j \in \mathcal{J}\}$ *denote the set of all deterministic online algorithms for this problem and let* p *be any probability distribution over request sequences* $\{\sigma_i : i \in \mathcal{I}\}$*. Then,*

$$\overline{\mathcal{R}}_{\text{OBL}}\left(\text{ALG}\right) \geq \max\left\{\min_{j\in\mathcal{J}} \frac{\mathbb{E}_p[\text{ALG}_j(\sigma_i)]}{\mathbb{E}_p[\text{OPT}(\sigma_i)]}, \min_{j\in\mathcal{J}} \mathbb{E}_p\left[\frac{\text{ALG}_j(\sigma_i)}{\text{OPT}(\sigma_i)}\right]\right\}, \tag{1.1}$$

where $\overline{\mathcal{R}}_{\text{OBL}}(\text{ALG})$ *is* ALG*'s competitive ratio against an oblivious adversary.*

For maximization problems, the theorem is slightly different:

Theorem 1.1.2 (Yao's Principle for Maximization Problems (Yao, 1977))**.** *Let* ALG *be any randomized online algorithm for an online maximization problem. Let* $\{\text{ALG}_j : j \in \mathcal{J}\}$ *denote the set of all deterministic online algorithms for this problem and let* p *be any probability distribution over request sequences* $\{\sigma_i : i \in \mathcal{I}\}$*. Then,*

$$\overline{\mathcal{R}}_{\text{OBL}}(\text{ALG}) \geq \max\left\{ \min_{j\in\mathcal{J}} \frac{\mathbb{E}_p[\text{OPT}(\sigma_i)]}{\mathbb{E}_p[\text{ALG}_j(\sigma_i)]}, \min_{j\in\mathcal{J}} \frac{1}{\mathbb{E}_p\left[\frac{\text{ALG}_j(\sigma_i)}{\text{OPT}(\sigma_i)}\right]} \right\}, \tag{1.2}$$

where $\overline{\mathcal{R}}_{\text{OBL}}(\text{ALG})$ *is* ALG*'s competitive ratio against an oblivious adversary.*

1.2 Outline and Contribution

In the following, we give an outline of this thesis and point out the main contributions of each chapter.

In Chapter 2, we consider an online knapsack problem with incremental capacity. In each time period, a set of items, each with a specific weight and value, is revealed and, without knowledge of future items, it has to be decided which of these items to accept. Additionally, the knapsack capacity is not fully available from the start but increases by a constant amount in each time period. The goal is to maximize the overall value of the accepted items. This setting extends the basic online knapsack problem by introducing a dynamic instead of a static knapsack capacity and is applicable to classic problems such as resource allocation or one-way trading.

For the case of unit weight requests and unit incremental capacity (i.e., one additional unit of capacity in each time period), we give a deterministic T-competitive online algorithm and a matching lower bound on the competitive ratio of any deterministic online algorithm. For unit weights and k-incremental capacity (where $k \geq 2$ additional units of capacity become available in each time period), a deterministic $(T+1)k/(2k-1)$-competitive algorithm is proposed that approaches the lower bound on the competitive ratio of any randomized (and deterministic) algorithm for $k \to \infty$. Moreover, a best possible randomized algorithm with a competitive ratio of $(T+1)/2$ is developed.

For the case that general nonnegative weights are allowed, we show that no competitive online algorithm exists for the problem. However, for limited weights in $\{1, \ldots, k\}$ and k-incremental capacity, we present a competitive deterministic online algorithm and a lower bound on the competitive ratio of any deterministic online algorithm that approaches the competitive ratio of the proposed algorithm for $k \to \infty$.[1] For the randomized case, we present a $3\,((T+1)/2)$-competitive online algorithm matching the lower bound of

[1]Note that, for unit incremental capacity $k = 1$, the case of limited weights in $\{1, \ldots, k\}$ coincides with the unit weight case, so the results shown for unit weights carry over to the limited weight setting if $k = 1$.

		unit weights		limited weights		
		$k=1$	$k \geq 2$	$k=1$	$k \geq 2$	removable items
det.	LB	T	$\frac{T+1}{2}$	T	$\left\lfloor \frac{Tk}{\lfloor \frac{k}{2} \rfloor + 1} \right\rfloor$	$\sqrt{2}$
	UB	T	$\frac{(T+1)k}{2k-1}$	T	$2T-1$	3
rand.	LB	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	–
	UB	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$\frac{T+1}{2}$	$3\left(\frac{T+1}{2}\right)$	–

Table 1.1: Main results for the online knapsack with incremental capacities: deterministic and randomized lower and upper bounds.

$(T+1)/2$ up to a factor of 3. The results for the online knapsack problem with incremental capacity are summarized in Table 1.1.

In order to achieve algorithms with competitive ratios independent of T, we study different settings in which the power of the online player is increased. In the setting of resource augmentation, the online player is allowed to use more resources than the adversary. We show that, if there exists a c-competitive algorithm for the online knapsack problem with incremental capacity, then resource augmentation by a multiplicative factor of r allows for a $(1 + c/r)$-competitive algorithm. Then, we discuss the setting of removable items, i.e., the online player is entitled to remove previously accepted items from the knapsack in any time period, and present a 3-competitive deterministic algorithm and a lower bound of $\sqrt{2}$ on the competitive ratio of any deterministic online algorithm. Furthermore, we analyze the setting of bounded values for the online knapsack problem with incremental capacity and unit weights, i.e., $v_i \in [m, M]$, and obtain a deterministic $\sqrt{M/m}$-competitive algorithm.

In addition to these results for the single knapsack case, we show that all our algorithms can be extended generically to multiple knapsacks while only increasing their competitiveness by one.

Finally, we study the setting of independent knapsacks with differing capacities, i.e., in each time period a certain capacity is available independently of the available capacity of other time periods and the accepted requests in other time periods. For this setting, we present a 2-competitive algorithm as well as a lower bound on the competitive ratio of any deterministic algorithm of $(1+\sqrt{5})/2$.

In Chapter 3, we introduce a general framework for the competitive analysis of multi-objective online problems which expands the known theory of competitive analysis for single-objective online problems to the multi-objective case. The fact that solutions to multi-objective optimization problems are sets rather than singletons as in the case of single-objective optimization problems requires a proper adaptation of the definition of competitiveness for multi-objective online problems.

For $c = \big(c_1, \ldots, c_n\big)^\mathsf{T}$, we define a multi-objective online algorithm ALG as c-competitive if, for each input sequence, there exists an efficient solution to the offline problem for which ALG is c_i-competitive (in the original sense) in the i-th component for $i = 1, \ldots, n$. Additionally, a multi-objective online algorithm ALG is labeled as *strongly* c-competitive if, for each input sequence, ALG is c_i-competitive (in the original sense) in the i-th component, for $i = 1, \ldots, n$, for *all* efficient solutions to the offline problem. For $f : \mathbb{R}^n \to \mathbb{R}$, the infimum over the set of all values $f(c)$ such that ALG is (strongly) c-competitive is then defined as the *(strong) competitive ratio with respect to* f of ALG.

Furthermore, we analyze the multi-objective time series search problem by means of the introduced notions of competitiveness with respect to functions $f_1(c) = \max_{i=1,\ldots,n} c_i$, $f_2(c) = \frac{1}{n}\sum_{i=1}^n c_i$, and $f_3(c) = \sqrt[n]{\prod_{i=1}^n c_i}$, and present deterministic algorithms featuring the best possible (strong) competitive ratios. Additionally, a randomized algorithm that improves on the best possible deterministic competitive ratio is presented.

The concept of competitive analysis of multi-objective online problems is further applied to multi-objective extensions of some well-known online problems, namely a bi-objective variant of the ski rental problem, the bi-objective 2-server problem in the plane, and the multi-objective k-Canadian traveller problem. We present multi-objective deterministic online algorithms for these problems featuring best possible deterministic competitive ratios. Finally, we discuss general relations between single- and multi-objective online problems and extend Yao's principle to multi-objective online problems.

In Chapter 4, we consider the linear search problem with turn costs: A searcher starts from the origin and tries to find an immobile object located on the real line in minimum time. Each time the searcher changes direction, a cost of d is incurred. Demaine et al. (2006) consider the sum of searching time and turn cost as objective function and present an algorithm which guarantees a solution smaller than $9 \cdot \text{OPT} + 2d$. The additive term $2d$ is minimal subject to the (optimal) competitive ratio 9. As mentioned in (Demaine et al., 2006), it may be desirable to improve the additive term, while allowing an increase in the competitive ratio. The determining tradeoff curve is obtained experimentally, but it is not characterized analytically. We present an analytical characterization of the tradeoff curve between the competitive factor and the turn cost and close this gap.

In Chapter 5, we present a real-world optimization problem, namely a cutting problem arising in the veneer industry. This problem features uncertainty in the input data but, for the sake of practicability, we depart from the concept of online optimization. We present methods to solve this problem by deterministic and (in a simplified version) robust optimization. We classify the problem, present a detailed single-objective optimization model and discuss the uncertainties in the problem formulation. Furthermore, we point out that these uncertainties are a result of the varying wood quality and describe the various factors influencing this quality. We present a quality distribution obtained from the experience of the manufacturer and provide computational results of the described problem. Finally, we present a simplified, yet multi-objective version of the optimization problem and discuss the uncertainties in this formulation. In order to hedge against these uncertainties, the concept of minmax robust efficiency is applied to this simplified version and robust efficient solutions to this problem are computed.

1.3 Credits

In the following, results accomplished in cooperation with other researchers are outlined:

The first ideas for the online knapsack problem with incremental capacity presented in Chapter 2 are due to a collaboration with Clemens Thielen and Stephan Westphal. The elaborations are individual work of the author. Most of the results from Chapter 2 have been submitted to *Mathematical Methods of Operations Research* (Thielen et al., 2015). Moreover, the implementation of the algorithms for the empirical analysis presented in Section 2.7 is cooperative work with Martin Dahmen.

The development of the concept of multi-objective online optimization as well as the analysis of the multi-objective time series search presented in Chapter 3 are joint work with Jonas Ide and Anita Schöbel and are to appear in the *Proceedings of the 9th Workshop on Algorithms and Computations* (Tiedemann et al., 2015). The remaining results given in Chapter 3 are the author's sole work.

The results on the linear search problem with turn cost presented in Chapter 4 are individual work of the author and have been submitted to *Information Processing Letters* (Tiedemann, 2015)

Finally, the application of deterministic and robust optimization to a problem in the wood cutting industry studied in Chapter 5 is joint work with Jonas Ide, Felix Haiduk, and Stephan Westphal and is published in *4OR* (Ide et al., 2015). While Stephan Westphal gave a first idea on the modeling of the optimization problem and Felix Haiduk provided information on the uncertainty in wood quality, most of the work in this chapter was accomplished by Jonas Ide and the author.

1.4 Acknowledgement

Before we dive into the world of mathematics, I embrace the opportunity to thank the people who supported me in the course of the preparation and completion of this thesis.

First of all, I am indebted to my supervisor Stephan Westphal for valuable support and boundless optimism that inspired me to never give up. Moreover, I thank Anita Schöbel for being my co-supervisor and always providing treasured advice. Additionally, my thanks go to Sven Krumke for taking the position as second examiner without hesitation.

I thank the German Research Foundation (DFG) and the DFG Research Training Group 1703 *Resource Efficiency in Interorganizational Networks* for the financial support and providing an interdisciplinary context for my work. Furthermore, I give thanks to the project *Optimization and its Applications in Learning and Industry* (OptALI) for providing the possibility of a research stay in New Zealand.

The optimization working group at the Institute for Numerical and Applied Mathematics in Göttingen contributed greatly to the completion of this work. Mathematics is beautiful, thrilling, and often fulfilling, but I enjoyed every day in the office most of all due to my colleagues. My heartfelt thanks go to Sönke Behrends, Marc Goerigk, Jonas Harbering, Ruth Hübner, Jonas Ide, Thorsten Krempasky, Corinna Krüger, Robert

Schieweck, and Marie Schmidt. I have to single out my twin brother and office mate Marco Bender who stayed the course with me and always found the right words to cheer me up. Thank you!

Finally, I thank Janneke for her unconditional support and reminding me every now and then just to settle back, have a coffee, and pause for a moment.

2 Online Knapsack Problems with Dynamic Capacity

2.1 Introduction

In this chapter, the online version of the classic knapsack problem is extended by the introduction of an incremental capacity. In each time period, a set of requests (items), each with a specific weight and value, is revealed and, without knowledge of future requests, it has to be decided which of these requests to accept (i.e., pack into the knapsack). Furthermore, the knapsack capacity changes dynamically over time, i.e., the capacity is not fully available from the start but an additional amount of capacity becomes available in each time period. Hence, if $k \geq 1$ denotes the amount of additional capacity that becomes available in each time period, the available capacity in time period $i \in \{1, \ldots, T\}$ is $k \cdot i$ minus the total weight of all requests that have been accepted in time periods 1 to $i-1$, where T denotes the total number of time periods considered. The goal is to maximize the overall value of the accepted requests while respecting the capacity constraint in each time period.

The idea of incremental capacity within the framework of online knapsack problems can be motivated, for example, by the problem of resource allocation in the context of renewable resources: In each time period, offerings with specific prices for certain amounts of the resource are revealed and, at the same time, additional resources become available. Which offers should be accepted in order to maximize the profit? Such questions occur, for example, at timber producers who frequently receive supply from the forest and have to deal with requests offering specific prices for different amounts of timber in each time period.

Another motivation for studying online knapsack problems with incremental capacity is the generalization of the classic one-way trading problem introduced by El-Yaniv et al. (1992) to dynamically increasing funds. In the classic one-way trading problem, an online player is given an initial amount of dollars that should be converted to yen over a given number of days. Each day, a new exchange rate is announced and the player has to decide how many dollars to convert in order to maximize the total amount of yen obtained after the last day. This can be seen as a special case of the online knapsack problem by viewing the initial amount of dollars as the knapsack capacity and introducing a request for each possible amount that could be traded on each day, where all requests on a specific day have the same value to weight ratio (which corresponds to the exchange rate on this day). Considering the online knapsack problem with incremental capacity then corresponds to receiving a certain amount of dollars for conversion every day instead of having the total amount of dollars available already on the first day.

Both applications above motivate a thorough investigation of the described setting. In the following, we discuss the setting of the online knapsack problem with incremental capacity, present deterministic and randomized online algorithms for the problem, and derive lower bounds on the solution quality achievable by online algorithms.

2.1.1 Previous Work

The offline version of the knapsack problem and a wide range of its variants have been studied for many years and are covered in the literature comprehensively. For a full-scale presentation of methods and techniques available for the solution of the knapsack problem, we refer to the textbooks (Martello and Toth, 1990; Kellerer et al., 2004). The classic 0/1 knapsack problem is $\mathcal{NP}$-hard as proven by Karp (1972), but it admits an FPTAS as first shown by Ibarra and Kim (1975).

The online knapsack problem was introduced by Marchetti-Spaccamela and Vercellis (1995), who also showed that the general online knapsack problem does not admit any competitive online algorithms. Consequently, researchers began to study the online knapsack problem under additional assumptions that allow the design of competitive algorithms or in a stochastic setting using average case analysis.

Marchetti-Spaccamela and Vercellis (1995) studied stochastic online knapsack problems by assuming that the profit and the size coefficients are independent and identically distributed random variables. The results for this setting were subsequently improved by Lueker (1998) who presented an online algorithm whose solution differs from the true optimum by an average of $\Theta(\log n)$, where n denotes the number of items. Further generalizations of the stochastic online knapsack problem were studied, for example, by Papastavrou et al. (1996); Kleywegt and Papastavrou (1998); van Slyke and Young (2000).

Babaioff et al. (2007) considered the online knapsack problem without the assumption of any knowledge regarding the distribution of weights, but made the assumption that items arrive in a random order and presented two e-competitive algorithms for the unweighted case and a $10e$-competitive algorithm for the weighted case as the number of items tends to infinity. Zhou et al. (2008) studied the online knapsack problem with two additional assumptions regarding the weights of the items and were able to deduce best possible algorithms with a competitive ratio of $\log(U/L) + 1$ in this setting, where U and L are the upper and lower bound for the value-to-weight ratio.

Further variants of the online knapsack problem considered in the literature contain settings with removable items, which means that accepted items can be removed to give way for newly arriving items. For the case of unit weights, this variant was considered in (Iwama and Taketomi, 2002) and a $\left((\sqrt{5}+1)/2\right)$-competitive algorithm was presented. For the case of general weights, no competitive algorithms exist. Furthermore, Babaioff et al. (2009) studied the case where a penalty has to be paid for removing an accepted item from the knapsack. Another possibility to circumvent the nonexistence of competitive algorithms for the online knapsack problem is the approach of resource augmentation. In this setting, the online player is allowed to use a knapsack of capacity $R \geq 1$ while the adversary uses a knapsack of capacity one. In (Iwama and Zhang, 2007), this approach was combined with the idea of removable items and, for general weights and $1 < R \leq 2$, a $1/(R-1)$-competitive algorithm was presented. In (Noga and Sarbua, 2005), the approach of resource augmentation was applied to the online partially fractional knapsack problem and, for $1 \leq R \leq 2$, a deterministic $2/R$-competitive algorithm that achieves the best possible competitive ratio was given.

Hajiaghayi et al. (2005) studied online auctions with re-usable goods. This setting can be viewed as an online knapsack problem in which the knapsack is emptied after each time period and the items are available from an arrival to a departure time. The problem analyzed by Hajiaghayi et al. (2005) provided the initial idea for the setting investigated in this chapter.

However, to the best of our knowledge, there is no previous work on the online knapsack problem with incremental capacity as studied in this chapter. The rest of this chapter is structured as follows: In Section 2.2, we formally introduce the online knapsack problem with incremental capacity. In Section 2.3, we derive lower bounds on the competitive ratio achievable by both deterministic and randomized online algorithms for the case of unit weight requests. Deterministic and randomized competitive online algorithms for this case are presented in Section 2.4. In Section 2.5, the cases of general weights as well as limited weights in $\{1, \ldots, k\}$ are considered. In Section 2.6, we discuss different approaches to increase the power of the online player in order to obtain competitive algorithms independent of the number of time periods T. Furthermore, in Section 2.7, we provide an empirical analysis of some of the algorithms presented in the previous sections and, in Section 2.8, we extend the online knapsack problem with incremental capacity to multiple knapsacks. Finally, we consider the setting of independent knapsacks with differing capacities in Section 2.9.

2.2 Problem Definition

In this section, we formally introduce the online knapsack problem with incremental capacity. We consider a time horizon $T \in \mathbb{N}^+$ and N requests $r_i = (d_i, v_i, w_i)$, each consisting of a time period $d_i \in \{1, \ldots, T\}$ in which the request is offered, a value $v_i \in \mathbb{R}^+$, and a weight $w_i \in \mathbb{N}^+$.

For the sake of simplicity, we first consider unit weights, i.e., $w_i \equiv 1$. The case of general weights is considered in Section 2.5. The time horizon T is known to an online algorithm, whereas the number N of requests is not. In each time period $t \in \{1, \ldots, T\}$, the requests r_i with $d_i = t$ are revealed and an online algorithm has to decide which of these requests to accept. The requests with $d_i = t$ that are not accepted in time period t are lost. Note that, even if one would consider requests that remain valid for several time periods, it would not be advantageous for the adversary to reveal requests that remain valid for more than one period. Therefore, this possibility is not considered in our model. The knapsack capacity is increased by a constant amount of $k \in \mathbb{N}^+$ units in each time period, where k is known to an online algorithm. Denoting the available knapsack capacity in time period t by c_t, this means that $c_1 = k$ and $c_t = c_{t-1} + k - |S_{t-1}|$ for $t \geq 2$, where S_{t-1} denotes the set of indices of requests accepted by the online algorithm in time period $t-1$. The objective is to maximize the total value of accepted requests over all time periods $1, \ldots, T$ while not accepting requests of total size larger than c_t in any time period t.

The problem described above is in the following referred to as the *online incremental knapsack problem* (OKIC). Before we proceed by developing lower bounds in Section 2.3, the problem is illustrated in Example 2.2.1.

Example 2.2.1. *Consider a time horizon $T = 3$, an additional capacity of $k = 1$ in each time period, and the following four requests: $r_1 = (1,1,1)$, $r_2 = (2,2,1)$, $r_3 = (3,3,1)$, and $r_4 = (3,3,1)$. We start with a capacity of $c_1 = 1$, and in each time period one additional unit of capacity becomes available. The optimal solution is to reject request r_1 and accept requests r_2, r_3, and r_4, resulting in an objective value of eight (the right part of Figure 2.1 gives an illustration).*

However, an online algorithm has to decide whether to accept or reject request r_1 without any knowledge of future requests. It is only known to the online algorithm that there is a time horizon of $T = 3$ and one additional unit of capacity in each time period. Let us assume that an online algorithm accepts request r_1 in the first time period and request r_2 in the second time period. Due to the available capacity, the online algorithm is then only able to accept either request r_3 or request r_4 in the third time period, which results in an objective value of six (the left part of Figure 2.1 gives an illustration).

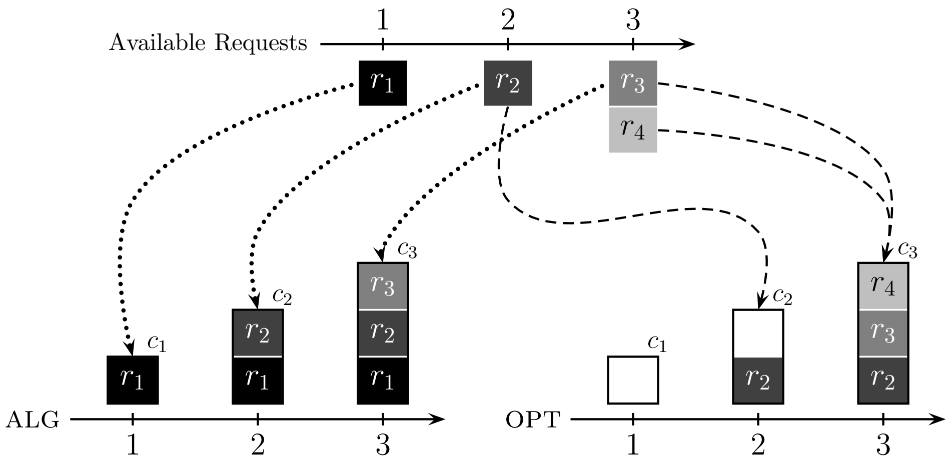


Figure 2.1: Example for OKIC.

The offline problem corresponding to OKIC is given as follows. For $i = 1, \dots, N$, let $x_i \in \{0, 1\}$ be a binary variable with

$$x_i = \begin{cases} 1, & \text{if request } i \text{ is accepted at time } d_i, \\ 0, & \text{otherwise.} \end{cases}$$

By means of these binary variables, we are able to state the following integer programming formulation for OKIC:

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} x_i v_i \\ \text{s.t.} \quad & \sum_{i \in I_t} x_i w_i \le kt \qquad \text{for } t = 1, \dots, T, \\ & \qquad x_i \in \{0, 1\} \quad \text{for } i = 1, \dots, N, \end{aligned}$$

where $I_t = \{i \in \{1, \dots, N\} \mid d_i \le t\}$.

2.3 Lower Bounds

Before we discuss competitive algorithms for OKIC in Section 2.4, lower bounds on the competitive ratio of any deterministic and randomized online algorithm are given in this section.

2.3.1 A Lower Bound for Deterministic Online Algorithms

For a lower bound on deterministic algorithms for OKIC, we first consider the case $k = 1$.

Theorem 2.3.1. *For $k = 1$, no deterministic online algorithm for* OKIC *can achieve a competitive ratio smaller than T.*

Proof. Consider the following sequence of requests: in each time period t, there are t identical requests $r_{t_1}, \dots, r_{t_t}$ with

$$r_{t_i} = (t, v^t, 1), \quad v \ge 1, \quad i = 1, \dots, t.$$

First, we show by induction that, for this sequence of requests, any deterministic online algorithm ALG must accept exactly one request in each time period in order to be competitive against the offline adversary: For $t = 1$, ALG must accept the only available request r_{1_1}, otherwise the adversary accepts r_{1_1} and reveals no further requests. Now, let $t = t'$ and assume that the statement holds for $t < t'$. Note that there are only two options for ALG: by induction hypothesis, ALG accepted exactly one request in each time period $t < t'$ and, since there is only one additional unit of capacity in each time period, the available capacity at time t' equals one. Therefore, ALG can either accept one request

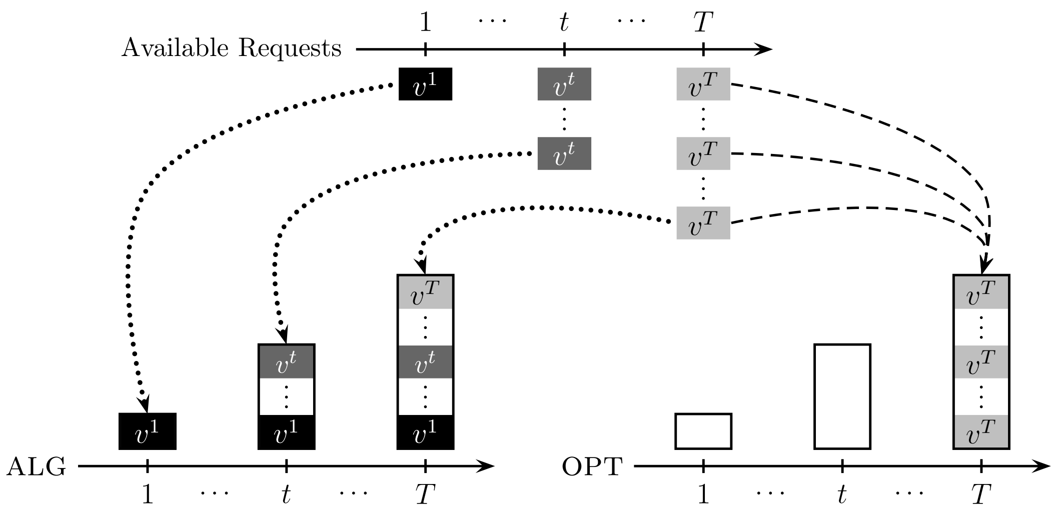


Figure 2.2: A lower bound for OKIC with $k = 1$.

or no request at all. If ALG accepts none of the available requests r_{t_i}, $i = 1, \dots, t$, the adversary accepts all requests r_{t_i}, $i = 1, \dots, t$, in time period t (and no requests in earlier time periods) and no further requests are revealed. For the competitive ratio, we have by induction hypothesis

$$\frac{\text{OPT}}{\text{ALG}} = \frac{tv^t}{\sum\limits_{j=1}^{t-1} v^j} = \frac{tv^t}{\frac{1-v^t}{1-v} - 1} = \frac{tv^t - tv^{t+1}}{v - v^t} = \frac{tv\left(1 - \frac{1}{v}\right)}{1 - v^{1-t}} \to \infty \quad \text{for } v \to \infty.$$

Consequently, any competitive deterministic online algorithm for OKIC accepts exactly one request r_{t_i} in each time period t (the left part of Figure 2.2 gives an illustration). The optimal offline algorithm, on the other hand, accepts no requests until time T and then accepts the requests $r_{T_1}, \dots, r_{T_T}$ (the right part of Figure 2.2 gives an illustration). Analogously to the calculation above, this leads to a competitive ratio of

$$\frac{\text{OPT}}{\text{ALG}} = \frac{Tv^T}{\sum\limits_{j=1}^{T} v^j} = \frac{Tv^T}{\frac{1-v^{T+1}}{1-v} - 1} = \frac{Tv^T - Tv^{T+1}}{v - v^{T+1}} = \frac{T\left(1 - v^{-1}\right)}{1 - v^{-T}} \to T \quad \text{for } v \to \infty.$$

Note that, by a sufficiently large choice of v, any additive constant in the competitive ratio can be eliminated. □

For unit incremental capacity, i.e., $k = 1$, the adversary is able to force the online player to accept one request in each time period, otherwise the online player cannot be competitive. For k-incremental capacity with $k \geq 2$, the online player still has to accept at least one request, but the remaining $k - 1$ additional units in each time period grant a certain amount of freedom to the online player. Therefore, better competitive ratios than T can be obtained for the case $k \geq 2$, see Section 2.4.2.

2.3.2 A Lower Bound for Randomized Online Algorithms

We now present a lower bound on the competitive ratio of any randomized online algorithm for OKIC. In order to prove the lower bound, we make use of Yao's principle (cf. Theorem 1.1.2).

Theorem 2.3.2. *For $T \geq 2$ and $k \in \mathbb{N}^+$, no randomized online algorithm for* OKIC *can achieve a competitive ratio smaller than* $(T+1)/2$.

Proof. For each $i \in \{1, \dots, T\}$, consider the request sequence σ_i consisting of $j \cdot k$ requests with value v^j and deadline j in each time period j, i.e., for each $j \in \{1, \dots, i\}$, we have $j \cdot k$ requests of the form $r_j = (j, v^j, 1)$ with $v > 0$. The optimal solution for sequence σ_i is obviously to save up capacity until time period i and then accept all requests of value v^i, resulting in $\text{OPT}(\sigma_i) = i \cdot k \cdot v^i$.

With respect to the request sequences described above, each deterministic online algorithm ALG is characterized by the number of requests accepted at time j, denoted by α_j. Using this notation, the profit ratio of any deterministic online algorithm ALG and the optimal offline algorithm OPT with respect to request sequence σ_i is given by

$$\frac{\text{ALG}(\sigma_i)}{\text{OPT}(\sigma_i)} = \frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i}.$$

In the following, we derive a probability distribution p over the request sequences σ_i such that

$$\mathbb{E}_p \left[\frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i} \right] \leq \frac{2}{T+1}$$

for each deterministic online algorithm ALG. By Yao's principle for maximization problems (cf. Theorem 1.1.2), more precisely the second term of the maximum in (1.2), the lower bound then follows.

For a given probability distribution p, we have

$$\mathbb{E}_p \left[\frac{\text{ALG}(\sigma_i)}{\text{OPT}(\sigma_i)} \right] = \sum_{i=1}^{T} p_i \frac{\sum_{j=1}^{i} \alpha_j v^j}{ikv^i} = \sum_{i=1}^{T} \sum_{j=1}^{i} \frac{p_i v^{j-i}}{ik} \alpha_j = \sum_{j=1}^{T} \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} \alpha_j,$$

where p_i denotes the probability that request sequence σ_i occurs. The sum of accepted requests up to time period i is at most $i \cdot k$, since there are k additional units of capacity in each time period, i.e., $\sum_{j=1}^{i} \alpha_j \leq ik$ for $i = 1, \dots, T$.

Thus, the maximum profit ratio of any deterministic online algorithm and the optimal offline algorithm is obtained by the following integer program in the variables α_j:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{T}\sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik}\alpha_j && \text{(P)}\\
\text{s.t.} \quad & \sum_{j=1}^{i} \alpha_j \leq ik \quad \text{for } i = 1,\dots,T,\\
& \alpha_j \in \mathbb{N}^+ \quad \text{for } j = 1,\dots,T.
\end{aligned}$$

In order to determine an upper bound on (P) it is sufficient to find a feasible solution to the dual of the linear relaxation of (P), which is given by

$$\begin{aligned}
\min \quad & \sum_{i=1}^{T} ik\Pi_i && \text{(D)}\\
\text{s.t.} \quad & \sum_{i=j}^{T} \Pi_i \geq \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} \quad \text{for } j = 1,\dots,T, && (2.1)\\
& \Pi_i \geq 0 \quad \text{for } i = 1,\dots,T.
\end{aligned}$$

We set

$$p_1 = \frac{2}{T(T+1)} \tag{2.2}$$

and

$$p_i = ip_1 \quad \text{for } i = 2,\dots,T. \tag{2.3}$$

This is at least a feasible choice for the p_i, $i = 1,\dots,T$, since $p_i \geq 0$ and

$$\sum_{i=1}^{T} p_i \overset{(2.3)}{=} \sum_{i=1}^{T} ip_1 = p_1\frac{T(T+1)}{2} \overset{(2.2)}{=} 1.$$

In the following, we will show that by this choice of the variables p_i a feasible solution of (D) with objective value $2/(T+1)$ can be found.

For this purpose, we replace the inequality constraints (2.1) in (D) by equality constraints, which only narrows the set of feasible solutions of (D). For the variables Π_j,

$j = 1, \dots, T$, we then obtain

$$\begin{aligned}
\Pi_j &\overset{(2.1)}{=} \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} - \sum_{i=j+1}^{T} \Pi_i \\
&\overset{(2.1)}{=} \sum_{i=j}^{T} \frac{p_i v^{j-i}}{ik} - \sum_{i=j+1}^{T} \frac{p_i v^{j+1-i}}{ik} \\
&\overset{(2.3)}{=} \sum_{i=j}^{T} \frac{i p_1 v^{j-i}}{ik} - \sum_{i=j+1}^{T} \frac{i p_1 v^{j+1-i}}{ik} \\
&= \frac{p_1}{k} \left(\sum_{i=j}^{T} v^{j-i} - \sum_{i=j+1}^{T} v^{j+1-i} \right) \\
&= \frac{p_1}{k} v^{j-T}. \qquad (2.4)
\end{aligned}$$

By means of the analysis above, the objective function of (D) becomes

$$\begin{aligned}
\sum_{j=1}^{T} jk\Pi_j &\overset{(2.4)}{=} \sum_{j=1}^{T} jk \frac{p_1}{k} v^{j-T} \\
&= \frac{p_1}{v^T} \sum_{j=1}^{T} j v^j \\
&= \frac{p_1}{v^T} \frac{v\left(T v^{T+1} - (T+1)v^T + 1\right)}{(v-1)^2} \\
&= p_1 \frac{v^{T+2}\left(T - (T+1)v^{-1} + v^{-(T+1)}\right)}{v^{T+2}\left(1 - 2v^{-1} + v^{-2}\right)} \\
&= p_1 \frac{T - (T+1)v^{-1} + v^{1-T}}{1 - 2v^{-1} + v^{-2}}.
\end{aligned}$$

For $v \to \infty$, we thus have

$$\sum_{j=1}^{T} jk\Pi_j \to p_1 T \overset{(2.2)}{=} \frac{2}{T+1}.$$

This completes the proof. □

2.4 Competitive Algorithms

In this section, deterministic and randomized online algorithms for the problem OKIC are discussed. The first, obvious choice for an online algorithm for OKIC is a greedy algorithm. In the following subsection, we show that the canonical greedy algorithm is T-competitive and, therefore, best possible for $k = 1$.

2.4.1 A Greedy Algorithm

In each time period, the greedy algorithm for OKIC accepts as many requests as possible in a greedy manner with respect to the value of the requests, see Algorithm 1.

Algorithm 1: Greedy algorithm for OKIC.

```
1 for $t = 1, \dots, T$ do
2     Accept the requests $r_i$ with $d_i = t$ in order of nonincreasing value until either
      no more requests are available or the capacity is fully utilized.
```

Theorem 2.4.1. *Algorithm 1 is T-competitive for* OKIC *with arbitrary $k \in \mathbb{N}^+$.*

Proof. Algorithm 1 always accepts the k requests of highest value denoted by $v_{(1)} \geq \dots \geq v_{(k)}$ among all requests as there are at least k units of capacity available in each time period. OPT can accept no more than $T \cdot k$ requests in total. Since

$$\text{OPT} \leq \sum_{i=1}^{k} v_{(i)} + (Tk - k)v_{(k)} \leq \sum_{i=1}^{k} v_{(i)} + (T-1)\sum_{i=1}^{k} v_{(i)} = T\sum_{i=1}^{k} v_{(i)},$$

and $\text{ALG} \geq \sum_{i=1}^{k} v_{(i)}$, we have

$$\max_{\sigma} \frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)} \leq \frac{T \sum\limits_{i=1}^{k} v_{(i)}}{\sum\limits_{i=1}^{k} v_{(i)}} = T.$$

□

For $k = 1$, the competitive ratio of Algorithm 1 matches the lower bound of T for any deterministic algorithm as given in Theorem 2.3.1. For $k \geq 2$, the online player has more reach of play in order to outsmart the adversary. Thus, we are able to develop a better deterministic online algorithm for the case $k \geq 2$, which is presented in the next subsection.

2.4.2 A Balancing Algorithm

For $k \geq 2$, a deterministic online algorithm for OKIC with competitive ratio smaller than T can be constructed. The idea of the algorithm is as follows: In each time period, we set an upper bound on the number of requests that may be accepted. This upper bound increases over time in order to maximize the competitive ratio. In the first half of the time horizon, less than k requests may be accepted. This way, the online player is able to save up some capacity in order to hedge against the advantage of the offline player, which increases over time. In the second half of the time horizon, the saved up capacity is utilized and more than k requests may be accepted. This policy is formally summarized in Algorithm 2.

Algorithm 2: Balancing algorithm for OKIC.

```
1 for $t = 1, \dots, T$ do
2     Set $R_t = \left\lceil \frac{t(2k-1)}{T+1} \right\rceil$;
3     Accept at most $R_t$ requests $r_i$ with $d_i = t$ in order of nonincreasing value until
      no more requests are available.
```

Theorem 2.4.2. *For $k \geq 2$, Algorithm 2 is $c(k)$-competitive with $c(k) = {}^{(T+1)k}/_{(2k-1)}$.*

Proof. The proof is partitioned in two steps. First of all, we show that Algorithm 2 outputs a feasible solution, i.e., it is feasible to accept up to R_t requests in each time period t. Secondly, we prove the competitive ratio of Algorithm 2.

To begin with, the feasibility of Algorithm 2 is established, i.e., in time period t at least R_t units of capacity are available. For this purpose, each time period $t \leq {}^T/_2$ is paired with the time period $T - t + 1$ and it is shown that in both time periods together not more than $2k$ requests and in time period t not more than k requests are accepted, which proves the feasibility. For $t \leq {}^T/_2$, we have

$$\frac{t(2k-1)}{T+1} + \frac{(T-t+1)(2k-1)}{T+1} = \frac{(T+1)(2k-1)}{T+1} = 2k - 1, \tag{2.5}$$

i.e., the sum of R_t and R_{T-t+1} without ceiling functions is integral. We define

$$x_1 = R_t - \frac{t(2k-1)}{T+1} \quad \text{and} \quad x_2 = R_{T-t+1} - \frac{(T-t+1)(2k-1)}{T+1},$$

and, due to (2.5) and $k \in \mathbb{N}^+$, $x_1 + x_2 = 1$ or $x_1 + x_2 = 0$. Then, R_t and R_{T-t+1} add up to

$$\begin{aligned} R_t + R_{T-t+1} &= \left\lceil \frac{t(2k-1)}{T+1} \right\rceil + \left\lceil \frac{(T-t+1)(2k-1)}{T+1} \right\rceil \\ &= \frac{t(2k-1)}{T+1} + x_1 + \frac{(T-t+1)(2k-1)}{T+1} + x_2. \end{aligned}$$

Thus, we have

$$R_t + R_{T-t+1} = 2k - 1 + x_1 + x_2 \leq 2k.$$

Consequently, when considering the sum of the number of accepted requests in two time periods t and $T - t + 1$ for $t \leq {}^T/_2$, at most $2k$ requests are accepted. Additionally, for $t \leq {}^T/_2$,

$$\left\lceil \frac{t(2k-1)}{T+1} \right\rceil \leq \left\lceil \frac{{}^T/_2(2k-1)}{T+1} \right\rceil = \left\lceil \frac{T\,(k - {}^1/_2)}{T+1} \right\rceil \leq \left\lceil k - \frac{1}{2} \right\rceil = k,$$

i.e., in each time period $t \leq {}^T/_2$, at most k requests are accepted. Since in each time period k additional units of capacity are available, it is feasible to accept at most R_t requests in time period t.

Now, the competitive ratio of Algorithm 2 is analyzed. Denote by $\textsc{opt}_t$ the total value of items accepted by $\textsc{opt}$ in time period t and by $\textsc{alg}_t$ the total value of items accepted by Algorithm 2 in time period t. Since the number of items chosen by $\textsc{opt}$ in time period t is at most $t \cdot k$, Algorithm 2 recovers at least $\textsc{opt}_t \cdot R_t/tk$. Thus, we have

$$\frac{\textsc{opt}_t}{\textsc{alg}_t} \leq \frac{\textsc{opt}_t}{\textsc{opt}_t \cdot R_t/tk} = \frac{tk}{R_t} = \frac{tk}{\left\lceil \frac{t(2k-1)}{T+1} \right\rceil} \leq \frac{(T+1)k}{2k-1}.$$

Since this holds for an arbitrary $t \in \{1, \dots, T\}$, the competitive ratio of Algorithm 2 is given by $(T+1)k/(2k-1)$. □

Note that for $k \to \infty$ the competitive ratio of Algorithm 2 matches the lower bound on the competitive ratio of any randomized (and deterministic) algorithm, i.e.,

$$\lim_{k\to\infty} \frac{(T+1)k}{2k-1} = \frac{T+1}{2}.$$

2.4.3 A Randomized Greedy Algorithm

In this section, a randomized online algorithm for the problem $\textsc{okic}$ is presented. The idea behind Algorithm 3 is to act greedily with a certain probability in each time period. This way, the algorithm is eventually able to save up some capacity and at the same time cannot be leveraged by the adversary.

The probability of being greedy is adjusted in each time period in order to maximize the competitive ratio. In fact, the competitive ratio of Algorithm 3 matches the lower bound for any randomized online algorithm for $\textsc{okic}$ given in Section 2.3.2. Basically, the probability of being greedy in a time period increases over time since we have to hedge against the capacity possibly saved up by the adversary, which also increases over time.

Algorithm 3: Randomized greedy algorithm for $\textsc{okic}$.

```
1 for $t = 1, \dots, T$ do
2     With probability $p_t = 2/(T-t+2)$, accept all requests $r_i$ with $d_i = t$ in order of
      nonincreasing value until either no more requests are available or the capacity
      is fully utilized. With probability $1 - p_t$, accept no requests at all.
```

Theorem 2.4.3. *Algorithm 3 is $(T+1)/2$-competitive for* $\textsc{okic}$ *with $k \in \mathbb{N}^+$.*

Proof. Denote by $\textsc{opt}_t$ the total value of items accepted by $\textsc{opt}$ in time period t and by α_t the number of items accepted by Algorithm 3 in time period t. Since the number of items chosen by $\textsc{opt}$ in time period t is at most $t \cdot k$, Algorithm 3 recovers at least $\textsc{opt}_t \cdot \alpha_t/tk$. Thus, the competitive ratio c is given by

$$c = \max_{\sigma} \frac{\textsc{opt}(\sigma)}{\mathbb{E}\left[\textsc{alg}(\sigma)\right]} \leq \max_{t=1,\dots,T} \frac{\textsc{opt}_t}{\mathbb{E}\left[\textsc{opt}_t \cdot {}^{\alpha_t}/_{tk}\right]} = \max_{t=1,\dots,T} \frac{tk}{\mathbb{E}\left[\alpha_t\right]},$$

where $\mathbb{E}[\alpha_t]$ denotes the expected number of accepted requests by ALG in time period t. We proceed by proving that

$$\mathbb{E}[\alpha_t] = p_t \left(\sum_{i=1}^{t-1} ikp_{t-i} \prod_{j=1}^{i-1} (1 - p_{t-j}) + tk \prod_{j=1}^{t-1} (1 - p_j) \right) \tag{2.6}$$

$$= \frac{2tk}{T+1}. \tag{2.7}$$

Equality (2.6) results from the following observation: in order to accept $i \cdot k$ requests in time period t, first of all ALG has to accept requests in time period t, which happens with probability p_t. Additionally, $i \cdot k$ units of capacity have to be available in time period t. Consequently, ALG has to reject accepting any requests in the previous $i-1$ time periods, which happens with probability $\prod_{j=1}^{i-1}(1-p_{t-j})$, and accept the requests in time period $t-i$, which happens with probability p_{t-i}. Furthermore, in order to accept $t \cdot k$ requests in time period t, ALG has to reject accepting any requests in all previous periods, which happens with probability $\prod_{j=1}^{t-1}(1-p_j)$. Altogether, we end up with (2.6).
As a preliminary result for the proof of (2.7), we show by induction that

$$\prod_{j=1}^{t-1}(1-p_j) = \frac{(T-t+1)(T-t+2)}{T(T+1)}, \quad \text{for } t = 2, \dots, T. \tag{2.8}$$

Base Case: (2.8) holds for $t = 2$:

$$\begin{aligned} \prod_{j=1}^{2-1}(1-p_j) &= 1 - p_1 \\ &= 1 - \frac{2}{T-1+2} \\ &= \frac{T-1}{T+1} \\ &= \frac{(T-t+1)(T-t+2)}{T(T+1)}. \end{aligned}$$

Inductive Step ($\star_1$): Let $t \geq 2$, $t \in \mathbb{N}$, be arbitrary and assume that (2.8) holds for t. Then, (2.8) also holds for $t+1$:

$$\prod_{j=1}^{(t+1)-1}(1-p_j) = \prod_{j=1}^{t}\left(1 - \frac{2}{T-j+2}\right)$$

$$\begin{aligned}
&= \prod_{j=1}^{t} \frac{T-j}{T-j+2} \\
&= \frac{T-t}{T-t+2} \prod_{j=1}^{t-1} \frac{T-j}{T-j+2} \\
&\overset{(\star_1)}{=} \frac{T-t}{T-t+2} \cdot \frac{(T-t+1)(T-t+2)}{T(T+1)} \\
&= \frac{(T-(t+1)+1)\,(T-(t+1)+2)}{T(T+1)}.
\end{aligned}$$

In a similar manner, we show that

$$\prod_{j=1}^{i-1} (1-p_{t-j}) = \frac{(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)}, \quad \text{for } i = 2, \dots, T-1. \tag{2.9}$$

Base Case: (2.9) holds for $i = 2$:

$$\begin{aligned}
\prod_{j=1}^{2-1} (1-p_{t-j}) &= 1 - p_{t-1} \\
&= 1 - \frac{2}{T-(t-1)+2} \\
&= \frac{T-t+1}{T-t+3} \\
&= \frac{(T-t+1)(T-t+2)}{(T-t+i)(T-t+i+1)}.
\end{aligned}$$

Inductive Step ($\star_2$): Let $i \geq 2$, $i \in \mathbb{N}$, be arbitrary and assume that (2.9) holds for i. Then, (2.9) also holds for $i+1$:

$$\begin{aligned}
\prod_{j=1}^{(i+1)-1} (1-p_{t-j}) &= \prod_{j=1}^{i} \left(1 - \frac{2}{T-(t-j)+2}\right) \\
&= \prod_{j=1}^{i} \frac{T-t+j}{T-t+j+2} \\
&= \frac{T-t+i}{T-t+i+2} \prod_{j=1}^{i-1} \frac{T-t+j}{T-t+j+2} \\
&\overset{(\star_2)}{=} \frac{T-t+i}{T-t+i+2} \cdot \frac{(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)} \\
&= \frac{(T-t+2)\,(T-t+1)}{(T-t+(i+1))(T-t+(i+1)+1)}.
\end{aligned}$$

Then, by means of (2.6), (2.8), and (2.9), we have

$$\begin{aligned}\mathbb{E}\left[\alpha_t\right] &= p_t\left(\sum_{i=1}^{t-1} ikp_{t-i}\prod_{j=1}^{i-1}(1-p_{t-j}) + tk\prod_{j=1}^{t-1}(1-p_j)\right)\\ &= kp_t\left(\sum_{i=1}^{t-1}\frac{ip_{t-i}(T-t+2)(T-t+1)}{(T-t+i)(T-t+i+1)} + \frac{t(T-t+1)(T-t+2)}{T(T+1)}\right). \qquad (2.10)\end{aligned}$$

Once again, we use induction to show that

$$\begin{aligned}\sum_{i=1}^{t-1}\frac{ip_{t-i}}{(T-t+i)(T-t+i+1)} &= \sum_{i=1}^{t-1}\frac{2i}{(T-t+i)(T-t+i+1)(T-t+i+2)}\\ &= \frac{t(1-t)}{T(T+1)(t-T-1)}, \quad \text{for } t=2,\dots,T. \qquad (2.11)\end{aligned}$$

Base Case: (2.11) holds for $t=2$:

$$\begin{aligned}\sum_{i=1}^{2-1}\frac{ip_{t-i}}{(T-t+i)(T-t+i+1)} &= \frac{p_1}{(T-1)T}\\ &= \frac{2}{(T-1)T(T+1)}\\ &= \frac{t(1-t)}{T(T+1)(t-T-1)}.\end{aligned}$$

Inductive Step ($\star_3$): Let $t \geq 2$, $t \in \mathbb{N}$, be arbitrary and assume that (2.11) holds for t. Then, (2.11) also holds for $t+1$:

$$\begin{aligned}&\sum_{i=1}^{(t+1)-1}\frac{ip_{(t+1)-i}}{(T-t-1+i)(T-t-1+i+1)}\\ &= \sum_{i=1}^{t}\frac{2i}{(T-t-1+i)(T-t+i)(T-t+1+i)}\\ &= \sum_{i=1}^{t}\frac{2i}{(T'-t+i)(T'-t+1+i)(T'-t+2+i)} \quad \text{with } T' := T-1\\ &= \sum_{i=1}^{t-1}\frac{2i}{(T'-t+i)(T'-t+i+1)(T'-t+i+2)} + \frac{2t}{T'(T'+1)(T'+2)}\\ &\overset{(\star_3)}{=} \frac{t(1-t)}{T'(T'+1)(t-T'-1)} + \frac{2t}{T'(T'+1)(T'+2)}\\ &= \frac{t\left((1-t)(T'+2)+2(t-T'-1)\right)}{T'(T'+1)(T'+2)(t-T'-1)}\end{aligned}$$

$$\begin{aligned}
&= \frac{t\,(T' - tT' - 2T')}{T'(T'+1)(T'+2)(t-T'-1)} \\
&= \frac{t(-1-t)}{(T'+1)(T'+2)(t-T'-1)} \\
&= \frac{t(-1-t)}{T(T+1)(t-T)} = \frac{(t+1)(1-(t+1)}{T(T+1)((t+1)-T-1)}.
\end{aligned}$$

By means of (2.10) and (2.11), the expected number of requests accepted by ALG in time period t with respect to the sequence σ is now given by

$$\begin{aligned}
\mathbb{E}\left[\alpha_t\right] = \; & kp_t \left(\frac{t(1-t)(T-t+2)(T-t+1)}{T(T+1)(t-T-1)} + \frac{t(T-t+1)(T-t+2)}{T(T+1)} \right) \\
= \; & -\frac{2tk(1-t)}{T(T+1)} + \frac{2tk(T-t+1)}{T(T+1)} \\
= \; & \frac{2tk}{T+1}.
\end{aligned}$$

Finally, by means of this result, the competitive ratio c becomes

$$c = \max_{t=1,\ldots,T} \frac{tk}{\mathbb{E}\left[\alpha_t\right]} = \frac{tk}{\frac{2tk}{T+1}} = \frac{T+1}{2},$$

which completes the proof. □

2.5 Limited Weights

In this section, we drop the assumption of *unit* weights and discuss the setting of k-incremental capacity with *limited* weights, i.e., we have weights $w_i \in \{1, \ldots, k\}$. Let S_t denote the set of indices of requests accepted by some algorithm in time period t. Then, the knapsack capacity c_t in time period t is now given as $c_t = c_{t-1} + k - \sum_{i \in S_{t-1}} w_i$ and $c_1 = k$.

It is reasonable to consider *limited* weights $w_i \in \{1, \ldots, k\}$ instead of *unlimited* weights $w_i \in \mathbb{N}^+$ since the setting with unlimited weights does not allow any competitive online algorithm as shown in the following theorem:

Theorem 2.5.1. *There does not exist a competitive algorithm for* OKIC *with unlimited weights $w_i \in \mathbb{N}^+$.*

Proof. For $\epsilon > 0$, consider the request sequence σ with requests $r_1 = (1, \epsilon, k)$ and $r_2 = (2, M, k+1)$ as illustrated in Figure 2.3. In the first time period, request r_1 is offered to any online algorithm and, in order to be competitive, any online algorithm has to accept this request. Otherwise, the adversary offers no further requests and accepts request r_1. In the second time period, request r_2 is offered to any online algorithm. The remaining capacity available for the online player is given by $c_2 = c_1 + k - \sum_{i \in S_1} w_i = k + k - k = k$. Thus, the online player does not have the capacity to accept request r_2. The offline player

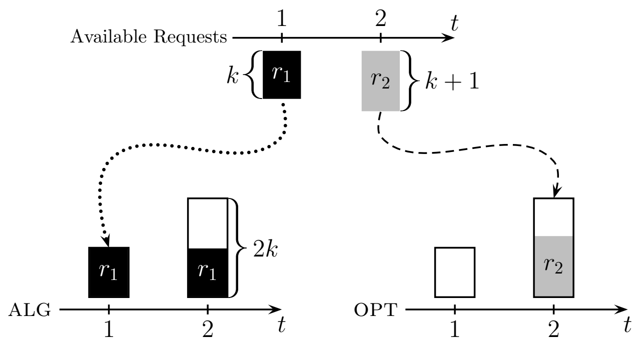


Figure 2.3: Request sequence with $w_2 > k$.

rejects request r_1, saves up the capacity and accepts request r_2. The competitive ratio is then given by

$$\frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)} = \frac{M}{\epsilon} \to \infty \quad \text{for } M \to \infty.$$

Consequently, there exists no competitive algorithm for OKIC with weights $w_i \in \mathbb{N}^+$. $\square$

2.5.1 Deterministic Online Algorithms

Now, we consider a lower bound for any deterministic algorithm for OKIC with limited weights $w_i \in \{1, \dots, k\}$. For $k = 1$, we have unit weights and the results from Sections 2.3 and 2.4 with $k = 1$ apply. For $k \geq 2$, we show the following lower bound on the competitive ratio of deterministic online algorithms:

Theorem 2.5.2. *For $k \geq 2$, no deterministic online algorithm for* OKIC *with limited weights can achieve a competitive ratio smaller than* $\left\lfloor Tk/(\lfloor \frac{k}{2} \rfloor + 1) \right\rfloor$.

Proof. The proof is analogous to the proof of Theorem 2.3.1. For each time period $t = 1, \dots, T-1$, the adversary presents t identical requests $r_{t_1}, \dots, r_{t_t}$ with

$$r_{t_i} = (t, v^t, k), \quad v \geq 1, \quad i = 1, \dots, t,$$

and, by the same argumentation as in the proof of Theorem 2.3.1, forces the online player to accept one request in each time period in order to be competitive.

In time period T, the adversary presents $\left\lfloor Tk/(\lfloor \frac{k}{2} \rfloor + 1) \right\rfloor$ identical requests with value v^T and weight $\lfloor \frac{k}{2} \rfloor + 1$. Since the adversary did not accept any request before, he is able to accept all of these requests, whereas the online player is able to accept exactly one of these requests. This leads to a competitive ratio of

$$\frac{\text{OPT}}{\text{ALG}} = \frac{\left\lfloor \frac{Tk}{\lfloor \frac{k}{2} \rfloor + 1} \right\rfloor v^T}{\sum\limits_{j=1}^{T-1} v^j + v^T} \to \left\lfloor \frac{Tk}{\lfloor \frac{k}{2} \rfloor + 1} \right\rfloor \quad \text{for } v \to \infty.$$

$\square$

Algorithm 4: Greedy algorithm for OKIC with limited weights.

```
1 for $t = 1, \dots, T$ do
2     Solve the knapsack problem with all requests $r_i$ with $d_i = t$ and the available
      capacity and accept the corresponding requests.
```

The lower bound given in Theorem 2.5.2 is matched by the competitive ratio of Algorithm 4 for $k \to \infty$, as shown in the following. Note that the running time of Algorithm 4 is only pseudo-polynomial in the encoding length of the problem input as the algorithm has to solve a knapsack problem exactly in each time period. This can be done, for example, by standard dynamic programming approaches (cf., for example, (Martello and Toth, 1990; Kellerer et al., 2004)).

Theorem 2.5.3. *Algorithm 4 is $2T-1$-competitive for* OKIC *with limited weights.*

Proof. In each time period $t = 1, \dots, T$, OPT has at most $t \cdot k$ units of capacity available. Hence, the requests accepted by OPT in time period t can be fractionally assigned to t knapsacks of size k each, such that at most $t-1$ requests overlap from one knapsack to the next, i.e., at most $t-1$ requests are fractionally assigned (cf. Figure 2.4).

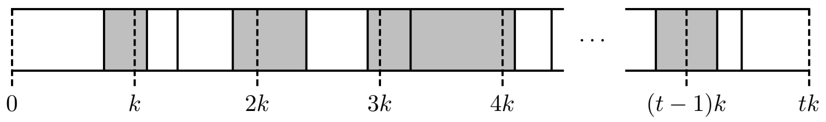


Figure 2.4: Fractional assignment to t knapsacks of size k each. Fractionally assigned requests are shown in grey.

Consequently, removing each of these requests and assigning it to its own additional knapsack yields an integral assignment of the requests accepted by OPT in time period t to at most $2t-1$ knapsacks. Since ALG has at least k units of capacity available in time period t, ALG obtains at least the value of the most valuable of these $2t-1$ knapsacks in period t. Denote by OPT_t the total value of items accepted by OPT in time period t and by ALG_t the total value of items accepted by ALG in time period t. We then have

$$\frac{\text{OPT}}{\text{ALG}} = \frac{\sum_{t=1}^{T} \text{OPT}_t}{\sum_{t=1}^{T} \text{ALG}_t} \leq \frac{\sum_{t=1}^{T} \text{OPT}_t}{\sum_{t=1}^{T} \frac{1}{2t-1} \text{OPT}_t} \leq \frac{\sum_{t=1}^{T} \text{OPT}_t}{\frac{1}{2T-1} \sum_{t=1}^{T} \text{OPT}_t} = 2T - 1.$$

□

For $k \to \infty$, the lower bound converges to the competitive ratio of Algorithm 4:

$$\left\lfloor \frac{Tk}{\left\lfloor \frac{k}{2} \right\rfloor + 1} \right\rfloor = \begin{cases} \left\lfloor \frac{2T}{1+\frac{2}{k}} \right\rfloor \to 2T-1 & \text{for } k \to \infty, \quad k \text{ even}, \\ \left\lfloor \frac{2T}{1+\frac{1}{k}} \right\rfloor \to 2T-1 & \text{for } k \to \infty, \quad k \text{ odd}. \end{cases}$$

2.5.2 Randomized Online Algorithms

Finally, we consider randomized algorithms for OKIC with limited weights. By combining several methods used in the previous sections, a lower bound on the competitive ratio of any randomized online algorithm and a competitive randomized online algorithm can be established.

Theorem 2.5.4. *For $T \geq 2$ and $k \in \mathbb{N}^+$, no randomized algorithm for* OKIC *with limited weights $w_i \in \{1, \dots, k\}$ can achieve a competitive ratio smaller than $\frac{T+1}{2}$.*

Proof. Consider the proof of Theorem 2.3.2. The request sequences σ_i consist of $j \cdot k$ requests of the form $r_j = (j, v^j, 1)$ for each $j \in \{1, \dots, i\}$. We replace these request sequences σ_i by request sequences consisting of j requests of the form $r_j = (j, v^j, k)$, for each $j \in \{1, \dots, i\}$. The remaining proof is then equivalent to the case of $k = 1$ in the proof of Theorem 2.3.2. □

In order to construct a competitive online algorithm for OKIC with limited weights, we combine the methods used in Algorithm 3 and Algorithm 4:

Algorithm 5: Randomized greedy algorithm for OKIC with limited weights.

```
1 for $t = 1, \dots, T$ do
2     With probability $p_t = 2/(T-t+2)$, solve the knapsack problem with all
      requests $r_i$ with $d_i = t$ and the available capacity and accept the
      corresponding requests. With probability $1 - p_t$, accept no requests at all.
```

Theorem 2.5.5. *Algorithm 5 is $3\,(T+1/2)$-competitive for* OKIC *with limited weights.*

Proof. First of all, note that, in each time period, it is advantageous for the adversary to reveal additional invaluable requests with appropriate weights such that the online player uses up all available capacity when solving the knapsack problem. Thus, the expected available capacity $\mathbb{E}_c^t$ in time period t is given by

$$\mathbb{E}_c^t = \sum_{i=1}^{t-1} ikp_{t-i} \prod_{j=1}^{i-1} (1 - p_{t-j}) + tk \prod_{j=1}^{t-1} (1 - p_j)\,.$$

See the proof of Theorem 2.4.3 for a detailed explanation. Due to (2.7), we have

$$\mathbb{E}_c^t = \frac{2tk}{(T+1)p_t} = \frac{tk\,(T-t+2)}{T+1}. \tag{2.12}$$

Now we apply the same line of argument as in the proof of Theorem 2.5.3, but incorporate the expected available capacity. In each time period $t = 1, \dots, T$, OPT has at most $t \cdot k$ units of capacity available. Hence, the requests accepted by OPT in time period t can be fractionally assigned to $\lceil tk/\mathbb{E}_c^t \rceil$ knapsacks of size $\mathbb{E}_c^t \geq k$ each, such that at most $\lceil tk/\mathbb{E}_c^t \rceil - 1$ requests overlap from one knapsack to the next, i.e., at most $\lceil tk/\mathbb{E}_c^t \rceil - 1$

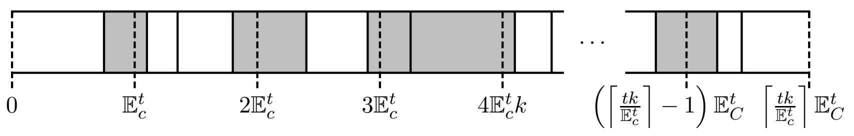

0
$\mathbb{E}_c^t$
$2\mathbb{E}_c^t$
$3\mathbb{E}_c^t$
$4\mathbb{E}_c^t k$
…
$\left(\left\lceil \frac{tk}{\mathbb{E}_c^t} \right\rceil - 1\right)\mathbb{E}_C^t$
$\left\lceil \frac{tk}{\mathbb{E}_c^t} \right\rceil \mathbb{E}_C^t$

Figure 2.5: Fractional assignment to $\lceil tk/\mathbb{E}_c^t \rceil$ knapsacks of size $\mathbb{E}_c^t$ each. Fractionally assigned requests are shown in grey.

requests are fractionally assigned (cf. Figure 2.5). Consequently, removing each of these requests and assigning it to its own additional knapsack yields an integral assignment of the requests accepted by OPT in time period t to at most $2\lceil tk/\mathbb{E}_c^t \rceil - 1$ knapsacks. Since the expected available capacity of ALG in time period t is given by $\lceil tk/\mathbb{E}_c^t \rceil$, ALG obtains at least the value of the most valuable of these $2\lceil tk/\mathbb{E}_c^t \rceil - 1$ knapsacks in period t. Denote by OPT_t the total value of items accepted by OPT in time period t and by ALG_t the total value of items accepted by ALG in time period t. We then have

$$\frac{\text{OPT}}{\mathbb{E}\left[\text{ALG}\right]} = \frac{\sum_{t=1}^{T} \text{OPT}_t}{\sum_{t=1}^{T} \mathbb{E}\left[\text{ALG}_t\right]} \leq \frac{\sum_{t=1}^{T} \text{OPT}_t}{\sum_{t=1}^{T} \frac{p_t}{2\lceil tk/\mathbb{E}_c^t \rceil - 1} \text{OPT}_t} \leq \frac{\sum_{t=1}^{T} \text{OPT}_t}{\frac{2}{3T+3} \sum_{t=1}^{T} \text{OPT}_t} = 3\left(\frac{T+1}{2}\right)$$

Here, the second inequality holds since, for all $1 \leq t \leq T$, we have:

$$\frac{p_t}{2\left\lceil \frac{tk}{\mathbb{E}_c^t} \right\rceil - 1} \overset{(2.12)}{=} \frac{\frac{2}{T-t+2}}{2\left\lceil \frac{T+1}{T-t+2} \right\rceil - 1} \geq \frac{\frac{2}{T-t+2}}{2\left(\frac{T+1}{T-t+2} + 1\right) - 1} = \frac{2}{3T+4-t} \geq \frac{2}{3T+3}.$$

□

2.6 Increasing the Power of the Online Player

The lower bounds presented in Section 2.3 and the algorithms presented in Sections 2.4 and 2.5 are all dependent on the time horizon T. For $T \to \infty$, all those algorithms are not competitive, and, even worse, there are no competitive algorithms at all due to the dependence of the lower bounds on T.

Therefore, we increase the power of the online player by different means in the following sections and investigate the impact on the competitiveness. We apply the concepts of resource augmentation (Section 2.6.1), removable items (Section 2.6.2), and bounded request values (Section 2.6.3). These concepts have also been applied to the classic online knapsack problem:

The concept of removable item gives the online player the ability to remove previously accepted items from the knapsack in order to give way for newly arriving items. Even if items are removable, there exists no competitive algorithm for the online knapsack problem (cf. (Iwama and Zhang, 2003)). The approach of resource augmentation allows the online player to use more resources than the adversary. Iwama and Zhang (2007) apply resource augmentation to the online knapsack problem with removable items and provide competitive algorithms: if the online player uses a knapsack of capacity $R > 1$, while the adversary uses a knapsack of capacity one, an algorithm with competitive ratio $1/(R-1)$ for $1 < R \leq 2$ is obtained. The approach of bounded request values is utilized by Zhou et al. (2008), who studied the online knapsack problem with two additional assumptions, namely a bounded value-to-weight ratio of each item and a small weight of each item with respect to the capacity of the knapsack. In this setting, the authors were able to deduce best possible algorithms with a competitive ratio of $\log(U/L) + 1$, where U and L are the upper and lower bound for the value-to-weight ratio.

In the following, we analyze the impact of these approaches on the online knapsack problem with incremental capacity.

2.6.1 Resource Augmentation

In this section, we consider the approach of resource augmentation for OKIC and analyze its impact on the competitiveness. In the setting of resource augmentation, the online player is allowed to use more resources than the adversary. We consider resource augmentation by a multiplicative factor of $r \in \mathbb{N}_+$, i.e., the offline player starts with capacity k, whereas the online player starts with capacity $r \cdot k$. In each subsequent time period, the capacity is increased by k units for the offline player, whereas the capacity is increased by $r \cdot k$ units for the online player.

Consider a c-competitive algorithm ALG for OKIC. By means of ALG, we are able to construct a $(1 + c/r)$-competitive algorithm for OKIC with resource augmentation by a factor of r. The proof follows a method proposed by Awerbuch et al. (1996, pp. 436-438).

Theorem 2.6.1. *Let* ALG *be a c-competitive algorithm for* OKIC*. Then, for $r \in \mathbb{N}_+$, there exists a $(1 + c/r)$-competitive algorithm for* OKIC *with resource augmentation by a factor of r.*

Proof. We proceed as follows to construct an algorithm ALG$'$ for OKIC with resource augmentation by a factor of r: Consider r copies of ALG, denoted by ALG$_1, \ldots,$ ALG$_r$. In each time period, the set of new requests is first presented to ALG$_1$. The set of requests is then reduced by the requests accepted by ALG$_1$, and passed on to ALG$_2$, and so forth.

Denote by R the set of requests presented to ALG$'$ and by R_i the requests presented to ALG$_i$. Moreover, denote by O the set of requests accepted by OPT and by T_i the set of requests accepted by ALG$_i$. Note that these sets apply to the requests from all time periods. The value of the requests associated with a set S of items is denoted by $v(S)$. By definition of ALG$'$, the set of requests presented to ALG$_i$ is given by

$$R_i = R \setminus \cup_{j<i} T_j.$$

Furthermore, we have

$$O \setminus \cup_{j<i} T_j \subseteq R_i.$$

Thus, the optimal value that can be obtained from requests R_i is at least

$$v\left(O \setminus \cup_{j<i} T_j\right).$$

Applying the fact that ALG$_i$ is c-competitive with respect to the set R_i, we therefore have

$$v\left(T_i\right) \geq \frac{1}{c} v\left(O \setminus \cup_{j<i} T_j\right) = \frac{1}{c} v\left(O\right) - \frac{1}{c} v\left(\cup_{j<i} T_j \cap O\right).$$

Therefore,

$$\begin{aligned}\sum_{i=1}^{r} v\left(T_i\right) \geq\; & \frac{1}{c}\sum_{i=1}^{r} v\left(O\right) - \frac{1}{c}\sum_{i=1}^{r} v\left(\cup_{j<i} T_j \cap O\right)\\ \geq\; & \frac{r}{c} v\left(O\right) - \frac{1}{c}\sum_{i=1}^{r} v\left(\cup_{j<i} T_j\right)\\ \geq\; & \frac{r}{c} v\left(O\right) - \frac{r}{c}\sum_{i=1}^{r} v\left(T_i\right).\end{aligned}$$

We thus have

$$\left(1+\frac{r}{c}\right)\sum_{i=1}^{r} v\left(T_i\right) \;\geq\; \frac{r}{c} v\left(O\right) \quad\Leftrightarrow\quad \left(1+\frac{c}{r}\right)\sum_{i=1}^{r} v\left(T_i\right) \;\geq\; v\left(O\right).$$

Consequently, ALG$'$ is $(1+{}^{c}\!/\!_{r})$-competitive. If ALG is a randomized algorithm, the analysis can be conducted in the same way as shown in (Awerbuch et al., 1996, pp. 437-438). This completes the proof. □

2.6.2 Removable Items

In this section, we increase the power of the online player by allowing the online player to remove previously accepted items from the knapsack in any time period. Once an item is removed from the knapsack it cannot be accepted again. We consider the case of k-incremental capacity and limited weights, i.e., $w_i \in \{1,\ldots,k\}$. The problem is in the following referred to as OKIC *with removable items.*

Consider Algorithm 6 for OKIC with removable items. This algorithm is based on the linear relaxation of the offline version of OKIC with removable items, which is hence an upper bound on the optimal offline solution of OKIC with removable items. For $1 \leq t \leq T$, we define the *incremental fractional knapsack problem with time horizon t*, denoted by IFK^t. For each time period $\tau \in \{1,\ldots,t\}$ and each item $i \in \{1,\ldots,n_\tau\}$, we introduce continuous variables $0 \leq \left(x_\tau^t\right)_i \leq 1$ that take value 1 if the corresponding item is accepted and 0 otherwise. Further, n_τ denotes the number of items in time period τ, $\left(v_\tau\right)_i$ denotes the value of the i-th item given in time period τ, and $\left(w_\tau\right)_i$ the corresponding weight. The problem IFK^t is then given by

$$\begin{aligned}\max\quad & \sum_{\tau=1}^{t}\sum_{i=1}^{n_\tau}\left(x_\tau^t\right)_i \left(v_\tau\right)_i & & (\text{IFK}^t)\\ \text{s.t.}\quad & \sum_{\tau=1}^{\bar{t}}\sum_{i=1}^{n_\tau}\left(x_\tau^t\right)_i \left(w_\tau\right)_i \leq k\cdot\bar{t} & & \text{for } \bar{t}=1,\ldots,t,\\ & 0 \leq \left(x_\tau^t\right)_i \leq 1 & & \text{for } i=1,\ldots,n_\tau,\ \tau=1,\ldots,t.\end{aligned}$$

Algorithm 6: Greedy algorithm for OKIC with removable items.

```
1  for $t = 1, \dots, T$ do
2      Let $\mathcal{N}^t$ be the set of new items (possibly fractionally) accepted by IFK$^t$ in time
       period $t$.
3      if $\alpha_t^t = 1$ then
4          Accept all items in $\mathcal{N}^t$.
5          Remove items accepted in previous time periods in order of nondecreasing
           efficiency such that the capacity constraints are satisfied.
6      else
7          if $\sum_{i=1}^{s_t^t - 1} (v_t)_i \geq (v_t)_{s_t^t}$ then
8              Accept items $i = 1, \dots, s_t^t - 1$.
9              Remove items accepted in previous time periods in order of
               nondecreasing efficiency such that the capacity constraints are satisfied.
10         else
11             Accept the split item $s_t^t$.
```

Note that IFK^t denotes the incremental fractional knapsack problem with t time periods, i.e., $\tau = 1, \dots, t$.

The optimal solution of IFK^t is given by accepting the most efficient items while respecting the capacity constraint of each time period. In the following, assume that, in each time period τ, the new items are sorted by nonincreasing efficiency, i.e., $(v_\tau)_1/(w_\tau)_1 \geq (v_\tau)_2/(w_\tau)_2 \geq \dots \geq (v_\tau)_{n_\tau}/(w_\tau)_{n_\tau}$. Then, the optimal solution vector x^t of IFK^t is given by $x^t = \left(x_1^t, \dots, x_t^t\right)$, where each $x_\tau^t \in \mathbb{R}_+^{n_\tau}$ for $1 \leq \tau \leq t$ is given by

$$x_\tau^t = \big(\,\underbrace{1, \dots, 1}_{s_\tau^t - 1}, \alpha_\tau^t, 0, \dots, 0\,\big), \text{ with } 0 < \alpha_\tau^t \leq 1, \tag{2.13}$$

for some $1 \leq s_\tau^t \leq n_\tau$ (note that $s_t^t = |\mathcal{N}^t|$, see Algorithm 6). Keep in mind that it is possible that x_τ^t does not use the full $k \cdot \tau$ units of capacity available in time period τ since it may be beneficial to save some capacity for items of higher efficiency arriving in later periods. The item s_τ^t (possibly fractionally) accepted to an amount α_τ^t in time period τ will be referred to as the *split item* of time period τ. Observe that $s_\tau^t \leq s_\tau^{t'}$ for all $t \geq t'$, since, for $t > t'$, IFK^t will never accept any item of time period τ that was not accepted by IFK$^{t'}$.

We now analyze the competitiveness of Algorithm 6, also referred to as ALG in the proof of the following theorem.

Theorem 2.6.2. *For $k \geq 2$, Algorithm 6 is 3-competitive for* OKIC *with removable items. For $k = 1$, Algorithm 6 finds the optimal offline solution.*

Proof. First of all, we consider the case $k = 1$. In this case, IFK^T never accepts any item fractionally, i.e., $\alpha_\tau^T = 1$ for all $\tau = 1, \dots, T$. It is easy to see that the solution produced by ALG is identical to the solution of IFK^T. Since $\text{IFK}^T \geq \text{OPT}$, ALG obtains an optimal solution for $k = 1$.

For $k \geq 2$, consider now the incremental fractional knapsack problem with time horizon T, i.e., IFK^T: since $\text{IFK}^T \geq \text{OPT}$, it suffices to prove $3 \cdot \text{ALG} \geq \text{IFK}^T$ for $k \geq 2$.

For $1 \leq t \leq T$, denote by ALG^t the online algorithm after time period t and by $y^t = \left(y_1^t, \ldots, y_t^t\right)$ the solution vector produced by ALG^t, where $y_\tau^t \in \mathbb{R}_+^{n_\tau}$ for $1 \leq \tau \leq t$. In the following, we prove by induction on t that, for each time period t and the corresponding program IFK^t, we have $3 \cdot \text{ALG}^t \geq \text{IFK}^t$. For $t = T$, we then have $3 \cdot \text{ALG} \geq \text{IFK}^T \geq \text{OPT}$. First of all, we show that this holds for $t = 1$ and $t = 2$. The step from an arbitrary $t - 1$ to t then works analogously.

Without loss of generality, assume that the weight of the new items in each time period is larger than k: If the weight of new items in time period t is at most k, $\mathcal{N}^t$ contains all new items of time period t and $\alpha_t^t = 1$. Hence, by Step 4 of Algorithm 6, ALG accepts all new items in addition to the items in the knapsack after the previous time period and the adversary is not able to gain a competitive edge over the online player (note that ALG does not remove any items in Step 5).

Consider $t = 1$ and the optimal solution vector $x^1 = \left(x_1^1\right)$ for IFK^1, where x_1^1 is given by

$$x_1^1 = (\, \underbrace{1, \ldots, 1}_{s_1^1 - 1}, \alpha_1^1, 0, \ldots, 0 \,)\,, \tag{2.14}$$

whereas the solution vector y^1 of ALG_1 is given by $y^1 = \left(y_1^1\right)$, where

$$y_1^1 = (\, \underbrace{1, \ldots, 1}_{s_1^1 - 1}, 0, 0, \ldots, 0 \,) \text{ if } \sum_{i=1}^{s_1^1 - 1} (v_1)_i \geq (v_1)_{s_1^1}\,, \tag{2.15}$$

$$y_1^1 = (\, \underbrace{0, \ldots, 0}_{s_1^1 - 1}, 1, 0, \ldots, 0 \,) \text{ if } \sum_{i=1}^{s_1^1 - 1} (v_1)_i < (v_1)_{s_1^1}\,. \tag{2.16}$$

Since ALG^1 accepts the more valuable solution, we have

$$2 \cdot \text{ALG}^1 \geq \text{IFK}^1. \tag{2.17}$$

Figures 2.6 and 2.7 depict the possible situations after the first time period. Note that we can neglect items that are rejected by both IFK^1 and ALG^1 since the online algorithm cannot use them in later time periods and $s_1^t \leq s_1^1$ for $t \geq 1$.

Now, consider $t = 2$. We distinguish two cases with respect to α_2^2: either the split item of the second time period is completely accepted, i.e., $\alpha_2^2 = 1$, or the split item of the second time period is fractionally accepted, i.e., $\alpha_2^2 < 1$.

Case 1: $\alpha_2^2 = 1$

In this case, the split item of the second time period is completely accepted by IFK^2.

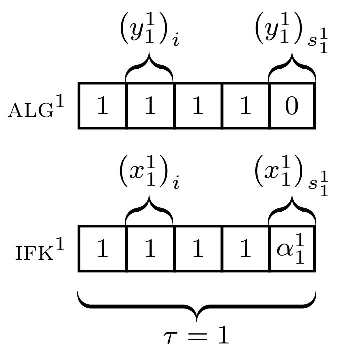


Figure 2.6: ALG^1 according to (2.15).

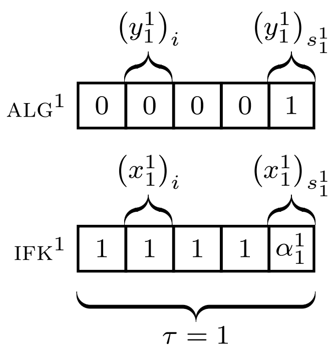


Figure 2.7: ALG^1 according to (2.16).

Consequently, ALG^2 accepts all items in $\mathcal{N}_2$, according to Step 4 of Algorithm 6, and we have

$$\underbrace{\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by ALG}^2}} = \underbrace{\sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items}\\ \text{accepted by IFK}^2}} . \tag{2.18}$$

Now, consider the items from the first time period. ALG^2 possibly has to remove some of the items accepted in the first time period in order to give way for the accepted items from the second time period (see Step 5 of Algorithm 6). We distinguish two cases with respect to the behavior of ALG^1:

Case 1.1: ALG^1 acted according to (2.15).

In this case, ALG^1 accepted all items that are accepted by IFK^1 in the first time period except for the split item s_1^1. If ALG^2 does not have to remove any item from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (2.17) and (2.18).

Therefore, assume that ALG^2 has to remove some items from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. The remaining value for ALG^2 of items from the first time period is then given by

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^1-1} (v_1)_i - \sum_{i=d}^{s_1^1-1} (v_1)_i \,,$$

for some $1 \leq d \leq s_1^1 - 1$ (note that the items in each time period are sorted by nonincreasing efficiency). However, since ALG^2 has to remove items $d, \dots, s_1^1 - 1$ from the first time period and both ALG^2 and IFK^2 accept the same items from the second time period, IFK^2 cannot accept more than items $1, \dots, d$ from the first time period due to capacity constraints, i.e., $s_1^2 = d$ (see Figure 2.8). Therefore, we have

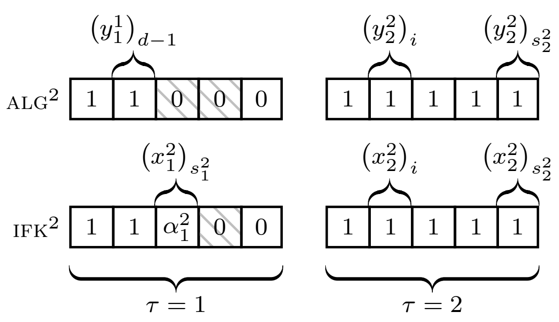


Figure 2.8: Solution vectors of ALG^2 and IFK^2 in Case 1.1. Hatched fields denote removed items.

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{d-1} (v_1)_i = \sum_{i=1}^{s_1^2-1} \left(x_1^2\right)_i (v_1)_i \,. \tag{2.19}$$

The weight of the items from the second time period accepted by ALG^2 is larger than k and, additionally, the efficiency of each item from the second time period accepted by ALG^2 is at least as large as the efficiency of item s_1^2 (otherwise, IFK^2 would not have accepted all new items completely). Since $s_1^2 \leq s_1^1$, this holds also for s_1^1. Thus, we have

$$\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i \geq (v_1)_{s_1^1} \quad \text{and} \quad \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i \geq (v_1)_{s_1^2} \,. \tag{2.20}$$

For the total value of ALG^2, we then have

$$\begin{aligned}
2 \cdot \text{ALG}^2 &= 2\left(\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i\right) \\
&\overset{(2.18)}{=} 2\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(2.19)}{=} 2\sum_{i=1}^{s_1^2-1} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(2.20)}{\geq} 2\sum_{i=1}^{s_1^2-1} \left(x_1^2\right)_i (v_1)_i + (v_1)_{s_1^2} + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\geq \sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&= \text{IFK}^2.
\end{aligned}$$

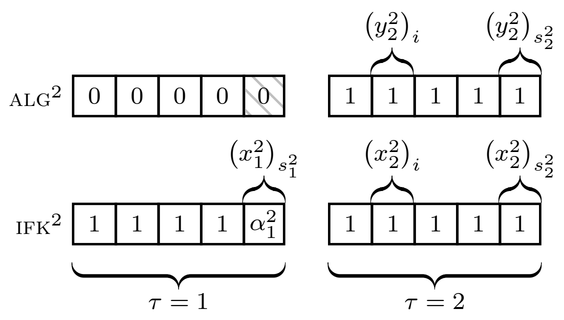


Figure 2.9: Solution vectors of ALG^2 and IFK^2 in Case 1.2. Hatched fields denote removed items.

Case 1.2: ALG^1 acted according to (2.16)**.**

In this case, ALG^1 accepted only the split item s_1^1 in the first time period. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (2.17) and (2.18).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. We then have

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = 0, \tag{2.21}$$

see also Figure 2.9. Furthermore, (2.20) holds by the same arguments as in Case 1.1. In this case, the total value of ALG^2 satisfies

$$\begin{aligned}
3 \cdot \text{ALG}^2 &= 3\left(\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i\right) \\
&\overset{(2.21)}{=} 3\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i \\
&\overset{(2.18)}{=} 2\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(2.20)}{\geq} (v_1)_{s_1^1} + (v_1)_{s_1^2} + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\overset{(2.16)}{\geq} \sum_{i=1}^{s_1^1-1} \left(x_1^1\right)_i (v_1)_i + (v_1)_{s_1^2} + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\
&\geq \sum_{i=1}^{s_1^2} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \quad (\text{since } s_1^2 \leq s_1^1) \\
&= \text{IFK}^2.
\end{aligned}$$

Case 2: $\alpha_2^2 < 1$

In this case, the split item of the second time period is only fractionally accepted by IFK^2. Therefore, ALG^2 has to decide whether to take the split item s_2^2, or all new items accepted by IFK^2 except for the split item. Since ALG^2 accepts the more valuable solution, we have

$$2 \underbrace{\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items} \\ \text{accepted by ALG}^2}} \geq \underbrace{\sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i}_{\substack{\text{value of new items} \\ \text{accepted by IFK}^2}} . \tag{2.22}$$

If ALG^2 accepts only the split item s_2^2, there is no need for ALG^2 to remove any of the items accepted in the first time period since the weight of each item is at most k and there are k units of additional capacity available. Thus, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (2.17) and (2.22).

Therefore, assume that ALG^2 accepts all new items of the second time period accepted by IFK^2 except for the split item s_2^2. Before we proceed, we make the following observation:

Observation 1. *Since* IFK^2 *accepted the split item* s_2^2 *fractionally, the efficiency of the split item is the lowest among all items accepted by* IFK^2*. In particular,* IFK^2 *accepts each item from the first time period accepted by* IFK^1 *to the same fraction (if its efficiency is higher than the efficiency of* s_2^2*), or* IFK^2 *does not accept the item at all (if its efficiency is lower than the efficiency of* s_2^2*). For items with equal efficiency, we can assume without loss of generality that* IFK^2 *is the offline solution that assigns the fractionality only to* s_2^2 *since this does not change the value of the solution of* IFK^2.

As in Case 1, we now distinguish two cases with respect to the behavior of ALG^1:

Case 2.1: ALG^1 acted according to (2.15).

In this case, ALG^1 accepted all items that are accepted by IFK^1 in the first time period except for the split item s_1^1. If ALG^2 does not have to remove any items from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (2.17) and (2.22).

Therefore, assume that ALG^2 has to remove some items from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. As in Case 1.1, the remaining value for ALG^2 of items from the first time period is then given by

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^1-1} (v_1)_i - \sum_{i=d}^{s_1^1-1} (v_1)_i ,$$

for some $1 \leq d \leq s_1^1-1$. Due to Observation 1, IFK^2 accepts at most items $1, \ldots, d-1$ in this case, i.e., $s_1^2 \leq d-1$, since IFK^2 uses at least as much capacity as ALG^2 for

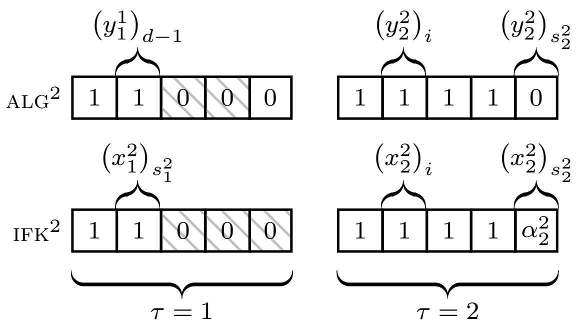


Figure 2.10: Solution vectors of ALG^2 and IFK^2 in Case 2.1. Hatched fields denote removed items.

items from the second time period (see Figure 2.10). Thus, we have

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = \sum_{i=1}^{s_1^2} \left(x_1^2\right)_i (v_1)_i \,, \tag{2.23}$$

and obtain $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ by means of (2.22) and (2.23).

Case 2.2: ALG^1 acted according to (2.16)**.**
In this case, ALG^1 accepted only the split item s_1^1 in the first time period. If ALG^2 does not have to remove item s_1^1 from the first time period, we have $2 \cdot \text{ALG}^2 \geq \text{IFK}^2$ due to (2.17) and (2.22).

Therefore, assume that ALG^2 has to remove item s_1^1 from the first time period and, hence, the weight of items from the second time period accepted by ALG^2 is larger than k. As in Case 1.2, we have

$$\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i = 0. \tag{2.24}$$

The weight of new items accepted by IFK^2 must also be larger than k and, due to Observation 1, IFK^2 does not accept s_1^1. Therefore, we have

$$\sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i \leq \sum_{i=1}^{s_1^1-1} (v_1)_i \,, \tag{2.25}$$

see also Figure 2.11. Additionally, the efficiency of each item from the second time period accepted by ALG^2 is at least as large as the efficiency of item s_1^1 (otherwise,

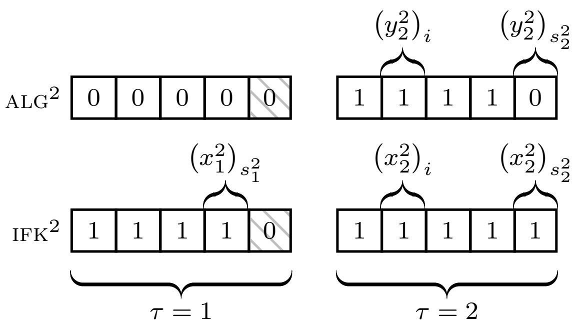


Figure 2.11: Solution vectors of ALG^2 and IFK^2 in Case 2.2. Hatched fields denote removed items.

IFK^2 would not have accepted new items with weight larger than k). Thus, we have

$$\begin{aligned}\sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_1)_i &\geq (v_1)_{s_1^1} \\ &\overset{(2.16)}{>} \sum_{i=1}^{s_1^1-1} (v_1)_i \\ &\overset{(2.25)}{\geq} \sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i . \end{aligned} \tag{2.26}$$

For the total value of ALG^2, we then have

$$\begin{aligned} 3 \cdot \text{ALG}^2 &= 3 \left(\sum_{i=1}^{n_1} \left(y_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i \right) \\ &\overset{(2.24)}{=} 3 \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i \\ &\overset{(2.22)}{\geq} \sum_{i=1}^{n_2} \left(y_2^2\right)_i (v_2)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\ &\overset{(2.26)}{\geq} \sum_{i=1}^{n_1} \left(x_1^2\right)_i (v_1)_i + \sum_{i=1}^{n_2} \left(x_2^2\right)_i (v_2)_i \\ &= \text{IFK}^2. \end{aligned}$$

Consider now the step from an arbitrary $t-1$ to t for $t > 2$. The basic analysis works analogously, but we have to make some comments concerning the details.

The analysis is identical with respect to the new items accepted in time period t by ALG^t and IFK^t. If no items accepted by ALG^{t-1} are removed by ALG^t, we are done. Thus, assume that ALG^t has to remove items accepted by ALG^{t-1}. In contrast to the step

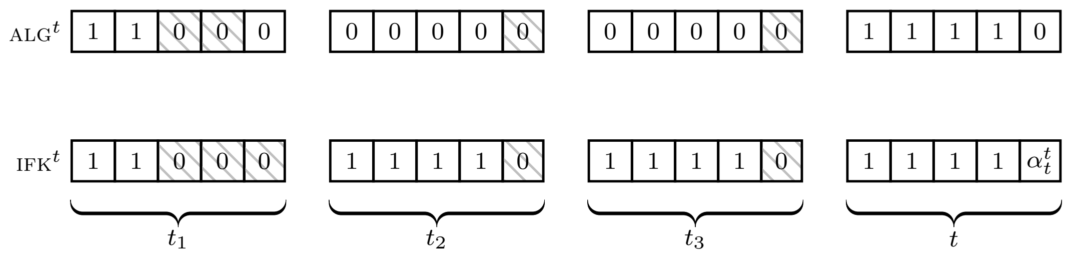


Figure 2.12: Solution vectors of ALG^t and IFK^t. Hatched fields denote removed items.

from the first to the second time period, ALG^t possibly removes items from several time periods. A generalization of Observation 1 helps us to analyze this situation:

Observation 2. *If* IFK^t *accepts the split item of time period t fractionally, each item accepted by* IFK^{t-1} *is either accepted by* IFK^t *to the same fraction, or not accepted at all by* IFK^t*.*

Observation 3. *If* IFK^t *accepts the split item of time period t completely, there is at most one item accepted by* IFK^{t-1} *that is accepted by* IFK^t *to a smaller but positive fraction compared to* IFK^{t-1}*.*

Both observations hold due to the same argumentation as for Observation 1. Now, consider the case that IFK^t accepts the split item of time period t fractionally. Due to Observation 2, each time period $t' < t$ for which ALG$^{t'}$ acted according to (2.15) can be analyzed as in Case 2.1 (see (2.23) and t_1 in Figure 2.12). Each time period for which ALG$^{t'}$ acted according to (2.16) is analyzed as in Case 2.2 (see t_2 and t_3 in Figure 2.12), with the following additional argument: Before ALG^t removes the split item from time period t', at least the k additional units of capacity from time period t are used up by new items with efficiency at least as high as the split item's efficiency. After ALG^t removed the split item, the corresponding block of k units of capacity is available. Note that this must be at least k units of capacity, since $\alpha_{t'}^{t'} < 1$ in time period t'. Thus, before the next split item is removed by ALG^t, again k units of capacity are used by new items with efficiency at least as high as the previous items, and so forth. Therefore, the total value of all split items removed by ALG^t is not larger than the total value of all new items accepted by ALG^t.

Finally, consider the case that IFK^t accepts the split item of time period t completely. Due to Observation 3, there is at most one item accepted by IFK^{t-1} that is accepted by IFK^t to a smaller but positive fraction compared to IFK^{t-1}. If we neglect this item, the analysis works as in the case above. Since ALG^t accepts all new items that are accepted by IFK^t, ALG^t obtains the same value from new items as IFK^t, instead of only half of the value as in the case above. Since the neglected item has value not larger than the total value of the new items of IFK^t, the value of the new items accepted by ALG^t is at least half of the value of both the new items accepted by IFK^t and the value of the neglected item. The value of the items from previous time periods except for the neglected item can be bounded as before. This concludes the proof. □

In the following, we prove that the analysis of Algorithm 6 is tight for $k \to \infty$.

Theorem 2.6.3. *For $k \to \infty$, the competitive ratio of Algorithm 6 is 3.*

Proof. Consider the requests $r_1 = (1, k, 1)$, $r_2 = (1, k + \epsilon, k)$, $\epsilon > 0$, and $r_3 = (1, k - 2, k - 2)$, that are presented to ALG in the first time period. Note that $k \geq 3$. We have an ordering with respect to efficiency of

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{(k+\epsilon)}{k} = \frac{v_2}{w_2} > 1 = \frac{v_3}{w_3},$$

for ϵ sufficiently small. Thus, IFK^1 accepts r_1 and a fraction of ${}^{k-1}/_{k}$ of r_2. Since $w_1 + w_2 = 1 + k > k$ and $v_2 = k + \epsilon > k = v_1$, ALG accepts r_2.
Assume that k is odd. Then, in the second time period, two identical requests

$$r_4 = r_5 = (2, \frac{(k+1)(k+2\epsilon)}{2k}, \frac{k+1}{2})$$

are revealed (see also Figure 2.13). The efficiency order is given by

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{v_4}{w_4} = \frac{v_5}{w_5} = \frac{\frac{(k+1)(k+2\epsilon)}{2k}}{\frac{k+1}{2}} = \frac{k+2\epsilon}{k} > \frac{k+\epsilon}{k} = \frac{v_2}{w_2}.$$

IFK^2 accepts r_1, r_4, r_5, and a fraction of ${}^{k-2}/_{k}$ of r_2. Thus, ALG accepts r_4 and r_5. Since $w_4 + w_5 + w_2 = 2k + 1 > 2k$, ALG has to remove r_2. The value of ALG is then given by

$$\text{ALG} = v_4 + v_5 = \frac{(k+1)(k+2\epsilon)}{k} = k + 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

The optimal offline solution is given by accepting r_1,r_3,r_4, and r_5. This solution is feasible since $w_1 + w_3 = k - 1$ and $w_1 + w_3 + w_4 + w_5 = 2k$, and the total value is given by

$$\text{OPT} = v_1 + v_3 + v_4 + v_5 = k + k - 2 + \frac{(k+1)(k+2\epsilon)}{k} = 3k - 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

Since $\epsilon > 0$ can be chosen small, the competitive ratio is in this case given by ${}^{\text{OPT}}/_{\text{ALG}} = {}^{(3k-1)}/_{(k+1)}$, and for $k \to \infty$, we have ${}^{\text{OPT}}/_{\text{ALG}} \to 3$.

It remains to prove the theorem for even k. For this case, we slightly change the weight and value of items r_4 and r_5, i.e., we set

$$r_4 = \left(2, \frac{k+2\epsilon}{2}, \frac{k}{2}\right) \quad \text{and} \quad r_5 = \left(2, \frac{k+2\epsilon}{2}\left(1 + \frac{2}{k}\right), \frac{k}{2} + 1\right).$$

Since

$$\frac{v_1}{w_1} = \frac{k}{1} > \frac{v_4}{w_4} = \frac{v_5}{w_5} = \frac{k+2\epsilon}{k} > \frac{k+\epsilon}{k} = \frac{v_2}{w_2},$$

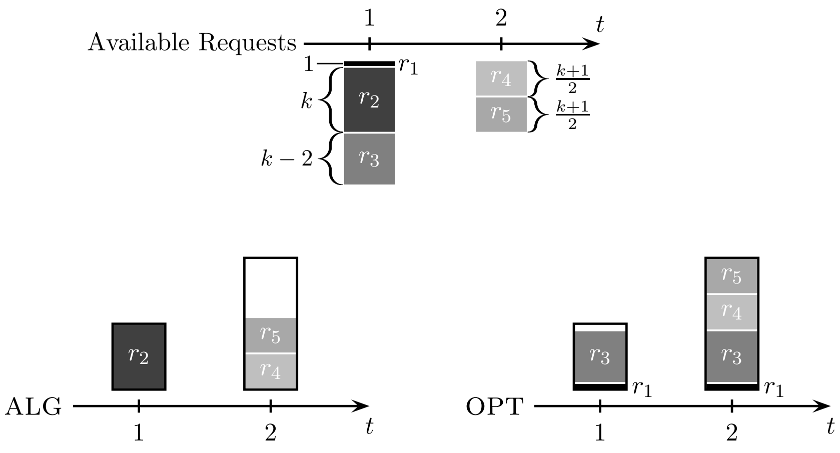


Figure 2.13: Lower bound for OKIC with removable items for $k \geq 2$ odd.

ALG again accepts r_4 and r_5 in the second time period and, due to $w_4 + w_5 + w_2 = 2k + 1 > 2k$, ALG has to remove r_2. The value of ALG is then given by

$$v_4 + v_5 = \frac{k + 2\epsilon}{2}\left(2 + \frac{2}{k}\right) = k + 1 + 2\epsilon + \frac{2\epsilon}{k}.$$

The optimal offline solution is given by accepting r_1,r_3,r_4, and r_5. This solution is still feasible and features the same total value as in the case of odd k. Therefore, the competitive ratio of Algorithm 6 is three. □

In the following, lower bounds on the competitive ratio of any deterministic online algorithm for OKIC with removable items are provided.

Theorem 2.6.4. *For $k \geq 4$, no deterministic online algorithm for OKIC with removable items can achieve a competitive ratio smaller than $\sqrt{2}$. For $k = 3$ and $k = 2$, lower bounds are given by $(1+\sqrt{10})/3 \approx 1.387$ and $(1+\sqrt{17})/4 \approx 1.281$, respectively.*

Proof. Consider the request sequence $\sigma = (a_1, b_1, a_2, b_2, \ldots, a_k, b_k)$. In the first time period $t = 1$, the adversary reveals $a_1 = (1, 1, 1)$ and $b_1 = (1, x, k)$ with $x > 1$. Obviously, the online player must accept either a_1 or b_1 in order to be competitive. If the online player accepts a_1, all further requests a_t and b_t, for $t = 2, \ldots, k$, will be worthless and the competitive ratio is given by x since the adversary chooses b_1.

Otherwise, the online player accepts b_1 and the adversary reveals $a_2 = (1, 1, 1)$ and $b_2 = (1, x, k)$. Again, if the online player accepts a_2, all further requests a_t and b_t, for $t = 3, \ldots, k$, will be worthless and the competitive ratio is given by $(x+2)/(x+1)$ since the adversary accepts a_1 in the first period and a_2 and b_2 in the second period. Otherwise, if the online player accepts b_2, the adversary reveals $a_3 = (1, 1, 1)$ and $b_3 = (1, x, k)$.

This procedure is repeated until the online player either accepts a_t for some $t < k$ or $t = k$. If the online player accepts a_t for some $2 \leq t < k$, the competitive ratio is given by

$$\frac{(t-1)x+t}{(t-1)x+1}. \tag{2.27}$$

For $2 \leq t < k$, (2.27) is increasing in t. Thus, the online player accepts a_1 in the first time period and ends up with a competitive ratio of x, or accepts a_2 and ends up with the abovementioned competitive ratio of $(x+2)/(x+1)$, or accepts b_t for $t = 1, \dots, k-1$. In the latter case, the online player obviously accepts b_k in the last time period since $x > 1$, and the competitive ratio is given by

$$\frac{(k-1)x+k}{kx}. \tag{2.28}$$

Set $x := \sqrt{2}$. Then, for $k \geq 4$, (2.28) is larger than $\sqrt{2}$, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{k-1}{k} + \frac{1}{\sqrt{2}} \geq \frac{3}{4} + \frac{1}{\sqrt{2}} \approx 1.457 > \sqrt{2},$$

and $(x+2)/(x+1) = (\sqrt{2}+2)/(\sqrt{2}+1) = \sqrt{2}$. Consequently, for $k \geq 4$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than $\sqrt{2}$.
For $k = 3$, set $x := (1+\sqrt{10})/3 \approx 1.387$. Then, (2.28) equals x, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{2}{3} + \frac{3}{1+\sqrt{10}} = \frac{1+\sqrt{10}}{3},$$

and $(x+2)/(x+1) \approx 1.418$. Thus, for $k = 3$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than $(1+\sqrt{10})/3$.
Finally, for $k = 2$, set $x := (1+\sqrt{17})/4 \approx 1.281$. Then, (2.28) equals x, i.e.,

$$\frac{(k-1)x+k}{kx} = \frac{1}{2} + \frac{4}{1+\sqrt{17}} = \frac{1+\sqrt{17}}{4}.$$

Therefore, for $k = 2$, no deterministic algorithm for OKIC with removable items can achieve a competitive ratio smaller than $(1+\sqrt{17})/4$. □

Instead of being greedy in each time period t, we could solve the knapsack problem with all available items and capacity $k \cdot t$ in each time period, see Algorithm 7. Theorem 2.6.5 states that this strategy cannot lead to a better competitive ratio than two. However, it remains subject to further research whether Algorithm 7 obtains a constant competitive ratio or even a competitive ratio smaller than three.

Theorem 2.6.5. *For $k \to \infty$, the competitive ratio of Algorithm 7 is at least two.*

Proof. Consider the following sequence of requests: in each time period t, there are two requests r_{t_1} and r_{t_2} with

$$r_{t_1} = (t, 1, k) \quad \text{and} \quad r_{t_2} = (t, 1-\epsilon, 1),$$

Algorithm 7: Knapsack algorithm for OKIC with removable items.

```
1 for $t = 1, \dots, T$ do
2     Solve the knapsack problem with capacity $t \cdot k$ and all available items, i.e.,
      both items $r_i$ with $d_i = t$ and items which have been accepted in previous
      periods. Accept the corresponding requests and remove requests which have
      been accepted in previous periods but are not part of the knapsack solution.
```

where $\epsilon > 0$. In the first time period, ALG accepts r_{1_1} since $v_{1_1} > v_{1_2}$ and the sum of the weights of r_{1_1} and r_{1_2} exceeds the available capacity of k.

In the second time period, ALG solves the knapsack problem with capacity $2k$ and the available requests r_{1_1} (which ALG accepted in the first time period), r_{2_1}, and r_{2_2}. Thus, ALG keeps r_{1_1} and accepts r_{2_1}.

This situation reoccurs in every following time period. Consequently, ALG accepts r_{t_1} in every time period and achieves a total value of T.

On the contrary, OPT accepts in the first time period r_{1_2} and in the following $k-1$ time periods both r_{t_1} and r_{t_2}. In the $(k+1)$-st time period, OPT again accepts only r_{t_2} and in the following $k-1$ time periods both r_{t_1} and r_{t_2}, and so forth. Consequently, OPT achieves a total value of

$$T(1-\epsilon) + \left(T - \left\lceil \frac{T}{k} \right\rceil\right) = 2T - T\epsilon - \left\lceil \frac{T}{k} \right\rceil .$$

Therefore, the competitive ratio is given by

$$\frac{\text{OPT}}{\text{ALG}} = \frac{2T - T\epsilon - \lceil \frac{T}{k} \rceil}{T} = 2 - \epsilon - \frac{\lceil \frac{T}{k} \rceil}{T} \geq 2 - \epsilon - \frac{1}{k}. \tag{2.29}$$

For $k \to \infty$, (2.29) converges to $2 - \epsilon$, and since ϵ is arbitrary the competitive ratio of Algorithm 6 is not smaller than two. □

2.6.3 Bounded Values

In this section, the request values v_i are bounded in order to give the online player a competitive edge over the adversary. Thus, consider bounded values $v_i \in [m, M]$ with $0 < m < M$. We confine ourselves to k-incremental capacity and unit weights. The problem is in the following referred to as OKIC *with bounded values.* Note that for arbitrary or limited weights the analysis of competitive algorithms is subject to further research.

In the previous sections, we observed that greedy algorithms are not the worst choice. Thus, consider Algorithm 8 that solves OKIC with bounded values in a greedy manner. However, Algorithm 8 achieves only the trivial competitive ratio M/m.

Theorem 2.6.6. *For $T \to \infty$, the competitive ratio of Algorithm 8 for* OKIC *with bounded values is given by M/m.*

Algorithm 8: Greedy algorithm for OKIC with bounded values.

```
1 for $t = 1, \dots, T$ do
2     Accept the requests $r_i$ with $d_i = t$ in order of nonincreasing value until either
      no more requests are available or the capacity is fully utilized.
```

Proof. Algorithm 8 always accepts the k requests of highest value among all requests as there are at least k units of capacity available in each time period. OPT can accept no more than $T \cdot k$ requests in total. Thus, $\text{OPT} \le T \cdot k \cdot M$ and $\text{ALG} \ge k \cdot M + (T-1)k \cdot m$, and we have

$$\max_{\sigma} \frac{\text{OPT}(\sigma)}{\text{ALG}(\sigma)} \le \frac{T \cdot k \cdot M}{k \cdot M + (T-1)k \cdot m} = \frac{M}{M/T + (1 - 1/T)m} \to \frac{M}{m} \quad \text{for } T \to \infty.$$

□

In order to find an algorithm for OKIC with bounded values with a better competitive ratio, we apply a threshold value policy (as proposed for the time series search problem by El-Yaniv et al. (2001)): Algorithm 9 accepts only items with value at least matching a certain threshold.

Algorithm 9: Threshold value policy for OKIC with bounded values.

```
1 for $t = 1, \dots, T$ do
2     Accept the requests $r_i$ with $d_i = t$ if $v_i \ge v^\star$ in order of nonincreasing value
      until either no more requests are available or the capacity is fully utilized,
      where
          $$v^\star = \frac{\sqrt{M^2 + 4T(T-1)Mm} - M}{2(T-1)}.$$
```

Theorem 2.6.7. *For $T \to \infty$, the competitive ratio of Algorithm 9 for* OKIC *with bounded values is given by $\sqrt{M/m}$.*

Proof. The value of each request is either smaller than the threshold or it matches at least the threshold. In the first case, the request is rejected by ALG, and, in the latter case, the request is accepted by ALG (if enough capacity is available). In order to maximize the competitive ratio, the adversary therefore reveals only requests with value $v^\star - \epsilon$, for some $\epsilon > 0$, or $v^\star$.

Depending on the threshold, the adversary now chooses between two strategies in order to outsmart the online player: Either the adversary first presents requests with a value matching the threshold such that the online player uses up all available capacity and then presents requests with maximal value M. Or the adversary presents only requests which are just below the threshold such that the online player has to settle for the lower bound m in the end. Note that the best result for the adversary (i.e., the worst case for the online player) is achieved by sticking to one of these two strategies for the whole game: otherwise, the online player is either able to accept additional requests with maximal value instead of requests with value just below the threshold, or the online player is able to accept additional request with value matching the threshold instead of requests with minimal value.

If the adversary chooses the first strategy, k requests with value $v^\star$ are revealed in time period $t = 1, \dots, T-1$, which are accepted by the online player. In the last time period, $k \cdot T$ requests with value M are revealed, from which the online player is able to accept k requests and the adversary accepts all of them. This leads to a competitive ratio of

$$\frac{k \cdot T \cdot M}{k \cdot M + k(T-1)v^\star} = \frac{T \cdot M}{M + (T-1)v^\star}. \tag{2.30}$$

If the adversary chooses the second strategy, k requests with value $v^\star - \epsilon$, for some $\epsilon > 0$, are revealed in each time period. Since the value of these requests is below the threshold $v^\star$, the online player rejects all of them and ends up with the lower bound m. Thus, ignoring the ϵ, the competitive ratio is given by

$$\frac{k \cdot T \cdot v^\star}{k \cdot T \cdot m} = \frac{v^\star}{m}. \tag{2.31}$$

Due to the choice of $v^\star$, we have (2.30) = (2.31), i.e.,

$$\frac{v^\star}{m} = \frac{T \cdot M}{M + (T-1)v^\star}.$$

Thus, the competitive ratio is given by

$$\begin{aligned} \frac{v^\star}{m} &= \frac{\sqrt{M^2 + 4T(T-1)Mm} - M}{2(T-1)m} \\ &= \frac{\sqrt{\frac{M^2}{T^2} + 4Mm - \frac{4Mm}{T}} - \frac{M}{T}}{2m - \frac{2m}{T}} \to \sqrt{\frac{M}{m}} \quad \text{for } T \to \infty. \end{aligned}$$

□

2.7 Empirical Analysis

In the previous sections, we have analyzed various settings of OKIC and presented competitive algorithms as well as lower bounds on the competitive ratio. Since we took the approach of competitive analysis, we have looked for the weak spot of the presented algorithms and analyzed their worst-case performance. In order to get a feeling for the performance of these algorithms on randomly generated instances, we implemented them and evaluated their average-case competitive ratio for various instances with increments $k \in \{2, 5, 10\}$, numbers of time periods $T \in \{5, 10, 20, 40\}$, and numbers of items $N = l \cdot k \cdot T$ where $l \in \{1, 2, 4\}$. The number of items is linked to the number of time periods and the increment, since otherwise there is a high probability that all new items fit in the additional k units of capacity in each time period and the problem is of no interest. The time period in which an item is revealed to the online algorithm is sampled uniformly from $\{1, \ldots, T\}$ and the value of each item is sampled uniformly from $[1, 100]$. Note that the impact of a change of the maximal possible value for each item was neglectable in all experiments that we carried out. Thus, we decided to fix the maximal value. For the limited weight case, the weight of each item is sampled uniformly from $\{1, \ldots, k\}$.

In Table 2.1, we present the simulation results for Algorithm 1 (greedy algorithm), Algorithm 2 (balancing algorithm), Algorithm 3 (randomized greedy algorithm), and Algorithm 9 (threshold value policy). Note that we deal with unit weights. Each row represents 100 randomly generated instances and the left part of Table 2.1 states the parameter settings, whereas the right part of Table 2.1 states the average competitivities along with the standard deviations. The empirical competitive ratio of the randomized algorithm is given by the mean value of 100 runs on the same instance.

Regardless of the different parameter settings, the greedy algorithm and the threshold value policy perform better than the balancing algorithm and the randomized greedy algorithm. The greedy algorithm and the threshold value policy achieve on average a competitivity of 1.031 and 1.029, respectively, while the balancing algorithm and the randomized greedy algorithm obtain on average a competitivity of 1.502 and 2.679, respectively. In comparison to the worst-case behaviour of these algorithms analyzed in the course of this chapter, the average-case performance of the greedy algorithm and the threshold value policy is quite good.

In Table 2.2, we present the simulation results for Algorithm 4 (greedy algorithm), Algorithm 5 (randomized greedy algorithm), Algorithm 6 (greedy algorithm with removal), and Algorithm 7 (knapsack algorithm with removal). Note that we deal with limited weights and Algorithm 6 and Algorithm 7 are allowed to remove previously accepted items. Again, each row represents 100 randomly generated instances and the left part of Table 2.2 states the parameter settings, whereas the right part of Table 2.2 states the average-case competitivities. The empirical competitive ratio of the randomized algorithm is given by the mean value of 100 runs on the same instance.

For the limited weights setting, the algorithms that are allowed to remove previously accepted items (Algorithm 6 and Algorithm 7) achieve on average a competitivity of 1.018 and 1.007, respectively. Due to their ability to remove previously accepted items, these algorithms outperform the greedy algorithm and the randomized greedy algorithm, regardless of the instance. The greedy algorithm achieves an average competitivity of 1.076 and is clearly better than the randomized greedy algorithm with an average competitivity of 2.167.

Finally, we consider the dependence of the competitive ratio of the greedy algorithm on the number of time periods as indicated by the theoretical analysis. Note that all other algorithms for OKIC without additional assumptions also feature a competitive ratio dependent on the number of time periods. However, we picked the greedy algorithm

k	T	N	Algorithm 1 greedy	Algorithm 2 balance	Algorithm 3 rand. greedy	Algorithm 9 threshold
2	5	10	1.016 ± 0.045	1.718 ± 0.350	1.738 ± 0.206	1.012 ± 0.030
2	5	20	1.063 ± 0.059	1.725 ± 0.154	1.487 ± 0.143	1.056 ± 0.055
2	5	40	1.037 ± 0.037	1.697 ± 0.073	1.263 ± 0.069	1.037 ± 0.037
2	10	20	1.016 ± 0.028	2.079 ± 0.351	2.500 ± 0.361	1.015 ± 0.022
2	10	40	1.086 ± 0.045	2.107 ± 0.191	1.938 ± 0.186	1.078 ± 0.042
2	10	80	1.053 ± 0.033	2.052 ± 0.080	1.492 ± 0.082	1.052 ± 0.032
2	20	40	1.019 ± 0.026	1.965 ± 0.205	3.765 ± 0.579	1.012 ± 0.013
2	20	80	1.108 ± 0.040	2.048 ± 0.133	2.858 ± 0.333	1.093 ± 0.038
2	20	160	1.065 ± 0.025	1.990 ± 0.067	1.879 ± 0.124	1.064 ± 0.024
2	40	80	1.017 ± 0.015	2.112 ± 0.177	6.079 ± 0.839	1.013 ± 0.010
2	40	160	1.115 ± 0.029	2.183 ± 0.112	4.606 ± 0.465	1.101 ± 0.026
2	40	320	1.069 ± 0.017	2.083 ± 0.053	2.682 ± 0.221	1.069 ± 0.016
5	5	25	1.005 ± 0.016	1.265 ± 0.118	1.703 ± 0.187	1.005 ± 0.013
5	5	50	1.030 ± 0.024	1.272 ± 0.076	1.472 ± 0.095	1.029 ± 0.022
5	5	100	1.016 ± 0.017	1.218 ± 0.029	1.244 ± 0.041	1.016 ± 0.017
5	10	50	1.006 ± 0.009	1.363 ± 0.108	2.491 ± 0.249	1.006 ± 0.006
5	10	100	1.042 ± 0.021	1.354 ± 0.060	1.922 ± 0.125	1.040 ± 0.020
5	10	200	1.020 ± 0.013	1.296 ± 0.023	1.480 ± 0.054	1.020 ± 0.012
5	20	100	1.008 ± 0.010	1.371 ± 0.064	3.730 ± 0.367	1.007 ± 0.005
5	20	200	1.052 ± 0.016	1.361 ± 0.043	2.875 ± 0.213	1.049 ± 0.015
5	20	400	1.024 ± 0.008	1.287 ± 0.018	1.866 ± 0.091	1.024 ± 0.008
5	40	200	1.009 ± 0.008	1.414 ± 0.060	6.161 ± 0.600	1.008 ± 0.004
5	40	400	1.056 ± 0.014	1.382 ± 0.033	4.523 ± 0.346	1.052 ± 0.012
5	40	800	1.026 ± 0.006	1.306 ± 0.014	2.663 ± 0.150	1.026 ± 0.006
10	5	50	1.002 ± 0.006	1.180 ± 0.063	1.720 ± 0.124	1.003 ± 0.004
10	5	100	1.020 ± 0.016	1.164 ± 0.039	1.442 ± 0.072	1.020 ± 0.016
10	5	200	1.007 ± 0.006	1.135 ± 0.016	1.232 ± 0.033	1.007 ± 0.006
10	10	100	1.003 ± 0.005	1.237 ± 0.058	2.453 ± 0.190	1.004 ± 0.003
10	10	200	1.026 ± 0.012	1.211 ± 0.036	1.903 ± 0.101	1.025 ± 0.012
10	10	400	1.010 ± 0.006	1.154 ± 0.017	1.464 ± 0.043	1.010 ± 0.006
10	20	200	1.004 ± 0.006	1.276 ± 0.041	3.782 ± 0.247	1.006 ± 0.004
10	20	400	1.027 ± 0.010	1.232 ± 0.030	2.866 ± 0.180	1.026 ± 0.009
10	20	800	1.012 ± 0.004	1.160 ± 0.011	1.855 ± 0.070	1.012 ± 0.004
10	40	400	1.005 ± 0.004	1.282 ± 0.036	6.084 ± 0.440	1.007 ± 0.002
10	40	800	1.029 ± 0.006	1.242 ± 0.021	4.578 ± 0.212	1.029 ± 0.006
10	40	1600	1.013 ± 0.003	1.162 ± 0.009	2.673 ± 0.130	1.013 ± 0.003

Table 2.1: Average competitivities for unit weights.

k	T	N	Algorithm 4 greedy	Algorithm 5 rand. greedy	Algorithm 6 greedy remove	Algorithm 7 knapsack remove
2	5	10	1.066 ± 0.088	1.623 ± 0.226	1.018 ± 0.036	1.004 ± 0.016
2	5	20	1.095 ± 0.078	1.413 ± 0.129	1.043 ± 0.055	1.016 ± 0.028
2	5	40	1.065 ± 0.051	1.309 ± 0.067	1.041 ± 0.048	1.008 ± 0.015
2	10	20	1.093 ± 0.066	2.231 ± 0.318	1.014 ± 0.022	1.006 ± 0.014
2	10	40	1.109 ± 0.054	1.762 ± 0.181	1.025 ± 0.024	1.013 ± 0.017
2	10	80	1.079 ± 0.036	1.547 ± 0.088	1.025 ± 0.025	1.005 ± 0.007
2	20	40	1.106 ± 0.055	3.401 ± 0.542	1.007 ± 0.009	1.005 ± 0.007
2	20	80	1.126 ± 0.041	2.397 ± 0.267	1.012 ± 0.010	1.009 ± 0.009
2	20	160	1.097 ± 0.031	1.904 ± 0.101	1.016 ± 0.013	1.004 ± 0.005
2	40	80	1.118 ± 0.038	5.476 ± 0.777	1.003 ± 0.003	1.004 ± 0.004
2	40	160	1.131 ± 0.029	3.711 ± 0.407	1.007 ± 0.006	1.007 ± 0.006
2	40	320	1.098 ± 0.023	2.474 ± 0.160	1.011 ± 0.008	1.004 ± 0.003
5	5	25	1.093 ± 0.063	1.461 ± 0.116	1.057 ± 0.053	1.019 ± 0.026
5	5	50	1.054 ± 0.041	1.364 ± 0.075	1.038 ± 0.040	1.010 ± 0.012
5	5	100	1.040 ± 0.024	1.316 ± 0.057	1.025 ± 0.028	1.003 ± 0.006
5	10	50	1.091 ± 0.044	1.845 ± 0.186	1.026 ± 0.024	1.015 ± 0.014
5	10	100	1.077 ± 0.026	1.672 ± 0.099	1.025 ± 0.021	1.011 ± 0.009
5	10	200	1.053 ± 0.018	1.584 ± 0.062	1.010 ± 0.010	1.004 ± 0.004
5	20	100	1.111 ± 0.033	2.493 ± 0.257	1.017 ± 0.012	1.012 ± 0.008
5	20	200	1.083 ± 0.022	2.095 ± 0.130	1.012 ± 0.010	1.008 ± 0.005
5	20	400	1.055 ± 0.015	1.977 ± 0.080	1.007 ± 0.006	1.003 ± 0.003
5	40	200	1.112 ± 0.021	3.863 ± 0.416	1.009 ± 0.006	1.012 ± 0.005
5	40	400	1.087 ± 0.013	2.814 ± 0.173	1.008 ± 0.004	1.007 ± 0.003
5	40	800	1.061 ± 0.010	2.524 ± 0.086	1.004 ± 0.003	1.002 ± 0.001
10	5	50	1.067 ± 0.039	1.396 ± 0.074	1.050 ± 0.049	1.013 ± 0.014
10	5	100	1.043 ± 0.023	1.362 ± 0.057	1.032 ± 0.028	1.007 ± 0.007
10	5	200	1.027 ± 0.016	1.322 ± 0.050	1.021 ± 0.019	1.003 ± 0.004
10	10	100	1.069 ± 0.024	1.742 ± 0.100	1.024 ± 0.022	1.013 ± 0.009
10	10	200	1.050 ± 0.020	1.657 ± 0.075	1.015 ± 0.011	1.007 ± 0.004
10	10	400	1.033 ± 0.013	1.607 ± 0.062	1.011 ± 0.009	1.003 ± 0.002
10	20	200	1.077 ± 0.019	2.239 ± 0.129	1.012 ± 0.008	1.011 ± 0.005
10	20	400	1.057 ± 0.014	2.114 ± 0.092	1.007 ± 0.006	1.005 ± 0.003
10	20	800	1.038 ± 0.008	2.014 ± 0.071	1.005 ± 0.004	1.002 ± 0.001
10	40	400	1.084 ± 0.014	2.994 ± 0.197	1.007 ± 0.005	1.009 ± 0.004
10	40	800	1.059 ± 0.010	2.711 ± 0.095	1.003 ± 0.002	1.004 ± 0.002
10	40	1600	1.040 ± 0.006	2.607 ± 0.080	1.002 ± 0.002	1.002 ± 0.001

Table 2.2: Average competitivities for limited weights.

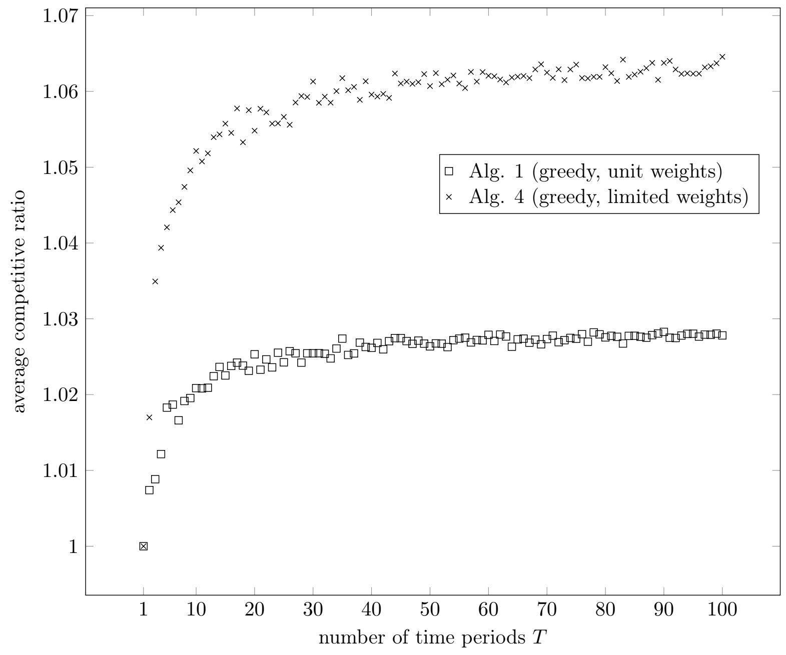


Figure 2.14: Average competitivity of the greedy algorithm for $T = 1, \dots, 100$.

since it performed best among these algorithms. In Figure 2.14, we depicted the average competitivity of the greedy algorithm for unit and limited weights on instances with $k = 5$, $N = 4 \cdot k \cdot T$, and $T = 1, \dots, 100$. Each dot represents the average competitivity of the greedy algorithm on 100 randomly generated instances (as described above).

For an increasing number of time periods and constant k as well as a constant number of items per unit of available capacity, the average competitivity of the greedy algorithm for both unit and limited weights increases, but the increase flattens with an increasing number of time periods.

2.8 Extension to Multiple Knapsacks

In this section, we consider the extension of the online knapsack problem with incremental capacity to *multiple knapsacks*. Instead of a single knapsack with increasing capacity, we now have multiple knapsacks, each with increasing capacity: at the beginning, there are m knapsacks and the capacity of each knapsack increases by k units in each time period, as in the case of a single knapsack.

By means of a result by Awerbuch et al. (1996), the algorithms presented in this work can be used to obtain competitive algorithms for the problem with multiple knapsacks: We run m copies of the single knapsack algorithm $\textsc{alg}$, one for each knapsack, denoted by $\textsc{alg}_1, \dots, \textsc{alg}_m$. In each time period, the set of new requests is first presented to $\textsc{alg}_1$. The set of requests is then reduced by the requests accepted by $\textsc{alg}_1$, and passed on to $\textsc{alg}_2$, and so forth.

If ALG is c-competitive, we obtain a $(c+1)$-competitive algorithm for multiple knapsacks:

Theorem 2.8.1 (Awerbuch et al. (1996)). *If there exists a c-competitive algorithm* ALG *for* OKIC *with a single knapsack, then there exists a $(c+1)$-competitive algorithm for* OKIC *with m knapsacks.*

The theorem in (Awerbuch et al., 1996) is given for a broad class of packing problems, but can be translated directly to the online knapsack problem with incremental capacity. In our setting, in each time period several requests are presented, whereas in the setting in (Awerbuch et al., 1996) one request is presented in each time period. However, the proof is based on the following argument: Denote the set of requests accepted by OPT and assigned to knapsack i by O_i, the set of requests accepted by ALG_i by T_i, and the set of requests presented to ALG_i by R_i. Then,

$$O_i \setminus \bigcup_{j<i} T_j \subseteq R_i.$$

This argument also holds if not only one but several requests are presented in each time period. The remainder of the proof is then executed in exactly the same way.

2.9 Independent Knapsacks

Up to now, we have always considered the online knapsack problem with *incremental* capacity. In this section, we modify this setting and consider independent knapsacks with differing capacities, i.e., in each time period a certain capacity is available independent of the available capacity of other time periods and the accepted requests in other time periods. Consequently, it is not possible to save up capacity for subsequent time periods. In this setting, the capacity of the knapsack in any time period can, for example, be regarded as a perishable resource. Note that, in this variant, it can be advantageous for the adversary to present requests that remain valid for more than one time period. Thus, the knapsacks of each time period are independent of each other, but the time periods are linked by the requests.

This setting is on the one hand closely related to the multiple knapsack problem, which is a generalization of the standard knapsack problem considering several knapsacks with possibly different capacities (cf. (Kellerer et al., 2004, Chapter 10)). However, in an online version of the multiple knapsack problem all knapsacks are available in each time period and the capacity of each knapsack is known in advance. Hence, this setting differs from the setting proposed in this section.

On the other hand, the setting of independent knapsacks is related to online auctions with re-usable goods as presented by Hajiaghayi et al. (2005). This problem can be viewed as an online knapsack problem whereby the knapsack is emptied after each time period and the items are available from an arrival to a departure time. However, the available capacity remains the same in each time period and is known in advance. Hence, this setting also differs from the setting of independent knapsacks discussed in this section.

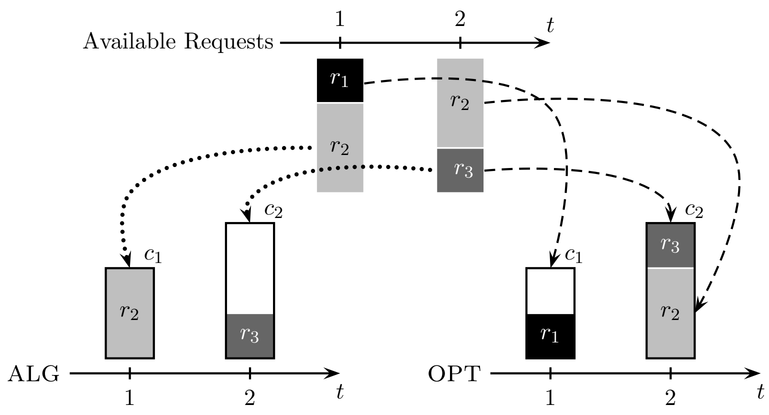


Figure 2.15: Example for OIK.

First, the model is introduced formally. Consider a time horizon $T \in \mathbb{N}^+$ and N requests $r_i = (a_i, d_i, v_i, w_i)$, each consisting of an arrival time $a_i \in \{1, \dots, T\}$, a deadline $d_i \in \{1, \dots, T\}$ with $d_i \geq a_i$, a value $v_i \in \mathbb{R}^+$, and a weight $w_i \in \mathbb{N}^+$.

The time horizon T is known to an online algorithm, whereas the number of requests N is not. In each time period $t \in \{1, \dots, T\}$, the available capacity c_t as well as the requests r_i with $a_i = t$ are revealed. In the setting considered in this section, the available capacity in each time period is independent from the available capacity in previous time periods, i.e., in each time period t there is a capacity of c_t, but if capacity remains unused in any time period, it is not available in the next time period.

The problem described above is in the following referred to as the *online independent knapsack problem* (OIK). The problem is illustrated in Example 2.9.1:

Example 2.9.1. *Consider a time horizon $T = 2$ and the following three requests:*

$$r_1 = (1, 1, 1, 1), \quad r_2 = (1, 2, 3, 2), \quad r_3 = (2, 2, 2, 1).$$

In Figure 2.15, this example is depicted graphically. In the first time period, requests r_1 and r_2 are revealed along with capacity $c_1 = 2$. An online algorithm can choose to accept either r_1, r_2, or neither of them. It is not feasible to accept both requests due to the capacity constraint. Assume that an online algorithm accepts request r_2. Then, request r_3 is revealed in the second time period along with capacity $c_2 = 3$. The online algorithm is only able to accept request r_3 and achieves an objective value of 5.

However, the optimal solution is to accept request r_1 in the first time period and requests r_2 and r_3 in the second time period leading to an objective value of 7. Note that request r_2 is still available in the second time period.

The offline problem corresponding to OIK can be formulated as an integer program. Let $\mathcal{R} = \{r_1, \dots, r_N\}$ be the set of requests and $x_{i,t} \in \{0, 1\}$ be a binary variable with

$$x_{i,t} = \begin{cases} 1, & \text{if request } i \text{ is accepted at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

where $a_i \leq t \leq d_i$ and $i = 1, \dots, N$. By means of these binary variables, we are able to state the following integer programming formulation:

$$\begin{aligned}
\max \quad & \sum_{t=1}^{T} \sum_{i=1}^{N} x_{i,t} v_i \\
\text{s.t.} \quad & \sum_{i \in I_t} x_{i,t} w_i \leq c_t && \text{for } t = 1, \dots, T, \\
& \sum_{t=a_i}^{d_i} x_{i,t} \leq 1 && \text{for } i = 1, \dots, N, \\
& x_{i,t} \in \{0,1\} && \text{for } a_i \leq t \leq d_i,\ i = 1, \dots, N,
\end{aligned}$$

where $I_t = \{i \in \{1, \dots, N\} \mid a_i \leq t \leq d_i\}$. Thus, we have T knapsacks with capacity c_t, $t \in T$, and each request r_i corresponds to an item with weight w_i and value v_i. Additionally, we are only able to pack item r_i in knapsacks t' with $a_i \leq t' \leq d_i$. So, we have T knapsack problems linked with each other.

2.9.1 A Lower Bound for Deterministic Online Algorithms

Before we discuss a competitive algorithm for OIK in the next section, a lower bound on the competitive ratio of any deterministic online algorithm is given in this section.

Theorem 2.9.1. *No deterministic online algorithm for* OIK *can achieve a competitive ratio smaller than the golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$ (≈ 1.61803).*

Proof. Consider the following four requests:

$$\begin{aligned}
r_1 &= (1, 1, v_1, 2), \\
r_2 &= (1, 3, v_2, 1), \\
r_3 &= (2, 2, v_3, 2), \\
r_4 &= (2, 3, v_4, 1),
\end{aligned}$$

where $v_1, \dots, v_4 \geq 0$. In the first time period, requests r_1 and r_2 are revealed along with capacity $c_1 = 2$ and an online algorithm can choose to accept either r_1, r_2, or nothing at all. If no request is accepted, the online algorithm is not competitive, since the adversary reveals no further requests and is able to accept either r_1 or r_2. If the online algorithm accepts request r_1, no further requests are revealed. The offline algorithm accepts r_2 and the competitive ratio is v_2/v_1.

Otherwise, if the online algorithm accepts request r_2, requests r_3 and r_4 are revealed together with capacity $c_2 = 2$ in the second time period. Now, the online algorithm can choose between r_3 and r_4, since r_1 is not available in the second time period. If the online algorithm accepts request r_3, capacity $c_3 = 0$ is revealed, the offline algorithm accepts r_1 in the first time period and r_2 and r_4 in the second time period, and the competitive ratio is given by $(v_1+v_2+v_4)/(v_2+v_3)$. Due to $c_3 = 0$ the online algorithm is not able to accept request r_4.

Finally, if the online algorithm accepts request r_4, capacity $c_3 = 2$ is revealed and the offline algorithm accepts r_1 in the first time period, r_3 in the second time period, and r_2 and r_4 in the third time period. The competitive ratio is given by $(v_1+v_2+v_3+v_4)/(v_2+v_4)$. The rejection of both r_3 and r_4 leads to a larger competitive ratio for the online algorithm due to the same argumentation as above.

Recapitulating, no deterministic online algorithm can achieve a competitive ratio smaller than

$$\max_{v_1,\dots,v_4} \min \left\{ \frac{v_2}{v_1}, \frac{v_1+v_2+v_4}{v_2+v_3}, \frac{v_1+v_2+v_3+v_4}{v_2+v_4} \right\}.$$

For $\varphi v_1 = v_2$ and $\varphi v_3 = v_4$ (for example, $v_1 = 1$, $v_2 = \varphi$, $v_3 = 1$, $v_4 = \varphi$) the three ratios above are equal to each other since $\varphi^2 = \varphi + 1$ for $\varphi = (1+\sqrt{5})/2$ and thus:

$$\begin{aligned} \frac{v_2}{v_1} &= \varphi, \\ \frac{v_1+v_2+v_4}{v_2+v_3} &= \frac{(1+\varphi)v_1+\varphi v_3}{\varphi v_1+v_3} = \frac{\varphi((1+\varphi)v_1+\varphi v_3)}{\varphi^2 v_1+\varphi v_3} = \varphi, \text{ and} \\ \frac{v_1+v_2+v_3+v_4}{v_2+v_4} &= \frac{(1+\varphi)(v_1+v_3)}{\varphi(v_1+v_3)} = \frac{\varphi((1+\varphi)(v_1+v_3))}{\varphi^2(v_1+v_3)} = \varphi. \end{aligned}$$

Therefore, no deterministic online algorithm can achieve a competitive ratio smaller than φ. □

2.9.2 Competitive Online Algorithms

In this section, a deterministic 2-competitive online algorithm for OIK is presented. Consider the greedy algorithm for OIK given by Algorithm 10.

Algorithm 10: Greedy algorithm for OIK.

```
1 for $t = 1, \dots, T$ do
2     Solve the deterministic knapsack problem with capacity $c_t$ for all available
      requests, i.e., for all requests $r_i \in \mathcal{R}$ such that $r_i$ has not been accepted before
      and $a_i \leq t \leq d_i$.
```

Theorem 2.9.2. *Algorithm 10 achieves a competitive ratio of two for* OIK.

Proof. In order to establish a competitive ratio for Algorithm 10, we use a charging argument: the requests accepted by an optimal solution are charged to the requests accepted by the online algorithm in such a way that each request accepted by the online algorithm is at most charged twice its value (see also Figure 2.16).

For a given sequence of requests, consider an optimal offline solution OPT and the solution ALG obtained by Algorithm 10. If request r_i is accepted by OPT in any time period, the value v_i is charged to a request that is accepted by ALG in the following way:

If request r_i is accepted both by OPT and ALG (in any time period), charge the value v_i to the request r_i itself.

Otherwise, let $\mathcal{A}_t^{\mathrm{OPT}}$ be the index set of requests available in time period t by OPT and $\mathcal{S}_t^{\mathrm{OPT}}$ be the index set of requests accepted in time period t by OPT. Correspondingly, let $\mathcal{A}_t^{\mathrm{G}}$ be the index set of requests available in time period t for ALG and $\mathcal{S}_t^{\mathrm{G}}$ the index set of requests accepted in time period t by ALG. Request $r_i \in \mathcal{R}$ is available in time period t if $a_i \le t \le d_i$ and the request has not been accepted before. So far, we considered the requests accepted both by OPT and ALG in any time period, denoted by A:

$$\bigcup_{t=1}^{T} \mathcal{S}_t^{\mathrm{OPT}} \cap \bigcup_{t=1}^{T} \mathcal{S}_t^{\mathrm{G}} =: A.$$

Consider now the requests B_τ that are accepted by OPT in time period τ, but never by ALG, i.e.,

$$\mathcal{S}_\tau^{\mathrm{OPT}} \setminus \bigcup_{t=1}^{T} \mathcal{S}_t^{\mathrm{G}} =: B_\tau.$$

Note that, thereby, all requests accepted by OPT are covered, i.e.

$$A \cup \bigcup_{\tau=1}^{T} B_\tau = \bigcup_{t=1}^{T} \mathcal{S}_t^{\mathrm{OPT}}. \tag{2.32}$$

Since the requests B_τ are never accepted by ALG, they are available in time period τ for ALG. Consequently, the sum of the values of all requests accepted by ALG in time period τ is greater than or equal to the sum of the values of requests that are accepted by OPT in time period τ but never accepted by ALG, since these requests are available to ALG in time period τ and ALG solves the knapsack problem of all available requests to optimality. Formally, we have

$$\sum_{i \in B_\tau} v_i \le \sum_{i \in \mathcal{S}_\tau^{\mathrm{G}}} v_i. \tag{2.33}$$

For each time period τ, charge the requests $r_i \in B_\tau$ to the requests $r_j \in \mathcal{S}_\tau^{\mathrm{G}}$ evenly. Hence, we have

$$\begin{aligned}
\mathrm{OPT} &= \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_t^{\mathrm{OPT}}} v_i \\
&\overset{(2.32)}{=} \sum_{i \in A} v_i + \sum_{t=1}^{T} \sum_{i \in B_\tau} v_i \\
&\overset{(2.33)}{\le} \mathrm{ALG} + \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_t^{\mathrm{G}}} v_i \\
&= 2 \cdot \mathrm{ALG}.
\end{aligned}$$

□

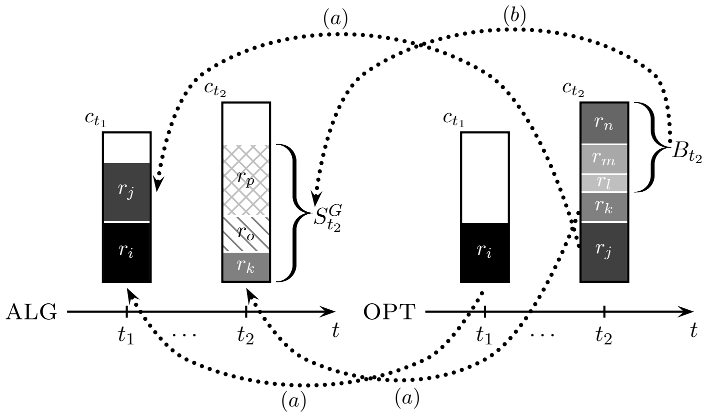


Figure 2.16: Charging of requests: (a) requests accepted by both ALG and OPT, (b) requests accepted only by OPT.

The analysis of Algorithm 10 is tight, which is established by Lemma 2.9.1.

Lemma 2.9.1. *The competitive ratio of Algorithm 10 is two.*

Proof. For $\epsilon > 0$, consider the requests

$$r_1 = (1, 2, 1 + \epsilon, 1),$$
$$r_2 = (1, 1, 1, 1),$$

and capacities $c_1 = c_2 = 1$. Algorithm 10 accepts r_1 in the first time period and cannot accept any request in the second time period. The optimal solution is to accept r_2 in the first time period and r_1 in the second time period. Therefore, the competitive ratio is given by ${}^{(2+\epsilon)}/_{(1+\epsilon)}$. Since ϵ is arbitrary this shows that the competitive ratio of Algorithm 10 is not smaller than two. □

The knapsack problem is an $\mathcal{NP}$-hard problem (Karp, 1972). Consequently, this raises the question, what happens to the competitive ratio of Algorithm 10 if the knapsack problem in each time period is not solved to optimality but by an α-approximation algorithm for the knapsack problem? A proper reconsideration of the proof of Theorem 2.9.2 gives the answer.

Corollary 2.9.1. *If Algorithm 10 solves the knapsack problem in each time period by an α-approximation algorithm, Algorithm 10 achieves a competitive ratio of $1 + \alpha$ for the problem* OIK.

Proof. The proof is analogous to the proof of Theorem 2.9.2, except for the charging of the requests B_τ that are accepted in time period τ by OPT, but never by ALG. For (2.33) we now have

$$\sum_{i \in B_\tau} v_i \leq \alpha \sum_{i \in \mathcal{S}_\tau^G} v_i, \tag{2.34}$$

since the requests B_τ are available to ALG in time period τ and ALG solves the knapsack problem of all available requests by an α-approximation algorithm. Again, for each time period, charge the requests $i \in B_\tau$ to the requests $j \in \mathcal{S}_\tau^G$ evenly. Due to (2.34) and since the value of each request in A was charged to itself as described in Theorem 2.9.2, each request accepted by ALG is charged at most $(1 + \alpha)$-times its value on average. □

From a theoretical point of view, a fully polynomial time approximation scheme (FPTAS) is the best approximation for an $\mathcal{NP}$-hard optimization problem, assuming $\mathcal{P} \neq \mathcal{NP}$. Since there exists an FPTAS for the knapsack problem (cf. (Ibarra and Kim, 1975)), Algorithm 10 achieves a competitive ratio of $2 + \epsilon$ for any given $\epsilon > 0$ in polynomial time.

2.10 Conclusion and Future Research

In this chapter, we considered the online knapsack problem with incremental capacity. For the restriction to unit weight items, we presented lower bounds on the competitive ratio and algorithms with competitive ratios matching these lower bounds exactly in the randomized case and for $k \to \infty$ in the deterministic case. For limited weights in $\{1, \dots, k\}$, a deterministic algorithm with a competitive ratio matching the lower bound for $k \to \infty$ and a randomized algorithm with a competitive ratio matching the lower bound up to a factor of 3 have been developed. In order to be able to develop algorithms with competitive ratios independent of T, we investigated the approaches of resource augmentation, removable items, and bounded values. Additionally, we provided average-case competitivities for the developed algorithms and showed how to obtain competitive algorithms for the problem with multiple knapsacks. Finally, we studied the setting of independent knapsacks and presented a 2-competitive algorithm and a lower bound on the competitive ratio for any deterministic algorithm of $(1+\sqrt{5})/2$.

For some problems presented in this chapter, there is still a gap between the lower and upper bound on the competitive ratio of any deterministic or randomized algorithm, which is subject to further research. Additionally, questions for future research include the study of the online knapsack problem with incremental capacity in a stochastic setting using average case analysis. Another direction for future research could be to study further generalizations of the online knapsack problem with incremental capacity, for example, the case where the increase in capacity is not identical in each time period, but varies over time.

3 Multi-Objective Online Optimization

3.1 Introduction

Online optimization is a helpful tool for various single-objective decision problems. However, online problems may also be of multi-objective nature. Imagine you want to sell your antique car and you are facing a sequence of offers by different people which you have to reject or accept immediately since the potential buyers are not willing to wait for a decision at a later time. On the one hand, you are eager to reach a high price, on the other hand, you want to know your antique car in safe keeping. Consequently, you evaluate each offer not only by the price, but also by your appreciation for the potential buyer. This leads to a bi-objective online problem.

In general, the concept of competitive analysis for online problems is only applied to single-objective online problems. However, the decision process of many online problems is subject to multiple objectives in real-world situations, but a uniform theory for the analysis of multi-objective online problems is not provided in the literature so far. In this chapter, we close this gap and expand the concept of competitive analysis to multi-objective online problems.

3.1.1 Previous Work

For an overview on the topic of multi-objective optimization, we refer to the textbook by Ehrgott (2005). The definitions from the field of multi-objective optimization that are relevant for the development of the concept of competitive analysis for multi-objective online problems are given in the course of this section when needed. To the best of our knowledge, there exists no general definition of competitive analysis for multi-objective online problems. However, some approaches in the field of online optimization are related to multi-objective online optimization:

Resource augmentation refers to a relaxed notion of competitive analysis, in which the online player is allowed more resources than the adversary, see for example (Kalyanasundaram and Pruhs, 2000; Phillips et al., 1997). Competitive ratios are then stated with respect to a fixed resource augmentation, which can be seen as a bi-objective online problem. Furthermore, for preemptive online scheduling problems, the trade-off between the competitive ratio and the cost of preemption is considered, for example, in (Motwani et al., 1994), which is also a bi-objective setting. However, the competitive ratios considered for these problems are the classical single-objective competitive ratios, while we derive a notion for a multi-objective competitive ratio, i.e., for defining a competitive ratio for a multi-objective online problem.

Furthermore, online algorithms are closely related to approximation algorithms. In the field of multi-objective optimization, a solution x of a multi-objective maximization problem is called a ρ-approximation of a solution x' if $f_i(x) \geq \rho \cdot f_i(x')$ for $i = 1, \ldots, n$, where f_i, $i = 1, \ldots, n$, are the components of the objective function and $0 < \rho \leq 1$. A set of feasible solutions X' is called a ρ-approximation of a set of efficient solutions if, for every feasible solution x, X' contains a feasible solution x' that is a ρ-approximation of x (see, for example, (Bazgan et al., 2012; Erlebach et al., 2002)). In (Papadimitriou and Yannakakis, 2000), the authors point out that, under very general conditions, there exists a $(1 - \epsilon)$-approximation of the set of efficient solutions, for any $\epsilon > 0$, whose size is polynomial both in the size of the instance and in $1/\epsilon$.

In our approach, the output generated by an online algorithm is, due to the online nature of the problem, a single solution instead of a set of solutions. Therefore, the competitiveness of a multi-objective online algorithm as introduced in this work is not directly deducible from the concept of ρ-approximation, but in some cases a close relation between our approach of competitive analysis for multi-objective online algorithms and multi-objective approximation algorithms is given.

The rest of this chapter is organized as follows: In Section 3.2, we introduce the notion of a multi-objective online problem and define the concept of competitive analysis for multi-objective online problems. In the following sections, we make use of the newly developed concept of multi-objective competitive analysis and present competitive multi-objective online algorithms for multi-objective variants of classical problems in the field of online optimization: in Section 3.3, we analyze the multi-objective time series search problem, in Section 3.4, we discuss a bi-objective variant of the ski rental problem, in Section 3.5, we investigate the bi-objective 2-server problem in the plane, and, in Section 3.6, we analyze the multi-objective k-Canadian traveller problem. In Section 3.7, we discuss general relations between single- and multi-objective online problems and, in Section 3.8, we extend Yao's principle to multi-objective online problems. Finally, we give concluding remarks in Section 3.9.

3.2 Competitive Analysis for Multi-Objective Online Algorithms

In this section, we first introduce the notion of a multi-objective online problem and, secondly, define the concept of competitive analysis for multi-objective online problems.

3.2.1 Multi-Objective Online Problems

In the following, we define the concept of competitive analysis for multi-objective online problems with respect to *minimization problems*. If not mentioned otherwise, the definition for the corresponding maximization problem is analogous. First of all, we define a *multi-objective optimization problem* $\mathcal{P}$ as a triple $(\mathcal{I}, \mathcal{X}, f)$, consisting of a set of inputs $\mathcal{I}$, a set of feasible outputs (or solutions) $\mathcal{X}(I)$ associated with every input $I \in \mathcal{I}$, and the objective function f given as $f : \mathcal{I} \times \mathcal{X} \to \mathbb{R}^n_+$ where, for $\mathbf{x} \in \mathcal{X}(I)$, $f(I, \mathbf{x})$ represents the objective value of the solution $\mathbf{x}$ with respect to input $I \in \mathcal{I}$.

Given input $I \in \mathcal{I}$, an *algorithm* $\textsc{alg}$ for a multi-objective optimization problem $\mathcal{P}$ computes a feasible solution $\textsc{alg}[I] \in \mathcal{X}(I)$. The objective associated with this feasible output is denoted by $\textsc{alg}(I) = f(I, \textsc{alg}[I])$. According to (Ehrgott, 2005, p. 24), a feasible solution $\hat{\mathbf{x}} \in \mathcal{X}(I)$ is called efficient if there is no other $\mathbf{x} \in \mathcal{X}(I)$ such that $f(I, x) \preceq f(I, \hat{x})$, where $\preceq$ denotes a component-wise order, i.e., for $x, y \in \mathbb{R}^n$, $x \preceq y :\Leftrightarrow x_i \leq y_i$, for $i = 1, \ldots, n$, and $x \neq y$. An *optimal algorithm* $\textsc{opt}$ for $\mathcal{P}$ is such that, for all inputs $I \in \mathcal{I}$, $\textsc{opt}[I]$ is the set of efficient solutions to $\mathcal{P}$, i.e.,

$$\textsc{opt}[I] = \{\mathbf{x} \in \mathcal{X}(I) \,|\, \mathbf{x} \text{ is an efficient solution to } \mathcal{P}\}\,.$$

The objective associated with a solution $\mathbf{x} \in \textsc{opt}[I]$ is denoted by $\textsc{opt}(\mathbf{x})$.

The definition of a multi-objective online problem is now given analogously to the definition of a single-objective online optimization problem given in (Borodin and El-Yaniv, 1998, p. 2). Accordingly, *multi-objective online problems* are multi-objective optimization problems in which the input is revealed bit by bit and an output must be produced in an online manner, i.e., after each new bit of input a decision affecting the output must be made.

3.2.2 The Competitive Ratio and Competitiveness

The study of online problems is concerned with assessing the quality of corresponding online algorithms and, ultimately, the question of which is the best algorithm. We carry this leading question forward to multi-objective online problems. In the following, we list conditions that are supposed to be met by an appropriate measure for the quality of multi-objective online algorithms:

Condition 1 (worst case model): Just as in the case of competitive analysis for single-objective algorithms (cf. (Fiat and Woeginger, 1998, p.4)), we aim for a worst case model for multi-objective competitive analysis that holds for any input distribution in order to avoid the problems of probabilistic models.

Condition 2 (worst case ratio): Furthermore, a standard worst case analysis of multi-objective online algorithms leads to the same pitfall as in the single-objective case (cf. (Fiat and Woeginger, 1998, p.3)): Due to the incomplete knowledge of the online algorithm, it is often possible to ensure that each decision made by an online algorithm is the worst possible decision with respect to all components. For example, consider the multi-objective time series search problem (see Section 3.3): if a sequence consisting only of the minimal price vector is revealed, the online player always ends up with the minimal price vector regardless of his strategy.

Therefore, following the underlying idea of competitive analysis, it is desirable to consider the ratio of the algorithm's performance and the optimal performance in every component on the same problem instance.

Condition 3 (independence from efficient solutions): Normally, the solution to a multi-objective optimization problem is given by a set of efficient solutions (see, for example, (Ehrgott, 2005)). However, due to the online nature of our approach and the corresponding urge to obtain an autonomous algorithm, we assume a multi-objective online algorithm to compute a single solution instead of a set of solutions. The competitive ratio should, nevertheless, be independent of a particular solution chosen from the set of efficient solutions of the offline problem.

Condition 4 (total order): In order to compare different multi-objective online algorithms, a total order on the competitive ratio of multi-objective online algorithms is necessary.

These requirements lead us to the following definition of c-competitiveness for multi-objective online algorithms:

Definition 3.2.1. *A multi-objective online algorithm* ALG *is* c-competitive *if, for all finite input sequences I, there exists an efficient solution $\mathbf{x} \in \text{OPT}[I]$ such that*

$$\text{ALG}(I)_i \leq c_i \cdot \text{OPT}(\mathbf{x})_i + \alpha_i, \ \textit{for } i = 1, \ldots, n\,,$$

where $c = \big(c_1, \ldots, c_n\big)^\intercal$ and $\alpha \in \mathbb{R}^n$ is a constant vector independent of I.

Note that c is a vector instead of a scalar as in the classic definition of competitiveness for single-objective online algorithms. A multi-objective online algorithm which accomplishes this postulation even for all efficient solutions is called *strongly c-competitive*:

Definition 3.2.2. *A multi-objective online algorithm* ALG *is* strongly c-competitive *if, for all finite input sequences I and all efficient solutions $\mathbf{x} \in \text{OPT}[I]$,*

$$\text{ALG}(I)_i \leq c_i \cdot \text{OPT}(\mathbf{x})_i + \alpha_i, \ \textit{for } i = 1, \ldots, n\,,$$

where $c = \big(c_1, \ldots, c_n\big)^\intercal$ and $\alpha \in \mathbb{R}^n$ is a constant vector independent of I.

Applying these definitions to single-objective problems results in the classical single-objective competitive ratio for both Definition 3.2.1 and Definition 3.2.2. Obviously, every strongly c-competitive multi-objective online algorithm is also c-competitive. For maximization problems, the inequalities in Definitions 3.2.1 and 3.2.2 are replaced by $\text{ALG}(I)_i \geq {}^1\!/_{c_i} \cdot \text{OPT}(\mathbf{x})_i + \alpha_i$.

The definition of competitiveness for multi-objective online algorithms is a worst case ratio due to the consideration of all finite input sequences as required by Conditions 1 and 2. Furthermore, the definition takes the set of all efficient offline solutions into account and hence does not rely on a particular efficient solution, as demanded by Condition 3. In order to achieve a comparable competitive ratio of multi-objective online algorithms as demanded by Condition 4, a total order on the competitiveness of an online algorithm is necessary. This gives rise to the following definition of the competitive ratio for multi-objective online algorithms:

Definition 3.2.3. *Let $f : \mathbb{R}^n \to \mathbb{R}_+$. The infimum over the set of all values $f(c)$ such that* ALG *is (strongly) c-competitive is called* the (strong) competitive ratio with respect to f *of* ALG *and is denoted by* $(\mathcal{R}^f_s(\text{ALG}))$ $\mathcal{R}^f(\text{ALG})$.

The choice of the function f grants a certain degree of freedom that is left to the analyst of the online algorithm (in the style of the decision maker in the field of multi-objective optimization). However, f has to be chosen such that $f(c) \le f(\hat{c})$ if $c_i \le \hat{c}_i$ for $i = 1, \dots, n$ in order to guarantee a reasonable setting.

In this work, we consider three intuitive choices for the function f. First of all, consider f_1 given as $f_1(c) := \max_{i=1,\dots,n} c_i$. By this choice, the competitive ratio is guaranteed for each component of the objective function. We label this choice as *worst-component competitive ratio.* Further, we consider f given by $f_2(c) := \frac{1}{n}\sum_{i=1}^n c_i$ and $f_3(c) := \sqrt[n]{\prod_{i=1}^n c_i}$. In these cases, the arithmetic and geometric mean value of the components' competitive ratios is taken, which is why these choices are labeled as *arithmetic-* and *geometric-mean-component competitive ratio.*

For randomized multi-objective online algorithms, the definition of the (strong) competitive ratio is given in the same way. Let ALG be a randomized multi-objective online algorithm. An oblivious adversary must choose a finite input I in advance, based on the knowledge of the probability distribution(s) ALG uses.

Definition 3.2.4. *A randomized multi-objective online algorithm* ALG *is* c-competitive against an oblivious adversary *if, for every input I chosen as described above, there exists an efficient solution $\mathbf{x} \in \text{OPT}[I]$ such that*

$$\mathbb{E}\left[\text{ALG}(I)_i\right] \le c_i \cdot \text{OPT}(\mathbf{x})_i + \alpha_i, \text{ for } i = 1, \dots, n,$$

where $c = \left(c_1, \dots, c_n\right)^\intercal$ and $\alpha \in \mathbb{R}^n$ is a constant independent of I.

Definition 3.2.5. *A randomized multi-objective online algorithm* ALG *is* strongly c-competitive against an oblivious adversary *if, for every input I chosen as described above and all efficient solution $\mathbf{x} \in \text{OPT}[I]$,*

$$\mathbb{E}\left[\text{ALG}(I)_i\right] \le c_i \cdot \text{OPT}(\mathbf{x})_i + \alpha_i, \text{ for } i = 1, \dots, n,$$

where $c = \left(c_1, \dots, c_n\right)^\intercal$ and $\alpha \in \mathbb{R}^n$ is a constant independent of I.

Note that $\mathbb{E}\left[\text{ALG}(I)_i\right]$ is the expected value of the i-th component of ALG with respect to its randomized decisions. Again, applying these definitions to single-objective problems results in the classical single-objective competitive ratio and, for maximization problems, the inequalities in Definitions 3.2.4 and 3.2.5 are replaced by $\mathbb{E}\left[\textsc{alg}(I)\right] \geq {}^1\!/\!_{c_i} \cdot \textsc{opt}(\mathbf{x}) + \alpha$.

The definition of the competitive ratio for randomized multi-objective online algorithms is given accordingly:

Definition 3.2.6. *Let $f : \mathbb{R}^n \to \mathbb{R}_+$. The infimum over the set of all values $f(c)$ such that* ALG *is (strongly) c-competitive against an oblivious adversary is called* ALG*'s* (strong) competitive ratio with respect to f against an oblivious adversary *and is denoted by* $(\overline{\mathcal{R}}^f_s(\textsc{alg}))$ $\overline{\mathcal{R}}^f(\textsc{alg})$.

In online optimization, two further adversary models for randomized algorithms are known, namely the adaptive-online adversary and the adaptive-offline adversary (see Section 1.1). The definitions of the competitive ratio for these adversaries can be accomplished analogously in a straightforward manner, just as it is done for the oblivious adversary. However, the concept of the oblivious adversary is the most widely used concept, and since the focus of this work is the initial establishment of multi-objective online optimization, the adaptive-online adversary and adaptive-offline adversary are not considered in this work.

3.3 The Multi-Objective Time Series Search Problem

In this section, the concept of competitive analysis for multi-objective online problems is applied to the classic time series search problem, where an online player is searching for the maximum (or minimum) price in a sequence of prices. At the beginning of each time period $t = 1, \dots, T$, a price p_t is revealed to the online player and the player must decide whether to accept or reject the price p_t. When the player accepts a price p_t, the game ends and the return for the player is p_t.

Within the framework of competitive analysis, the time series search problem is initially investigated in (El-Yaniv et al., 2001) and competitive search algorithms are provided. Here, it is assumed that prices are chosen from the real interval $[m, M]$, where $0 < m \leq M$ and the online player can always end the game by accepting the minimum price m; otherwise the adversary is too powerful and there exist no competitive algorithms for the problem. The ratio $\varphi := {}^M\!/\!_m$ is defined as the fluctuation ratio of possible prices. El-Yaniv et al. prove that, if only the fluctuation ratio φ is known to the online player, no better ratio than the trivial one of φ is achievable. Therefore, suppose that both m and M are known to the online player. Then, the best possible deterministic online algorithm is the reservation price policy RPP, where the algorithm accepts the first price greater than or equal to $p^\star = \sqrt{Mm}$, achieving a competitive ratio of $\sqrt{\varphi}$, as shown in (El-Yaniv et al., 2001).

The competitive ratio can be improved by means of randomized algorithms. Assume that $\varphi = 2^k$. The randomized algorithm EXPO presented in (El-Yaniv et al., 2001), which chooses the reservation price policy with reservation price $m2^l$, $l = 0, \dots, k-1$, with probability $1/k$ each before the start of the game, is $\mathcal{O}\left(\log(\varphi)\right)$-competitive, which is within a constant factor of the best possible competitive ratio. This result also holds when φ is not a power of 2. The best possible competitive ratio is achieved by algorithms that obey threat-based policies (cf. (El-Yaniv et al., 2001)).

In order to investigate the time series search problem in a multi-objective setting, a price *vector* is introduced, i.e., at each time period $t = 1, \dots, T$, a request $r_t \in \mathbb{R}^n_+$ is revealed to the online player, where $r_t = \left(p_t^1, \dots, p_t^n\right)^\intercal$, and the player must decide whether to accept or reject r_t. When the player accepts a price vector r_t, the game ends and the return for the player is r_t. It is assumed that, for $i = 1, \dots, n$, p_t^i is chosen from the real interval $[m_i, M_i]$, where $0 < m_i \leq M_i$. For $i = 1, \dots, n$, the ratios $\varphi_i := {}^{M_i}/_{m_i}$ are defined as the fluctuation ratios of possible prices for the price component i. The online player can always end the game by accepting the minimum price vector $\left(m_1, \dots, m_n\right)$. Without loss of generality, we assume for the fluctuation ratios that ${}^{M_1}/_{m_1} \geq {}^{M_2}/_{m_2} \geq \dots \geq {}^{M_n}/_{m_n}$.

3.3.1 Worst-Component Competitive Analysis

In this section, a competitive analysis with respect to $f_1(c) := \max_{i=1,\dots,n} c_i$, i.e., a worst-component competitive analysis, for the multi-objective time series search problem is presented.

In order to develop a deterministic algorithm for the multi-objective time series search problem the idea of a reservation price policy as for the single-objective time series search problem is transferred to the multi-objective setting. A first (obvious) approach is to apply the single-objective reservation price policy to multiple components. We start by considering not only the component with the highest fluctuation ratio, but also the component with the second highest fluctuation ratio, and apply the single-objective reservation price policy to both components, see Figure 3.1. This policy is formally captured by Algorithm 11, also denoted by RPP-AND. Remember that ${}^{M_1}/_{m_1} \geq {}^{M_2}/_{m_2} \geq \dots \geq {}^{M_n}/_{m_n}$

Algorithm 11: Multi-objective reservation price policy RPP-AND.

```
1 for $t = 1, \dots, T$ do
2     Accept $r_t = \left(p_t^1, \dots, p_t^n\right)^\intercal$ if $p_t^1 \geq p_\star^1$ and $p_t^2 \geq p_\star^2$.
```

As shown by the following theorem, RPP-AND is only φ_1-competitive which is achievable by any algorithm.

Theorem 3.3.1. *The (strong) competitive ratio with respect to $f_1(c) = \max_{i=1,\dots,n} c_i$ of* RPP-AND *is given by $\varphi_1 = {}^{M_1}/_{m_1}$.*

Proof. Consider the request sequence $\sigma = (r_1)$ consisting of only one request,

$$r_1 = \left(M_1, \, p_\star^2 - \epsilon, \, m_3, \dots, m_n\right)^\intercal, \; \epsilon > 0.$$

The online player obviously rejects request r_1 since $p_\star^2 - \epsilon < p_\star^2$ and has to settle for the lower bound in each component, i.e., ALG $= \left(m_1, \dots, m_n\right)^\intercal$. However, the adversary

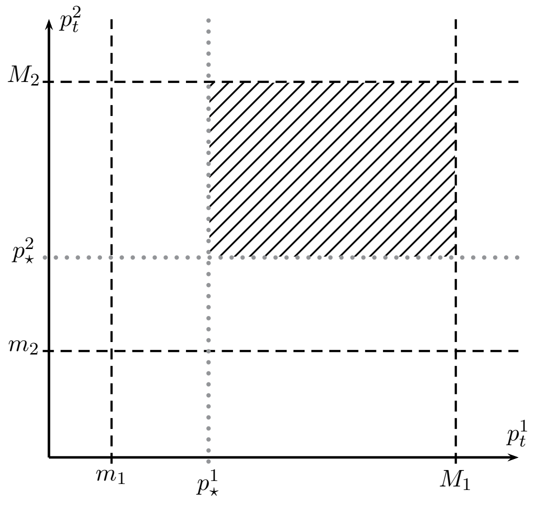


Figure 3.1: Acceptance region of RPP-AND.

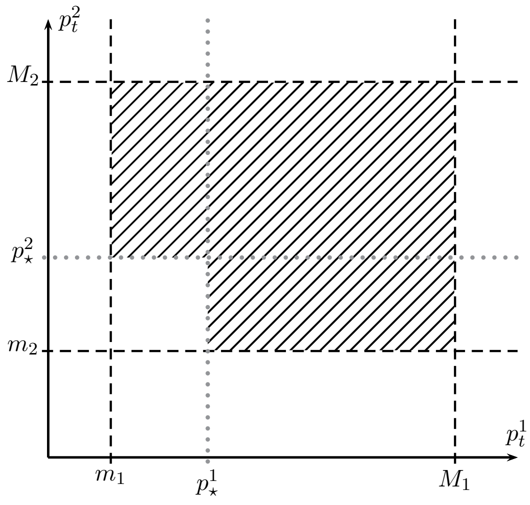


Figure 3.2: Acceptance region of RPP-OR.

accepts the request, which is also the only efficient solution, and, therefore, the optimal solution for the adversary, i.e., OPT $= \big(M_1, p_\star^2 - \epsilon, m_3, \ldots, m_n\big)^\intercal$.

The multi-objective time series search problem is a maximization problem. According to Definition 3.2.1, a multi-objective online algorithm ALG is c-competitive if, for all finite input sequences I, there exists an efficient solution $\mathbf{x} \in \text{OPT}[I]$ such that

$$\text{ALG}(I)_i \geq \frac{1}{c_i} \cdot \text{OPT}(\mathbf{x})_i + \alpha_i, \text{ for } i = 1, \ldots, n,$$

where ALG $= \big(m_1, m_2, m_3, \ldots, m_n\big)^\intercal$ and OPT $= \big(M_1, p_\star^2 - \epsilon, m_3, \ldots, m_n\big)^\intercal$. By Definition 3.2.3, the competitive ratio with respect to $f_1(c) = \max_{i=1,\ldots,n} c_i$ is given by the infimum over the set of all values $f_1(c)$ such that ALG is c-competitive, i.e.,

$$\mathcal{R}^{f_1}(\text{RPP-AND}) = \max\left\{\frac{M_1}{m_1}, \frac{p_\star^2 - \epsilon}{m_2}, \frac{m_3}{m_3}, \ldots, \frac{m_n}{m_n}\right\} = \frac{M_1}{m_1},$$

since any reasonable choice for $p_\star^2$ is smaller than or equal to M_2 and, by assumption, $M_1/m_1 \geq M_2/m_2 \geq \cdots \geq M_n/m_n$.

Since an algorithm's strong competitive ratio is never better than its competitive ratio and RPP-AND achieves only the worst possible competitive ratio M_1/m_1, the strong competitive ratio of RPP-AND is also given by M_1/m_1. □

The analysis of Algorithm 11 is independent of $p_\star^1$ and $p_\star^2$ and, therefore, holds for any choice of $p_\star^1$ and $p_\star^2$. Furthermore, it is not advantageous for the online player to include further components of the price vector and accept a request only if *all* included components exceed certain reservations prices: the request sequence σ in the proof of Theorem 3.3.1 would lead to the same result.

Another approach is to accept a request if the corresponding reservation price policy is satisfied in *at least one* component, see Figure 3.2. In the following, this policy is again applied to the components with the best and the second best fluctuation ratio. Formally, this policy is given by Algorithm 12, also denoted by RPP-OR.

Algorithm 12: Multi-objective reservation price policy RPP-OR.

```
1 for $t = 1, \dots, T$ do
2     Accept $r_t = (p_t^1, \dots, p_t^n)^\intercal$ if $p_t^1 \geq p_\star^1$ or $p_t^2 \geq p_\star^2$.
```

Nevertheless, RPP-OR also achieves only the trivial competitive ratio $\varphi_1 = M_1/m_1$, as shown by the following theorem.

Theorem 3.3.2. *The (strong) competitive ratio with respect to $f_1(c) = \max_{i=1,\dots,n} c_i$ of* RPP-OR *is given by $\varphi_1 = M_1/m_1$.*

Proof. Consider the request sequence $\sigma = (r_1, r_2)$, consisting of two requests,

$$r_1 = \big(m_1,\, p_\star^2,\, m_3, \dots, m_n\big)^\intercal \text{ and } r_2 = \big(M_1, \dots, M_n\big)^\intercal .$$

The online player accepts request r_1, i.e., ALG $= \big(m_1,\, p_\star^2,\, m_3, \dots, m_n\big)^\intercal$. The adversary rejects request r_1 and accepts r_2, which is the only efficient solution and, therefore, the optimal solution for the adversary, i.e., OPT $= \big(M_1, \dots, M_n\big)^\intercal$. Consequently, the competitive ratio with respect to f_1 is in this case given by

$$\mathcal{R}^{f_1}(\text{RPP-OR}) = \max\left\{\frac{M_1}{m_1}, \frac{M_2}{p_\star^2}, \frac{M_3}{m_3}, \dots, \frac{M_n}{m_n}\right\} = \frac{M_1}{m_1},$$

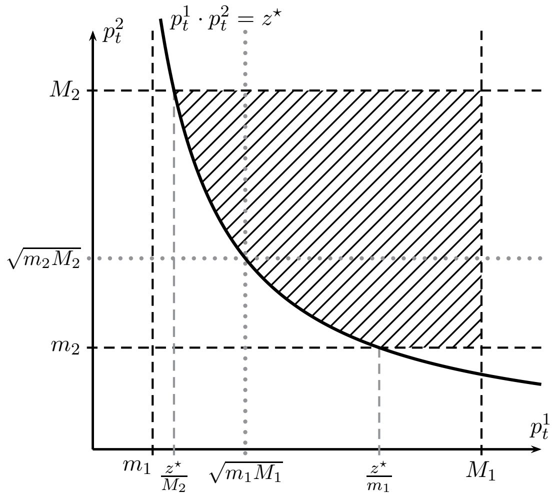
since any reasonable choice for $p_\star^2$ is greater than or equal to m_2 and, by assumption, $M_1/m_1 \geq M_2/m_2 \geq \dots \geq M_n/m_n$. Obviously, the strong competitive ratio of RPP-OR is also given by M_1/m_1. □

As for RPP-AND, the analysis of Algorithm 12 is independent of $p_\star^1$ and $p_\star^2$ and, therefore, holds for any choice of $p_\star^1$ and $p_\star^2$. Additionally, it is again not advantageous for the online player to include further components of the price vector and accept a request only if *at least one* of the included components exceeds a certain reservation price: the request sequence σ in the proof of Theorem 3.3.2 would lead to the same result.

It seems that RPP-AND's policy is too reserved and RPP-OR's policy is too greedy, see also Figures 3.1 and 3.2. Thus, we define an algorithm with acceptance region "in between" RPP-AND and RPP-OR in order to balance being reserved and being greedy.

Algorithm 13: Multi-objective reservation price policy RPP-PROD.

```
1 for $t = 1, \dots, T$ do
2     Accept $r_t = (p_t^1, \dots, p_t^n)^\intercal$ if $p_t^1 \cdot p_t^2 \geq z^\star$, where $z^\star = \sqrt{m_1 M_1 m_2 M_2}$.
```



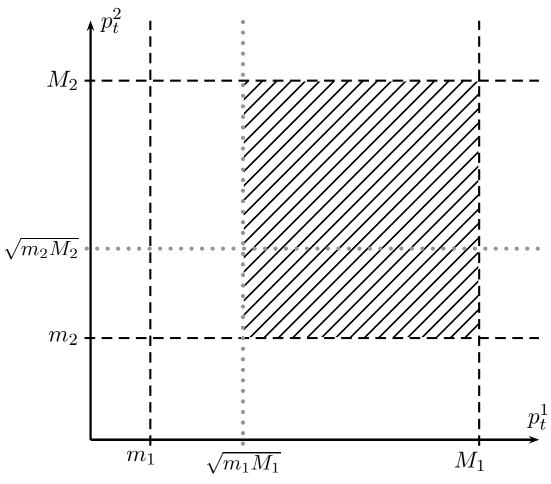


Figure 3.3: Acceptance region of RPP-PROD. Figure 3.4: Acceptance region of RPP-HIGH.

In order to illustrate the behavior of RPP-PROD, the acceptance region is depicted in Figure 3.3.

Theorem 3.3.3. *The strong competitive ratio with respect to* $f_1(c) = \max_{i=1,\ldots,n} c_i$ *of* RPP-PROD *is given by*

$$\mathcal{R}_s^{f_1}(\text{RPP-PROD}) = \sqrt{M_1 M_2 / m_1 m_2}.$$

Proof. We distinguish two cases with respect to the request sequence $\sigma = (r_1, \ldots, r_T)$ revealed by the adversary:

Case 1: there exists a request $r_{t'}$ with $p_{t'}^1 \cdot p_{t'}^2 \geq z^\star$.

In this case, the online player accepts the first request r_t with $p_t^1 \cdot p_t^2 \geq z^\star$, i.e., ALG $= \big(p_t^1, \ldots, p_t^n\big)^\mathsf{T}$. However, the adversary is able to reveal a further request r_j with $p_j^i = M_i$ for $i = 1, \ldots, n$, which is then the only efficient offline solution and, therefore, the optimal solution for the adversary, i.e., OPT $= \big(M_1, \ldots, M_n\big)^\mathsf{T}$.

In the worst case with respect to all sequences, the request r_t accepted by the online player is such that $p_t^1 \cdot p_t^2 = z^\star$. The set of all points $\big(p_t^1, \ldots, p_t^n\big)^\mathsf{T}$ in $[m_1, M_1] \times \cdots \times [m_n, M_n]$ satisfying $p_t^1 \cdot p_t^2 = z^\star$ is given by

$$\mathcal{I}_1 = \Bigg\{ \Big(x_1, \tfrac{z^\star}{x_1}, x_3, \ldots, x_n\Big)^\mathsf{T} \mid \frac{z^\star}{M_2} \leq x_1 \leq \frac{z^\star}{m_2} \text{ and } x_i \in [m_i, M_i] \text{ for } i = 3, \ldots, n \Bigg\}, \tag{3.1}$$

since

$$\frac{z^\star}{M_2} = \frac{\sqrt{m_1 M_1 m_2 M_2}}{M_2} \geq m_1 \quad (\text{due to } \frac{M_1}{m_1} \geq \frac{M_2}{m_2})$$

and

$$\frac{z^\star}{m_2} = \frac{\sqrt{m_1 M_1 m_2 M_2}}{m_2} \leq M_1 \quad (\text{due to } \frac{M_1}{m_1} \geq \frac{M_2}{m_2}).$$

Thus, the competitive ratio is in this case given by

$$\begin{aligned} \max_{x \in \mathcal{I}_1} \max \left\{ \frac{M_1}{x_1}, \frac{M_2 x_1}{z^\star}, \frac{M_3}{x_3}, \dots, \frac{M_n}{x_n} \right\} &\overset{(3.1)}{\leq} \max \left\{ \frac{M_1 M_2}{z^\star}, \frac{M_2}{m_2} \right\} \\ &= \sqrt{\frac{M_1 M_2}{m_1 m_2}}, \end{aligned} \tag{3.2}$$

since ${M_1}/{m_1} \geq {M_2}/{m_2} \geq \cdots \geq {M_n}/{m_n}$.

Case 2: for all r_t, $t = 1, \dots, T$, we have $p_t^1 \cdot p_t^2 < z^\star$.
In this case, the online player does not accept any request and has to settle for the lower bounds in each component, i.e., ALG $= \big(m_1, \dots, m_n\big)^\mathsf{T}$.

The adversary is able to offer (and accept) any request r_j for which the product $p_t^1 \cdot p_t^2$ is smaller than but arbitrarily close to $z^\star$, i.e., $p_t^1 \cdot p_t^2 = z^\star - \epsilon$, $\epsilon > 0$. The set of efficient solutions for OPT is given by

$$\begin{aligned} \mathcal{I}_2 = \Bigg\{ \Big(x_1, \tfrac{z^\star - \epsilon}{x_1}, x_3, \dots, x_n \Big)^\mathsf{T} \mid \\ \frac{z^\star - \epsilon}{M_2} \leq x_1 \leq \frac{z^\star - \epsilon}{m_2} \text{ and } x_i \in [m_i, M_i] \text{ for } i = 3, \dots, n \Bigg\}, \end{aligned} \tag{3.3}$$

due to the same argumentation as in Case 1. Now, ignoring the ϵ, the competitive ratio is in this case given by

$$\begin{aligned} \max_{x \in \mathcal{I}_2} \max \left\{ \frac{x_1}{m_1}, \frac{z^\star}{x_1 m_2}, \frac{x_3}{m_3}, \dots, \frac{x_n}{m_n} \right\} &\overset{(3.3)}{\leq} \max \left\{ \frac{z^\star}{m_1 m_2}, \frac{M_2}{m_2} \right\} \\ &= \sqrt{\frac{M_1 M_2}{m_1 m_2}}, \end{aligned} \tag{3.4}$$

since ${M_1}/{m_1} \geq {M_2}/{m_2} \geq \cdots \geq {M_n}/{m_n}$.

By means of (3.2) and (3.4), the competitive ratio $\mathcal{R}^{f_1}$(RPP-PROD) results in

$$\mathcal{R}^{f_1}(\text{RPP-PROD}) = \sqrt{\frac{M_1 M_2}{m_1 m_2}}.$$

This result holds for all efficient solutions since, in the first case, there is exactly one efficient solution for the adversary and, in the second case, we considered the maximum over all $x \in \mathcal{I}_2$. Consequently, we have $\mathcal{R}^{f_1}$(RPP-PROD) $= \mathcal{R}_s^{f_1}$(RPP-PROD). □

If the fluctuation ratios φ_1 and φ_2 are equal, i.e., $M_1/m_1 = M_2/m_2$, RPP-PROD obtains only the trivial competitive ratio M_1/m_1, just as RPP-AND and RPP-OR. For $M_1/m_1 > M_2/m_2$, RPP-PROD's competitive ratio is better than the trivial competitive ratio. However, the best possible algorithm with respect to a worst case competitive analysis is given by considering only the component with the highest fluctuation ratio and applying the best single-objective strategy to this component. This policy is formally given by Algorithm 14 and is denoted by RPP-HIGH (see also Figure 3.4):

Algorithm 14: Multi-objective reservation price policy RPP-HIGH.

```
1 for $t = 1, \dots, T$ do
2     Accept $r_t = \left(p_t^1, \dots, p_t^n\right)^\mathsf{T}$ if $p_t^1 \geq \sqrt{m_1 M_1}$.
```

Theorem 3.3.4. *The strong competitive ratio with respect to* $f_1(c) = \max_{i=1,\dots,n} c_i$ *of* RPP-HIGH *is given by*

$$\mathcal{R}_s^{f_1}(\text{RPP-HIGH}) = \max\left\{\sqrt{\frac{M_1}{m_1}}, \frac{M_2}{m_2}\right\}.$$

Proof. We distinguish two cases with respect to the sequence $\sigma = (r_1, \dots, r_T)$ revealed by the adversary:

Case 1: there exists a request $r_{t'}$ with $p_{t'}^1 \geq \sqrt{m_1 M_1}$.

In this case, the online player accepts the first request r_t with $p_t^1 \geq \sqrt{m_1 M_1}$. However, the adversary is able to reveal a further request r_j with $p_j^i = M_i$ for $i = 1, \dots, n$, which is then the only efficient offline solution and, therefore, the optimal solution for the adversary, i.e., OPT $= \left(M_1, \dots, M_n\right)^\mathsf{T}$.

In the worst case with respect to all sequences, the request r_t accepted by the online player is such that $p_t^1 = \sqrt{m_1 M_1}$ and $p_t^i = m_i$ for $i = 2, \dots, n$. According to Definition 3.2.3, the competitive ratio is in this case given by

$$\max\left\{\frac{M_1}{\sqrt{m_1 M_1}}, \frac{M_2}{m_2}, \dots, \frac{M_n}{m_n}\right\} = \max\left\{\sqrt{\frac{M_1}{m_1}}, \frac{M_2}{m_2}\right\}, \tag{3.5}$$

since $M_1/m_1 \geq M_2/m_2 \geq \dots \geq M_n/m_n$.

Case 2: for all r_t, $t = 1, \dots, T$, we have $p_t^1 < \sqrt{m_1 M_1}$.

In this case, the online player does not accept any request and has to settle in the worst case for the lower bounds in each component, i.e., ALG $= \left(m_1, \dots, m_n\right)^\mathsf{T}$. The adversary is able to offer (and accept) any request r_j for which the first price component is smaller than but arbitrarily close to $\sqrt{m_1 M_1}$, i.e., $p_j^1 = \sqrt{m_1 M_1} - \epsilon$, $\epsilon > 0$. For the other components the upper bound is chosen, i.e., $p_j^i = M_i$ for $i = 2, \dots, n$. Ignoring the ϵ, the competitive ratio is in this case given by

$$\max\left\{\frac{\sqrt{m_1 M_1}}{m_1}, \frac{M_2}{m_2}, \dots, \frac{M_n}{m_n}\right\} = \max\left\{\sqrt{\frac{M_1}{m_1}}, \frac{M_2}{m_2}\right\}, \tag{3.6}$$

since ${}^{M_1}/_{m_1} \geq {}^{M_2}/_{m_2} \geq \cdots \geq {}^{M_n}/_{m_n}$.

The analysis above holds for all efficient solutions. Due to (3.5) and (3.6), the strong competitive ratio $\mathcal{R}_s^{f_1}(\text{RPP-HIGH})$ results in

$$\mathcal{R}_s^{f_1}(\text{RPP-HIGH}) = \max\left\{\sqrt{\frac{M_1}{m_1}}, \frac{M_2}{m_2}\right\}.$$

□

With respect to a worst-component competitive analysis, RPP-HIGH is the best possible deterministic algorithm for the multi-objective time series search problem which is proven by the following theorem:

Theorem 3.3.5. *No deterministic online algorithm for the multi-objective time series search problem can achieve a competitive ratio with respect to* $f_1(c) = \max_{i=1,\dots,n} c_i$ *smaller than* $\max\left\{\sqrt{{}^{M_1}/_{m_1}}, {}^{M_2}/_{m_2}\right\}$.

Proof. If $\sqrt{{}^{M_1}/_{m_1}} \leq {}^{M_2}/_{m_2}$, the adversary offers the request

$$r_1 = \left(m_1, M_2, m_3, \dots, m_n\right)^{\mathsf{T}}.$$

If the online player rejects r_1, no further requests are revealed, the online player has to settle for the lower bounds $\left(m_1, \dots, m_n\right)^{\mathsf{T}}$, and the adversary accepts r_1. In this case, the competitive ratio is given by the trivial competitive ratio ${}^{M_1}/_{m_1}$. Otherwise, if the online player accepts r_1, the adversary reveals another request $r_2 = \left(M_1, \dots, M_n\right)^{\mathsf{T}}$ and accepts this request. Thus, the competitive ratio is in this case given by

$$\max\left\{\frac{m_1}{m_1}, \frac{M_2}{m_2}, \frac{m_3}{m_3}, \dots, \frac{m_n}{m_n}\right\} = \frac{M_2}{m_2}. \tag{3.7}$$

If $\sqrt{{}^{M_1}/_{m_1}} > {}^{M_2}/_{m_2}$, the adversary offers the request

$$r_1 = \left(\sqrt{m_1 M_1}, m_2, \dots, m_n\right)^{\mathsf{T}}.$$

If the online player rejects r_1, no further requests are revealed, the online player has to settle for the lower bounds $\left(m_1, \dots, m_n\right)^{\mathsf{T}}$, and the adversary accepts r_1. In this case, the competitive ratio is given by

$$\max\left\{\frac{\sqrt{m_1 M_1}}{m_1}, \frac{m_2}{m_2}, \dots, \frac{m_n}{m_n}\right\} = \sqrt{\frac{M_1}{m_1}}. \tag{3.8}$$

Otherwise, if the online player accepts r_1, the adversary reveals another request $r_2 = \left(M_1, \dots, M_n\right)^{\mathsf{T}}$ and accepts this request. Thus, the competitive ratio is in this case given by

$$\max\left\{\frac{M_1}{\sqrt{m_1 M_1}}, \frac{M_2}{m_2}, \dots, \frac{M_n}{m_n}\right\} = \sqrt{\frac{M_1}{m_1}}. \tag{3.9}$$

By means of (3.7), (3.8), and (3.9), no deterministic algorithm for the time series search problem can achieve a smaller worst-component competitive ratio than

$$\max\left\{\sqrt{\frac{M_1}{m_1}},\, \frac{M_2}{m_2}\right\} = \mathcal{R}^{f_1}(\text{RPP-HIGH})\,.$$

□

With respect to a worst-component competitive analysis, only the component with the highest fluctuation ratio is decisive, and, therefore, the best possible deterministic single-objective policy applied to this component achieves the best competitive ratio.

3.3.2 Mean-Component Competitive Analysis

In this section, we present a competitive analysis with respect to $f_2(c) = \frac{1}{n}\sum_{i=1}^{n} c_i$ and $f_3(c) = \sqrt[n]{\prod_{i=1}^{n} c_i}$ for the multi-objective time series search problem, i.e., an arithmetic-mean-component and a geometric-mean-component competitive analysis. We consider the case $n = 2$, i.e., the bi-objective time series search problem. Note that the analysis of mean-component competitive algorithms for the multi-objective time series search problem for dimensions $n \geq 3$ is subject to further research.

The algorithm RPP-PROD achieves the best possible arithmetic-mean-component competitive ratio (see also Figure 3.3):

Theorem 3.3.6. *The strong competitive ratio with respect to $f_2(c) = \frac{1}{n}\sum_{i=1}^{n} c_i$ of* RPP-PROD *for the bi-objective time series search problem is given by*

$$\mathcal{R}_s^{f_2}(\text{RPP-PROD}) = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,.$$

Proof. We distinguish two cases with respect to the sequence $\sigma = (r_1, \ldots, r_T)$ revealed by the adversary:

Case 1: there exists a request $r_{t'}$ with $p_t^1 \cdot p_t^2 \geq z^\star$.

In this case, the online player accepts the first request r_t with $p_t^1 \cdot p_t^2 \geq z^\star$, i.e., ALG $= \left(p_t^1,\, p_t^2\right)^\intercal$. However, the adversary is able to reveal a further request r_j with $p_j^i = M_i$ for $i = 1, 2$, which is then the only efficient offline solution and, therefore, the optimal solution for the adversary, i.e., OPT $= \left(M_1,\, M_2\right)^\intercal$.

In the worst case with respect to all sequences, the request r_t accepted by the online player is such that $p_t^1 \cdot p_t^2 = z^\star$. The set of all points in $[m_1, M_1] \times [m_2, M_2]$ satisfying $p_t^1 \cdot p_t^2 = z^\star$ is given by (cf. (3.1))

$$\mathcal{I}_1 = \left\{ \left(x,\, \tfrac{z^\star}{x}\right)^\intercal \mid \frac{z^\star}{M_2} \leq x \leq \frac{z^\star}{m_2} \right\}\,. \tag{3.10}$$

Since $f_2(c) = \frac{1}{n}\sum_{i=1,\dots,n} c_i$ and due to (3.10), the competitive ratio is given by

$$\begin{aligned}\frac{1}{2}\max_{x\in\mathcal{I}_1}\left\{\frac{M_1}{x}+\frac{M_2 x}{z^\star}\right\} &= \frac{1}{2}\left(\frac{M_1}{\sqrt{\frac{M_1 z^\star}{M_2}}}+\frac{M_2\sqrt{\frac{M_1 z^\star}{M_2}}}{z^\star}\right)\\ &= \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,. \qquad (3.11)\end{aligned}$$

Case 2: for all r_t, $t = 1,\dots,T$, we have $p_t^1 \cdot p_t^2 < z^\star$.

In this case, the online player does not accept any request and has to settle in the worst case for the lower bounds in each component, i.e., ALG $= \big(m_1,\, m_2\big)^\mathsf{T}$.

The adversary is able to offer (and accept) any request r_j for which the product $p_t^1 \cdot p_t^2$ is smaller than but arbitrarily close to $z^\star$, i.e., $p_t^1 \cdot p_t^2 = z^\star - \epsilon$, $\epsilon > 0$. The set of efficient solutions for OPT is given by (cf. (3.1))

$$\mathcal{I}_2 = \left\{\big(x,\, \tfrac{z^\star-\epsilon}{x}\big)^\mathsf{T} \mid \frac{z^\star-\epsilon}{M_2} \le x \le \frac{z^\star-\epsilon}{m_2}\right\}. \qquad (3.12)$$

Now, ignoring the ϵ and due to (3.12), the competitive ratio is given by

$$\begin{aligned}\frac{1}{2}\max_{x\in\mathcal{I}_2}\left\{\frac{x}{m_1}+\frac{z^\star}{m_2 x}\right\} &= \frac{1}{2}\left(\frac{\sqrt{\frac{m_1 z^\star}{m_2}}}{m_1}+\frac{z^\star}{m_2\sqrt{\frac{m_1 z^\star}{m_2}}}\right)\\ &= \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,. \qquad (3.13)\end{aligned}$$

Due to (3.11) and (3.13), the arithemtic-mean-component competitive ratio results in

$$\mathcal{R}^{f_2}(\text{RPP-PROD}) = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,. \qquad (3.14)$$

This result holds for all efficient solutions since, in the first case, there is exactly one efficient solution for the adversary and, in the second case, we considered the maximum over all $x \in \mathcal{I}_2$. Consequently, we have $\mathcal{R}_s^{f_2}(\text{RPP-PROD}) = \mathcal{R}^{f_2}(\text{RPP-PROD})$. □

Note that the arithmetic-mean-component competitive ratio of RPP-PROD is closely related to the competitive ratio of the corresponding single-objective algorithm RPP which is given by $\sqrt{M/m}$. In the following, we prove that RPP-PROD achieves the best possible arithmetic-mean-component competitive ratio.

Theorem 3.3.7. *No deterministic algorithm for the bi-objective time series search problem can achieve a competitive ratio with respect to $f_2(c) = \frac{1}{n}\sum_{i=1}^{n} c_i$ smaller than $\sqrt[4]{M_1 M_2/m_1 m_2}$.*

Proof. The adversary offers a request r with

$$r = \left(\tilde{x},\ \tfrac{z^\star}{\tilde{x}}\right)^\mathsf{T},\ \text{where } \tilde{x} = \sqrt{\frac{M_1 z^\star}{M_2}}\,.$$

If the online player accepts this request, another request $\left(M_1,\ M_2\right)^\mathsf{T}$ is revealed and the competitive ratio is given by

$$\mathcal{R}^{f_2}(\textsc{rpp-prod}) = \frac{1}{2}\left(\frac{M_1}{\sqrt{\frac{M_1 z^\star}{M_2}}} + \frac{M_2\sqrt{\frac{M_1 z^\star}{M_2}}}{z^\star}\right) = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,. \tag{3.15}$$

Otherwise, the adversary only reveals r and the online player has to settle for the lower bounds m_1 and m_2. The competitive ratio is then given by

$$\mathcal{R}^{f_2}(\textsc{rpp-prod}) = \frac{1}{2}\left(\frac{\sqrt{\frac{m_1 z^\star}{m_2}}}{m_1} + \frac{z^\star}{\sqrt{\frac{m_1 z^\star}{m_2}}m_2}\right) = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,. \tag{3.16}$$

Due to (3.15) and (3.16), no deterministic algorithm for the bi-objective time series search problem can achieve a competitive ratio with respect to f_2 smaller than $\sqrt[4]{M_1 M_2/m_1 m_2}$. Obviously, the same lower bound holds for the strong competitive ratio. $\square$

The competitive analysis with respect to $f_3(c) = \sqrt[n]{\prod_{i=1}^{n} c_i}$, i.e., the geometric-mean-component competitive analysis, yields the same results: For f_3, (3.11) and (3.13) are given by

$$\max_{x\in\mathcal{I}_1}\left\{\sqrt{\frac{M_1}{x}\cdot\frac{M_2 x}{z^\star}}\right\} = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}} \quad\text{and}\quad \max_{x\in\mathcal{I}_2}\left\{\sqrt{\frac{x}{m_1}\cdot\frac{z^\star}{m_2 x}}\right\} = \sqrt[4]{\frac{M_1 M_2}{m_1 m_2}}\,.$$

For the lower bound, a request $r = \left(\tilde{x},\ \tfrac{z^\star}{\tilde{x}}\right)^\mathsf{T}$, where $\tilde{x}$ is an arbitrary feasible value, leads to the same result as given in Theorem 3.3.7.

Summarizing, the best possible deterministic algorithm for the multi-objective time series search problem with respect to a worst-component competitive analysis is given by RPP-HIGH, whereas the best possible deterministic algorithm for the bi-objective time series search problem with respect to a arithmetic/geometric-mean-component competitive analysis is given by RPP-PROD.

Finally, we give a short example for the bi-objective time series search problem in order to illustrate the behavior of the multi-objective online algorithms and the difference to the single-objective algorithm.

Example 3.3.1. *As in the example given at the beginning of this section, you would like to sell your antique car and aim for both a high price and a buyer you appreciate. You are not willing to sell your car for less than* \$ 5000 *and some market research yielded a* *peak price of* \$ 20000 *for your car, i.e.,* $m_1 = 5000$ *and* $M_1 = 20000$. *Furthermore, you rate the potential buyers on a scale from 1 to 5, 1 is your arch enemy and 5 is your best friend, i.e.,* $m_2 = 1$ *and* $M_2 = 5$.

In the single-objective case, you would have accepted any offer which features either a price of at least \$ $10000 = \sqrt{m_1 M_1}$ *or an appreciation of at least* $2.24 \approx \sqrt{m_2 M_2}$.

First of all, consider now the multi-objective case with respect to a worst-component competitive analysis. As shown in this section, the best course of action is to apply the single-objective strategy to the component with the highest fluctuation ratio. Since $20000/5000 \geq 5/2$, *the algorithm* RPP-HIGH *tells you to accept any price that is at least* \$ $10000 = \sqrt{m_1 M_1}$ *and not to care about your appreciation of the buyer at all.*

The situation looks different for the less conservative arithmetic/geometric-mean-component competitive analysis: Here, the best possible algorithm is given by RPP-PROD *that tells you to accept any request for which the product of price and appreciation is at least* $31622.776 \approx \sqrt{m_1 M_1 m_2 M_2}$. *So, for example, you would accept a request with a moderate price of* \$ 9000 *(which you would not have accepted in the previous case), if your appreciation of the buyer is at least* 3.51.

This concludes the analysis of deterministic multi-objective online algorithms for the multi-objective time series search problem. In the next section, a randomized algorithm for the multi-objective time series search problem is presented.

3.3.3 A Randomized Bi-Objective Online Algorithm

In this section, a randomized algorithm for the bi-objective time series search problem is presented and analyzed by worst-component competitive analysis. As for the single-objective time series search problem, the competitive ratio is reduced by randomization. Note that we consider the bi-objective time series search problem, i.e., $n = 2$. Additionally, we consider only the competitive ratio, rather than the *strong* competitive ratio. Competitive randomized algorithms for the multi-objective time series search problem with $n \geq 3$ are subject to further research, as well as strongly competitive randomized algorithms for $n \geq 2$.

We follow the same approach as presented for the single-objective case in (El-Yaniv et al., 2001) and assume that $M_i/m_i = 2^{k_i}$ for $k_i \in \mathbb{N}_+$, $i = 1, 2$. Without loss of generality, suppose that $k_1 \geq k_2$. For the fluctuation ratio φ of the multi-objective time series search problem we then have $\varphi = (M_1 M_2)/(m_1 m_2) = 2^k$, where $k = k_1 + k_2$. Consider the deterministic algorithms RPP-PROD$_l$ for $l = 0, \ldots, k-1$:

Algorithm 15: Multi-objective reservation price policy RPP-PROD$_l$.

```
1 for $t = 1, \ldots, T$ do
2     Accept $r_t = (p_t^1, p_t^2)^\intercal$ if $p_t^1 \cdot p_t^2 \geq m_1 m_2 2^l$.
```

The randomized algorithm EXPO-MULT chooses strategy RPP-PROD$_l$, $l = 0, \ldots, k-1$, with probability $1/k$ each before the first request is revealed.

Theorem 3.3.8. *For $k_1 \geq k_2$, the competitive ratio with respect to f_1 of* EXPO-MULT *is given by*

$$\mathcal{R}^{f_1}(\text{EXPO-MULT}) = \log(\varphi).$$

Proof. The idea for the worst case sequence presented to the online player by the adversary is analogous to the worst case sequence in the single-objective variant: The efficient offline solution for the adversary is supposed to be given by a request $r_{j+1} = \left(p_j^1, p_j^2\right)^\mathsf{T}$ with $p_t^1 \cdot p_t^2$ just below the threshold $m_1 m_2 2^{j+1}$ for some j. This request will not be accepted by the online player when a policy RPP-PROD$_l$ with $l \geq j+1$ is selected. In order to prevent the online player from accepting this request in case a policy RPP-PROD$_l$ with $l \leq j$ is selected, the adversary initially presents requests $r_i = \left(p_i^1, p_i^2\right)^\mathsf{T}$ that match the threshold for policy RPP-PROD$_l$, i.e., with $p_i^1 \cdot p_i^2 = m_1 m_2 2^i$ for $i = 0, \dots, j$. This way, the expected value of the request accepted by the online player is minimized. Choosing the index j which maximizes the competitive ratio then gives the worst case instance.

Note that, in order to maximize the competitive ratio (rather than the *strong* competitive ratio), the requests triggering the thresholds $m_1 m_2 2^i$ with $i \leq j$ have to be chosen such that the request r_{j+1} remains the *only* efficient solution. Otherwise, the algorithm's value would also be compared to the other efficient solutions leading to a smaller competitive ratio. See also Figures 3.5 and 3.6 for an illustration of the worst case instance.

Let $r_{j+1} = \left(p_{j+1}^1, p_{j+1}^2\right)^\mathsf{T}$ be an efficient offline solution and let j be an integer such that

$$m_1 m_2 2^j \leq p_{j+1}^1 \cdot p_{j+1}^2 < m_1 m_2 2^{j+1}.$$

For the maximal choice of $j = k$, we have $p_{j+1}^i = M_i$, $i = 1, \dots, n$, due to bounded prices and $m_1 2^{k_1} \cdot m_2 2^{k_2} = m_1 m_2 2^k = M_1 M_2$. This choice is not advantageous for the adversary, since the online player obtains this price with probability $1/k$ and any other choice for j gives the adversary the possibility to prevent the online player from obtaining the best price.

So, for any particular choice of $j < k$, the adversary selects $p_{j+1}^1 \cdot p_{j+1}^2$ arbitrarily close to $m_1 m_2 2^{j+1}$, since, thereby, the adversary's return is increased whereas the return of the online player remains unchanged. Therefore,

$$p_{j+1}^1 \cdot p_{j+1}^2 = m_1 m_2 2^{j+1} - \epsilon,\ \epsilon > 0. \tag{3.17}$$

Ignoring the ϵ, efficient offline solutions are given by

$$r_{j+1} = \left(p_{j+1}^1, p_{j+1}^2\right)^\mathsf{T} = \left(x_{j+1}, \tfrac{m_1 m_2 2^{j+1}}{x_{j+1}}\right)^\mathsf{T} \text{ for } x_{j+1} \text{ feasible.} \tag{3.18}$$

The feasible values for x_{j+1} are restricted by $x_{j+1} \in [m_1, M_1]$ and

$$\frac{m_1 m_2 2^{j+1}}{x_{j+1}} \in [m_2, M_2] \ \Leftrightarrow\ \frac{m_1 m_2 2^{j+1}}{M_2} \leq x_{j+1} \leq m_1 2^{j+1}.$$

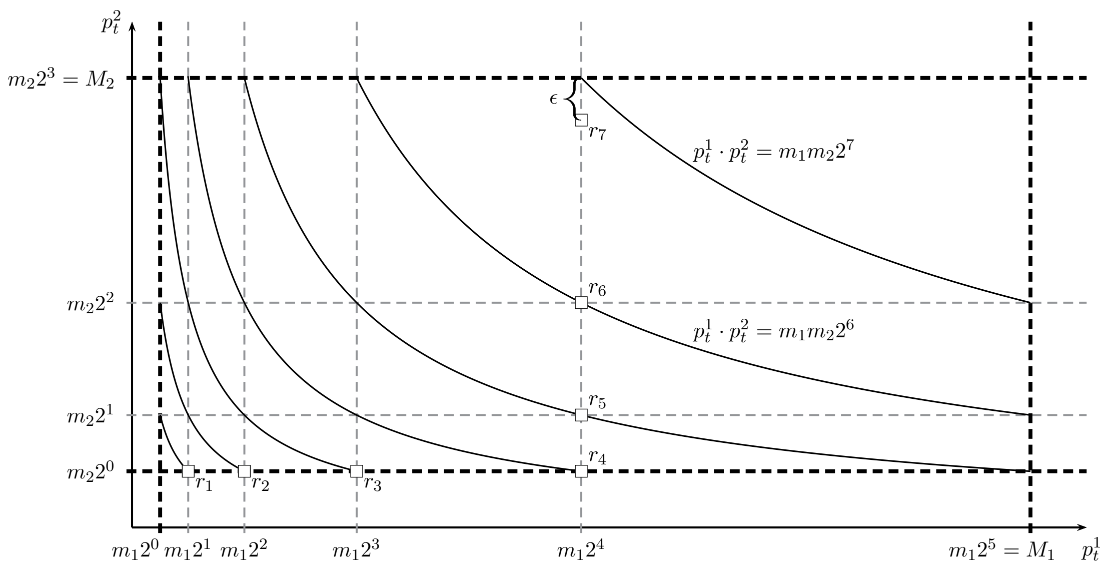


Figure 3.5: Example for a worst case sequence in Case 1.1: $k_1 = 5$, $k_2 = 3$, $j = 6$, and $x_7 = m_1 2^4$, the squares mark the requests.

Since $M_2 = m_2 2^{k_2}$, we have $x_{j+1} \in \mathcal{I}_{j+1}$ with

$$\mathcal{I}_{j+1} := \left[\max\left\{m_1, m_1 2^{j+1-k_2}\right\}, \min\left\{M_1, m_1 2^{j+1}\right\}\right]$$

For the following analysis, we consider three cases with respect to the relation of j, k_1, and k_2:

Case 1: $j + 1 \geq k_1$

By assumption, we have $k_1 \geq k_2$, and, due to $j+1 \geq k_1$, we have $j+1 \geq k_2$. Therefore, $m_1 2^{j+1-k_2} \geq m_1$ and $M_1 = m_1 2^{k_1} \leq m_1 2^{j+1}$. Thus, the interval $\mathcal{I}_{j+1}$ is given by

$$\mathcal{I}_{j+1} = \left[m_1 2^{j+1-k_2}, M_1\right].$$

Due to (3.18), the competitive ratio with respect to $f_1(c) = \max_{i=1,\dots,n} c_i$ is maximized by choosing the minimal or maximal feasible value for x_{j+1}, i.e., $x_{j+1} = m_1 2^{j+1-k_2}$ or $x_{j+1} = M_1$:

Case 1.1: $x_{j+1} = m_1 2^{j+1-k_2}$

Consider the minimal feasible value for x_{j+1}, i.e., $x_{j+1} = m_1 2^{j+1-k_2}$ (see r_7 in Figure 3.5). Then, r_{j+1} is given by $r_{j+1} = \left(m_1 2^{j+1-k_2},\, m_2 2^{k_2}\right)^\intercal$. Request r_{j+1} will not be accepted by the online player when a policy RPP-PROD$_l$ with $l \geq j+1$ is selected. Note that for $l = j+1$, the request r_{j+1} is not accepted by the online player since $p_{j+1}^1 \cdot p_{j+1}^2 = m_1 m_2 2^{j+1} - \epsilon$ (see (3.17), we ignored the ϵ to improve the readability).

In order to prevent the online player from accepting this request r_{j+1} in case a policy RPP-PROD$_l$ with $0 \le l \le j$ is selected, the adversary initially presents requests $r_i = \left(p_i^1,\, p_i^2\right)^\mathsf{T}$ with $p_i^1 \cdot p_i^2 = m_1 m_2 2^i$ for $i = 0, \ldots, j$ (see also Figure 3.5). Thus, request r_i is given by

$$r_i = \left(p_i^1,\, p_i^2\right)^\mathsf{T} = \left(x_i,\, \tfrac{m_1 m_2 2^i}{x_i}\right)^\mathsf{T}, \quad x_i \in \mathcal{I}_i.$$

The interval $\mathcal{I}_i$ of feasible values for x_i is in this case determined by the following conditions: In order to obtain a feasible request, we need to guarantee $x_i \in [m_1, M_1]$ and

$$\frac{m_1 m_2 2^i}{x_i} \in [m_2, M_2] \;\Leftrightarrow\; m_1 2^{i-k_2} \le x_i \le m_1 2^i.$$

Furthermore, we have to ensure that $r_i \leqq r_{j+1}$, i.e., that r_i does not represent another efficient solution. Otherwise, the algorithm's value would also be compared to this efficient solution leading to a smaller competitive ratio (see Definition 3.2.1). Hence, we have $x_i \le m_1 2^{j+1-k_2}$ and

$$\frac{m_1 m_2 2^i}{x_i} \le m_2 2^{k_2} \;\Leftrightarrow\; x_i \ge m_1 2^{i-k_2}.$$

Since $j + 1 \le k = k_1 + k_2$ and, hence, $M_1 = m_1 2^{k_1} \ge m_1 2^{j+1-k_2}$, the interval of feasible values $\mathcal{I}_i$ is given by

$$\mathcal{I}_i = \left[\max\left\{m_1, m_1 2^{i-k_2}\right\},\, \min\left\{m_1 2^i, m_1 2^{j+1-k_2}\right\}\right]. \tag{3.19}$$

x_{j+1} is chosen as the *minimal* feasible value in order to maximize the second component p_{j+1}^2 for the adversary. Thus, we are looking for the *maximal* feasible value for x_i in order to minimize the second component p_i^2 for the online player and, hence, maximize the competitive ratio with respect to f_1.

Due to (3.19), we have $x_i \le m_1 2^i$ for $i = 0, \ldots, j + 1 - k_2$ and $x_i \le m_1 2^{j+1-k_2}$ for $i = j + 2 - k_2, \ldots, j$. Thus, r_i is chosen as

$$r_i = \left(m_1 2^i,\; m_2\right)^\mathsf{T} \text{ for } i = 0, \ldots, j + 1 - k_2, \text{ and}$$
$$r_i = \left(m_1 2^{j+1-k_2},\; m_2 2^{i-j-1+k_2}\right)^\mathsf{T} \text{ for } i = j + 2 - k_2, \ldots, j.$$

For the expected value of the i-th component of ALG with respect to its randomized decisions, denoted by $\mathbb{E}\left[\text{ALG}_i\right]$, we have

$$\begin{aligned}
\mathbb{E}\left[\text{ALG}_1\right] &= \frac{1}{k}\sum_{i=0}^{j+1-k_2} m_1 2^i + \frac{1}{k}\sum_{i=j+2-k_2}^{j} m_1 2^{j+1-k_2} + \frac{k-(j+1)}{k} m_1 \\
&= \frac{m_1}{k}\Big(2^{j-k_2+2} - 1 + (k_2-1)\,2^{j-k_2+1} + k - j - 1\Big) \\
&= \frac{m_1}{k}\Big((k_2+1)\,2^{j+1-k_2} + k - j - 2\Big),
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}\left[\text{ALG}_2\right] &= \frac{1}{k}\sum_{i=0}^{j+1-k_2} m_2 + \sum_{i=j+2-k_2}^{j} m_2 2^{i-j-1+k_2} + \frac{k-(j+1)}{k} m_2 \\
&= \frac{m_2}{k}\Big(j + 2 - k_2 + 2^{k_2} - 2 + k - j - 1\Big) \\
&= \frac{m_2}{k}\Big(2^{k_2} + k_1 - 1\Big).
\end{aligned}$$

The competitive ratio with respect to f_1 is then given by

$$\begin{aligned}
&\max_{k\geq j+1\geq k_1} \max\left\{\frac{m_1 2^{j+1-k_2}}{\mathbb{E}\left[\text{ALG}_1\right]}, \frac{m_2 2^{k_2}}{\mathbb{E}\left[\text{ALG}_2\right]}\right\} \\
= &\max_{k\geq j+1\geq k_1} \max\left\{\frac{k 2^{j+1-k_2}}{(k_2+1)\,2^{j+1-k_2} + k - j - 2}, \frac{k 2^{k_2}}{2^{k_2} + k_1 - 1}\right\} \\
\overset{(j+1=k)}{=} &\max\left\{\frac{\log(\varphi)}{k_2 + 1 - \frac{1}{2^{k_1}}}, \frac{\log(\varphi)}{1 + \frac{k_1 - 1}{2^{k_2}}}\right\}. \qquad (3.20)
\end{aligned}$$

Note that the continuous maximum with respect to j is obtained for $j = k - 2 + \frac{1}{\log(2)}$. However, for integer $j \leq k-1$, the maximum is given by $j = k-1$.

Case 1.2: $x_{j+1} = M_1$

Now, consider the maximal value for x_{j+1}, i.e., $x_{j+1} = M_1$ (see r_7 in Figure 3.6). Then, r_{j+1} is given by $r_{j+1} = \left(M_1,\, m_2 2^{j+1-k_1}\right)^\mathsf{T}$. By the same argumentation as in the previous case, the adversary previously presents requests

$$r_i = \left(p_i^1,\, p_i^2\right)^\mathsf{T} = \left(x_i,\, \tfrac{m_1 m_2 2^i}{x_i}\right)^\mathsf{T}, \quad x_i \in \mathcal{I}_i,$$

where $\mathcal{I}_i$ is again the set of feasible values for x_i (see also Figure 3.6). In order to obtain a feasible request, we need to guarantee $x_i \in [m_1, M_1]$ and

$$\frac{m_1 m_2 2^i}{x_i} \in [m_2, M_2] \;\Leftrightarrow\; m_1 2^{i-k_2} \leq x_i \leq m_1 2^i.$$

As above, we also have to ensure that $r_i \leqq r_{j+1}$, i.e., $x_i \leq M_1$ and

$$\frac{m_1 m_2 2^i}{x_i} \leq m_2 2^{j+1-k_1} \;\Leftrightarrow\; x_i \geq m_1 2^{i-j-1+k_1}.$$

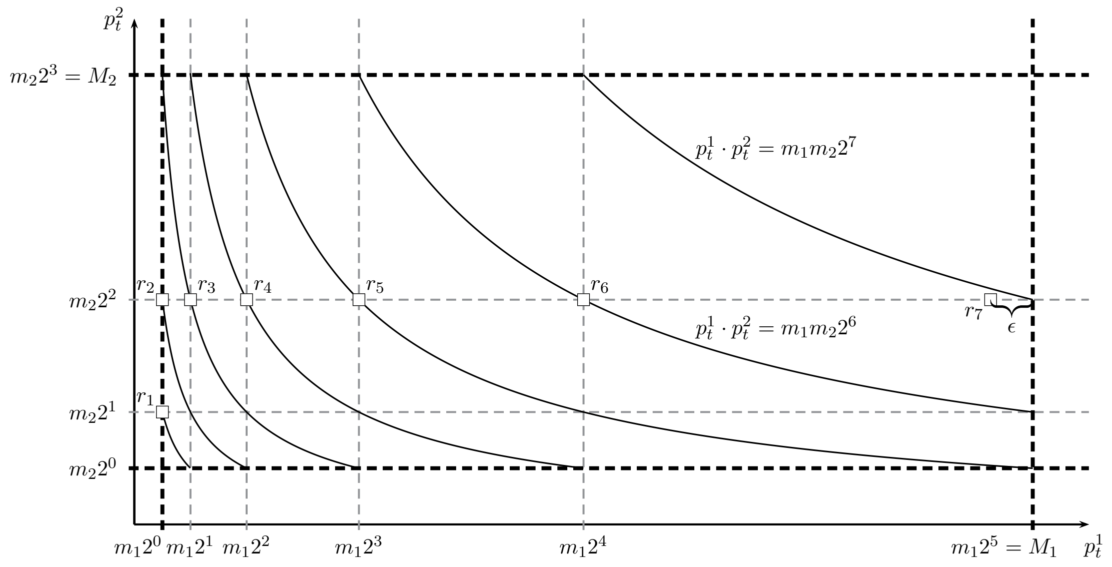


Figure 3.6: Example for a worst case sequence in Case 1.2: $k_1 = 5$, $k_2 = 3$, $j = 6$, and $x_7 = m_1 2^5$, the squares mark the requests.

Since $j + 1 \leq k = k_1 + k_2$ and, hence, $m_1 2^{i-k_2} \leq m_1 2^{i-(j+1-k_1)}$, the interval of feasible values $\mathcal{I}_i$ is given by

$$\mathcal{I}_i = \left[\max\left\{m_1, m_1 2^{i-j-1+k_1}\right\}, \min\left\{M_1, m_1 2^i\right\}\right]. \tag{3.21}$$

Since x_{j+1} is in this case chosen as the *maximal* feasible value in order to maximize the first component p_{j+1}^1 for the adversary, we are looking for the *minimal* feasible value for x_i in order to minimize the first component p_i^1 for the online player and, hence, maximize the competitive ratio with respect to f_1. Thus, r_i is given by

$$\begin{aligned}
r_i &= \left(m_1,\ m_2 2^i\right)^\intercal \text{ for } i = 0, \dots, j+1-k_1, \text{ and} \\
r_i &= \left(m_1 2^{i-j-1+k_1},\ m_2 2^{j+1-k_1}\right)^\intercal \text{ for } i = j+2-k_1, \dots, j.
\end{aligned}$$

For the expected value of ALG we then have

$$\begin{aligned}
\mathbb{E}\left[\text{ALG}_1\right] &= \frac{1}{k} \sum_{i=0}^{j+1-k_1} m_1 + \frac{1}{k} \sum_{i=j+2-k_1}^{j} m_1 2^{i-j-1+k_1} + \frac{k-(j+1)}{k} m_1 \\
&= \frac{m_1}{k} \left(j + 2 - k_1 + 2^{k_1} - 2 + k - j - 1\right) \\
&= \frac{m_1}{k} \left(2^{k_1} + k_2 - 1\right),
\end{aligned}$$

$$\mathbb{E}\left[\text{ALG}_2\right] = \frac{1}{k} \sum_{i=0}^{j+1-k_1} m_2 2^i + \frac{1}{k} \sum_{i=j+2-k_1}^{j} m_2 2^{j+1-k_1} + \frac{k-(j+1)}{k} m_2$$

$$
\begin{aligned}
&= \frac{m_2}{k}\left(2^{j-k_1+2} - 1 + (k_1-1)\,2^{j+1-k_1} + k - j - 1\right)\\
&= \frac{m_2}{k}\left((k_1+1)2^{j+1-k_1} + k - j - 2\right).
\end{aligned}
$$

The competitive ratio with respect to f_1 is then given by

$$
\begin{aligned}
&\max_{k\geq j+1\geq k_1} \max\left\{\frac{M_1}{\mathbb{E}\left[\text{ALG}_1\right]}, \frac{m_2 2^{j+1-k_1}}{\mathbb{E}\left[\text{ALG}_2\right]}\right\}\\
= &\max_{k\geq j+1\geq k_1} \max\left\{\frac{k2^{k_1}}{2^{k_1}+k_2-1}, \frac{k2^{j+1-k_1}}{(k_1+1)2^{j+1-k_1}+k-j-2}\right\}\\
\overset{(j+1=k)}{=} &\max\left\{\frac{\log(\varphi)}{1+\frac{k_2-1}{2^{k_1}}}, \frac{\log(\varphi)}{k_1+1-\frac{1}{2^{k_2}}}\right\}. \qquad (3.22)
\end{aligned}
$$

Note that the competitive ratio is symmetrical to the competitive ratio in Case 1.1 with respect to k_1 and k_2.

Case 2: $j+1 \leq k_2$
By assumption, we have $k_1 \geq k_2$, and, due to $j+1 \leq k_2$, we have $j+1 \leq k_1$. Therefore, $m_1 \geq m_1 2^{j+1-k_2}$ and $m_1 2^{j+1} \leq m_1 2^{k_1} = M_1$. Thus, the interval $\mathcal{I}_{j+1}$ is given by

$$
\mathcal{I}_{j+1} = \left[m_1, m_1 2^{j+1}\right].
$$

Due to (3.18), the competitive ratio with respect to f_1 is maximized by choosing the minimal or maximal feasible value for x_{j+1}, i.e., $x_{j+1} = m_1$ or $x_{j+1} = m_1 2^{j+1}$. We proceed with the analysis as in Case 1:

Case 2.1: $x_{j+1} = m_1$
Consider the minimal feasible value for x_{j+1}, i.e., $x_{j+1} = m_1$. Then, r_{j+1} is given by $r_{j+1} = \left(m_1,\ m_2 2^{j+1}\right)^\intercal$. By the same argumentation as in the previous cases, the adversary previously presents requests

$$
r_i = \left(p_i^1,\, p_i^2\right)^\intercal = \left(x_i,\ \tfrac{m_1 m_2 2^i}{x_i}\right)^\intercal, \quad x_i \in \mathcal{I}_i.
$$

The interval $\mathcal{I}_i$ is determined by $x_i \in [m_1, M_1]$, $m_1 2^{i-k_2} \leq x_i \leq m_1 2^i$, and $r_i \leqq r_{j+1}$, i.e., $m_1 2^{i-j-1} \leq x_i \leq m_1$ (for a detailed explanation see the previous cases). Consequently, $\mathcal{I}_i$ is given by $\mathcal{I}_i = [m_1,\ m_1]$, i.e., $x_i = m_1$ and we have

$$
r_i = \left(m_1,\ m_2 2^i\right)^\intercal \text{ for } i = 0, \ldots, j.
$$

For the expected value of ALG we then have

$$\mathbb{E}\left[\text{ALG}_1\right] = \frac{1}{k}\sum_{i=0}^{j} m_1 + \frac{k-(j+1)}{k} m_1 = m_1,$$

$$\begin{aligned}\mathbb{E}\left[\text{ALG}_2\right] &= \frac{1}{k}\sum_{i=0}^{j} m_2 2^i + \frac{k-(j+1)}{k} m_2 \\ &= \frac{m_2}{k}\left(2^{j+1} - 1 + k - j - 1\right) \\ &= \frac{m_2}{k}\left(2^{j+1} + k - j - 2\right).\end{aligned}$$

The competitive ratio with respect to f_1 is then given by

$$\begin{aligned}&\max_{j+1\le k_2} \max\left\{\frac{m_1}{\mathbb{E}\left[\text{ALG}_1\right]}, \frac{m_2 2^{j+1}}{\mathbb{E}\left[\text{ALG}_2\right]}\right\} \\ = &\max_{j+1\le k_2} \max\left\{1, \frac{k2^{j+1}}{2^{j+1}+k-j-2}\right\} \\ \overset{(j+1=k_2)}{=} &\frac{\log(\varphi)}{1+\frac{k_1-1}{2^{k_2}}}.\end{aligned} \tag{3.23}$$

Case 2.2: $x_{j+1} = m_1 2^{j+1}$

Consider the maximal value for x_{j+1}, i.e., $x_{j+1} = m_1 2^{j+1}$. Then, r_{j+1} is given by $r_{j+1} = \left(m_1 2^{j+1}, m_2\right)^\intercal$. As seen in Case 1.1 and Case 1.2, this case is symmetrical with respect to k_1 and k_2 to Case 2.1. Consequently, we have for the expected value of ALG

$$\mathbb{E}\left[\text{ALG}_1\right] = \frac{m_1}{k}\left(2^{j+1} + k - j - 2\right),$$

$$\mathbb{E}\left[\text{ALG}_2\right] = m_2,$$

and the competitive ratio with respect to f_1 is given by

$$\max_{j+1\le k_2} \max\left\{\frac{m_1 2^{j+1}}{\mathbb{E}\left[\text{ALG}_1\right]}, \frac{m_2}{\mathbb{E}\left[\text{ALG}_2\right]}\right\} = \frac{\log(\varphi)}{1+\frac{k_2-1}{2^{k_1}}}. \tag{3.24}$$

Case 3: $k_1 \ge j+1 \ge k_2$

Since, in this case, $m_1 2^{j+1-k_2} \ge m_1$ and $m_1 2^{j+1} \le m_1 2^{k_1} = M_1$, the interval $\mathcal{I}_{j+1}$ is given by

$$\mathcal{I}_{j+1} = \left[m_1 2^{j+1-k_2}, m_1 2^{j+1}\right].$$

Due to (3.18), the competitive ratio with respect to f_1 is maximized by choosing the minimal or maximal feasible value for x_{j+1}, i.e., $x_{j+1} = m_1 2^{j+1-k_2}$ or $x_{j+1} = m_1 2^{j+1}$. These cases have been analyzed before, see Case 1.1 and Case 2.2.

Now, we bring together the competitive ratios of the different cases analyzed above. By (3.20), (3.22), (3.23), and (3.24), the competitive ratio with respect to f_1 of EXPO-MULT is given by

$$\mathcal{R}^{f_1}(\text{EXPO-MULT}) = \max\left\{ \frac{\log(\varphi)}{k_2+1-\frac{1}{2^{k_1}}}, \frac{\log(\varphi)}{1+\frac{k_1-1}{2^{k_2}}}, \frac{\log(\varphi)}{1+\frac{k_2-1}{2^{k_1}}}, \frac{\log(\varphi)}{k_1+1-\frac{1}{2^{k_2}}} \right\}.$$

Then, we have

$$1+\frac{k_2-1}{2^{k_1}} \leq k_2+1-\frac{1}{2^{k_1}} \quad \text{and} \quad 1+\frac{k_1-1}{2^{k_2}} \leq k_1+1-\frac{1}{2^{k_2}},$$

and, due to $k_1 \geq k_2$,

$$\frac{k_2-1}{2^{k_1}} \leq \frac{k_1-1}{2^{k_2}}.$$

Therefore, we have

$$\mathcal{R}^{f_1}(\text{EXPO-MULT}) = \frac{\log(\varphi)}{1+\frac{k_2-1}{2^{k_1}}} \leq \log(\varphi).$$

□

3.4 The Bi-Objective Ski Rental Problem

In the context of online optimization and competitive analysis, the ski rental problem (originally suggested by Larry Rudolph, cf. (Fiat et al., 1998, p. 374) and (Karlin et al., 1988)) is often used as an analogy to introduce the concept of online problems. Imagine you are about to go skiing for the first time in your life and you are faced with the question of whether to buy skis or to rent them. If you knew how often you would go skiing in the future, the optimal decision could be calculated based on the rental costs and the buying costs:

Assume, renting skis costs \$ 2 and buying skis costs \$ 50. If you knew that you would go skiing at least 25 times in the future, buying skis at the very beginning would be the right decision. Otherwise, if you would go skiing less then 25 times in the future, renting skis every time you go skiing would be the optimal decision with respect to your costs.

Obviously, the online manner of the problem lies in the number of days you will go skiing in the future. Suppose that renting skis costs \$ 1 and buying skis costs \$ B and consider the online algorithm which rents $B-1$ times and then buys skis. In the worst case, you go skiing B times, occasioning costs of \$ $2B-1$, wheres the optimal solution would have been to buy skis for \$ B in the very beginning. Consequently, this algorithm is $2-1/B$-competitive and it can be proven that this is the best possible competitive ratio (cf. (Karlin et al., 1988)).

Now, what if you are not only interested in your expenditures but also in your personal comfort? Maybe you prefer having your own skis in your garage instead of renting skis every time and having to deal with the renting effort and probably slightly varying skis. Or, maybe you do not want to make room in your garage in order to store all the skiing gear and you prefer renting skis. Thus, your decision of renting or buying skis depends not only on financial costs but also on your personal comfort. This motivates the introduction of the *bi-objective ski rental problem.*

As for the classical ski rental problem, renting skis costs 1 and buying skis costs $B > 1$; if $B \leq 1$, the problem would be trivial. The (unknown) number of skiing days is denoted by n. Furthermore, we now introduce *inconvenience.* Analogously to the financial costs, buying skis induces a onetime inconvenience of $C > 0$ and renting skis induces normalized inconvenience of 1 for each rental transaction. Depending on the preferences of the online player, C could be smaller or greater than 1. The goal is to minimize financial costs and inconvenience simultaneously. Note that the units of financial costs and inconvenience are not comparable and, thus, a bi-objective approach to the problem is necessary, instead of simply optimizing the sum of both costs.

In the following, we present competitive algorithms and lower bounds for the bi-objective ski rental problem. First, we consider the worst-component competitive ratio, i.e., the competitive ratio with respect to $f_1 = \max_{i=1,\dots,n} c_i$. Without loss of generality, assume that $B \geq C$. For $B \leq C$ the same analysis holds with B and C exchanged. Consider Algorithm 16 for the bi-objective ski rental problem, also denoted by SKI.

Algorithm 16: Bi-objective ski rental algorithm SKI for $B \geq C$.

```
1 Rent skis $r$ times, then buy skis, where
      $$r = \begin{cases} \lfloor r_1^\star \rfloor & \text{if } \frac{\lfloor r_1^\star \rfloor + B}{\lfloor r_1^\star \rfloor + 1} \leq \frac{\lceil r_1^\star \rceil + C}{C}, \\ \lceil r_1^\star \rceil & \text{otherwise,} \end{cases}$$
  and
      $$r_1^\star = \sqrt{\frac{1}{4} + C(B-1)} - \frac{1}{2}.$$
```

Before we analyze the competitive ratio of SKI, we give a numerical example in order to illustrate the algorithm:

Example 3.4.1. *Assume renting regular skis costs \$ 25 a day and buying skis costs \$ 350. If your only objective is the minimization of your financial expenses and you act according to the optimal online algorithm for the ski rental problem, you would rent skis 13 times and buy them when you go skiing for the* 14*-th time (for normalized renting costs of \$ 1 the buying costs amount to \$ 14). In the worst case, you never go skiing after the* 14*-th time again and pay* $13 \cdot \$25 + \$350 = \$675$*. The optimal offline solution would have been to buy skis in the very beginning and pay only \$ 350. The competitive ratio is then given by* $\$\,675/\$\,350 = 2 - 1/14 \approx 1.928$*.*

Now you would like to apply the bi-objective ski rental algorithm SKI*. Assume your inconvenience for buying skis is given by $C = 0.5$ and the inconvenience for renting skis is given by 1, i.e., you prefer owning skis over renting skis every time. Then,*

$$r^\star = \sqrt{\frac{1}{4} + 0.5\,(14-1)} - \frac{1}{2} \approx 2.098$$

Since $(2+14)/(2+1) < (3+0.5)/(0.5)$*, you rent skis two times and buy skis on the third skiing day. In the worst case, you never go skiing again after the third time and pay* $2 \cdot \$\ 25 + \$\ 350 = \$\ 400$*. Additionally, your inconvenience amounts to* $2 + 0.5 = 2.5$*. There are two efficient offline solutions, namely renting skis three times leading to a cost vector of* $\big(75,\ 3\big)^\intercal$*, and buying skis in the very beginning leading to a cost vector of* $\big(350,\ 0.5\big)^\intercal$*. The worst-component competitive ratio is thus given by*

$$\max\left\{\frac{400}{75}, \frac{2.5}{3}, \frac{400}{350}, \frac{2.5}{0.5}\right\} = 5.\overline{3},$$

which is the optimal worst-component competitive ratio as shown by Theorem 3.4.1. Due to the low value of C, you buy skis much earlier than in the single-objective case.

For a higher value of C, for example $C = 20$, the situation is different: In this case, your inconvenience of buying skis equals the inconvenience ascribed to renting skis twenty times, i.e., compared to the case above, you do not mind renting skis. Once again, you apply SKI*. Note that we have to exchange B and C since now $C \geq B$. Thus*

$$r^\star = \sqrt{\frac{1}{4} + 14\,(20-1)} - \frac{1}{2} \approx 15.817,$$

and, since $(15+20)/(15+1) \geq (16+14)/(14)$*, you rent skis* 16 *times and buy skis on the* 17*-th skiing day. In the worst case, you never go skiing after the* 17*-th time and pay* $16 \cdot \$\ 25 + \$\ 350 = \$750$*. Additionally, your inconvenience amounts to* $16 + 0.5 = 16.5$*. As above, there are two efficient offline solutions, namely renting skis* 17 *times leading to a cost vector of* $\big(425,\ 17\big)^\intercal$*, and buying skis in the very beginning leading to a cost vector of* $\big(350,\ 20\big)^\intercal$*. The worst-component competitive ratio is thus given by*

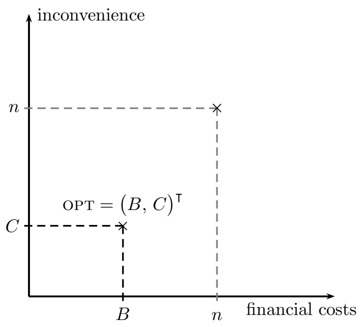
$$\max\left\{\frac{750}{425}, \frac{16.5}{17}, \frac{750}{350}, \frac{16.5}{20}\right\} \approx 2.142.$$

Due to the high value of C, you buy skis later than in the single-objective case.

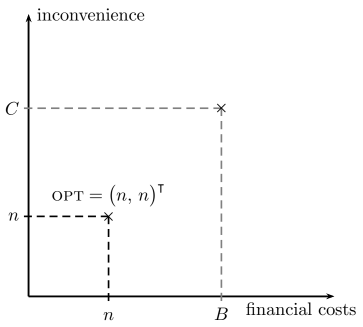
Theorem 3.4.1. *The strong competitive ratio with respect to $f_1 = \max_{i=1,\dots,n} c_i$ of* SKI *is given by*

$$\mathcal{R}_s^{f_1}(\textsc{ski}) = \min\left\{\frac{\lfloor r_1^\star\rfloor + B}{\lfloor r_1^\star\rfloor + 1}, \frac{\lceil r_1^\star\rceil + C}{C}\right\}.$$

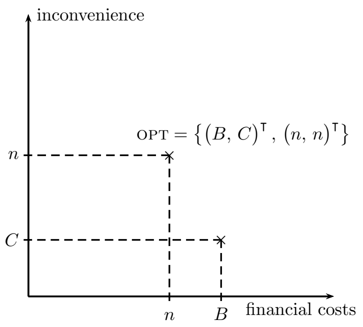
Proof. The proof is a basic case distinction with respect to the relations of B, C, and the number of skiing days n. The analysis of each case is then similar to the analysis of the well-known (single-objective) ski rental problem.



(a) $n \geq B$ and $n \geq C$.



(b) $n < B$ and $n < C$.



(c) $n < B$ and $n \geq C$.

Figure 3.7: Relations of B, C, and n.

Case 1: $n \geq B$

If the number of skiing days n is greater than or equal to B, i.e., $n \geq B$, it follows that $n \geq C$ since by assumption $B \geq C$. Consequently, the only efficient offline solution is given by buying skis at the very beginning leading to financial costs of B and inconvenience of C, i.e., OPT $= \big(B,\, C\big)^{\mathsf{T}}$, see also Figure 3.7a.

If the number of skiing days n is greater than r, i.e., $n > r$, SKI rents r times and then buys skis. Consequently, SKI's costs result in ALG $= \big(r + B,\, r + C\big)^{\mathsf{T}}$. Since there is exactly one efficient solution in this case, the strong competitive ratio with respect to $f_1 = \max_{i=1,\ldots,n} c_i$ is given by

$$\max\left\{\frac{r+B}{B},\, \frac{r+C}{C}\right\} \overset{(B \geq C)}{=} \frac{r+C}{C}. \tag{3.25}$$

Otherwise, if the number of skiing days n is smaller than or equal to r, i.e., $n \leq r$, SKI rents n times. Consequently, SKI's costs result in ALG $= \big(n,\, n\big)^{\mathsf{T}}$ and the strong competitive ratio with respect to f_1 is given by

$$\max\left\{\frac{n}{B},\, \frac{n}{C}\right\} \overset{(n \leq r)}{\leq} \max\left\{\frac{r}{B},\, \frac{r}{C}\right\} \overset{(B \geq C)}{=} \frac{r}{C}.$$

Since $C > 0$, the strong competitive ratio is in this case always smaller than in (3.25) and, hence, is neglected in the following.

Case 2: $n < B$ **and** $n < C$

In this case, the number of skiing days n is smaller than both B and C. Consequently, there is again exactly one efficient solution for the adversary given by OPT $= \big(n,\, n\big)^{\mathsf{T}}$, see also Figure 3.7b.

If the number of skiing days is smaller than or equal to r, i.e., $n \leq r$, the solution of ALG would be the same as OPT's solution, leading to a competitive ratio of 1. Therefore, assume $n > r$. The competitive ratio with respect to f_1 is then given by

$$\max\left\{\frac{r+B}{n},\, \frac{r+C}{n}\right\} \overset{(n>r)}{\leq} \max\left\{\frac{r+B}{r+1},\, \frac{r+C}{r+1}\right\} \overset{(B \geq C)}{=} \frac{r+B}{r+1} \tag{3.26}$$

Case 3: $n < B$ and $n \geq C$

Again, the number of skiing days n is smaller than B, but now $n \geq C$. Thus, both buying at the very beginning and renting for the whole time represent efficient solutions for the adversary, i.e., the set of efficient offline solutions is given by

$$\textsc{opt} = \left\{ \begin{pmatrix} B \\ C \end{pmatrix}, \begin{pmatrix} n \\ n \end{pmatrix} \right\},$$

see also Figure 3.7c.

If $n \leq r$, the strong competitive ratio with respect to f_1 is now given by

$$\max\left\{\max\left\{\frac{n}{B}, \frac{n}{C}\right\}, \max\left\{\frac{n}{n}, \frac{n}{n}\right\}\right\} \overset{(n\leq r)}{\leq} \max\left\{\frac{r}{B}, \frac{r}{C}\right\} \overset{(B\geq C)}{=} \frac{r}{C}, \tag{3.27}$$

which can be neglected due to (3.25).

Otherwise, if $n > r$, SKI rents r times and then buys skis. The strong competitive ratio with respect to f_1 is given by

$$\begin{aligned} &\max\left\{\max\left\{\frac{r+B}{B}, \frac{r+C}{C}\right\}, \max\left\{\frac{r+B}{n}, \frac{r+C}{n}\right\}\right\} \\ \overset{(B\geq C)}{=}\ &\max\left\{\frac{r+C}{C}, \frac{r+B}{n}\right\} \\ \overset{(n>r)}{\leq}\ &\max\left\{\frac{r+C}{C}, \frac{r+B}{r+1}\right\}. \end{aligned} \tag{3.28}$$

Due to (3.25), (3.26), and (3.28), the overall strong competitive ratio with respect to f_1 of SKI is given by

$$\max\left\{\frac{r+C}{C}, \frac{r+B}{r+1}\right\}. \tag{3.29}$$

In order to obtain the best possible strong competitive ratio with respect to f_1, we calculate the solution of

$$\frac{r_1^\star + C}{C} = \frac{r_1^\star + B}{r_1^\star + 1},\ r^\star > 0 \quad \Leftrightarrow \quad r_1^\star = \sqrt{\frac{1}{4} + C(B-1)} - \frac{1}{2}.$$

Note that $r_1^\star$ is, in general, not integer. Thus, we set the number of rental transactions to $\lfloor r_1^\star \rfloor$ or $\lceil r_1^\star \rceil$, depending on (3.29). Since $\frac{r_1^\star+C}{C}$ is monotonically increasing in $r_1^\star$ and $\frac{r_1^\star+B}{r_1^\star+1}$ is monotonically decreasing in $r_1^\star$, we establish the number of rental transactions r_1 as $\lfloor r_1^\star \rfloor$ if

$$\frac{\lfloor r_1^\star \rfloor + B}{\lfloor r_1^\star \rfloor + 1} \leq \frac{\lceil r_1^\star \rceil + C}{C},$$

and $\lceil r_1^\star \rceil$ otherwise (see also Figure 3.8). Consequently, the strong competitive ratio with

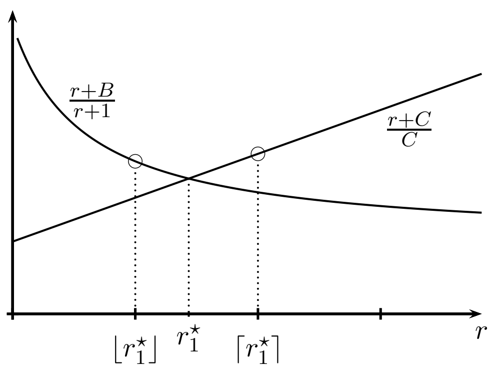


Figure 3.8: Number of rental transactions r.

respect to f_1 of SKI is given by

$$\mathcal{R}_s^{f_1}(\text{SKI}) = \min\left\{\frac{\lfloor r_1^\star\rfloor + B}{\lfloor r_1^\star\rfloor + 1}, \frac{\lceil r_1^\star\rceil + C}{C}\right\}.$$

□

The analysis of the competitive ratios with respect to f_2 and f_3 is conducted analogously to the proof of Theorem 3.4.1 (replacing the maximum by the arithmetic mean and the geometric mean). The optimal number of rental transactions corresponding to a competitive ratio with respect to f_2 is then given by

$$r_2^\star = \sqrt{\frac{1}{4} + \frac{BC(B+C-2)}{B+C}} - \frac{1}{2}.$$

For the competitive ratio with respect to f_3, the optimal number of rental transactions is given by

$$r_3^\star = \sqrt{BC} - 1.$$

The competitive ratios with respect to f_2 and f_3 are then calculated analogously to the competitive ratio with respect to f_1. We omit these calculations (since the steps are exactly the same as in the proof of Theorem 3.4.1) and just state the competitive ratios with respect to f_2 and f_3:

$$\mathcal{R}_s^{f_2}(\text{SKI}) = \min\left\{\frac{1}{2}\left(\frac{\lceil r_2^\star\rceil + B}{B} + \frac{\lceil r_2^\star\rceil + C}{C}\right), \frac{1}{2}\left(\frac{2\lfloor r_2^\star\rfloor + B + C}{\lfloor r_2^\star\rfloor + 1}\right)\right\},$$

$$\mathcal{R}_s^{f_3}(\text{SKI}) = \min\left\{\sqrt{\frac{(\lceil r_3^\star\rceil + B)(\lceil r_3^\star\rceil + C)}{BC}}, \sqrt{\frac{(\lfloor r_3^\star\rfloor + B)(\lfloor r_3^\star\rfloor + C)}{(\lfloor r_3^\star\rfloor + 1)^2}}\right\}.$$

Now, we have a closer look at $r_1^\star$, $r_2^\star$, and $r_3^\star$: Some basic calculations show that $r_1^\star \geq r_2^\star \geq r_3^\star$, i.e., the optimal algorithm with respect to f_3 does not rent more often than the optimal algorithm with respect to f_2 which in turn does not rent more often than the optimal algorithm with respect to f_1. This matches the fact that f_1 is the most pessimistic way (compared to f_2 and f_3) of measuring the competitive ratio, and renting skis is the more cautious action compared to buying skis.

The algorithm SKI presented above achieves the best possible competitive ratio with respect to f_1 for the bi-objective ski rental problem with $B \geq C$, which is formally proven by the following theorem.

Theorem 3.4.2. *No deterministic algorithm for the bi-objective ski rental problem with $B \geq C$ can achieve a smaller (strong) competitive ratio with respect to f_1 than*

$$\mathcal{R}_s^{f_1}(\text{SKI}) = \min\left\{\frac{\lfloor r_1^\star \rfloor + B}{\lfloor r_1^\star \rfloor + 1}, \frac{\lceil r_1^\star \rceil + C}{C}\right\}.$$

Proof. Denote by ALG_r the deterministic algorithm that rents skis r times and then buys skis. Obviously, there are no other reasonable deterministic algorithms for the bi-objective ski rental problem and, in order to be competitive, $r < +\infty$ must hold.

The cruel adversary chooses $n = r + 1$, i.e., the adversary waits until the online player buys skis and then ends the game. Thus, the online player's costs amount to $\text{ALG} = \big(r + B,\, r + C\big)^\mathsf{T}$. This case has been considered in the proof of Theorem 3.4.1 and Algorithm 16 chooses the optimal value for r with respect to this case in order to minimize the competitive ratio. Consequently, there is no deterministic algorithm for the bi-objective ski rental problem with $B \geq C$ that achieves a competitive ratio with respect to f_1 smaller than $\mathcal{R}_s^{f_1}(\text{SKI})$. □

3.5 The Bi-Objective 2-Server Problem in the Plane

Consider k servers which are located on points of a metric space $\mathcal{M} = (M, d)$, where d is a metric over the set of points M with $|M| > k$. In every time period, a request $r \in M$ is presented to the online player and the request r is served if one of the servers is located at r. The goal of the online player is to minimize the total distance traveled by the servers in order to serve all requests sequentially.

This problem is known as the *k-server problem* and was introduced in (Manasse et al., 1990, 1988). Since then, the k-server problem attracted attention of many researchers in the area of online algorithms and fostered the development of competitive analysis substantially. Manasse et al. (1990) proved that k is a lower bound on the competitive ratio for any deterministic algorithm for the k-server problem on any metric space with at least $k + 1$ points and presented a 2-competitive algorithm for the 2-server problem. Furthermore, they gave a k-competitive algorithm on any metric space with exactly $k+1$ points. The question of the existence of a k-competitive algorithm solving the k-server problem in an arbitrary metric space, also known as the k-server conjecture, is still unsolved.

For arbitrary metric spaces, Koutsoupias and Papadimitriou (1995) proved a competitive ratio of $(2k - 1)$ for the so-called work function algorithm for the k-server problem for all k. More recently, Bansal et al. (2011) presented a randomized algorithm for the k-server problem that achieves a competitive ratio of $\mathcal{O}(\log^2 k \log^3 n \log\log n)$ on any metric space with n points, which improves upon the deterministic $(2k-1)$-competitive algorithm given by Koutsoupias and Papadimitriou (1995) whenever n is sub-exponential in k. The gap between the lower bound of k and the competitive ratio of $(2k-1)$ of the work function algorithm is only closed in some special cases, such as when M is a tree (Chrobak and Larmore, 1996) or M has at most $k+2$ points (Koutsoupias and Papadimitriou, 1996). However, for arbitrary metric spaces, the k-server conjecture remains unsolved.

Consider the k-server problem on the line. For this problem, Chrobak et al. (1990) gave a k-competitive (and hence optimal) deterministic algorithm DOUBLE-COVERAGE (see Algorithm 17) which moves the server nearest to the request if the request is on one side of all servers. If the request is between two servers, both adjacent servers are moved closer to the request until one of them covers the request.

Algorithm 17: DOUBLE-COVERAGE (Chrobak et al., 1990).

```
1 if the request is between two servers then
2     Move both adjacent servers at the same speed closer to the request until one of
      them covers the request.
3 else if the request is on one side of all servers then
4     Move the server nearest to the request.
```

Now, we transfer this problem to the plane and restrict ourselves to two servers that are allowed to move horizontally and vertically. Additionally, we introduce a bicriterial objective function which aims to minimize the distance traveled by both servers in the first dimension and the second dimension. More precisely, we consider the positive quadrant of the euclidean plane $\mathbb{R}^2_+$ and the pseudo-metrics d_1 and d_2, where

$$d_i(\mathbf{x},\mathbf{y}) = |x_i - y_i|, \quad \text{for } \mathbf{x},\mathbf{y} \in \mathbb{R}^2_+.$$

Note that d_1 and d_2 are only pseudo-metrics since, in general, $d_i(\mathbf{x},\mathbf{y}) = 0$ does not imply that $\mathbf{x} = \mathbf{y}$ (hence, the identity of indiscernibles is not satisfied).

Thus, in every time period, a request $\mathbf{r} \in \mathbb{R}^2_+$ is presented to the online player and the request is served if either server is located at $\mathbf{r}$. Both servers are allowed to move horizontally and vertically. The goal of the online player is to minimize the total distance traveled by the servers in both dimensions in order to serve all requests sequentially. This problem is labeled as the *bi-objective 2-server problem in the plane.*

In the following, we present a deterministic online algorithm for the bi-objective 2-server problem in the plane which is best possible with respect to a worst-component competitive analysis and a mean-component competitive analysis.

The positions of the online player's servers are in the following denoted by $\mathbf{s}$ and $\mathbf{t}$. We subdivide the positive quadrant of the euclidean plane into four sections A, B, C, and

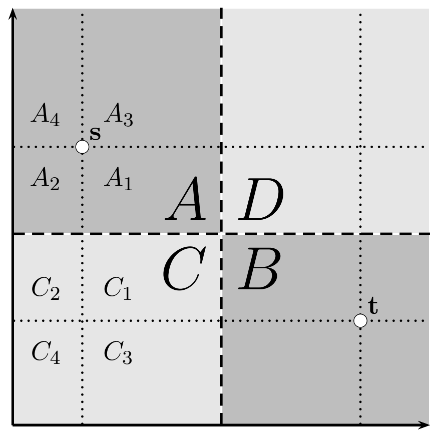


Figure 3.9: Subdivision of the positive quadrant of the euclidean plane according to the current positions s and t of the two servers.

D, according to the positions $\mathbf{s}$ and $\mathbf{t}$ of the online player's servers (see also Figure 3.9):

$$\begin{aligned}
A &= \left\{\mathbf{x} \in \mathbb{R}^2_+ \mid d_1(\mathbf{x},\mathbf{s}) \le d_1(\mathbf{x},\mathbf{t}) \ \wedge \ d_2(\mathbf{x},\mathbf{s}) \le d_2(\mathbf{x},\mathbf{t})\right\}, \\
B &= \left\{\mathbf{x} \in \mathbb{R}^2_+ \mid d_1(\mathbf{x},\mathbf{s}) \ge d_1(\mathbf{x},\mathbf{t}) \ \wedge \ d_2(\mathbf{x},\mathbf{s}) \ge d_2(\mathbf{x},\mathbf{t})\right\}, \\
C &= \left\{\mathbf{x} \in \mathbb{R}^2_+ \mid d_1(\mathbf{x},\mathbf{s}) \le d_1(\mathbf{x},\mathbf{t}) \ \wedge \ d_2(\mathbf{x},\mathbf{s}) \ge d_2(\mathbf{x},\mathbf{t})\right\}, \\
D &= \left\{\mathbf{x} \in \mathbb{R}^2_+ \mid d_1(\mathbf{x},\mathbf{s}) \ge d_1(\mathbf{x},\mathbf{t}) \ \wedge \ d_2(\mathbf{x},\mathbf{s}) \le d_2(\mathbf{x},\mathbf{t})\right\}.
\end{aligned}$$

By means of this subdivision, the algorithm DC-PLANE for the bi-objective 2-server problem in the plane is given by:

Algorithm 18: Bi-objective 2-server problem in the plane: DC-PLANE.

```
1 if request $\mathbf{r} \in A, B$ then
2     Apply DOUBLE-COVERAGE to both components.
3 else if request $\mathbf{r} \in C, D$ then
4     Move both servers at the same speed and in horizontal and vertical direction
      closer to the request as long as each server also moves closer to the starting
      point of the other server.
5     if no server covers the request then
6         Cover the request with the server closer to $\mathbf{r}$ with respect to the sum of the
          distances in the first and the second component between the server's
          position and the request's position.
```

For an illustration of the functioning of DC-PLANE, see Figures 3.10a-3.11b.

Theorem 3.5.1. *The strong competitive ratio with respect to* $f_1 = \max_{i=1,\dots,n} c_i$ *of* DC-PLANE *is given by 2.*

Proof. Consider a sequence of requests $\sigma = (\mathbf{r_1}, \dots, \mathbf{r_n})$ and a corresponding efficient offline solution $\mathbf{x} \in \textsc{opt}[I]$ with objective value $\textsc{opt}(\mathbf{x})$. Furthermore, let $\textsc{alg}^i$ denote the algorithm's cost for the i-th step and let $\textsc{opt}(\mathbf{x})^i$ denote the cost of the i-th step of the efficient solution $\mathbf{x}$. The proof is now based on a potential function $\Phi := \Gamma + \Delta$ where

$$\Gamma := 2\begin{pmatrix} M_{\min} \\ M_{\min} \end{pmatrix} \quad \text{and} \quad \Delta := \begin{pmatrix} d_1(\mathbf{s},\mathbf{t}) \\ d_2(\mathbf{s},\mathbf{t}) \end{pmatrix}.$$

Note that $\mathbf{s}$ and $\mathbf{t}$ are the positions of the online player's servers. $M_{\min}$ is the minimum sum of the components of the 2-dimensional minimum cost matching between the online player's servers and the adversary's servers with respect to the efficient solution $\mathbf{x}$: Denote the position of the adversary's servers with respect to the efficient solution $\mathbf{x}$ by $\mathbf{u}$ and $\mathbf{v}$. Additionally, for $i = \mathbf{s}, \mathbf{t}$ and $j = \mathbf{u}, \mathbf{v}$, consider binary variables $x_{i,j}$ that take value one if the online player's server i is matched to adversary's server j. $M_{\min}$ is then the optimal objective value of the following integer program:

$$\begin{aligned} \min \quad & \sum_{i \in \{\mathbf{s},\mathbf{t}\}} \sum_{j \in \{\mathbf{u},\mathbf{v}\}} x_{i,j} \left(d_1(i,j) + d_2(i,j)\right) && \text{(M)} \\ \text{s.t.} \quad & \sum_{i \in \{\mathbf{s},\mathbf{t}\}} x_{i,j} = 1 \quad \text{for } j \in \{\mathbf{u},\mathbf{v}\}, \\ & \sum_{j \in \{\mathbf{u},\mathbf{v}\}} x_{i,j} = 1 \quad \text{for } i \in \{\mathbf{s},\mathbf{t}\}, \\ & \qquad x_{i,j} \in \{0,1\} \quad \text{for } i \in \{\mathbf{s},\mathbf{t}\} \text{ and } j \in \{\mathbf{u},\mathbf{v}\}. \end{aligned}$$

Note that $\Phi \in \mathbb{R}^2_+$ and denote by $\Phi^i \in \mathbb{R}^2_+$ the value of the potential function after step i. Now, we use a potential function argument in order to prove the competitive ratio. In particular, we utilize the *interleaving moves style* (see (Borodin and El-Yaniv, 1998, Section 1.4)), i.e., we show for each step that

(I) if the online player moves $\mathbf{v} \in \mathbb{R}^2_+$, Φ decreases by at least $\mathbf{v}$, i.e., $\Phi^i - \Phi^{i-1} \leqq -\mathbf{v}$.

(II) if the offline player moves $\mathbf{v} \in \mathbb{R}^2_+$, Φ increases by at most $2\mathbf{v}$, i.e., $\Phi^i - \Phi^{i-1} \leqq 2\mathbf{v}$.

By means of (I) and (II), we have

$$\begin{aligned} & \Phi^i - \Phi^{i-1} \leqq 2 \cdot \textsc{opt}(\mathbf{x})^i - \textsc{alg}^i \\ \Rightarrow \quad & \textsc{alg}^i + \Phi^i - \Phi^{i-1} \leqq 2 \cdot \textsc{opt}(\mathbf{x})^i \\ \Rightarrow \quad & \sum_{i=1}^{n} \textsc{alg}^i + \Phi^n - \Phi^0 \leqq 2 \cdot \sum_{i=1}^{n} \textsc{opt}(\mathbf{x})^i \\ \Rightarrow \quad & \textsc{alg} \leqq 2 \cdot \textsc{opt}(\mathbf{x}) + \Phi_0, \end{aligned}$$

since $\Phi^n \geq 0$. Consequently, DC-PLANE is 2-competitive in both components due to the independence of Φ^0 of the sequence σ. Therefore, the competitive ratio with respect to f_1 is given by 2.

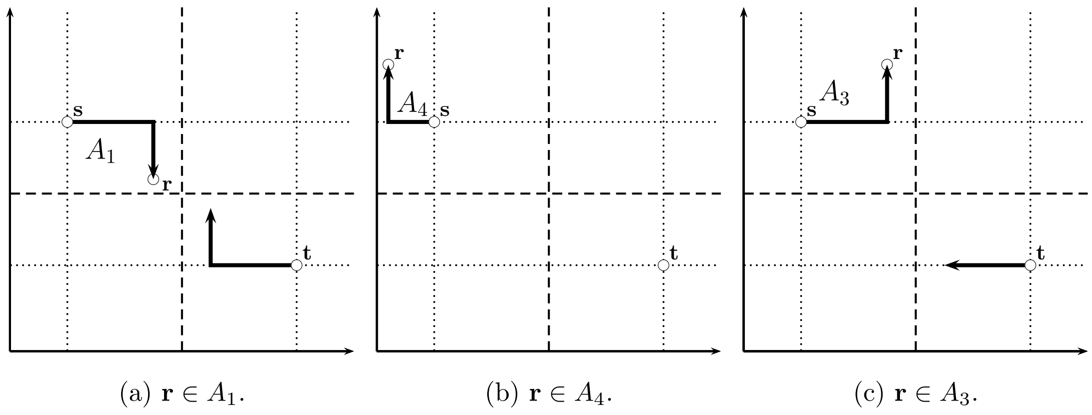

s
A_1
r
t
(a) $\mathbf{r} \in A_1$.
r
A_4
s
t
(b) $\mathbf{r} \in A_4$.
s
A_3
r
t
(c) $\mathbf{r} \in A_3$.

Figure 3.10: Servers' movements according to DC-PLANE in case $\mathbf{r} \in A$.

It is obvious that (II) holds in every step, since Δ is independent of the moves of the offline player and Γ increases by at most $2\mathbf{v}$ if the offline player moves $\mathbf{v}$. It remains to prove that (I) holds. For the following analysis, assume that a new request $\mathbf{r}$ is revealed and the offline player has already made its move. We distinguish two cases depending on the relation of the location of the new request $\mathbf{r}$ to the servers $\mathbf{s}$ and $\mathbf{t}$ of the online player:

Case 1: $r \in A, B$

Without loss of generality, we assume that $\mathbf{r} \in A$, since, due to the symmetry of the problem, the analysis is performed analogously for $\mathbf{r} \in B$. According to DC-PLANE, the single-objective algorithm DOUBLE-COVERAGE is applied to both components.

If $\mathbf{r} \in A_1$ (see Figure 3.9), both servers are moved in both components since the request is in between both servers in both components (see Figure 3.10a). Due to $\mathbf{r} \in A_1$ and, hence, $d_1(\mathbf{s}, \mathbf{r}) \leq d_1(\mathbf{t}, \mathbf{r})$ and $d_2(\mathbf{s}, \mathbf{r}) \leq d_2(\mathbf{t}, \mathbf{r})$, $\mathbf{s}$ covers $\mathbf{r}$. Since the online player's servers move towards each other, and $\mathbf{s}$ covers $\mathbf{r}$, Δ is decreased by $2\,\big(d_1(\mathbf{s}, \mathbf{r}),\, d_2(\mathbf{s}, \mathbf{r})\big)^{\intercal}$.

Furthermore, since both servers move towards $\mathbf{r}$, $M_{\min}$ does not increase: If $\mathbf{s}$ is matched to the adversary's server covering $\mathbf{r}$ before the online player's move, $M_{\min}$ decreases by the distance moved by $\mathbf{s}$ and increases at most by the distance moved by $\mathbf{t}$. Since $\mathbf{s}$ and $\mathbf{t}$ move the same total distance, $M_{\min}$ does not increase. Otherwise, if $\mathbf{t}$ is matched to the adversary's server covering $\mathbf{r}$ before the online player's move, the situation is analyzed analogously due to symmetry. The total change of Φ is hence bounded by

$$2\begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}-2d_1(\mathbf{s},\mathbf{r})\\-2d_2(\mathbf{s},\mathbf{r})\end{pmatrix} = \begin{pmatrix}-2d_1(\mathbf{s},\mathbf{r})\\-2d_2(\mathbf{s},\mathbf{r})\end{pmatrix},$$

which implies that (I) holds.

If $\mathbf{r} \in A_4$, only the server $\mathbf{s}$ is moved by $\big(d_1(\mathbf{s}, \mathbf{r}),\, d_2(\mathbf{s}, \mathbf{r})\big)^{\intercal}$ in order to cover $\mathbf{r}$ since the request is on one side of both servers with respect to both components (see Figure 3.10b). Due to $\mathbf{r} \in A_4$, there exists an optimal solution to (M), such that $\mathbf{s}$ is matched to the adversary's server covering $\mathbf{r}$ before and after its move. Thus, Δ increases by $\big(d_1(\mathbf{s},\mathbf{r}),\, d_2(\mathbf{s},\mathbf{r})\big)^\intercal$ and $M_{\min}$ decreases by $d_1(\mathbf{s},\mathbf{r}) + d_2(\mathbf{s},\mathbf{r})$. The total change of Φ is thus bounded by

$$2\begin{pmatrix}-(d_1(\mathbf{s},\mathbf{r})+d_2(\mathbf{s},\mathbf{r}))\\-(d_1(\mathbf{s},\mathbf{r})+d_2(\mathbf{s},\mathbf{r}))\end{pmatrix}+\begin{pmatrix}+d_1(\mathbf{s},\mathbf{r})\\+d_2(\mathbf{s},\mathbf{r})\end{pmatrix}=\begin{pmatrix}-d_1(\mathbf{s},\mathbf{r})-2d_2(\mathbf{s},\mathbf{r})\\-2d_1(\mathbf{s},\mathbf{r})-d_2(\mathbf{s},\mathbf{r})\end{pmatrix},$$

which again implies that (I) holds.

Finally, if $\mathbf{r} \in A_3$ (the case of $\mathbf{r} \in A_2$ is analyzed analogously due to symmetry), both servers are moved towards $\mathbf{r}$ in the first component since the request is in between both servers with respect to the first component, and only $\mathbf{s}$ is moved towards $\mathbf{r}$ in the second component since the request is on one side of both servers with respect to the second component (see Figure 3.10c). Due to $\mathbf{r} \in A$, $\mathbf{s}$ covers $\mathbf{r}$. Thus, Δ changes by $\big(-2d_1(\mathbf{s},\mathbf{r}),\, d_2(\mathbf{s},\mathbf{r})\big)^\intercal$.

$M_{\min}$ does not increase by the move in the first component since both servers move towards $\mathbf{r}$ (see the argumentation in the case $\mathbf{r} \in A_1$). Thus, after the move *in the first component*, $M_{\min}$ did not increase and we have the following situation: $\mathbf{s}$ and $\mathbf{r}$ are equal in the first component and $d_2(\mathbf{r},\mathbf{s}) \leq d_2(\mathbf{r},\mathbf{t})$. Consequently, there exists an optimal solution to (M) such that $\mathbf{s}$ is matched to $\mathbf{r}$. Therefore, $M_{\min}$ decreases by at least $d_2(\mathbf{s},\mathbf{r})$ due to the move in the second component and the overall change in the potential function Φ is bounded by

$$2\begin{pmatrix}-d_2(\mathbf{s},\mathbf{r})\\-d_2(\mathbf{s},\mathbf{r})\end{pmatrix}+\begin{pmatrix}-2d_1(\mathbf{s},\mathbf{r})\\d_2(\mathbf{s},\mathbf{r})\end{pmatrix}=\begin{pmatrix}-2d_1(\mathbf{s},\mathbf{r})-2d_1(\mathbf{s},\mathbf{r})\\-d_2(\mathbf{s},\mathbf{r})\end{pmatrix}.$$

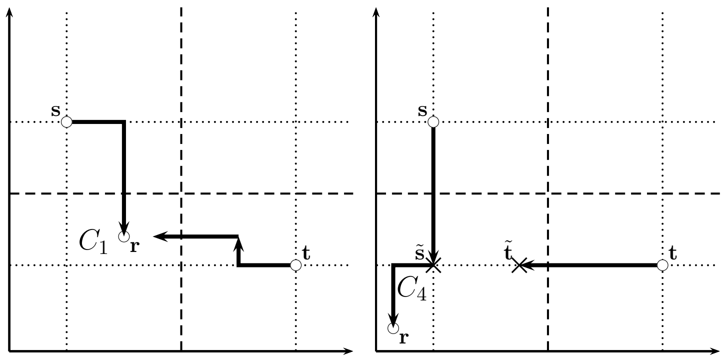
Since the online player's move is given by $\big(2d_1(\mathbf{s},\mathbf{r}),\, d_2(\mathbf{s},\mathbf{r})\big)^\intercal$, (I) holds.

Case 2: $\mathbf{r} \in C, D$

As before, we assume, without loss of generality, that $\mathbf{r} \in C$ since, due to the symmetry of the problem, the analysis is performed analogously for $\mathbf{r} \in D$.

If $\mathbf{r} \in C_1$, i.e., $\mathbf{r}$ lies between both servers in both components (see Figure 3.11a) and either $\mathbf{s}$ or $\mathbf{t}$ will cover $\mathbf{r}$ during Step 4 of Algorithm 18. Thus, Δ decreases exactly by the distance of the online move, since both servers move towards each other for the whole move. Figure 3.11a gives an example for the movement of both servers. Note that server $\mathbf{t}$ could also be moved first in the second component and then in the first component, or first in the first component, continuing in this direction longer than depicted in the example, as long as the rule of DC-PLANE is obeyed. Furthermore, $M_{\min}$ does not increase since both servers move the same overall distance towards $\mathbf{r}$ (see the argumentation in the case $\mathbf{r} \in A_1$). Consequently, (I) holds.

If $\mathbf{r} \in C_2, C_3,$ or C_4, first of all both servers are moved at the same speed and in horizontal and vertical direction closer to the request as long as each server is also moved closer to the starting point of the other server. Figure 3.11b gives an example for the movement of both servers: the end point of Step 4 of Algorithm 18 is marked by a cross for both servers and, during this step, $\mathbf{s}$ cannot move to the left since this would



(a) $\mathbf{r} \in C_1$. (b) $\mathbf{r} \in C_4$.

Figure 3.11: Servers' movements according to DC-PLANE in case $\mathbf{r} \in C$.

mean moving away from $\mathbf{t}$, and $\mathbf{t}$ cannot move down since this would mean moving away from $\mathbf{s}$. As described above for the case $\mathbf{r} \in C_1$, Δ decreases exactly by the distance of the online move since both servers move towards each other for the whole move, and $M_{\min}$ does not increase since both servers move the same overall distance towards $\mathbf{r}$.

Denote the positions of $\mathbf{s}$ and $\mathbf{t}$ after Step 4 of Algorithm 18 by $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$. Now, consider (M). There exists an optimal solution to (M) such that the server closer to $\mathbf{r}$ with respect to the sum of the distances in the first and the second component between the server's position and the request's position is matched to $\mathbf{r}$ since $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ are equal in at least one component (otherwise, the servers could move closer to the request). According to DC-PLANE, the server closer to $\mathbf{r}$ with respect to the sum of the distances in the first and the second component between the server's position and the request's position is moved to cover $\mathbf{r}$. Assume, without loss of generality, that $\tilde{\mathbf{s}}$ is the closer server, the other case is analyzed analogously due to symmetry. Then, Δ increases by $\big(d_1(\tilde{\mathbf{s}},\mathbf{r}),\, d_2(\tilde{\mathbf{s}},\mathbf{r})\big)^\intercal$ and $M_{\min}$ decreases by $d_1(\tilde{\mathbf{s}},\mathbf{r}) + d_2(\tilde{\mathbf{s}},\mathbf{r})$. Thus, the overall change of Φ is bounded by

$$\begin{aligned} &2 \cdot \begin{pmatrix} -d_1(\tilde{\mathbf{s}},\mathbf{r}) - d_2(\tilde{\mathbf{s}},\mathbf{r}) \\ -d_1(\tilde{\mathbf{s}},\mathbf{r}) - d_2(\tilde{\mathbf{s}},\mathbf{r}) \end{pmatrix} + \begin{pmatrix} -d_1(\mathbf{s},\tilde{\mathbf{s}}) - d_1(\mathbf{t},\tilde{\mathbf{t}}) + d_1(\tilde{\mathbf{s}},\mathbf{r}) \\ -d_2(\mathbf{s},\tilde{\mathbf{s}}) - d_2(\mathbf{t},\tilde{\mathbf{t}}) + d_2(\tilde{\mathbf{s}},\mathbf{r}) \end{pmatrix} \\ &= - \begin{pmatrix} d_1(\mathbf{s},\tilde{\mathbf{s}}) + d_1(\mathbf{t},\tilde{\mathbf{t}}) + d_1(\tilde{\mathbf{s}},\mathbf{r}) + 2d_2(\tilde{\mathbf{s}},\mathbf{r}) \\ d_2(\mathbf{s},\tilde{\mathbf{s}}) + d_2(\mathbf{t},\tilde{\mathbf{t}}) + d_2(\tilde{\mathbf{s}},\mathbf{r}) + 2d_1(\tilde{\mathbf{s}},\mathbf{r}) \end{pmatrix}, \end{aligned}$$

which is, in both components, a decrease by at least the distance of the online move which is given by

$$\begin{pmatrix} d_1(\mathbf{s},\tilde{\mathbf{s}}) + d_1(\mathbf{t},\tilde{\mathbf{t}}) + d_1(\tilde{\mathbf{s}},\mathbf{r}) \\ d_2(\mathbf{s},\tilde{\mathbf{s}}) + d_2(\mathbf{t},\mathbf{r}) + d_2(\tilde{\mathbf{s}},\mathbf{r}) \end{pmatrix}.$$

Thus, (I) holds also in this case.

By the reasoning above, DC-PLANE is 2-competitive in both components. Therefore, the competitive ratio with respect to f_1 is given by 2. Since the proof is conducted for an arbitrary efficient offline solution, the competitive ratio holds for all efficient offline solutions, and, therefore, the strong competitive ratio with respect to f_1 of DC-PLANE is also given by 2. □

Note that this result also holds for the strong competitive ratio with respect to f_2 and f_3 since DC-PLANE is 2-competitive in both components. Due to Theorem 3.7.1 (see Section 3.7), there is no algorithm for the bi-objective 2-server problem in the plane with a strong competitive ratio with respect to f_1 smaller than 2, since the lower bound on the competitive ratio of any algorithm for the 2-server problem on the line is given by 2. The same holds for the strong competitive ratio with respect to f_2 and f_3.

3.6 The Multi-Objective k-Canadian Traveller Problem

In this section, we introduce the multi-objective k-Canadian traveller problem and perform a worst-component competitive analysis. The Canadian traveller problem (CTP) is a variant of the shortest path problem and was initially introduced by Papadimitriou and Yannakakis (1991) in the following way:

Consider an undirected graph $G = (V, E)$ with non-negative edge-weights $w : E \to \mathbb{R}_+$ and distinguished nodes $s \in V$ and $t \in V$. The goal is to find a shortest path from s to t with respect to w. However, some of the edges may be blocked and an online algorithm only learns about the blockade of an edge when reaching one of its endpoints. In the k-Canadian traveller problem (k-CTP), the number of blocked edges is bounded from above by k.

The k-CTP is firstly studied in the framework of competitive analysis by Westphal (2008). Here, a lower bound of $2k + 1$ for the competitive ratio of any deterministic algorithm is given and, additionally, the deterministic algorithm BACKTRACK achieving the optimal competitive ratio of $2k + 1$, is presented. BACKTRACK repeatedly computes the shortest path in G, returns to s when a blockade is reached, and erases the blocked edge from G:

Algorithm 19: BACKTRACK for the k-CTP (Westphal, 2008).

```
   Data: Graph $G = (V, E)$ with non-negative edge-weights $w : E \to \mathbb{R}_+$,
         distinguished start and end nodes $s$ and $t$.
   Result: Shortest path $p$ from $s$ to $t$.
1  while $t$ is not reached do
2      Compute shortest path $p$ with respect to $w$ from $s$ to $t$ in $G = (V, E)$ and
       traverse it.
3      if edge $e$ on path $p$ is blocked then
4          Return to $s$ and set $E \leftarrow E \setminus e$.
```

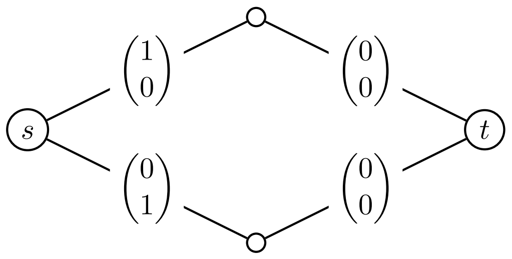


Figure 3.12: A worst case example for the multi-objective k-CTP.

Furthermore, a lower bound of $k+1$ for any randomized algorithm for the k-CTP is proven. For node-disjoint paths, this lower bound is matched by a randomized version of BACKTRACK given in (Bender and Westphal, 2013). The algorithm computes the $k+1$ shortest node-disjoint paths in the graph and chooses one of them according to an appropriately defined probability distribution. If the path is blocked, the procedure is repeated for a smaller set of paths.

We introduce a multi-objective version of the k-CTP by replacing the non-negative edge-weights w by a non-negative weight vector $\mathbf{w} : E \to \mathbb{R}^n_+$. The goal is to find a shortest path from s to t with respect to $\mathbf{w}$. Note that the optimal offline solution is a set of efficient paths. This problem is denoted as the *multi-objective k-CTP*.

First of all, we observe that, without any further restrictions, there exist no competitive algorithms for the multi-objective k-CTP:

Theorem 3.6.1. *There is no competitive randomized algorithm for the multi-objective k-CTP for any $n \geq 2$ and $k \geq 1$.*

Proof. Consider the graph in Figure 3.12. An online player chooses the upper path from s to t with probability p_1 and the lower path from s to t with probability p_2, which characterizes all randomized algorithms for this instance. ALG's costs are then given by

$$\text{ALG} = \begin{cases} p_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + p_2 \begin{pmatrix} 1 \\ 2 \end{pmatrix} & \text{if the lower edge is blocked,} \\ p_1 \begin{pmatrix} 2 \\ 1 \end{pmatrix} + p_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if the upper edge is blocked.} \end{cases}$$

Since OPT is given by $\big(1,\, 0\big)^\intercal$ if the lower edge is blocked and $\big(0,\, 1\big)^\intercal$ otherwise, the competitive ratio is always unbounded in at least one component. □

Thus, similar to the time series search problem investigated in Section 3.3, we assume that the weight $\mathbf{w}(p)$ of each path p from s to t (without any cycles) is bounded in each component by $0 < m \leq \mathbf{w}(p)_i \leq M$, for $i = 1, \ldots, n$, and define $\varphi = {}^{M}\!/_{m}$ as the fluctuation ratio of the weight of possible paths. This problem is denoted as the *multi-objective k-CTP with bounded paths.*

In the following, a competitive deterministic algorithm for the multi-objective k-CTP with bounded paths is presented, which is, basically, a multi-objective version of BACKTRACK:

Algorithm 20: MULTI-BACKTRACK.

```
  Data: Graph $G = (V, E)$ with non-negative edge-weights $\mathbf{w} : E \to \mathbb{R}^n_+$,
        distinguished start and end nodes $s$ and $t$.
  Result: Shortest path $p$ from $s$ to $t$.
1 Calculate the set of efficient solutions $\mathcal{P}$ for shortest paths from $s$ to $t$ with respect
  to the edge-weight vector $\mathbf{w}$.
2 Set $i \leftarrow 0$.
3 while $t$ is not reached do
4     Choose an efficient solution $p \in \mathcal{P}$ that minimizes component $(i \mod n) + 1$
      and traverse it.
5     if edge $e$ on path $p$ is blocked then
6         Return to $s$ and set $E \leftarrow E \setminus e$.
7         Recompute the set of efficient solutions $\mathcal{P}$ in $G = (V, E)$.
8         Set $i \leftarrow i + 1$.
```

Theorem 3.6.2. *The worst-component competitive ratio of* MULTI-BACKTRACK *for the multi-objective k-*CTP *with bounded paths is given by*

$$\mathcal{R}^{f_1}(\text{MULTI-BACKTRACK}) = 1 + 2\left(k\varphi - \left\lfloor \frac{k}{n} \right\rfloor (\varphi - 1)\right).$$

Proof. In the worst case, the first k paths chosen by ALG are blocked by the adversary and ALG reaches t via the $(k+1)$-st path. Denote the costs of the paths by $\mathbf{w}(p_j)$, for $j = 1, \ldots, k+1$. Then, the costs of ALG are given by

$$\text{ALG} = \mathbf{w}(p_{k+1}) + 2\sum_{j=1}^{k} \mathbf{w}(p_j). \tag{3.30}$$

For some efficient solution $\mathbf{x} \in \text{OPT}[I]$, we then have

$$\text{ALG} \overset{(3.30)}{=} \text{OPT}(\mathbf{x}) + 2\sum_{j=1}^{k} \mathbf{w}(p_j), \tag{3.31}$$

since ALG always computes the current set of efficient solutions and, consequently, there exists an efficient offline solution $\mathbf{x} \in \text{OPT}[I]$ such that $\mathbf{w}(p_{k+1})_i = \text{OPT}(\mathbf{x})_i$, for $i = 1, \ldots, n$. The competitive ratio with respect to f_1 of ALG is now given by

$$\mathcal{R}^{f_1}(\text{MULTI-BACKTRACK}) \le \max_{i=1,\ldots,n} \left\{ \frac{\text{OPT}(\mathbf{x})_i + 2\sum_{j=1}^{k} \mathbf{w}(p_j)_i}{\text{OPT}(\mathbf{x})_i} \right\}, \tag{3.32}$$

where $\mathbf{w}\left(p_j\right)_i$ is the i-th component of the weight of path p_j. The path p_j chosen by ALG is selected from the set of (currently available) efficient paths such that component $i' := (j \mod n) + 1$ is minimized (see Step 4 of Algorithm 20). Thus, for every efficient offline solution $\tilde{\mathbf{x}}$, we have $\textsc{opt}(\tilde{\mathbf{x}})_{i'} \geq \mathbf{w}(p_j)_{i'}$ and, in particular,

$$\textsc{opt}(\mathbf{x})_{i'} \geq \mathbf{w}\left(p_j\right)_{i'}. \tag{3.33}$$

For the other components $i \neq i'$, we have

$$\frac{\mathbf{w}(p_j)_i}{\textsc{opt}(\mathbf{x})_i} \leq \varphi. \tag{3.34}$$

For $i = 1, \ldots, n$, we then have

$$\begin{aligned}\frac{\textsc{opt}(\mathbf{x})_i + 2\sum_{j=1}^{k} \mathbf{w}\left(p_j\right)_i}{\textsc{opt}(\mathbf{x})_i} &\leq 1 + \overbrace{2\left\lfloor\frac{k}{n}\right\rfloor}^{(3.33)} + \overbrace{2\left(k - \left\lfloor\frac{k}{n}\right\rfloor\right)\varphi}^{(3.34)} \\ &= 1 + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi - 1)\right). \end{aligned} \tag{3.35}$$

This completes the proof. $\square$

The worst-component competitive ratio of MULTI-BACKTRACK is best possible, as shown by the following theorem:

Theorem 3.6.3. *No deterministic algorithm for the multi-objective k-CTP with bounded paths can achieve a smaller worst-component competitive ratio than*

$$\mathcal{R}^{f_1}(\textsc{multi-backtrack}) = 1 + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi - 1)\right).$$

Proof. Consider the graph given in Figure 3.13, where $\mathbf{w}^j \in \mathbb{R}^n$ for $1 \leq j \leq k+1$ is given by

$$\mathbf{w}^j = \left(w_1^j, \ldots, w_n^j\right)^\mathsf{T} = (\underbrace{M, \ldots, M}_{j \mod n}, m, M, \ldots, M)^\mathsf{T},$$

with m at the $1 + (j \mod n)$-th position. Note that the weight for each path p from s to t is bounded by $m \leq \mathbf{w}(p)_i \leq M$, $i = 1, \ldots, n$.

For any deterministic algorithm ALG, the first k choices are blocked, forcing the online player to return to s. The last remaining possibility cannot be blocked by the adversary. Assume that the successful path has cost m in component j', and M in all other components. Since there is only one path left, the efficient solution chosen by the adversary denoted by $\mathbf{x}$ is the path ultimately chosen by the online player.

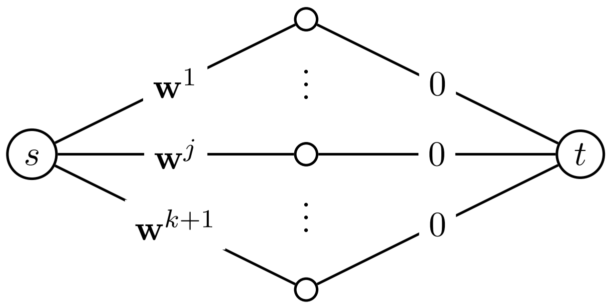


Figure 3.13: A worst case example for the multi-objective k-CTP with bounded paths.

If $n > k$, all other paths previously chosen by ALG have cost M in component j' (see the definition of $\mathbf{w}^j$). Otherwise, if $n \le k$, there are at most $\lfloor \frac{k}{n} \rfloor$ paths previously chosen by ALG that have cost m in component j. Therefore, we have

$$\begin{aligned}\max_{i=1,\dots,n}\left\{\frac{\text{ALG}_i}{\text{OPT}(\mathbf{x})_i}\right\} &\ge \frac{m + \overbrace{2\left\lfloor\frac{k}{n}\right\rfloor m}^{\substack{\text{blocked paths } p\\ \text{with } \mathbf{w}(p)_{j'}=m}} + \overbrace{2\left(k-\left\lfloor\frac{k}{n}\right\rfloor\right)M}^{\substack{\text{blocked paths } p\\ \text{with } \mathbf{w}(p)_{j'}=M}}}{m}\\ &= 1 + 2\left\lfloor\frac{k}{n}\right\rfloor + 2\left(k-\left\lfloor\frac{k}{n}\right\rfloor\right)\varphi\\ &= 1 + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi-1)\right).\end{aligned}$$

□

Note that, for $n > k$, the worst-component competitive ratio of MULTI-BACKTRACK is given by $1 + 2k\varphi$ and, for $n = 1$, MULTI-BACKTRACK coincides with the single-objective algorithm BACKTRACK. Furthermore, Theorem 3.6.2 considers the competitive ratio rather than the strong competitive ratio of MULTI-BACKTRACK. In the course of the proof of Theorem 3.6.2 the choice of a specific efficient solution is crucial in order to obtain the stated competitive ratio. If all efficient solutions are taken into account, the competitive ratio with respect to f_1 of MULTI-BACKTRACK increases:

Theorem 3.6.4. *The strong competitive ratio with respect to $f_1(c) = \max_{i=1,\dots,n} c_i$ of* MULTI-BACKTRACK *for the multi-objective k-*CTP *with bounded paths is given by*

$$\mathcal{R}_s^{f_1}(\text{MULTI-BACKTRACK}) = \varphi + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi-1)\right).$$

Proof. Consider (3.31) in the proof of Theorem 3.6.2. Here, we chose the efficient solution $\mathbf{x} \in \text{OPT}[I]$ such that $\text{OPT}(\mathbf{x})_i = \mathbf{w}(p_{k+1})_i$ for $i = 1, \dots, n$, i.e., the efficient solution coincides with the last path of the online player. If the efficient solution $\mathbf{x}$ chosen by the adversary differs from the path chosen by the online player in component i', the ratio of the weight of the adversary's and the online player's path in component i' is only bounded by φ. Analogous to the proof of Theorem 3.6.2, the strong competitive ratio with respect to f_1 is then given by

$$\begin{aligned}\mathcal{R}_s^{f_1}(\text{MULTI-BACKTRACK}) &= \max_{i=1,\dots,n}\left\{\frac{\mathbf{w}(p_{k+1})_i + 2\sum_{j=1}^{k}\mathbf{w}(p_j)_i}{\text{OPT}(\mathbf{x})_i}\right\}\\ &\le \frac{M}{m} + \frac{2\sum_{j=1}^{k}\mathbf{w}(p_j)_i}{m}\\ &= \varphi + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi-1)\right).\end{aligned}$$

□

The worst case instance given in Theorem 3.6.3 can be extended in order to provide a lower bound for the strong competitive ratio with respect to f_1 for any deterministic algorithm for the multi-objective k-CTP with bounded paths:

Theorem 3.6.5. *No deterministic algorithm for the multi-objective k-CTP with bounded paths can achieve a strong competitive ratio smaller than*

$$\mathcal{R}_s^{f_1}(\text{MULTI-BACKTRACK}) = \varphi + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi-1)\right).$$

Proof. Consider the graph in Figure 3.13 and add another path from s to t consisting of an edge with cost $\mathbf{w}^{k+2}$ and a subsequent edge with cost 0. The weight vector $\mathbf{w}^{k+2}$ is defined analogously to the weight vectors $\mathbf{w}^j$ in the proof of Theorem 3.6.3.

The adversary can always choose between two remaining paths after blocking the first k paths and obviously selects the one not chosen by the online player. By the same argumentation as in the proof of Theorem 3.6.3, we then have

$$\begin{aligned}\max_{i=1,\dots,n}\left\{\frac{\text{ALG}_i}{\text{OPT}(\mathbf{x})_i}\right\} &\ge \frac{M + 2\left\lfloor\frac{k}{n}\right\rfloor m + 2\left(k - \left\lfloor\frac{k}{n}\right\rfloor\right)M}{m}\\ &= \varphi + 2\left(k\varphi - \left\lfloor\frac{k}{n}\right\rfloor(\varphi-1)\right).\end{aligned}$$

□

3.7 Relations between Single- and Multi-Objective Online Problems

In this section, general relations between single- and multi-objective online problems are considered. More precisely, we consider questions of the following type: If, for $i = 1,\dots,n$, a c_i-competitive algorithm for the single-objective online problem associated with the i-th component of a multi-objective online problem's objective function is known, which statements about the competitiveness of algorithms for the corresponding multi-objective online problem can be deduced? Vice versa, if a (strongly) c-competitive algorithm for the multi-objective online problem is known, what can be stated about the competitiveness of algorithms for the associated single-objective online problems?

First of all, we formally define the notion of a *corresponding single-objective online problem to a multi-objective online problem*:

Definition 3.7.1. *Given a multi-objective online problem $\mathcal{P}$, a set $\mathcal{I}$ of inputs and an objective function $g : I \times \mathcal{X}(I) \to \mathbb{R}^n$, where $I \in \mathcal{I}$ and $\mathcal{X}(I)$ is the set of feasible outputs with respect to input I. For $i = 1, \ldots, n$, denote by g_i the i-th component of the objective function g. Then, the i-th corresponding single-objective online problem to $\mathcal{P}$ is given by $\mathcal{P}$, the set $\mathcal{I}$ of inputs and the objective function g_i.*

Consider a multi-objective online problem and the corresponding single-objective online problems. If we know the competitive ratios of the corresponding single-objective online problems, which conclusions can be drawn for the competitive ratio of the multi-objective online problem?

Theorem 3.7.1. *Let $\mathcal{P}$ be a multi-objective online problem and, for $i = 1, \ldots, n$, let c_i be the optimal competitive ratio for any algorithm for the i-th corresponding single-objective online problem. Then, we have for the strong competitive ratio with respect to $f_1(\tilde{c}) = \max_{i=1,\ldots,n} \tilde{c}_i$, $f_2(\tilde{c}) = \frac{1}{n} \sum_{i=1}^{n} \tilde{c}_i$, and $f_3 = \sqrt[n]{\prod_{i=1}^{n} \tilde{c}_i}$ of any multi-objective online algorithm* ALG *for $\mathcal{P}$,*

$$\mathcal{R}_s^{f_j}(\textsc{alg}) \geq f_j(c) \quad \text{for } j = 1, \ldots, 3,$$

where $c = \big(c_1, \ldots, c_n\big)^\mathsf{T}$.

Proof. First, we show that the functions f_1, f_2, and f_3 have the following common property:

$$\text{If } f_j(\tilde{c}) < f_j(c), \text{ then there exists an index } i' \text{ such that } \tilde{c}_{i'} < c_{i'}. \tag{3.36}$$

For f_1, we have $\tilde{c}_k \leq f_1(\tilde{c}) < f_1(c) = \max_{i=1,\ldots,n} c_i$ for all $k = 1, \ldots, n$ and, hence, (3.36) holds. For f_2, assume that $\tilde{c}_i \geq c_i$ for all $i = 1, \ldots, n$. Then, $f_2(\tilde{c}) \geq f_2(c)$ due to $\tilde{c}_i \geq 1$ and $c_i \geq 1$, and, hence, (3.36) holds. For f_3, (3.36) holds due to the same argumentation as for f_2.

For $j = 1, \ldots, 3$, suppose, there is an algorithm ALG for $\mathcal{P}$ with a strong competitive ratio $f_j(\tilde{c})$ such that $f_j(\tilde{c}) < f_j(c)$. By the definition of strong c-competitiveness (see Definition 3.2.2), we have, for all finite input sequences I and all efficient solutions $\mathbf{x} \in \textsc{opt}[I]$,

$$\textsc{alg}(I)_{i'} \leq \tilde{c}_{i'} \cdot \textsc{opt}(\mathbf{x})_{i'} + \alpha_{i'}. \tag{3.37}$$

Since (3.37) holds for all efficient solutions, this is also true for the optimal offline solution to the i'-th corresponding single-objective online problem denoted by $\mathbf{x}^\star$. For all finite input sequences I, we then have

$$\text{ALG}(I)_{i'} \overset{(3.37)}{\leq} \tilde{c}_{i'} \cdot \text{OPT}(\mathbf{x}^\star)_{i'} + \alpha_{i'}.$$

Consequently, there exists a $\tilde{c}_{i'}$-competitive algorithm for the i'-th corresponding single-objective online problem which is a contradiction since $\tilde{c}_{i'} < c_{i'}$ due to (3.36). Therefore, $f_j(\tilde{c}) \geq f_j(c)$. □

In particular, if one of the corresponding single-objective online problems is not competitive, the multi-objective online problem is not strongly competitive either. Note that Theorem 3.7.1 holds for any function f with property (3.36). An immediate implication of Theorem 3.7.1 is given by the following corollary:

Corollary 3.7.1. *Consider a multi-objective online problem $\mathcal{P}$. If there exists an algorithm for $\mathcal{P}$ with a strong competitive ratio $f_j(\tilde{c})$, $j = 1, \ldots, 3$, then there exists an algorithm for each corresponding single-objective online problem to the multi-objective online problem $\mathcal{P}$ with competitive ratio smaller than or equal to $f_j(\tilde{c})$.*

Note that Theorem 3.7.1 only holds for the *strong* competitive ratio but not for the competitive ratio:

Example 3.7.1. *Consider the time series search problem without any bounds on the prices p_t and objective function $\max \left(-p_t,\, p_t\right)^\mathsf{T}$. Due to the lack of bounds, there exists no competitive algorithm both for the single-objective problem corresponding to the first component of the objective function, i.e., $\max -p_t^1$, and for the single-objective problem corresponding to the second component of the objective function, i.e., $\max p_t^1$.*

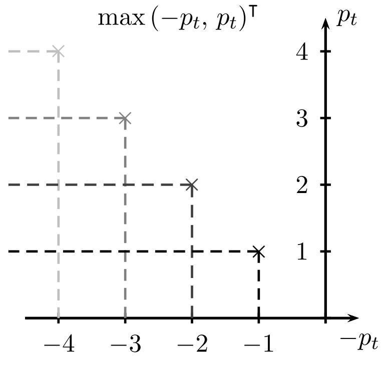


Figure 3.14: Every price is an efficient solution.

However, for the bi-objective problem, every solution is an efficient solution due to the objective function $\max \left(-p_t,\, p_t\right)^\mathsf{T}$. Consequently, any multi-objective online algorithm achieves a competitive ratio of 1 for the bi-objective problem, since any accepted price is also an efficient offline solution. We consider the competitive ratio with respect to any reasonable function f, i.e., a function that maps $\left(1,\, 1\right)^\mathsf{T}$ to 1. The strong competitive ratio, on the other hand, is unbounded for any multi-objective online algorithm.

Example 3.7.1 leads to the following observation:

Observation 4. *In general, the existence of a competitive algorithm for the multi-objective online problem does not imply the existence of competitive algorithms for the corresponding single-objective online problems.*

Now, we consider upper bounds on the competitive ratio of any multi-objective online algorithm based on the competitive ratios of the corresponding single-objective problems. The analysis of the multi-objective k-CTP in Section 3.6 showed that, even if there exists a competitive deterministic algorithm for each corresponding single-objective online problem, in general, this does not imply the existence of a competitive deterministic multi-objective online algorithm. This observation can also be made for the bi-objective online layered graph traversal problem:

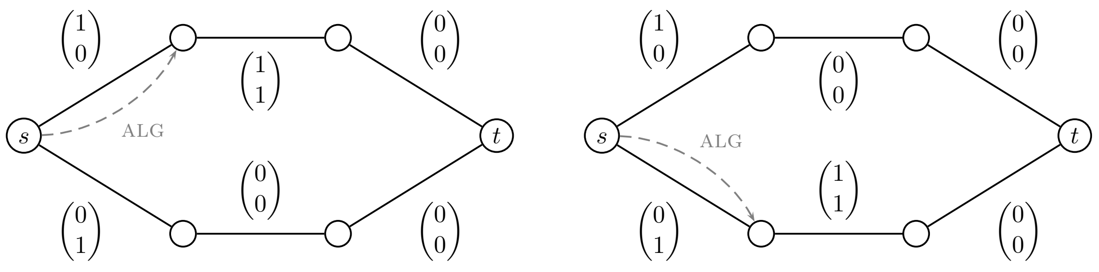
The bi-objective online layered graph traversal problem.

Consider the online layered graph traversal problem which was introduced by Papadimitriou and Yannakakis (1991). A layered graph is a connected graph $G = (V, E)$ whose set of vertices V is partitioned into subsets $L_0 = \{s\}, L_1, L_2, \ldots$ and whose edges $e \in E$ have nonnegative integral weights and run between L_i and L_{i-1} for some i. The width w of a layered graph is defined as $w := \max\{|L_i|\}$.

In the online layered graph traversal problem, a searcher starts at a source node s in a layered graph of unknown width w and tries to reach a target node t using the shortest possible path. However, the nodes in layer L_i and the edges from layer L_{i-1} to layer L_i with corresponding weights are only revealed when the searcher reaches layer L_{i-1}. In (Papadimitriou and Yannakakis, 1991), the authors present an algorithm with the optimal competitive ratio of 9 for the graph traversal problem on layered graphs of width 2, i.e., $|L_i| \leq 2$. For a layered graph consisting of w disjoint paths, i.e., w paths that are vertex disjoint except for the source s, Baeza-Yates et al. (1991) derive an optimal deterministic algorithm for all w with competitive ratio $1+2w\left(1 + {}^{1}\!/\!{}_{(w-1)}\right)^{w-1}$ which is asymptotic to $2ew$. Furthermore, in (Fiat et al., 1998), an algorithm with competitive ratio $\mathcal{O}(9^w)$ on layered graphs of arbitrary width w is presented, and, additionally, a lower bound on the competitive ratio of any deterministic online algorithm for the layered graph traversal problem of 2^{w-2} is given.

We consider the online layered graph traversal problem on graphs of width 2 and introduce a second weight component for each edge, in order to obtain a bi-objective online problem. As stated above, for both corresponding single-objective online problems there exists a (best possible) deterministic online algorithm with competitive ratio 9. However, as shown by the following example, there exists no competitive deterministic multi-objective online algorithm for the bi-objective online layered graph traversal problem on graphs of width 2.

Example 3.7.2. *Consider the graphs G_1 and G_2 given in Figure 3.15a and Figure 3.15b, respectively. Note that the layer L_1 and the corresponding edges from s to L_1 are identical in both graphs.*



(a) Graph G_1. (b) Graph G_2.

Figure 3.15: The bi-objective online layered graph traversal problem on graphs of width 2.

If an online algorithm decides to choose the upper edge from s to the first layer L_1, the adversary reveals L_2 and the corresponding edges from L_1 to L_2 as given in G_1. Regardless of whether the online player turns around and chooses the lower path or continues on the upper path, the cost of the online algorithm results in $\big(2,\, 1\big)^\intercal$. The optimal offline solution is given by the lower path with a cost of $\big(0,\, 1\big)^\intercal$. Consequently, this strategy is not competitive in the first component.

Otherwise, if an online algorithm decides to choose the lower edge from s to the first layer L_1, the adversary reveals L_2 and the corresponding edges from L_1 to L_2 as given in G_2. The online player finds himself in the same dilemma as in the first case and the cost of the online solution results in $\big(1,\, 2\big)^\intercal$, compared to the optimal offline solution with cost $\big(1,\, 0\big)^\intercal$. Consequently, this strategy is not competitive in the second component. Therefore, there exists no competitive online algorithm for the bi-objective online layered graph traversal problem on graphs of width 2.

The analysis of the multi-objective k-CTP and the bi-objective online layered graph traversal problem lead to the following observation:

Observation 5. *Even if there exists a competitive deterministic algorithm for each corresponding single-objective online problem to a multi-objective online problem, in general, this does not imply the existence of a (strongly) competitive deterministic multi-objective online algorithm.*

Finally, consider Example 3.7.3: Here, a bi-objective multi-objective online problem does not allow for an algorithm with finite competitive ratio. For one of the corresponding single-objective problems, there exists no competitive algorithm either. For the other corresponding single-objective problem, there exists a competitive algorithm.

Example 3.7.3. *Consider the online dial-a-ride problem in which objects are to be transported by a server between points in a metric space. Transportation requests arrive online, specifying the objects to be transported and the corresponding source and destination. Moving at a constant speed, the server starts its work at a designated origin and returns to its origin after all requests are served.*

For the minimization of the completion time there exist competitive algorithms (see, for example, (Ascheuer et al., 2000)). However, for the minimization of the maximal flow *time, i.e., the maximal difference between the point in time a request becomes known and the point in time this request is served, there exist no competitive algorithms (Hauptmeier et al., 2000):*

Consider the nonnegative real line $\mathbb{R}_+$ as the metric space and 0 as the starting point for the server. Assume that an online algorithm ALG *is c-competitive. At time $t = c$, the position $s(t)$ of the server of* ALG *is somewhere in $[0, c]$ and the adversary reveals a request at time $t = c$ with source x and destination $x + \epsilon$, where $\epsilon > 0$ and $x = \arg\max_{p \in \{0,c\}} \{d(s(t), p)\}$. Consequently, the cost for* ALG *is at least $c/2$ and the optimal offline solution is given by* OPT $= \epsilon$. *Therefore, the competitive ratio is given by $c/2\epsilon$, and, since ϵ can be chosen arbitrarily small,* ALG *cannot be c-competitive.*

Now, we consider the bi-objective online dial-a-ride problem with the minimization of both the completion time and the maximal flow time. By means of the same analysis as above, ALG*'s cost are at least $\left(c + c/2,\ c/2\right)^\intercal$ and the optimal offline solution is given by* OPT $= \left(c + \epsilon,\ \epsilon\right)^\intercal$, *resulting in a competitive ratio of*

$$\max\left\{\frac{c + c/2}{c + \epsilon}, \frac{c}{2\epsilon}\right\}.$$

Again, since ϵ can be chosen arbitrarily small, ALG *cannot achieve a finite competitive ratio with respect to a reasonable function f, i.e., a function f that maps a vector with at least one unbounded component to ∞.*

3.8 Yao's Principle for Multi-Objective Online Problems

Yao's principle is a method for obtaining lower bounds on the competitive ratio of any randomized online algorithm. In order to obtain a lower bound on the competitive ratio of any randomized online algorithm with this method, it is sufficient to choose a probability distribution q over request sequences and then consider the ratio of the expected optimal offline value and the expected value of any *deterministic* online algorithm, where the expectation is taken with respect to q. An upper bound on this ratio then gives a lower bound on the competitive ratio of any randomized online algorithm.

Instead of considering all possible randomized online algorithms, it suffices to consider the expected value of any deterministic online algorithm and, hence, this method often facilitates the search for lower bounds on the competitive ratio of any randomized online algorithm. In this section, we extend Yao's principle to multi-objective online algorithms, i.e., we present a method for obtaining lower bounds on the (strong) competitive ratio of any multi-objective randomized online algorithm. As for single-objective online algorithms, the cases of profit maximization and cost minimization online problems are treated separately. We start with profit maximization problems:

Theorem 3.8.1. *Let* ALG *be any randomized multi-objective online algorithm for a multi-objective online maximization problem and let $\{\text{ALG}_i : i \in \mathcal{I}\}$ denote the finite set of all deterministic multi-objective online algorithms for this problem. Let q be any probability distribution over the finite set of possible request sequences $\{\sigma_j : j \in \mathcal{J}\}$. Let $f : \mathbb{R}^n \to \mathbb{R}_+$*

be a monotonically increasing function and, for $j \in \mathcal{J}$, let $\mathbf{x}^j$ be an efficient solution with respect to request sequence σ_j, i.e., $\mathbf{x}^j \in \textsc{opt}[\sigma_j]$. Then,

$$\overline{\mathcal{R}}_s^f(\textsc{alg}) \geq f(c),$$

where $c = \left(c_1, \ldots, c_n\right)^\mathsf{T} \in \mathbb{R}^n$ is given by

$$c_k = \min_i \frac{\mathbb{E}_{q(j)}\left[\textsc{opt}\left(\mathbf{x}^j\right)_k\right]}{\mathbb{E}_{q(j)}\left[\textsc{alg}_i\left(\sigma_j\right)_k\right]}, \quad \text{for } k = 1, \ldots, n.$$

For the worst-component competitive ratio $\overline{\mathcal{R}}_s^{f_1}(\textsc{alg})$, where $f_1(c) = \max_{i=1,\ldots,n} c_i$, we additionally have

$$\overline{\mathcal{R}}_s^{f_1}(\textsc{alg}) \geq \min_i \frac{1}{\mathbb{E}_{q(j)}\left[\max_k \frac{\textsc{alg}_i(\sigma_j)_k}{\textsc{opt}(\mathbf{x}^j)_k}\right]}.$$

Proof. First of all, we prove that any randomized multi-objective online algorithm $\textsc{alg}$ is at most c-competitive with $c = \left(c_1, \ldots, c_n\right)^\mathsf{T} \in \mathbb{R}^n$ given by

$$c_k = \min_i \frac{\mathbb{E}_{q(j)}\left[\textsc{opt}(\mathbf{x}^j)_k\right]}{\mathbb{E}_{q(j)}\left[\textsc{alg}_i(\sigma_j)_k\right]} \quad \text{for } k = 1, \ldots, n.$$

Since f is a monotonically increasing function and the strong competitive ratio is given by the infimum over the set of all values $f(c)$ such that $\textsc{alg}$ is strongly c-competitive, we then have $\overline{\mathcal{R}}_s^f(\textsc{alg}) \geq f(c)$. The proof given below differs only slightly from the proof of Yao's principle for single-objective online problems, cf. (Borodin and El-Yaniv, 1998, pp. 117-118, Theorem 8.3).

Consider the constant vector $\tilde{c} = \left(\tilde{c}_1, \ldots, \tilde{c}_n\right)^\mathsf{T} \in \mathbb{R}^n$. For $k = 1, \ldots, n$, define the two-person zero-sum game $G_k(\tilde{c})$ between the online player and the adversary. The payoff function $h_k(i, j)$ for the online player is given by

$$h_k(i, j) = \tilde{c}_k \cdot \textsc{alg}_i(\sigma_j)_k - \textsc{opt}(\mathbf{x}^j)_k, \quad k = 1, \ldots, n, \tag{3.38}$$

i.e., $h_k(i, j)$ gives the payoff for the online player choosing strategy i against strategy j of the adversary. Note that $\textsc{alg}_i(\sigma_j)_k$ denotes the k-th component of the solution of the deterministic multi-objective online algorithm $\textsc{alg}_i$ with respect to request sequence σ_j and $\textsc{opt}(\mathbf{x}^j)_k$ denotes the k-th component of the efficient offline solution $\mathbf{x}^j$ with respect to request sequence σ_j. Due to (Borodin and El-Yaniv, 1998, p.112, Theorem 8.2), every finite two-person zero-sum game has a value $V(\tilde{c}_k)$. Furthermore, due to (Borodin and El-Yaniv, 1998, p.113, Lemma 8.2), $V(\tilde{c}_k)$ is given by

$$V(\tilde{c}_k) = \max_{p(i)} \min_j \mathbb{E}_{p(i)}\left[h_k(i, j)\right], \quad k = 1, \ldots, n, \tag{3.39}$$

where $p(i)$ is a mixed strategy for the online player. If $V(\tilde{c}_k) < 0$, for $k = 1, \ldots, n$, then, due to (3.38), the best randomized online algorithm is not strongly $\tilde{c}$-competitive. For $k = 1, \ldots, n$, suppose that

$$\begin{aligned} & \min_i \frac{\mathbb{E}_{q(j)}\left[\text{OPT}(\mathbf{x}^j)_k\right]}{\mathbb{E}_{q(j)}\left[\text{ALG}_i(\sigma_j)_k\right]} > \tilde{c}_k \\ \Leftrightarrow \quad & \mathbb{E}_{q(j)}\left[\text{OPT}\left(\mathbf{x}^j\right)_k\right] > \tilde{c}_k \cdot \max_i \mathbb{E}_{q(j)}\left[\text{ALG}_i(\sigma_j)_k\right] \\ \Leftrightarrow \quad & 0 > \max_i \mathbb{E}_{q(j)}\left[h_k(i,j)\right]. \end{aligned}$$

Due to Yao's inequality, cf. (Borodin and El-Yaniv, 1998, p.113), we hence have

$$0 > \max_i \mathbb{E}_{q(j)}\left[h_k(i,j)\right] \geq \max_{p(i)} \min_j \mathbb{E}_{q(j)}\left[h_k(i,j)\right] \overset{(3.39)}{=} V(\tilde{c}_k).$$

Therefore, ALG is at most c-competitive and since f is a monotonically increasing function, we have $\overline{\mathcal{R}}_s^f(\text{ALG}) \geq f(c)$.

Now, we consider the worst-component competitive ratio, i.e., $f_1(c) = \max_{k=1,\ldots,n} c_k$, and the payoff function

$$h(i,j) = \max_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}. \tag{3.40}$$

Consider some distribution $q(j)$ over request sequences and some $\mathbf{x}^j \in \text{OPT}\left[\sigma_j\right]$ for $j \in \mathcal{J}$. By applying Yao's inequality (cf. (Borodin and El-Yaniv, 1998, p.113)), we get

$$\begin{aligned} \max_{p(i)} \mathbb{E}_{q(j)}\left[\max_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_k}\right] &\geq \max_{p(i)} \min_j \mathbb{E}_{p(i)}\left[\max_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}\right] \\ &\geq \max_{p(i)} \min_j \max_k \frac{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]}{\text{OPT}\left(\mathbf{x}^j\right)_k} \qquad (3.41) \\ &\geq \max_{p(i)} \min_j \min_k \frac{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]}{\text{OPT}\left(\mathbf{x}^j\right)_k} \qquad (3.42) \\ &= \frac{1}{\min_{p(i)} \max_j \max_k \frac{\text{OPT}(\mathbf{x}^j)_k}{\mathbb{E}_{p(i)}[\text{ALG}_i(\sigma_j)_k]}}. \qquad (3.43) \end{aligned}$$

Here, (3.41) is due to the convexity of the maximum and, in (3.42), $\max_k$ is just replaced by $\min_k$. Since the optimal strong competitive ratio for maximization problems with respect to f_1 is given by

$$\overline{\mathcal{R}}_s^{f_1}(\text{ALG}) = \min_{p(i)} \max_j \max_{\mathbf{x}^j \in \text{OPT}[\sigma_j]} \max_k \frac{\text{OPT}\left(\mathbf{x}^j\right)_k}{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]},$$

we have, by (3.43),

$$\overline{\mathcal{R}}_s^{f_1}(\text{ALG}) \geq \min_i \frac{1}{\mathbb{E}_{q(j)}\left[\max_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}(\mathbf{x}^j)_k}\right]}.$$

This completes the proof. □

For cost minimization problems, the lower bounds are given in the same line. However, the proof differs slightly from the one given for profit maximization problems.

Theorem 3.8.2. *Let* ALG *be any randomized multi-objective online algorithm for a multi-objective online minimization problem and let $\{\text{ALG}_i : i \in \mathcal{I}\}$ denote the finite set of all deterministic multi-objective online algorithms for this problem. Let q be any probability distribution over the finite set of possible request sequences $\{\sigma_j : j \in \mathcal{J}\}$. Let $f : \mathbb{R}^n \to \mathbb{R}_+$ be a monotonically increasing function and, for $j \in \mathcal{J}$, let $\mathbf{x}^j$ be an efficient solution with respect to request sequence σ_j, i.e., $\mathbf{x}^j \in \text{OPT}[\sigma_j]$. Then,*

$$\overline{\mathcal{R}}_s^f(\text{ALG}) \geq f(c),$$

where $c = \big(c_1, \ldots, c_n\big)^\intercal \in \mathbb{R}^n$ is given by

$$c_k = \min_i \frac{\mathbb{E}_{q(j)}\left[\text{ALG}_i\left(\sigma_j\right)_k\right]}{\mathbb{E}_{q(j)}\left[\text{OPT}\left(\mathbf{x}^j\right)_k\right]}, \quad \textit{for } k = 1, \ldots, n.$$

For the worst-component competitive ratio $\overline{\mathcal{R}}_s^{f_1}(\text{ALG})$, where $f_1(c) = \max_{i=1,\ldots,n} c_i$, we additionally have

$$\overline{\mathcal{R}}_s^{f_1}(\text{ALG}) \geq \min_i \mathbb{E}_{q(j)}\left[\min_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}\right].$$

Proof. The lower bounds for $\overline{\mathcal{R}}_s^f(\text{ALG})$ and $\overline{\mathcal{R}}^f(\text{ALG})$ can be proven by defining the payoff function $h_k(i,j)$ as

$$h_k(i,j) = \tilde{c}_k \cdot \text{OPT}(\mathbf{x}^j)_k - \text{ALG}_i(\sigma_j)_k, \quad k = 1, \ldots, n,$$

and applying Yao's inequality for cost minimization problems. The remaining steps can be performed analogously to Theorem 3.8.1

For the worst-component competitive ratio, i.e., $f_1(c) = \min_{i=1,\ldots,n} c_i$, we define the payoff function

$$h(i,j) = \min_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}. \tag{3.44}$$

Consider some distribution $q(j)$ over request sequences and some $\mathbf{x}^j \in \text{OPT}\left[\sigma_j\right]$ for $j \in \mathcal{J}$. By applying Yao's inequality for cost minimization problems, we get

$$\begin{aligned}
\min_i \mathbb{E}_{q(j)}\left[\min_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}\right] &\leq \min_{p(i)} \max_j \mathbb{E}_{p(i)}\left[\min_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}\right] \\
&\leq \min_{p(i)} \max_j \min_k \frac{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]}{\text{OPT}\left(\mathbf{x}^j\right)_k} && (3.45) \\
&\leq \min_{p(i)} \max_j \max_k \frac{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]}{\text{OPT}\left(\mathbf{x}^j\right)_k}. && (3.46)
\end{aligned}$$

Here, (3.45) is due to the concavity of the minimum, and, in (3.46) $\min_k$ is just replaced by $\max_k$. Since the optimal strong competitive ratio with respect to f_1 for cost minimization problems is given by

$$\overline{\mathcal{R}}_s^{f_1}(\text{ALG}) = \min_{p(i)} \max_j \max_{\mathbf{x}^j \in \text{OPT}[\sigma_j]} \max_k \frac{\mathbb{E}_{p(i)}\left[\text{ALG}_i(\sigma_j)_k\right]}{\text{OPT}\left(\mathbf{x}^j\right)_k},$$

we have, by (3.46),

$$\overline{\mathcal{R}}_s^{f_1}(\text{ALG}) \geq \min_i \mathbb{E}_{q(j)}\left[\min_k \frac{\text{ALG}_i(\sigma_j)_k}{\text{OPT}\left(\mathbf{x}^j\right)_j}\right].$$

This completes the proof. □

The lower bounds obtained in Theorem 3.8.1 and Theorem 3.8.2 for the strong competitive ratio depend on the efficient offline solution chosen for each sequence. If *all* efficient offline solutions for each sequence are taken into account, a lower bound on the competitive ratio $\overline{\mathcal{R}}^f(\text{ALG})$ is obtained.

In the following example, we show how to apply Theorem 3.8.2 to a multi-objective online minimization problem in order to obtain a lower bound on the (strong) competitive ratio for any randomized algorithm.

Example 3.8.1. *Consider the bi-objective ski rental problem as introduced in Section 3.4 and set $B = 2$ and $C = 1$, i.e., buying skis costs $\left(2,\, 1\right)^\intercal$ and renting skis costs $\left(1,\, 1\right)^\intercal$. Now, we choose the distribution q over the request sequences (the number of skiing days) as $q(1) = 1/2$ and $q(3) = 1/2$, i.e, with probability $1/2$ there is one skiing day and with probability $1/2$ there are three skiing days. For $n = 1$, the only efficient offline solution is given by renting skis with costs $\left(1,\, 1\right)^\intercal$, since buying skis would cost $\left(2,\, 1\right)^\intercal$. For $n = 3$, the only efficient solution is given by buying skis with costs $\left(2,\, 1\right)^\intercal$, since renting skis would cost $\left(3,\, 3\right)^\intercal$. Thus, we have*

$$\mathbb{E}_{q(j)}\left[\text{OPT}\left(x^j\right)\right] = \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix} + \frac{1}{2}\begin{pmatrix}2\\1\end{pmatrix} = \begin{pmatrix}3/2\\1\end{pmatrix}.$$

Let ALG_i denote the deterministic online algorithm that rents skis $i-1$ times and buys skis on the i-th skiing day. The expected value with respect to q of ALG_i is then given by

$$\begin{aligned}\mathbb{E}_{q(j)}\left[\text{ALG}_i\left(\sigma_j\right)\right] &= \begin{cases}\begin{pmatrix}2\\1\end{pmatrix} & \text{for } i = 1,\\ \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix} + \frac{1}{2}\left(\begin{pmatrix}i-1+2\\i-1+1\end{pmatrix}\right) & \text{for } i = 2, 3,\\ \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix} + \frac{1}{2}\begin{pmatrix}3\\3\end{pmatrix} & \text{for } i > 3.\end{cases}\\ &\geq \begin{pmatrix}2\\1\end{pmatrix}.\end{aligned}$$

Therefore, we have

$$\min_i \frac{\mathbb{E}_{q(j)}\left[\textsc{alg}_i\left(\sigma_j\right)\right]}{\mathbb{E}_{q(j)}\left[\textsc{opt}\left(\mathbf{x}^j\right)\right]} = \begin{pmatrix} 4/3 \\ 1 \end{pmatrix},$$

and, by Theorem 3.8.2,

$$\overline{\mathcal{R}}_s^{f_1}(\textsc{alg}) \geq \max\left\{\frac{4}{3}, 1\right\} = \frac{4}{3}.$$

Consequently, the strong competitive ratio with respect to $f_1(c) = \max_{i=1,\dots,n} c_i$ *for any randomized online algorithm for the bi-objective ski rental problem is at least* $4/3$*. Since there is only one efficient solution in both cases (*$n = 1$ *and* $n = 3$*), this holds also for the competitive ratio for any randomized online algorithm. Considering the competitive ratio with respect to* f_2 *or* f_3*, we get the following lower bounds on the (strong) competitive ratio of any randomized algorithm: for* $f_2(c) = \frac{1}{n}\sum_{i=1}^n c_i$*, we have a lower bound of* $7/6$ *and, for* $f_3(c) = \sqrt[n]{\prod_{i=1}^n c_i}$*, we have a lower bound of* $\sqrt{4/3}$*.*

3.9 Conclusion and Future Research

In this chapter, we introduced a general framework for the competitive analysis of multi-objective online problems which expands the known theory of competitive analysis for online problems in a straightforward manner. In the course of this chapter, we gave several examples for the application of competitive analysis for multi-objective online problems and demonstrated that the analysis of multi-objective online problems by means of the introduced notions of competitiveness yields reasonable results which are closely related to the single-objective algorithms and their competitive ratios. Furthermore, we discussed relations between multi-objective online problems and the corresponding single-objective online problems and extended Yao's principle to multi-objective online problems.

The concept of competitive analysis for multi-objective online problems seems highly promising and provides further insight into the nature of online problems. Questions for future research include a more in-depth analysis of the problems proposed in this chapter such as randomized algorithms for the multi-objective k-CTP, and the analysis of multi-objective counterparts of other well-known online problems such as scheduling problems. Additionally, the definition of the competitive ratio given in this work serves as a basis for further extensions such as a vector of competitive ratios with respect to different functions: for example, if the analyst of the online problem wants the worst component and the average of the components to be reasonably small at the same time, the vector of both competitive ratios could be analyzed in the sense of multi-objective optimization.

4 The Linear Search Problem with Turn Costs

4.1 Introduction

The linear search problem is an optimal search problem independently introduced by Bellman (1963) and Beck (1964) in which an immobile object is located on the real line according to a probability distribution. A searcher starts from the origin and tries to find the object in minimum expected time. It is assumed that the searcher cannot see the object until she reaches the point at which the object is located. Originally, it is also assumed that the location of the object is given by a known probability distribution. However, in this work we consider the problem without knowledge about the probability distribution, leading to a basic online problem which is then analyzed by competitive analysis. The optimal competitive ratio for deterministic algorithms solving the linear search problem is 9, as first shown by Beck and Newman (1970). The optimal strategy for the searcher is to alternate between going to the right and to the left, doubling the step size in each iteration.

In (Demaine et al., 2006), the linear search problem (in the context of competitive analysis) is expanded by turn costs, i.e., each time the searcher changes direction a cost of d is incurred. The authors consider the sum of searching time and turn cost as objective function and present an algorithm which guarantees a solution smaller than $9 \cdot \textsc{opt} + 2d$. Note that, for deterministic algorithms, a minimum cost of d is required, regardless of $\textsc{opt}$, by placing the object arbitrarily close to the origin on the side not chosen by the searcher to start with. The additive term $2d$ is minimal subject to the (optimal) competitive ratio 9. As mentioned in (Demaine et al., 2006), it may be desirable to improve the additive term, while allowing an increase in the competitive ratio. The determining tradeoff curve is obtained experimentally, but it is not characterized analytically.

In this chapter, we close this gap by presenting an analytical characterization of the tradeoff curve between the competitive ratio and the additive term for the linear search problem with turn costs. We apply the analysis of the linear search problem presented in (Demaine et al., 2006) to an arbitrary competitive factor $\tilde{c}$ and determine the minimal additive factor analytically.

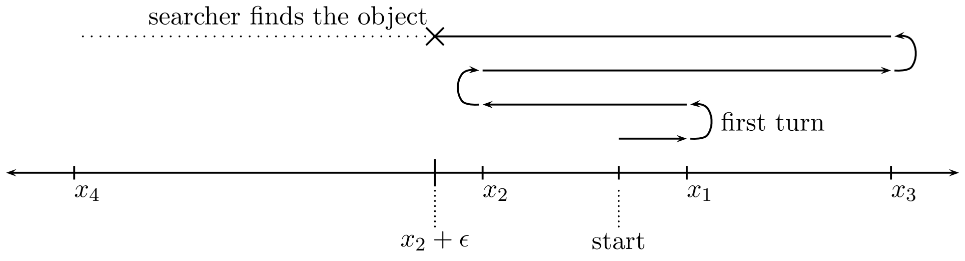


Figure 4.1: Path of the searcher for an object placed at $x_2 + \epsilon$.

4.2 Tradeoff between Competitive Factor and Turn Costs

In the following, we apply the method of infinite linear programs, introduced by Demaine et al. (2006), in order to solve the single-objective linear search problem with turn costs. In particular, we give an analytic characterization of the tradeoff curve between the competitive factor and the additive term.

A search strategy is given by a sequence of step lengths x_i, $i = 1, \dots$, each starting from the origin, where $x_1, x_3, \dots$ are increasing step lengths to the right and $x_2, x_4, \dots$ are increasing step lengths to the left. Consequently, the turn positions are given by $p_i = (-1)^{i+1} x_i$, see also Figure 4.1. The following theorem states the minimal additive term $\alpha(\tilde{c})$ for a competitive factor $\tilde{c} \geq 9$. The corresponding search strategy is given in the course of the proof.

Theorem 4.2.1. *For a competitive factor $\tilde{c} \geq 9$, the minimal additive term $\alpha(\tilde{c})$ for the linear search problem with turn cost is given by*

$$\alpha(\tilde{c}) = \frac{2d}{1 + \sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}.$$

Proof. In (Demaine et al., 2006), the authors determine the minimal possible additive term for the competitive factor 9 (which is the optimal competitive factor). In the following, we apply this method in order to determine the minimal additive term for an arbitrary competitive factor $\tilde{c} \geq 9$.

Suppose that the adversary places the object at $x_n + \epsilon$, $\epsilon > 0$, which is the worst case placement with respect to a search strategy as described above. Furthermore, let $\alpha(\tilde{c})$ be the minimal additive term for the competitive factor $\tilde{c}$. Then, OPT is given by OPT $= x_n + \epsilon$. Now, we are looking for an algorithm ALG such that ALG $\leq \tilde{c} \cdot$ OPT $+ \alpha(\tilde{c})$.

Let x_i be ALG's search strategy. The adversary's set of possible (worst case) locations is then given by $y_i = (-1)^{i+1}(x_i + \epsilon)$. Suppose, the adversary places the object at y_n. Then, it will take the online player $\sum_{i=1}^{n+1} 2x_i + x_n + \epsilon$ time and $n + 1$ turns to reach the object. Now, we would like to achieve a competitive factor of $\tilde{c}$ and determine the corresponding minimal additive cost α, i.e.,

$$\sum_{i=1}^{n+1} 2x_i + x_n + \epsilon + (n+1)d \leq \tilde{c}(x_n + \epsilon) + \alpha$$
$$\Leftrightarrow \quad \sum_{i=1}^{n+1} 2x_i + (1-\tilde{c})x_n + (n+1)d \leq \alpha,$$

since $x_i > 0$ and $\epsilon > 0$ is chosen arbitrarily. This gives us the following infinite linear program (ILP):

$$\begin{array}{llllll}
 & \min \alpha & & & & \\
\text{s.t.} & 2x_1 & & & & +d \leq \alpha \\
 & (3-\tilde{c})x_1 + & 2x_2 & & & +2d \leq \alpha \\
 & 2x_1 + (3-\tilde{c})x_2 + & & 2x_3 & & +3d \leq \alpha \\
 & 2x_1 + & 2x_2 + (3-\tilde{c})x_3 + 2x_4 & & & +4d \leq \alpha \\
 & \vdots & \vdots & \vdots & & \vdots \\
 & 2x_1 + & 2x_2 + & \cdots + 2x_{i-2} + (3-\tilde{c})x_{i-1} + 2x_i + id \leq \alpha & & \\
 & \vdots & \vdots & \vdots & & \vdots \\
 & & & & & x_i \geq 0.
\end{array} \tag{ILP}$$

Infinite linear programs can be regarded as the limit of finite linear programs successively extended by variables and constraints. Consequently, the solution to an infinite linear program is an infinite sequence. The dual to an infinite linear program is defined analogously to the finite case. Demaine et al. (2006) establish duality results for certain types of infinite linear programs, in particular, weak duality:

Theorem 4.2.2 (Demaine et al. (2006)). *Let $c = (c_i)_{i\in\mathbb{N}}$ and $b = (b_j)_{j\in\mathbb{N}}$ be sequences of real numbers, let $A = (a_{ij})_{i,j\in\mathbb{N}}$ be a doubly indexed sequence of real numbers, and let $x = (x_i)_{i\in\mathbb{N}}$ and $y = (y_i)_{i\in\mathbb{N}}$ be sequences of real variables.*

Then, let $\{\max c^\intercal x, Ax \leq b, x \geq 0\}$ be an infinite linear program. Assume that the set $I := \{i \in \mathbb{N} \mid c_i \neq 0\}$ is finite, and that for $i \in \mathbb{N}$, all but finitely many a_{ij} have the same sign. Let x be a feasible solution for the infinite linear program and let y be a feasible dual solution. Then, $c^\intercal x \leq b^\intercal y$ and if $c^\intercal x = b^\intercal y$, x is optimal.

(ILP) is an infinite linear program and the set of nonzero coefficients of the objective function of (ILP) is finite since the objective function is given by α. Furthermore, for each constraint, there is exactly one negative coefficient, due to $\tilde{c} \geq 9$. Thus, the conditions of Theorem 4.2.2 are fulfilled for (ILP) and weak duality holds. Consider the dual multipliers

$$y_j = q^j, \quad \text{where } q = \frac{1 - \sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}{2}.$$

Now, multiplying the j-th constraint of (ILP) with y^j and adding up all constraints, yields

$$\begin{aligned} x_1 \left(2\sum_{j=1}^{\infty} y_j - (\tilde{c}-1)y_2\right) + x_2 \left(2\sum_{j=2}^{\infty} y_j - (\tilde{c}-1)y_3\right) + \cdots \\ + x_i \left(2\sum_{j=i}^{\infty} y_j - (\tilde{c}-1)y_{i+1}\right) + \cdots + d\sum_{j=1}^{\infty}(y_j \cdot j) \leq \alpha \sum_{j=1}^{\infty} y_j. \end{aligned} \tag{4.1}$$

Furthermore, a closer examination of the coefficient of x_i results in

$$\begin{aligned} 2\sum_{j=i}^{\infty} y_j - (\tilde{c}-1)y_{i+1} &= 2\sum_{j=i}^{\infty} q^j - (\tilde{c}-1)q^{i+1} \\ &= q^i \left(2\sum_{j=i}^{\infty} q^{j-i} - (\tilde{c}-1)q\right) \\ &\overset{(|q|<1)}{=} q^i \left(\frac{2}{1-q} - (\tilde{c}-1)q\right) \\ &= q^i \left(\frac{2}{1-\frac{1-\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}{2}} - (\tilde{c}-1)\frac{1-\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}{2}\right) \\ &= q^i \left(\frac{4}{1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}} - \frac{(\tilde{c}-1)\left(1-\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)}{2}\right) \\ &= q^i \left(\frac{8-(\tilde{c}-1)\left(1-\frac{\tilde{c}-9}{\tilde{c}-1}\right)}{2\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)}\right) \\ &= q^i \left(\frac{8-\tilde{c}+1+\tilde{c}-9}{2\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)}\right) \\ &= 0. \end{aligned}$$

Thus, the coefficients of the x_i converge to 0 and (4.1) yields

$$\begin{aligned} & d\sum_{j=1}^{\infty}(y_j \cdot j) \leq \alpha \sum_{j=1}^{\infty} y_j \\ \Leftrightarrow \quad & \frac{d\sum_{j=1}^{\infty}(y_j \cdot j)}{\sum_{j=1}^{\infty} y_j} \leq \alpha. \end{aligned}$$

Therefore,

$$\alpha \geq \frac{d\sum\limits_{j=1}^{\infty}(y_j \cdot j)}{\sum\limits_{j=1}^{\infty} y_j} \overset{(|q|<1)}{=} \frac{d\frac{q}{(1-q)^2}}{\frac{q}{1-q}} = \frac{d}{1-q} = \frac{d}{1-\frac{1-\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}{2}} = \frac{2d}{1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}},$$

which gives us a lower bound for α. It remains to prove that $2d/\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)$ is also an upper bound for α. To this end, consider the feasible solution sequence x_i for (ILP) with

$$x_i = \frac{d\left(p^i-1\right)}{2}, \quad \text{where } p = \frac{2}{1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}.$$

Thereby, the i-th constraint, $i > 1$, yields

$$\begin{aligned}
& \sum_{j=1}^{i} 2x_j - (\tilde{c}-1)x_{i-1} + id \leq \alpha \\
\Leftrightarrow\quad & d\sum_{j=1}^{i}\left(p^j-1\right) - \frac{d(\tilde{c}-1)\left(p^{i-1}-1\right)}{2} + id \leq \alpha \\
\Leftrightarrow\quad & \sum_{j=1}^{i} p^j - \frac{(\tilde{c}-1)\left(p^{i-1}-1\right)}{2} \leq \frac{\alpha}{d} \\
\Leftrightarrow\quad & \frac{p\left(p^i-1\right)}{p-1} - \frac{(\tilde{c}-1)\,p^{i-1}}{2} + \frac{\tilde{c}-1}{2} \leq \frac{\alpha}{d} \\
\Leftrightarrow\quad & \frac{p^{i+1}}{p-1} - \frac{p}{p-1} - \frac{(\tilde{c}-1)\,p^{i-1}}{2} + \frac{\tilde{c}-1}{2} \leq \frac{\alpha}{d}.
\end{aligned} \tag{4.2}$$

The first and the third term of (4.2) vanish due to

$$\begin{aligned}
& \frac{p^{i+1}}{p-1} - \frac{(\tilde{c}-1)\,p^{i-1}}{2} \\
&= \frac{p^{i-1}}{p-1}\left(p^2 - \frac{(\tilde{c}-1)(p-1)}{2}\right) \\
&= \frac{p^{i-1}}{p-1}\left(\frac{4}{\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)^2} - \frac{\tilde{c}-1}{1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}} + \frac{\tilde{c}-1}{2}\right) \\
&= \frac{p^{i-1}}{2(p-1)\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)^2}\left(\underbrace{8 - 2(\tilde{c}-1)\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right) + (\tilde{c}-1)\left(1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}\right)^2}_{=0}\right) \\
&= 0.
\end{aligned}$$

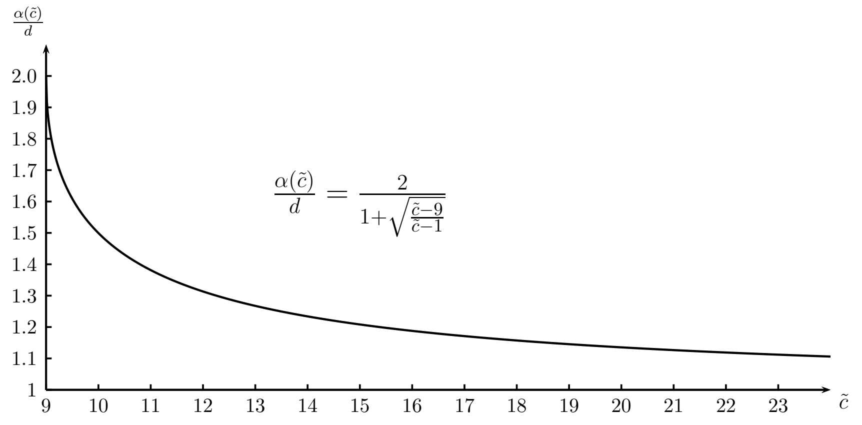


Figure 4.2: Tradeoff between competitive factor and additive term.

The second and the fourth term of (4.2) collapse to p, since

$$\begin{aligned}\frac{\tilde{c}-1}{2} - \frac{p}{p-1} &= \frac{(p-1)(\tilde{c}-1)-2p}{2(p-1)} \\ &= \frac{p - \frac{(p-1)(c-1)}{2}}{1-p} \\ &= \underbrace{\frac{p^2 - \frac{(p-1)(\tilde{c}-1)}{2}}{1-p}}_{=0} + \frac{p-p^2}{1-p} \\ &= p,\end{aligned}$$

where $p^2 - \frac{(p-1)(\tilde{c}-1)}{2}$ vanishes as shown in the previous calculation. Therefore, for $i > 1$, the i-th constraint results in $\alpha \geq dp$, and the first constraint is given by

$$2x_1 + d \leq \alpha \Leftrightarrow dp \leq \alpha.$$

Thus, we have the trivial linear program $\min\{\alpha \mid \alpha \geq pd\}$ with optimal objective value

$$\alpha = pd = \frac{2d}{1+\sqrt{\frac{\tilde{c}-9}{\tilde{c}-1}}}.$$

Since the objective value matches the lower bound calculated above, the solution sequence x_i is indeed an optimal solution to (ILP). $\square$

The analysis above gives us the minimal additive term $\alpha(\tilde{c})$ for an arbitrary competitive factor $\tilde{c} \geq 9$ and the corresponding search strategy x_i. For $\tilde{c} \geq 9$, the curve $\frac{\alpha(\tilde{c})}{d}$ is given in Figure 4.2. For $\tilde{c} = 9$, we have $\alpha(\tilde{c}) = 2$, as in the analysis of Demaine et al. (2006). For $\tilde{c} \to \infty$, we have $\alpha(\tilde{c}) \to d$: in order to find the object with only one turn, the searcher has to walk infinitely long into one direction before making the turn.

4.3 Conclusion

In this chapter, we gave an analytical characterization of the tradeoff curve between the competitive ratio and the additive term for the linear search problem with turn costs (note that the result matches the experimental characterization given in (Demaine et al., 2006)). Furthermore, the optimal search strategy corresponding to a competitive ratio $\tilde{c} \geq 9$ is given. Therefore, we now have a complete characterization of the linear search problem with turn costs.

5 Optimization in the Wood Cutting Industry

5.1 Introduction

The problem considered in this chapter is a real-world application from the veneer cutting industry, which we were introduced to by Fritz Becker KG, a manufacturer of shaped wood components from Northern Germany, who also provided us with real-world data. In the application problem, tree trunks are peeled into thin veneer strips which are cut, glued together and pressed into bentwood pieces for seats, backrests, armrests, chair legs, etc. The production process of these veneers is to be optimized with respect to a minimal wood offcut.

Currently, the production process is planned manually. On the one hand, this enables the planner to utilize his experience and certain rules of thumb, especially with respect to the wood quality, which is an important aspect of the problem. On the other hand, with an increasing number of orders, the problem becomes hardly comprehensible and understandable and, consequently, optimization tools have the potential to increase the quality of the production process significantly, especially with respect to long term planning periods. In the first part of this chapter, we develop a model for the problem at hand that both computes an optimal solution in reasonable time and incorporates all restrictions and features of the production process.

In the second part of this chapter, we have a closer look at the inherent uncertainties of the problem attributable to the quality of the wood. Before production, the quality of the wood is not known with certainty, but can only be estimated. In general, dealing with uncertainties in optimization problems is an important issue as disturbances or fluctuations in the problem formulation might significantly change the value of a formerly optimal solution. In fact, formerly feasible solutions may even become infeasible. With respect to the considered problem, the quality of the used wood is subject to fluctuations and it is only possible to determine the quality during the production process itself, which makes it necessary to take these uncertainties into account already during the planning of the production process. Due to the complex structure of the problem and for the sake of practicability, we depart from the concept of online optimization in this chapter.

Different ways of dealing with uncertain input data are commonly known throughout the literature such as stochastic optimization (for an overview, we refer to (Birge and Louveaux, 2011)). While stochastic optimization assumes some kind of probability distribution for the realizations of the uncertain parameters, the problem at hand calls for a different approach since the manufacturer could not provide any probabilistic information about the quality distribution of the wood. Therefore, we decided to approach the problem via robust optimization where we do not need such probabilistic information. The aim of robust optimization is to find solutions which remain feasible and of good quality in all scenarios, whereby a scenario is a realization of the uncertain input data.

For single-objective optimization problems several definitions of robustness, i.e., when a solution is seen as robust against uncertainties, have been analyzed thoroughly. One of these concepts is the concept of minmax robustness, introduced by Soyster (1973) and extensively researched by Ben-Tal and Nemirovski (1998, 1999); Ben-Tal et al. (2009). Here, a solution is called robust if it is feasible for every scenario and minimizes the objective function in the worst case. Very close to this concept is the concept of regret robustness, suggested, for example, by Kouvelis and Yu (1997), where the worst case regret is to be minimized and the solution has to be feasible in every scenario. Both of these concepts are quite strict with respect to the requirement that a solution has to be feasible in every scenario. To loosen this strict requirement, several other concepts have been proposed, such as the concept of light robustness (see, for example, (Fischetti and Monaci, 2009; Schöbel, 2014)) or the concept of recovery robustness (see, for example, (Liebchen et al., 2009; Erera et al., 2009; Goerigk and Schöbel, 2011)). Since the manufacturer's goal is to hedge against the worst case, we will follow the concept of minmax robustness throughout this chapter.

In applications of mathematical optimization and especially in the application presented in this chapter, there is often more than just one objective to consider. Therefore, we have to deal with uncertain multi-objective optimization for which several definitions of robustness have been presented in the literature, see for instance (Branke, 1998; Deb and Gupta, 2006).

Since we would like to hedge against the *worst case*, we follow the concept of minmax robust efficiency for multi-objective optimization problems. This concept is an extension of the concept of minmax robustness for single-objective optimization problems and has been presented by Ehrgott et al. (2014). Since in multi-objective optimization the term worst case is not that clear, as there is no total order on $\mathbb{R}^k$, the authors replace the worst case with a multi-objective maximization problem and define a dominance relation between the resulting sets, namely a set dominates another if it is completely contained in the other set minus the positive orthant of $\mathbb{R}^k$.

The rest of this chapter is organized as follows: In Section 5.2, the problem is presented explicitly and classified with respect to cutting stock problems. Then, in Section 5.3, the real-world cutting problem is modeled as a deterministic single-objective optimization problem and results for instances with practical relevance are presented. Then, we apply the concept of minmax robust efficiency to the cutting problem in Section 5.4. After clarifying the notation for uncertain multi-objective optimization and recalling the concept of minmax robust efficiency in Section 5.4.1, we simplify the application problem in Section 5.4.2 in order to be able to apply the concept of minmax robust efficiency properly in Section 5.4.3. We discuss the value of the minmax robust efficient solutions in practice and, finally, give concluding remarks and an outlook to future research in Section 5.5.

5.2 The Cutting Problem

In the following, the cutting problem is described in more detail. The manufacturer receives N orders $\{1, \ldots, N\}$ from different customers. Each order i is characterized by length l_i, width w_i, thickness t_i, and quality q_i of the requested veneer pieces which have to be cut from the veneer strip that is peeled off the tree trunk. The length dimension of a veneer piece corresponds to the length of the tree trunk and the width dimension of a veneer piece corresponds to the footage that is peeled off the tree trunk. The (previously fitted) length of a tree trunk ranges from 800mm to 2000mm and, depending on the diameter of the tree trunk, up to 150 meters of veneer strip can be peeled off the tree trunk. Length, width, and thickness are given in millimeters, the quality ranges from very bad (quality 10) to very good (quality 1). For the quality, certain wood characteristics such as wood color, vain, or knotholes are taken into account. The quality of the veneer strip can be determined by the veneer peeling machine during the cutting process.

Furthermore, each order specifies the number of required pieces n_i and a deadline d_i. The main goal of the manufacturer is to find cutting patterns on each day that fulfill all orders with deadline at this day and minimize the wood offcut. Further objectives are described in Section 5.3.2.

Consequently, we are dealing with a multi-objective optimization problem. The manufacturer provided preferences for the different objectives, from which we were able to deduce reasonable weights. Therefore, we applied the concept of weighted-sum scalarization and obtained a single-objective optimization problem as described in Section 5.3. In contrast, in Section 5.4.2, a multi-objective simplified version of the cutting problem is considered.

For a better understanding of the problem and the notion of a cutting pattern, the production process is described in the following: Depending on the orders, the tree trunks are cut to lengths and the bark is removed. Subsequently, the tree trunk is peeled into a thin veneer strip which is then cut down to the required veneer pieces. This process is visualized in Figure 5.1. In this phase, only cuts in the width dimension can be made. Thus, the length of the pieces cannot be changed. If necessary, the length is manually cut to size in an additional working step. In order to plan the production process, it has to be decided which tree trunk lengths are used and how the veneer strips are cut down to pieces, both in the width and the length dimension. Each length corresponds to a cutting scheme and due to certain characteristics of the production process, the number of different cutting schemes per day is limited to $C_{\max}$. Furthermore, there is a limit of $P_{\max}$ on the volume of wood in cubic meters that can be processed on each day. The difficulty of the problem is further increased by the following two aspects:

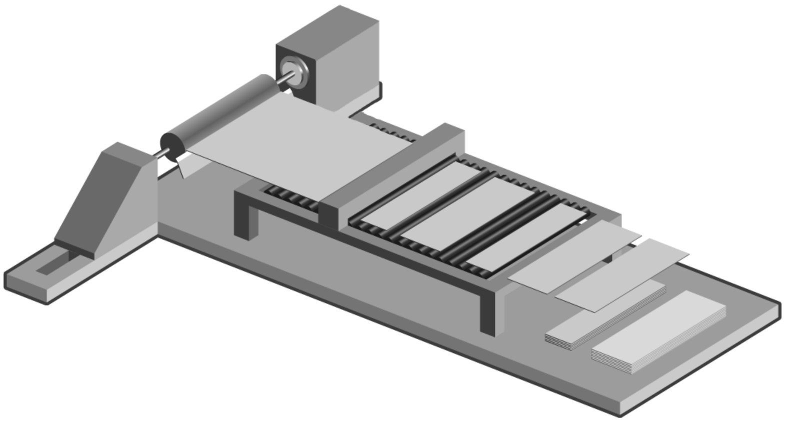


Figure 5.1: The tree trunk is peeled into a thin veneer strip and cut into veneer pieces. © Becker KG

First of all, the quality of the wood is uncertain. Wood of European beech (bot. Fagus sylvatica), which is commonly used by the manufacturer, has several characteristics which have to be taken into account when producing rotary cut veneer. These characteristics have an influence on optical and mechanical properties such as color and bending strength. In our case, the most important characteristics are heartwood and knottiness, which will be briefly explained in the following.

Red heartwood labels the appearance of red colored wood in the center of the beech tree trunk. Unlike other species, such as oak, where all trees have heartwood, not all beech trees have a heartwood area. If they do, this area is colored red to reddish-brown. Another difference to, for example, oak wood, is that heartwood of beech wood does not have other mechanical properties like strength. There have been some investigations (see, for example. (Ràcz, 1961)) but not much can be said about the likelihood of the appearance of heartwood in beech. It can be said though that heartwood appears more often with increasing age and diameter at breast height (cf. (Lohmann, 2003)).

Knottiness describes the frequency of occurrence of knots in round wood and timber. Apart from the frequency, form and type of knots is important. A forest tree can be divided into three parts: the lower part of the tree which is more or less free of external branches (ground tree trunk), the middle part with thick dead branches and branch stubs, and the top end of the tree trunk with living branches. For the production of veneer mostly the ground tree trunk is used. Even though it is free of external branches, it contains inner knots. During the growing of the tree it builds branches which are dying and falling off very early. Knotless wood is then growing over these branch stubs, so that the ground tree trunk wood is more or less free of knots except for its inner part. The number and size of these knots cannot be estimated easily before cutting the tree. Beech trees have so-called "Chinese beards", oval marks on the bark, which allow making rough estimations about inner knots (cf. (Lohmann, 2003)).

Due to these problems in determining the quality of the wood, the distribution of the different qualities can only be estimated and is therefore an uncertain parameter in

	orders					cutting pattern				
order	l_i	w_i	t_i	q_i	n_i	l_i	w_i	t_i	q_i	n_i
1	400	390	1.5	2	50	800	390	1.5	1,2	25
2	400	470	1.5	4	200	800	470	1.5	3,4	100
3	800	490	1.1	5	200	800	490	1.1	5	200
						800	450	1.1	> 5	400

Table 5.1: Orders and a cutting pattern for length 800mm.

the problem formulation. In Section 5.3, a distribution of the wood quality provided by the manufacturer based on historical values is used to model the uncertainties. In Section 5.4, we take into account the uncertainty of this quality distribution and, due to the requirements of the manufacturer, the concept of robust optimization is chosen to be applied to a simplified version of the problem. Furthermore, orders can always be satisfied with veneer pieces of a higher quality than requested. Obviously, satisfying orders with higher qualities than requested is a loss of profit for the manufacturer and is, therefore, to be minimized.

Secondly, the manufacturer has the possibility to manually cut down the veneer pieces to the appropriate size. For example, there is an order for 100 pieces of length 800mm and width 350mm and another order for 200 pieces with length 300mm and width 390mm. The manufacturer could schedule to cut 200 pieces of length 800mm and width 390mm, and cut down the width from 390mm to 300mm for the first order and the length from 800mm to two pieces with length 300mm for the second order. Of course, the manual cutting incurs additional working time, therefore the number of manually cut pieces is limited for each day.

In Table 5.1, three exemplary orders and an appropriate cutting pattern are given. Note that, due to the distribution of the wood quality in the tree trunk, certain amounts of each quality have to be used. Therefore, there is an additional line in the cutting pattern, covering all qualities worse than five: the first three lines are sufficient to meet the three orders on the left, but, in order to deal with qualities worse than five, which are inevitably peeled off the tree trunk as well, the last line is added. Furthermore, qualities one and two are used for order one, and qualities three and four are used for order two. Finally, orders one and two are cut with length 800mm and then manually cut down to 400mm. Therefore, only half the number of pieces of orders one and two is needed.

This example is supposed to give a brief glimpse of the planning problem of the manufacturer. The real-world problem is comprised of several hundreds of available orders with up to 50 different lengths and widths, 4 thicknesses and 10 qualities.

5.2.1 Classification of the Cutting Problem

The problem presented in this work is basically a cutting stock problem. The ordered veneer pieces are characterized by length and width, leading to a two-dimensional problem.

Furthermore, all ordered veneer pieces have to be assigned to a selection of tree trunks of different lengths and thicknesses. Finally, there are several different tree trunks and many ordered veneer pieces of many different shapes. Therefore, according to Dyckhoff's typology for cutting and packing problems (Dyckhoff, 1990), the problem is classified as 2/V/D/M. But, in addition, we have to deal with heterogeneous tree trunks, i.e., each tree trunk consists of different qualities. Furthermore, deadlines add a temporal dimension to the problem and certain production restrictions, such as a limited production for each day, increase the difficulty of the problem.

According to the classification of cutting and packing problems by Wäscher et al. (2007), we are faced with a two-dimensional input minimization problem where all ordered veneer pieces (strongly heterogeneous) have to be accommodated to several tree trunks for which one dimension is considered as a variable, resulting in an open dimension problem. Still, we have to deal with the aforementioned additional aspects of the problem, making it significantly more difficult. To the best of our knowledge, the cutting problem as described above is not discussed in the literature. The basic cutting stock problem on a strip, which is the core of our problem, is for example considered by Benati (1997); Zhiping et al. (1997).

5.3 Modeling the Deterministic Cutting Problem

Consider N orders $\{1,\dots,N\}$ characterized by length l_i, width w_i, thickness t_i, quality q_i, the number of required pieces n_i, and a deadline d_i. The goal is to determine cutting patterns for the next n^d (working) days, starting from day d', minimizing the wood offcut. We assume that all orders i are due on day d' or later, i.e., $d_i \geq d'$. Furthermore, the deadline of all orders with deadline later than day $d'+n^d$ will be set to day $d'+n^d$. The primary goal is to satisfy all orders i with deadline $d_i < d' + n^d$ before their deadline. All other orders with deadline $d_i = d' + n^d$ can be used to minimize the wood offcut and to increase the workload of the production.

The sets of all lengths $\mathcal{L}$, widths $\mathcal{W}$, thicknesses $\mathcal{T}$, qualities $\mathcal{Q}$, and relevant days $\mathcal{D}$ are given by $\mathcal{L} = \{l_1,\dots,l_N\}$, $\mathcal{W} = \{w_1,\dots,w_N\}$, $\mathcal{T} = \{t_1,\dots,t_N\}$, $\mathcal{Q} = \{q_1,\dots,q_N\}$, and $\mathcal{D} = \left[d', d' + n^d\right] \cap \mathbb{N}$. Furthermore, the last day, i.e., $d'+n^d$, is not to be planned, so we define the set of working days $\mathcal{D}^w$ as $\mathcal{D}^w = \mathcal{D} \setminus \left\{d' + n^d\right\}$.

5.3.1 Variables and Constraints

First of all, binary variables

$$z_{l,w,t,q,d^1,d^2} \in \{0,1\} \quad \text{for all } l \in \mathcal{L},\ w \in \mathcal{W},\ t \in \mathcal{T},\ q \in \mathcal{Q},$$
$$d^1, d^2 \in \mathcal{D}^w,\ d^2 \geq d^1,$$

representing the fulfillment of orders are introduced. If z_{l,w,t,q,d^1,d^2} equals one, order i with $l_i = l$, $w_i = w$, $t_i = t$, $q_i = q$, $d_i = d^1$ is completed on day d^2. For $d^2 = d^1$, the order is fulfilled before or at the deadline (an earlier fulfillment does not grant any benefits for the manufacturer), for $d^2 > d^1$, the order is fulfilled after the deadline. We require each order to be fulfilled at most once, i.e., for all $l \in \mathcal{L},\ w \in \mathcal{W},\ t \in \mathcal{T},\ q \in \mathcal{Q}$, and $d^1 \in \mathcal{D}^w$ we have

$$\sum_{d^2 \geq d^1} z_{l,w,t,q,d^1,d^2} \leq 1. \tag{5.1}$$

If an order is not accomplished at all or after its deadline, corresponding penalty terms are added to the objective function as described in Section 5.3.2. We choose to model the fulfillment of orders as soft constraints in order to be able to optimize sets of orders that are not all simultaneously satisfiable within their deadlines due to the limited production capacity.
Further, we introduce variables

$$x_{l,w,t,q,d} \in \mathbb{N} \quad \text{for all } l \in \mathcal{L},\ w \in \mathcal{W},\ t \in \mathcal{T},\ q \in \mathcal{Q},\ d \in \mathcal{D}^w,$$

representing the number of veneer pieces with length l, width w, thickness t, and quality q produced on day d without an additional manual cutting step.

For the variables $x_{l,w,t,q,d}$, the quality distribution of the wood is required. Denote by $p_{l,q,t}$ the probability of quality q for a veneer strip with length l and thickness t. Note that the distribution depends on the length and the thickness. The manufacturer provided us with the necessary estimates for the distributions. Now, we introduce the variables

$$y_{l,t,d} \in \mathbb{R} \quad \text{for all } l \in \mathcal{L},\ t \in \mathcal{T},\ d \in \mathcal{D}^w,$$

representing the total width cut from length l with thickness t on day d, and demand

$$\sum_{w \in \mathcal{W}} w \cdot x_{l,w,t,q,d} \leq y_{l,t,d} \cdot p_{l,q,t}. \tag{5.2}$$

The veneer pieces produced on day d without an additional manual cutting step represented by $x_{l,w,t,q,d}$ are either correctly sized and in the right quality, or will manually be cut down or used for an order with lower quality, respectively. In order to model this situation, we introduce transfer variables

$$\begin{aligned}\tau_{l^1,q^1,w,t,d}^{l^2,q^2} \in \mathbb{R} \quad & \text{for all } l^1 \in \mathcal{L},\ q^1 \in \mathcal{Q},\ w \in \mathcal{W},\ t \in \mathcal{T},\ d \in \mathcal{D}^w,\\ & l^1 \geq l^2,\ q^1 \leq q^2, \text{ and } (l^2, w, t, q^2) \in \mathcal{S},\end{aligned}$$

for the number of pieces that are manually cut down from length l^1 to length l^2 with $l^2 \leq l^1$ or used for an order of lower quality, i.e., $q^1 \leq q^2$. Additionally, $\mathcal{S}$ is the set of all stacks, i.e., the combinations of length, width, thickness, and quality derived from the orders $\{1, \ldots, N\}$. Note that we use only the combinations of length, width, thickness, and quality that appear in the original orders $\{1, \ldots, N\}$ for potential stacks.

On the first day d', the newly cut pieces $x_{l,w,t,q,d}$ are distributed among all possible transfer variables, i.e., for all $l^1 \in \mathcal{L}$ such that $l^1 \geq l_{\min}$, $w \in \mathcal{W},\ t \in \mathcal{T}$, and $q^1 \in \mathcal{Q}$, we have

$$\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau_{l^1,q^1,w,t,d'}^{l^2,q^2} = x_{l^1,w,t,q^1,d'}, \tag{5.3}$$

where $l_{\min}$ is given by the minimal tree trunk length the veneer peeling machine can process. All orders with a length smaller than $l_{\min}$ have to be cut down manually and, therefore, for all $l^1 \in \mathcal{L}$ such that $l^1 < l_{\min}$, $w \in \mathcal{W}$, $t \in \mathcal{T}$, and $q^1 \in \mathcal{Q}$, we have

$$\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^2,q^2}_{l^1,q^1,w,t,d'} = 0. \tag{5.4}$$

Imagine stacks for each configuration (length, width, thickness, quality) from which the orders have to be satisfied. The stacks change over time, since new pieces are produced and added to the corresponding stack and some pieces are manually cut down and therefore change their stack (see also Figure 5.2). The transfer variables model this *stack-transfer-concept* which is the basis for a compact model for our complex problem.

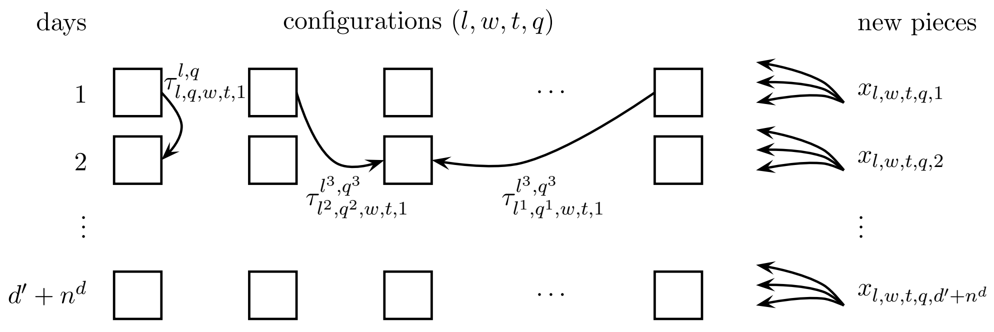


Figure 5.2: Stack-transfer-concept.

On the following days, the stack for a certain configuration (the left hand side of (5.5)) is composed of the stack for that configuration of the previous day and the newly produced pieces for that configuration reduced by the satisfied orders (the right hand side of (5.5)), i.e., for all $l^1 \in \mathcal{L}$ such that $l^1 \geq l_{\min}$, $w \in \mathcal{W}$, $t \in \mathcal{T}$, and $q^1 \in \mathcal{Q}$, we have

$$\begin{aligned}\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^2,q^2}_{l^1,q^1,w,t,d} = &\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^1,q^1}_{l^2,q^2,w,t,d-1} \cdot \left\lfloor \frac{l^2}{l^1} \right\rfloor \\ &- \sum_{i \in I} n_i \cdot z_{l,w,t,q,d_i,d-1} + x_{l,w,t,q,d},\end{aligned} \tag{5.5}$$

where

$$I = \{i \in \{1, \dots, N\} \mid l_i = l,\ w_i = w,\ t_i = t,\ q_i = q,\ d_i < d - 1\}.$$

Again, for lengths smaller than the minimal length $l_{\min}$, (5.5) has to be altered slightly, i.e., for all $l^1 \in \mathcal{L}$ such that $l^1 < l_{\min}$, $w \in \mathcal{W}$, $t \in \mathcal{T}$, and $q^1 \in \mathcal{Q}$, we have

$$\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^2,q^2}_{l^1,q^1,w,t,d} = \sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^1,q^1}_{l^2,q^2,w,t,d-1} \cdot \left\lfloor \frac{l^2}{l^1} \right\rfloor - \sum_{i \in I} n_i \cdot z_{l,w,t,q,d_i,d-1}. \tag{5.6}$$

Furthermore, it is possible to produce beyond the number of ordered pieces and store these pieces in a warehouse, but the manufacturer only wants to store a limited amount of frequently ordered combinations of length, width, thickness and quality. This set of combinations of length, width, thickness and quality is specified by the manufacturer and denoted by $\mathbb{W}$. Additionally, the manufacturer specified the corresponding limits on the amount of stored pieces, which is denoted by $n_{(l,w,t,q)}$, for all $(l,w,t,q) \in \mathbb{W}$. We consider the storage of certain kinds of veneer pieces up to a limited amount as additional orders with the corresponding length, width, thickness, and quality and include them in the set of orders. The deadline is set to $d' + n^d$ and the upper limits on the amount of stored pieces is considered as the number of required pieces for these *storage orders.*

On the last production day, i.e., $d' + n^d - 1$, we have to make sure that all produced pieces can be assigned to any of the regular orders or orders with deadlines set to $d' + n^d$. Thus, for all $j \in \{1, \dots, N\}$ such that $l_j = l,\ w_j = w,\ t_j = t,\ q_j = q,\ d_j = d' + n^d$, we have

$$\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l,q}_{l^2,q^2,w,t,d'+n^d-1} \cdot \left\lfloor \frac{l^2}{l} \right\rfloor - \sum_{i \in I} n_i \cdot z_{l,w,t,q,d_i,d'+n^d-1} \leq n_j, \tag{5.7}$$

where

$$I = \Big\{ i \in \{1, \dots, N\} \mid l_i = l,\ w_i = w,\ t_i = t,\ q_i = q,\ d_i \leq d' + n^d - 1 \Big\}.$$

The production of regular orders is supposed to take priority over the production of storage orders. Therefore, we introduce variables $s_{(l,w,t,q)}$, for all $(l,w,t,q) \in \mathbb{W}$, and require, for all $(l,w,t,q) \in \mathbb{W}$,

$$\begin{aligned} s_{(l,w,t,q)} \geq & \sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l,q}_{l^2,q^2,w,t,d'+n^d-1} \cdot \left\lfloor \frac{l^2}{l} \right\rfloor - \sum_{i \in I} n_i \cdot z_{l,w,t,q,d_i,d'+n^d-1} \\ & - \left(n_j - n_{(l,w,t,q)} \right), \end{aligned} \tag{5.8}$$

where j is such that $l_j = l,\ w_j = w,\ t_j = t,\ q_j = q,\ d_j = d' + n^d - 1$. The variables $s_{(l,w,t,q)}$ represent the number of pieces produced for storage orders and will be penalized in the objective function, see Section 5.3.2.

Finally, we make sure that all orders are fulfilled by means of the binary variables z_{l,w,t,q,d^1,d^2}. The stack for a certain configuration on some day d^2 has to be large enough in order to fulfill the order for that configuration, otherwise the variable z_{l,w,t,q,d^1,d^2} has to be set to zero:

$$\sum_{l^2 \in \mathcal{L}, q^2 \in \mathcal{Q}} \tau^{l^1,q^1}_{l^2,q^2,w,t,d} \cdot \left\lfloor \frac{l^2}{l^1} \right\rfloor \geq \sum_{i \in I} n_i \cdot z_{l,w,t,q,d_j,d}. \tag{5.9}$$

If z_{l,w,t,q,d^1,d^2} is set to zero, a penalty is included in the objective function, see Section 5.3.2.

Furthermore, certain restrictions with respect to the production capacity have to be considered. On each day, at most $P_{\max}$ cubic meters may be processed and the manufacturer aims for a production at full capacity. Consequently, we introduce the variable $g_d \in \mathbb{R}$ for all $d \in \mathcal{D}^w$ for the gap between the actual production and the capacity and have, for all $d \in \mathcal{D}^w$,

$$g_d + \sum_{l \in \mathcal{L}, t \in \mathcal{T}} y_{l,t,d} \cdot l \cdot t = P_{\max}. \tag{5.10}$$

In order to ensure a minimal production $P_{\min}$ on each day, we have, for all $d \in \mathcal{D}^w$,

$$g_d \leq P_{\max} - P_{\min}. \tag{5.11}$$

Additionally, on each day $d \in \mathcal{D}^w$, the number of used lengths, i.e., the number of cutting patterns, is limited by $C_{\max}$. In order to model this situation, we introduce binary variables $c_{l,d} \in \{0,1\}$, for all $l \in \mathcal{L},\ d \in \mathcal{D}^w$, taking value one if length l is cut on day d, and zero otherwise. For all $l \in \mathcal{L},\ d \in \mathcal{D}^w$, we model $c_{l,d}$ by

$$\sum_{t \in \mathcal{T}} y_{l,t,d} \leq c_{l,d} \cdot M, \tag{5.12}$$

where M is given by $M = {}^{P_{\max}}\!/\!\left(\min\limits_{l \in \mathcal{L}} l \cdot \min\limits_{t \in \mathcal{T}} t\right)$, i.e., the maximum width cut for length l, and restrict the number of cutting patterns for each day $d \in \mathcal{D}^w$ by

$$\sum_{l \in \mathcal{L}} c_{l,d} \leq C_{\max}. \tag{5.13}$$

It remains to bound the number of manually cut down pieces by $\mathcal{T}_{\max}$ in order to comply with the production capacity. We therefore define

$$\widetilde{L} := \left\{ \left(l^1 \in \mathcal{L},\ w \in \mathcal{W},\ t \in \mathcal{T},\ q^1 \in \mathcal{Q},\ l^2 \in \mathcal{L},\ q^2 \in \mathcal{Q}\right) \mid l^1 \neq l^2 \right\}.$$

Then, for all $d \in \mathcal{D}^w$, we require

$$\sum_{\widetilde{L}} \left(\tau^{l^2,q^2}_{l^1,w,t,q^1,d} \cdot \left(\left\lceil \frac{l^1}{l^2} \right\rceil - 1 \right) \right) \leq \mathcal{T}_{\max}. \tag{5.14}$$

This completes the description of the constraints of the cutting problem. In the next section the objectives of the cutting problem are discussed.

5.3.2 Objectives

The objective for the model is comprised of several aspects such as the minimization of the wood offcut in cubic meters denoted by c_1, the total number of delayed or unfulfilled orders denoted by c_2, the use of high quality pieces for lower quality orders in cubic meters weighted by a penalty function denoted by c_3, the number of manually cut down pieces denoted by c_4, and the overproduction, i.e., produced veneer pieces that will be stored in the warehouse, in cubic meters denoted by c_5. In the following, these five aspects are described in more detail. First of all, the wood offcut is to be minimized. The total wood offcut c_1 is given by the total amount of processed wood minus the fulfilled orders given by

$$\sum_{l,t,d} y_{l,t,d} \cdot l \cdot s - \sum_{\substack{i \in \{1,\dots,N\} \\ \text{s.t. } d_i \in \mathcal{D}^w}} \left(n_i \cdot l \cdot w \cdot t \cdot \sum_{d^2 \geq d_i} z_{l,w,t,q,d_i,d^2} \right),$$

minus the veneer pieces that are cut on the last production day for orders with deadline $d' + n^d$ given by

$$\sum_{l^1,w,t,q^1} \left(\sum_{l^2,q^2} \left(\tau^{l^1,q^1}_{l^2,w,t,q^2,d'+n^d-1} \cdot \left\lfloor \frac{l^2}{l^1} \right\rfloor \right) - \sum_{i \in I_1} n_i \cdot z_{l^1,w,t,q^1,d_i,d'+n^d-1} \right) l \cdot w \cdot t$$

where

$$I_1 = \{ i \in \{1,\dots,N\} \mid l_i = l^1,\, w_i = w,\, t_i = t, q_i = q^1,\, d_i \leq d' + n^d - 1 \}.$$

Secondly, we want to minimize delayed or unfulfilled orders c_2. A delayed order is penalized proportionally to the number of days the order is late, and if the order is not fulfilled at all during the planned period, the penalty is $n^d + 1$. Thus, c_2 is given by

$$c_2 = \sum_{\substack{i \in \{1,\dots,N\} \\ \text{s.t. } d_i \in \mathcal{D}^w}} \left(\sum_{d^2 > d_i} \left((d^2 - d_i) \cdot z_{l,w,t,q,d_i,d^2} \right) \right.$$
$$\left. + \left(n^d + 1 \right) \cdot \left(1 - \sum_{\substack{d^2 \in \mathcal{D}^w \\ \text{s.t. } d^2 \geq d_i}} z_{l,w,t,q,d_i,d^2} \right) \right).$$

Thirdly, we want to minimize the use of high quality pieces for lower quality orders c_3, which is given by summing up the corresponding transfer variables,

$$c_3 = \sum_{\substack{l^1,l^2,q^1,q^2,w,t,d: \\ q^1 < q^2 \wedge l^1 \geq l^2}} \tau^{l^2,q^2}_{l^1,q^1,w,t,d} \cdot l^2 \cdot \left\lfloor \frac{l^1}{l^2} \right\rfloor \cdot w \cdot s \cdot \hat{p}(q^1,q^2),$$

where $\hat{p}(q^1,q^2)$ is a fixed penalty for using wood of quality q_1 for an order of quality q^2. Note that c_3 is given in cubic meters in order to interrelate this part of the objective function with part c_1.

The number of manually cut down pieces is also given by summing up the corresponding transfer variables,

$$c_4 = \sum_{\substack{l^1,l^2,q^1,q^2,d,w,t: \\ l^1 \neq l^2}} \left(\tau^{l^2,q^2}_{l^1,q^1,w,t,d} \cdot \left(\left\lceil \frac{l^1}{l^2} \right\rceil - 1 \right) \right).$$

Finally, the number of produced veneer pieces that will be stored in the warehouse is given by

$$c_5 = \sum_{(l,w,t,q) \in \mathbb{W}} s_{(l,w,t,q)} \cdot l \cdot w \cdot t.$$

The objective function is then given by

$$\min \sum_{i=1}^{5} \omega_i \cdot c_i,$$

where the weighs $\omega_i,\ i = 1, \ldots, 5$ are chosen with respect to the preferences of the manufacturer. In interaction with (5.1) - (5.14), the mixed integer programming formulation for the cutting problem is given.

5.3.3 Computational Results

The model for the cutting problem presented in Section 5.3 is implemented with FICO Xpress Mosel Version 3.4.0 and solved with the FICO Express Optimizer Version 23.01.05. The model is tested on three real-world instances provided by the manufacturer and all computations were performed on a PC with an Intel Core i3-2350M 2.30GHz, 6.00 GB RAM. The weights w_i, $i = 1, \ldots, 5$, of the objectives were chosen with respect to the requirements of the manufacturer as $w_1 = 1$, $w_2 = P_{\max}$, $w_3 = 0.1$, $w_4 = 0.0001$, and $w_5 = 0.5$. Note that by choosing w_2 in cubic meters, w_1c_1, w_2c_2, w_3c_3, and w_5c_5 become comparable. w_4 is chosen sufficiently small in order to obtain a lexicographic solution, minimizing w_4c_4 after minimizing the sum of the other four objectives.

For example, an instance which is typical for the problem of the manufacturer consists of 466 orders leading to 47 lengths, 37 widths, 4 thicknesses, and 9 qualities. For a single day, the model was solved to optimality in 52 seconds on average. Note that in the case of a single day still all available orders are considered, but cutting patterns are only generated for the first day. For two days, the model is solved to optimality in 355 seconds on average and for three days in 915 seconds on average. In practice, it is not reasonable to plan ahead more than at most three days, due to eventually necessary adjustments caused by the uncertainty in the wood quality. For 6 days the model is solved to optimality in 1297 seconds on average.

The optimization model fulfills all production requirements and is able to generate cutting patterns that consider the full set of available orders. This leads to an improvement over the manual planning for which the set of considered orders is limited to a certain extent. Due to a lack of data, a direct comparison of generated solutions to the manual process is not possible. The generated solutions feature approximately 10 % wood offcut, whereas, according to the manufacturer, the manual production process exhibits about 20 % wood offcut.

The wood offcut of the generated solutions can be further decreased by increasing the value of the corresponding multiplier w_1 in the objective function. However, a trade-off between the wood offcut (c_1) and the use of high quality pieces for lower quality orders (c_3) and the volume of produced veneer pieces that will be stored in the warehouse (c_5) can be observed, and, at some point, even the number of delayed or unfulfilled orders (c_2) increases if w_1 is increased, see Table 5.2. Note that, in this example, c_4 always obtains the upper bound of $\mathcal{T}_{\max} = 1000$.

w_1	c_1	c_2	c_3	c_4	c_5
0.1	3.12	0	30.12	1000	4.49
1	2.94	0	33.24	1000	4.44
10	2.11	0	50.97	1000	9.11
100	1.62	0	96.14	1000	14.92
1000	0.26	10	73.22	1000	16.48

Table 5.2: Sensitivity of the wood offcut c_1 to changes in w_1.

5.4 Robust Cutting Patterns

As mentioned in Section 5.2, it is difficult to formulate the cutting problem in a deterministic way because the distribution of qualities in the used wood is not known at the time the pattern is determined. Therefore, it is not possible to know how bad a pattern becomes if the quality distribution of the wood differs from the expected quality distribution. In order to handle these uncertainties we now apply the concept of robust optimization to the cutting problem. Since, in fact, the problem is a multi-objective one, which for computational purposes we modeled as a single-objective one, we will use multi-objective robust optimization, namely the concept of minmax robust efficiency hedging against the set of worst cases, introduced by Ehrgott et al. (2014). To this end, we first introduce the general idea of an uncertain multi-objective optimization problem and the concept of minmax robust efficiency, and apply this concept to a simplified version of the cutting problem.

5.4.1 Uncertain Multi-Objective Optimization

First of all, we need to define an uncertain multi-objective optimization problem. Here, our assumption is that the uncertain data is given by an *uncertainty set* $\mathcal{U} \in \mathbb{R}^m$, i.e., a set of scenarios $\xi \in \mathcal{U}$ representing the various possible realizations of the uncertain data. The definition of this uncertainty set $\mathcal{U}$ is a crucial step in formulating the model, as we will see in Section 5.4.2.

For every possible realization of the uncertain parameters, we obtain a single (deterministic) multi-objective optimization problem $\mathcal{P}(\xi)$, $\xi \in \mathcal{U}$, which we denote in the following way:

Notation 5.4.1. *Given a set of feasible solutions $\mathcal{X} \in \mathbb{R}^n$, a vector-valued objective function $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^k$, an uncertainty set $\mathcal{U}$, and a scenario $\xi \in \mathcal{U}$, we denote the* multi-objective optimization problem *of minimizing $f(x, \xi)$ over $\mathcal{X}$ by*

$$\begin{aligned} \mathcal{P}(\xi) \quad & \min \ f(x,\xi) \\ & s.t. \ x \in \mathcal{X}. \end{aligned}$$

Note that, due to the lack of a total order on $\mathbb{R}^k$, the minimization of a vector-valued objective function is dependent on the definition of dominance. Here, we use the definition which goes back to Pareto (1896) and has been extensively studied throughout the literature. For an overview see (Ehrgott, 2005). Here a solution $x \in \mathcal{X}$ is said to be *efficient* if there does not exist an $\overline{x} \in \mathcal{X}$ such that $f(\overline{x})$ is at least as good as $f(x)$ in every component and better in at least one component.

By means of Notation 5.4.1, we can now define an uncertain multi-objective optimization problem $\mathcal{P}(\mathcal{U})$:

Definition 5.4.1 (Uncertain multi-objective optimization problem)**.** *Given a set of feasible solutions $\mathcal{X} \in \mathbb{R}^n$, a vector-valued objective function $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^k$, and an uncertainty set $\mathcal{U}$, an* uncertain multi-objective optimization problem *$\mathcal{P}(\mathcal{U})$ is defined as the family of multi-objective optimization problems $(\mathcal{P}(\xi), \xi \in \mathcal{U})$.*

Since it is not clear, when to call a solution to the family $\mathcal{P}(\mathcal{U})$ of optimization problems a "desired" solution, we need an interpretation of a "desired" solution. As mentioned before, we follow the concept of minmax robust efficiency.

The concept of minmax robust efficiency we use throughout this section is an extension of the concept of minmax robustness, originally introduced by Soyster (1973) and extensively studied by Ben-Tal et al. (2009). This concept was originally designed for single-objective functions and follows the general idea that a solution is robust if it is feasible in every scenario and minimizes the worst case of the objective value under all scenarios. It has later been extended to multi-objective optimization problems by Ehrgott et al. (2014). Here, the following definition is presented:

Definition 5.4.2 (Minmax Robust Efficiency)**.** *Given an uncertain multi-objective optimization problem $\mathcal{P}(\mathcal{U})$, for every $x \in \mathcal{X}$ we define $f_{\mathcal{U}}(x) := \{f(x, \xi) : \xi \in \mathcal{U}\}$. A solution $x \in \mathcal{X}$ is called* minmax robust efficient *if there is no $\overline{x} \in \mathcal{X}$ such that*

$$f_{\mathcal{U}}(\overline{x}) \subseteq f_{\mathcal{U}}(x) - \mathbb{R}^k_{\geq}.$$

Here $\mathbb{R}^k_{\geq}$ is the (closed) positive orthant of $\mathbb{R}^k$, without the 0.

The intuition behind this concept is the following: The worst case of a single-objective optimization problem is in fact a maximization problem for a fixed feasible solution over the uncertainty set. Therefore, for an uncertain multi-objective problem, this "worst case" becomes a multi-objective maximization problem. This would yield a set of efficient solutions for this maximization problem and now, minmax robust efficiency calls a solution "robust" if its set of worst cases (i.e., the efficient solutions of the multi-objective maximization problem over $\mathcal{U}$) is not dominated by the set of worst cases of another solution. This is formally described by Definition 5.4.2.

Several algorithms for calculating minmax robust efficient solutions have been provided by Ehrgott et al. (2014). We will use some of them later on for obtaining the computational results.

5.4.2 A Simplified Version of the Multi-Objective Cutting Problem

In order to obtain computational results in reasonable time, we first simplify the deterministic formulation of the cutting problem. Furthermore, we design new (also simpler) objective functions in order to obtain a multi-objective optimization problem. This is motivated by the nature of the application problem, which is in fact a multi-objective one. We did not use the multi-objective approach before, because the manufacturer had specific ideas about the weighting of the different objective functions, but for research purposes, a more general approach is of greater value.

Simplified Deterministic Formulation.

The differences to the full size single-objective problem are the following:

Let orders $1, \ldots, N$ be given, each defined by length l_i, quality q_i, and number of items n_i. In contrast to the full size single-objective problem, width and thickness of the items are considered standardized for simplification of the program. For instance, an order i now could be 400 items of length 960mm in quality 4 (the qualities are given in classes from 1 to 10, 1 being the best quality). The output of the optimization program again is a cutting pattern defined by a length $l \in \mathcal{L}$ and a set of orders $\mathcal{I}$ with $l_i \leq l$, for all $i \in \mathcal{I}$. This problem can be formulated in the following way:

Input: A set of orders $\mathcal{I}$, every order i with length l_i, quality q_i, and number of items n_i, $\mathcal{L}$ and $\mathcal{Q}$ as before, and a distribution $p \in \mathbb{R}^{10\times|\mathcal{L}|}_{\geq 0}$ of the qualities in the veneer strip of length l ($\sum_{q=1}^{10} p_{q,l} = 1$ for every $l \in \mathcal{L}$). Note that, due to the simplification, orders do not feature a width and a thickness anymore, i.e., width and thickness are standardized.

Decision variables: $x \in \mathbb{Z}^{|\mathcal{I}|\times|\mathcal{L}|}$, where $x_{i,l}$ indicates how many pieces of order i are satisfied by the cutting pattern for length l.

$c \in \mathbb{B}^{|\mathcal{L}|}$, where c_l indicates whether there is a cutting pattern for length l or not.

Objective functions: Note that the length of the veneer strip produced by the stripping machine can be modeled with an additional decision variable $w \in \mathbb{R}^{|\mathcal{L}|}$, where w_l

indicates the width of the veneer strip cut in the cutting pattern of length l (measured in standard units which is the width of one veneer). With the additional constraints

$$\sum_{i \in \mathcal{I}: q_i \leq q} x_{i,l} \leq w_l \cdot \sum_{q' \in \mathcal{Q}: q' \leq q} p_{q',l} \quad \text{for all} \quad q \in \mathcal{Q},\ l \in \mathcal{L}, \tag{5.15}$$

w_l is modeled correctly. Since, in the objective function, w_l is to be minimized, (5.15) is equivalent to

$$w_l = \max_{q \in \mathcal{Q}} \left(\frac{\sum_{i \in \mathcal{I}: q_i \leq q} x_{i,l}}{\sum_{q' \in \mathcal{Q}: q' \leq q} p_{q',l}} \right), \tag{5.16}$$

for all $l \in \mathcal{L}$ and, therefore, we can drop (5.15) and use the explicit expression (5.16) for w_l in order to clarify that the uncertainty only lies in the objective functions.

Minimize wood offcut:

$$f_1(x,p) := \sum_{l \in \mathcal{L}} l \cdot \max_{q \in \mathcal{Q}} \left(\frac{\sum_{i \in \mathcal{I}: q_i \leq q} x_{i,l}}{\sum_{q' \in \mathcal{Q}: q' \leq q} p_{q',l}} \right) - \sum_{i \in \mathcal{I}} l_i \cdot n_i$$

The wood offcut of a cutting pattern can be determined by the length of the veneer strip the machine has to peel from the tree in order to get enough wood to satisfy the orders assigned to the current cutting pattern minus all produced veneers.

Minimize lost high quality wood:

$$f_2(x,p) := \sum_{l \in \mathcal{L}} \left(l \cdot \max_{q \in \mathcal{Q}} \left(\frac{\sum_{i \in \mathcal{I}: q_i \leq q} x_{i,l}}{\sum_{q' \in \mathcal{Q}: q' \leq q} p_{q',l}} \right) \cdot \sum_{q \in \{1,2,3\}} p_{q,l} \right) - \sum_{i \in \mathcal{I}: q_i \leq 3} l_i \cdot n_i$$

Qualities 1, 2, and 3 are of high value and should not be wasted or used to satisfy orders of lower quality.

Now, the objective function is given

$$\min \begin{pmatrix} f_1(x,p) \\ f_2(x,p) \end{pmatrix}.$$

Note that p is an input parameter and therefore constant.

Constraints: The following constraints have to be met:

$$\sum_{l \in \mathcal{L}} x_{i,l} = n_i \qquad \text{for all } \; i \in \mathcal{I} \tag{5.17}$$

$$x_{i,l} = 0 \qquad \text{for all } \; i \in \mathcal{I}, \; l < l_i \tag{5.18}$$

$$\sum_{l \in \mathcal{L}} c_l \leq 2 \tag{5.19}$$

$$\sum_{i \in \mathcal{I}} x_{i,l} \leq c_l \cdot \sum_{i \in \mathcal{I}} n_i \qquad \text{for all } \; l \in \mathcal{L} \tag{5.20}$$

$$x_{i,l} \in \mathbb{N} \qquad \text{for all } \; i \in \mathcal{I}, \; l \in \mathcal{L} \tag{5.21}$$

$$c_l \in \mathbb{B} \qquad \text{for all } \; l \in \mathcal{L} \tag{5.22}$$

(5.17) mean that all items are produced (where it is possible to cut a veneer from a higher length down to a smaller length). (5.18) mean that produced veneers can only be used for orders with a length that is smaller than or equal to the length of the produced veneer. (5.19) limits the total number of cutting patterns since changing the cutting length of the machine is very time consuming. (5.21) and (5.22) define the decision space of the variables.

5.4.3 Minmax Robust Efficiency Applied to the Optimiziation Model

Now, after formulating the simplified deterministic optimization problem, we apply the concept of minmax robust efficiency. Obviously, to do so, we first need to determine the uncertain parameters in the formulation.

Uncertainties in the Problem Formulation.

The uncertainties in the problem formulation are due to fluctuations of the qualities of the used wood. As the machine only uncovers the true quality of the veneer strip at the time of production, this quality distribution is unknown at the time of creating the cutting patterns. We now consider the distribution of the qualities to be uncertain, i.e., we work with an uncertainty set $\mathcal{U}_p$ in which each scenario represents another quality distribution. By means of this uncertainty set, we formulate the minmax robust version of the deterministic formulation of the simplified problem given in Section 5.4.2.

Formulating the Robust Version of the Problem.

Formulating the robust version of our problem from Section 5.4.2 is fairly simple:

Input: Instead of a single distribution p the input is the whole uncertainty set $\mathcal{U}_p$. The rest of the input remains unchanged.

Decision variables: The decision variables remain unchanged.

Objective functions: We re-formulate the objective function as proposed by Ehrgott et al. (2014) in the following way:

$$\min \max_{p \in \mathcal{U}} \begin{pmatrix} f_1(x,p) \\ f_2(x,p) \end{pmatrix}$$

Constraints: All constraints remain unchanged since they are not affected by the uncertain parameters.

Note that the solutions to this problem are the minmax robust efficient solutions as defined in Section 5.4.1. To this end, we use the techniques proposed by Ehrgott et al. (2014) in this section.

Modeling the Uncertainty Set.

Modeling the uncertainty set is a crucial point in the formulation of the uncertain multi-objective optimization problem. With too strict uncertainty sets the robust version can become arbitrarily bad. Thus, the modeling of the uncertainty set has to be done carefully.

The uncertainty set we use was developed together with Fritz Becker KG who provided the application problem using empirically established quality distributions. In Figure 5.3, we give a rough idea of what the quality distributions look like. Here, the quality distribution for length 390mm is given, meaning that, for example, in Scenario 1 20 % of the veneer strip will be of quality 5. The uncertainty sets for the other lengths look similar even though the graphs are shifted a little to the right since obtaining good qualities becomes more unlikely with increasing lengths of the tree trunk.

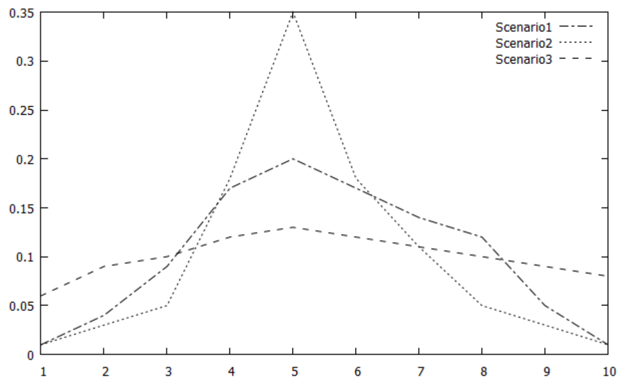


Figure 5.3: Quality distributions for length 390mm.

Computational Results.

Calculating minmax robust efficient solution is done using the weighted-sum scalarization method presented by Ehrgott et al. (2014). Here, both objective functions are weighted with a scalar $\lambda \in \mathbb{R}^2_+$ and added, and the worst-case of this sum is to be minimized:

$$\min \max_{p \in \mathcal{U}} \left(\lambda_1 \cdot f_1(x,p) + \lambda_2 \cdot f_2(x,p)\right), \tag{$\mathcal{WS}(\lambda)$}$$

such that x satisfies (5.17) - (5.22) from Section 5.4.2. This will yield minmax robust efficient solutions as defined in Section 5.4.1. We compare the results in the following way:

First, we choose weights $\lambda \in \mathbb{R}^2_+$. These weights represent a trade-off between the two objective functions. For instance, $\lambda = \big(1,\, 0.1\big)^{\mathsf{T}}$ represents a decision makers preference that saving 1m of wood offcut is worth loosing 10m of high quality wood. In order to get a first insight into the benefit of the different strategies of a decision maker, we choose the ratios 1 : 10, 1 : 5, 1 : 1, 5 : 1, and 10 : 1. For each of the resulting weights, we calculate a minmax robust efficient solution x^{rob}_λ obtained by optimizing $(\mathcal{WS}(\lambda))$. Furthermore, we calculate for every scenario $p \in \mathcal{U}$ an optimal solution x^p_λ to the corresponding deterministic multi-objective optimization problem via the deterministic weighted-sum scalarization with weight λ via the optimization problem

$$\min \left(\lambda_1 \cdot f_1(x,p) + \lambda_2 \cdot f_2(x,p)\right). \tag{$\mathcal{WS}_p(\lambda)$}$$

Then, for each $p \in \mathcal{U}$, we compare the two objective values

$$\max_{p' \in \mathcal{U}} \Big(\lambda_1 \cdot f_1(x^{rob}_\lambda, p') + \lambda_2 \cdot f_2(x^{rob}_\lambda, p')\Big) \tag{5.23}$$

(independent of p) and

$$\max_{p' \in \mathcal{U}} \big(\lambda_1 \cdot f_1(x^p_\lambda, p') + \lambda_2 \cdot f_2(x^p_\lambda, p')\big). \tag{5.24}$$

For a given λ, (5.23) is the worst case of the minmax robust efficient solution in the weighted sum and (5.24) is the worst case of x^p_λ in the weighted sum. We chose this comparison strategy for a reason: Usually, in real world applications, some knowledge about the past is available and, therefore, the scenario which seems most likely is used as reference. Since we do not know which scenario is seen as most likely, we assume all scenarios to be equally realistic. Therefore, we compute the optimal solutions to the model in the different scenarios, since those are the solutions most likely to be used in application. However, since our motivation is to hedge against the worst case, we compare these solutions in their respective worst cases (5.24) to the worst case of the minmax robust efficient solution in the same setting (5.23).

The used data sets were obtained from the full-size data, described in Section 5.3.3, by randomized selection leading to 59 (smaller) instances of order sets. Furthermore, 5 different weights $\lambda \in \mathbb{R}^2_+$ were used for the weighted-sum scalarization. We then computed the average and the maximum gain of x^{rob}_λ against x^p_λ for all 59 instances. Due to the simplified problem structure and smaller data sets, the computational time of each of the four problems (5.23) and (5.24) (the latter has to be solved once for each of the three scenarios) decreased to 1.4 seconds per weight where the choice of λ had only marginal impact.

λ	$\begin{pmatrix}1\\0.1\end{pmatrix}$	$\begin{pmatrix}1\\0.5\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}0.5\\1\end{pmatrix}$	$\begin{pmatrix}0.1\\1\end{pmatrix}$
avg gain	1.11	0.79	0.68	0.58	0.37
max gain	35.66	34.31	33.69	32.65	7.94

Table 5.3: Gain of x_λ^{rob} against x_λ^p in percentage.

The computational results for the different weights λ are stated in Table 5.3. On average, the gain of using minmax robust efficient solutions does not seem to matter very much, i.e., the gain ranges from 0.37 % to 1.11 %. More interesting are the results on the maximal gain. Using a minmax robust efficient solution over an optimal solution for some scenario leads in our example to a significant gain of up to 35 %. The minimal gain is omitted in the presentation of the results, since it equals zero for most of the instances.

Furthermore, one of the reasons for the quite low average values is that a lot of the optimal solutions to the different scenarios are also minmax robust efficient solutions themselves. If we neglect those instances for which this is the case and only have a look at the instances for which not all solutions to the different scenarios are also minmax robust efficient, we obtain different results, stated in Table 5.4.

λ	$\begin{pmatrix}1\\0.1\end{pmatrix}$	$\begin{pmatrix}1\\0.5\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}0.5\\1\end{pmatrix}$	$\begin{pmatrix}0.1\\1\end{pmatrix}$
avg gain	5.45	3.87	3.32	2.64	1.45

Table 5.4: Gain of x_λ^{rob} against x_λ^p, for instances for which not all solutions to the different scenarios are also minmax robust efficient, in percentage.

One might argue that comparing (5.23) to (5.24) does not reflect a realistic gain, since in practice the manufacturer does not optimize with respect to an arbitrary scenario, but with respect to a specific scenario. In our case, this is Scenario 1 from Figure 5.3, as this is the scenario based on empirical values as described in Section 5.3. Instead of comparing objective (5.23) to (5.24), we can also compare objective (5.23) to

$$\max_{p'\in\mathcal{U}} \left(\lambda_1 \cdot f_1(x_\lambda^{p_1}, p') + \lambda_2 \cdot f_2(x_\lambda^{p_1}, p')\right), \tag{5.25}$$

where p_1 is Scenario 1 from Figure 5.3 and $x_\lambda^{p_1}$ is an optimal solution to $(\mathcal{WS}_{p_1}(\lambda))$. The results we obtained for all instances are stated in Table 5.5.

As we can see, the benefit of a minmax robust efficient solution against an optimal one is (even though lower than before) quite significant in the maximal case. Furthermore, if we again neglect the instances where the optimal solution to $(\mathcal{WS}_{p_1}(\lambda))$ is not a minmax robust efficient solution itself, we obtain different average values, stated in Table 5.6.

These results strengthen the concept of minmax robust efficiency as they show the advantage of using minmax robust efficient solutions.

λ	$\begin{pmatrix}1\\0.1\end{pmatrix}$	$\begin{pmatrix}1\\0.5\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}0.5\\1\end{pmatrix}$	$\begin{pmatrix}0.1\\1\end{pmatrix}$
avg gain	0.64	0.44	0.39	0.39	0.28
max gain	8.80	7.59	6.60	5.78	3.50

Table 5.5: Gain of x_λ^{rob} against $x_\lambda^{p_1}$ in percentage.

λ	$\begin{pmatrix}1\\0.1\end{pmatrix}$	$\begin{pmatrix}1\\0.5\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}0.5\\1\end{pmatrix}$	$\begin{pmatrix}0.1\\1\end{pmatrix}$
avg gain	4.72	3.22	2.87	2.59	1.36

Table 5.6: Gain of x_λ^{rob} against $x_\lambda^{p_1}$ for instances for which not all solutions to the different scenarios are also minmax robust efficient, in percentage.

5.5 Conclusion and Future Research

In this chapter we presented a real-world optimization problem, namely a cutting problem arising in the veneer industry. We classified the problem, presented a detailed single-objective optimization model, and discussed the uncertainties in the problem formulation. We pointed out that these uncertainties are a result of the varying wood quality, described the various factors influencing this quality, and presented a quality distribution obtained from the experience of the manufacturer. We concluded the deterministic section with computational results of the described problem.

Then we presented a simplified, yet multi-objective version of the optimization problem and discussed the uncertainties in this formulation. In order to hedge against these uncertainties, the concept of minmax robust efficiency was applied to this simplified version and robust efficient solutions to this problem were computed. We discussed and analyzed the results. The results motivate an application of this concept to the original, much more complicated problem.

Summing up, we thoroughly examined a complex real-world cutting problem which was formerly only approached by heuristics and presented methods to solve this problem by deterministic and (in a simplified version) robust optimization, generating applicable solutions. Furthermore, the analysis substantiates the relatively novel concept of minmax robust efficiency and motivates its application to more complex problems.

Bibliography

Ascheuer, N., Krumke, S. O., and Rambau, J. (2000). Online Dial-a-Ride Problems: Minimizing the Completion Time. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 639–650.

Awerbuch, B., Azar, Y., Fiat, A., Leonardi, S., and Rosén, A. (1996). On-line competitive algorithms for call admission in optical networks. In *Proceedings of the 4th Annual European Symposium on Algorithms (ESA)*, volume 1136 of *LNCS*, pages 431–444.

Babaioff, M., Hartline, J. D., and Kleinberg, R. D. (2009). Selling ad campaigns: Online algorithms with cancellations. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, pages 61–70.

Babaioff, M., Immorlica, N., and Kempe, D. (2007). A knapsack secretary problem with applications. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 16–28.

Baeza-Yates, R. A., Culberson, J. C., and Rawlins, G. (1991). Searching in the Plane. *Information and Computation*, 106(2):234–252.

Bansal, N., Buchbinder, N., Madry, A., Joseph, A., and Naor, A. (2011). A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings of the 52nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 267–276.

Bazgan, C., Gourvès, L., and Monnot, J. (2012). Approximation with a Fixed Number of Solutions of Some Biobjective Maximization Problems. In *Proceedings of 9th Workshop on Approximation and Online Algorithms (WAOA)*, volume 7164 of *LNCS*, pages 233–246. Springer.

Beck, A. (1964). On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228.

Beck, A. and Newman, D. J. (1970). Yet more on the linear search problem. *Israel Journal of Mathematics*, 8(4):419–429.

Bellman, R. (1963). An optimal search. *SIAM Review*, 5(3):274.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization.* Princeton University Press.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13.

Benati, S. (1997). An algorithm for a cutting stock problem on a strip. *The Journal of the Operational Research Society*, 48(3):288–294.

Bender, M. and Westphal, S. (2013). An optimal randomized online algorithm for the k-Canadian Traveller Problem on node-disjoint paths. *Journal of Combinatorial Optimization*, Published online: 07 June 2013:1–10.

Birge, J. and Louveaux, F. (2011). *Introduction to Stochastic Programming.* Springer.

Borodin, A. and El-Yaniv, R. (1998). *Online Computation and Competitive Analysis.* Cambridge University Press.

Branke, J. (1998). Creating robust solutions by means of evolutionary algorithms. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 119–128.

Chrobak, M., Karloff, H., Payne, T., and Vishwanathan, S. (1990). New results on server problems. In *SIAM Journal on Discrete Mathematics*, pages 291–300.

Chrobak, M. and Larmore, L. L. (1996). An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20:144–148.

Deb, K. and Gupta, H. (2006). Introducing robustness in multi-objective optimization. *Evolutionary Computation*, 14(4):463–494.

Demaine, E. D., Fekete, S. P., and Shmuel, G. (2006). Online searching with turn cost. *Theoretical Computer Science*, 361(2-3):342–355.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159.

Ehrgott, M. (2005). *Multicriteria Optimization.* Springer.

Ehrgott, M., Ide, J., and Schöbel, A. (2014). Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, 239:17–31.

El-Yaniv, R., Fiat, A., Karp, R. M., and Turpin, G. (2001). Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139.

El-Yaniv, R., Turpin, G., Karp, R., and Fiat, A. (1992). Competitive analysis of financial games. In *Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 327–333.

Erera, A., Morales, J., and Savelsbergh, M. (2009). Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483.

Erlebach, T., Kellerer, H., and Pferschy, U. (2002). Approximating Multiobjective Knapsack Problems. *Management Sciences*, 48(12):1603–1612.

Fiat, A., Foster, D. P., Karloff, H., Rabani, Y., Ravid, Y., and Vishwanathan, S. (1998). Competitive Algorithms for Layered Graph Traversal. *SIAM Journal on Computing*, 28(2):447–462.

Fiat, A. and Woeginger, G., editors (1998). *Online Algorithms - The State of the Art*, volume 1442 of *LNCS*. Springer.

Fischetti, M. and Monaci, M. (2009). Light robustness. In *Robust and Online Large-scale Optimization*, pages 61–84. Springer.

Goerigk, M. and Schöbel, A. (2011). A scenario-based approach for robust linear optimization. In *Theory and Practice of Algorithms in (Computer) Systems*, pages 139–150.

Hajiaghayi, M. T., Kleinberg, R. D., Mahdian, M., and Parkes, D. C. (2005). Online auctions with re-usable goods. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, pages 165–174.

Hauptmeier, D., Krumke, S. O., and Rambau, J. (2000). The Online Dial-a-Ride Problem under Reasonable Load. In *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, pages 125–136.

Ibarra, O. H. and Kim, C. E. (1975). Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468.

Ide, J., Tiedemann, M., Westphal, S., and Haiduk, F. (2015). An application of deterministic and robust optimization in the wood cutting industry. *4OR*, 13(1):35–57.

Iwama, K. and Taketomi, S. (2002). Removable online knapsack problems. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 293–305.

Iwama, K. and Zhang (2003). Removable online knapsack - weighted case. In *Proceedings of the 7th Japan-Korea Workshop on Algorithms and Computation (WAAC)*, pages 223–227.

Iwama, K. and Zhang, G. (2007). Optimal resource augmentations for online knapsack. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 180–188.

Kalyanasundaram, B. and Pruhs, K. (2000). Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643.

Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. (1988). Competitive snoopy caching. *Algorithmica*, 3(1-4):79–119.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems.* Springer.

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

Koutsoupias, E. and Papadimitriou, C. (1995). On the k-server conjecture. *Journal of the ACM*, 42:507–511.

Koutsoupias, E. and Papadimitriou, C. (1996). The 2-evader problem. *Information Processing Letters*, 57(5):473–482.

Kouvelis, P. and Yu, G. (1997). *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers, Dordrecht.

Liebchen, C., Lübbecke, M., Möhring, R., and Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and Online Large-Scale Optimization*, pages 1–27. Springer.

Lohmann, U. (2003). *Holzlexikon (in German).* DRW-Verlag Weinbrenner GmbH & Co., Leinefelden-Echterdingen.

Lueker, G. S. (1998). Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305.

Manasse, M., McGeoch, L., and Sleator, D. (1988). Competitive algorithms for on-line problems. In *Proceedings of the 20th ACM Symposium on the Theory of Computing (STOC)*, pages 322–333.

Manasse, M. S., McGeoch, L. A., and Sleator, D. D. (1990). Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230.

Marchetti-Spaccamela, A. and Vercellis, C. (1995). Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1-3):73–104.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester.

Motwani, R., Phillips, S., and Torng, E. (1994). Non-clairvoyant scheduling. *Theoretical Computer Science*, 130:17–47.

Noga, J. and Sarbua, V. (2005). An online partially fractional knapsack problem. In *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 108–112.

Papadimitriou, C. H. and Yannakakis, M. (1991). Shortest paths without a map. *Theoretical Computer Science*, 84:127–150.

Papadimitriou, C. H. and Yannakakis, M. (2000). On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 86–92.

Papastavrou, J. D., Rajagopalan, S., and Kleywegt, A. J. (1996). The dynamic and stochastic knapsack problem with deadlines. *Management Sciences*, 42(12):1706–1718.

Pareto, V. (1896). *Manuel d'économie politique (in French).* F. Rouge, Lausanne.

Phillips, C. A., Stein, C., Torng, E., and Wein, J. (1997). Optimal time-critical scheduling via resource augmentation. In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, pages 140–149.

Ràcz, J. (1961). *Untersuchungen über das Auftreten des Buchenrotkerns in Niedersachsen (in German).* PhD thesis, University of Göttingen, Faculty of Forest Sciences.

Schöbel, A. (2014). Generalized light robustness and the trade-off between robustness and nominal quality. *Mathematical Methods of Operations Research*, 80(2):161–191.

Soyster, A. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157.

Thielen, C., Tiedemann, M., and Westphal, S. (2015). The online knapsack problem with incremental capacity. *Mathematical Methods of Operations Research.* Submitted.

Tiedemann, M. (2015). A note on online searching with turn cost. *Information Processing Letters.* Submitted.

Tiedemann, M., Ide, J., and Schöbel, A. (2015). Competitive analysis for multi-objective online algorithms. In *Proceedings of the 9th Workshop on Algorithms and Computations (WALCOM)*, volume 8973 of *LNCS*, pages 210–221.

van Slyke, R. and Young, Y. (2000). Finite horizon stochastic knapsacks with applications to yield management. *Operations Research*, 48(1):155–172.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130.

Westphal, S. (2008). A note on the k-Canadian Traveller Problem. *Information Processing Letters*, 106(3):87–89.

Yao, A. C. (1977). Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 222–227.

Zhiping, C., Hurkens, C., and Jong, J. (1997). A branch-and-price algorithm for solving the cutting strips problem. *Applied Mathematics - A Journal of Chinese Universities*, 12(2):215–224.

Zhou, Y., Chakrabarty, D., and Lukose, R. (2008). Budget constrained bidding in keyword auctions and online knapsack problems. In *Proceedings of the 4th Workshop on Internet and Network Economics (WINE)*, pages 566–576.